JUNIPER NETWORKS

Network Configuration Example

Configuring Policy-Based VPNs Using J Series Routers and SRX Series Devices

Release

10.4

Published: 2010-10-08
Revision 1

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos® OS Network Configuration Example Configuring Policy-Based VPNs using J Series Routers and SRX Series Devices
Release 10.4

Writing: Richard Kim
Editing: Roy Spencer, Katie Smith
Illustration: Faith Bradford
Cover Design: Edmonds Design

Revision History
October 2010—R1 Junos 10.4

The information in this document is current as of the date listed in the revision history.

YEAR 2000 NOTICE

Juniper Networks hardware and software products are Year 2000 compliant. The 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

READ THIS END USER LICENSE AGREEMENT ("AGREEMENT") BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. **The Parties.** The parties to this Agreement are (i) Juniper Networks, Inc. (if the Customer's principal office is located in the Americas) or Juniper Networks (Cayman) Limited (if the Customer's principal office is located outside the Americas) (such applicable entity being referred to herein as "Juniper"), and (ii) the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software ("Customer") (collectively, the "Parties").

2. **The Software.** In this Agreement, "Software" means the program modules and features of the Juniper or Juniper-supplied software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller, or which was embedded by Juniper in equipment which Customer purchased from Juniper or an authorized Juniper reseller. "Software" also includes updates, upgrades and new releases of such software. "Embedded Software" means Software which Juniper has embedded in or loaded onto the Juniper equipment and any updates, upgrades, additions or replacements which are subsequently embedded in or loaded onto the equipment.

3. **License Grant.** Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

a. Customer shall use Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.

b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius or IMS AAA software on multiple computers or virtual machines (e.g., Solaris zones) requires multiple licenses, regardless of whether such computers or virtualizations are physically contained on a single chassis.

c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer's use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface, processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer's use of the Software shall be subject to all such limitations and purchase of all applicable licenses.

d. For any trial copy of the Software, Customer's right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.

e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer's enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. **Use Prohibitions.** Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any 'locked' or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the

Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use Embedded Software on non-Juniper equipment; (j) use Embedded Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.

5. **Audit.** Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. **Confidentiality.** The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.

7. **Ownership.** Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. **Warranty, Limitation of Liability, Disclaimer of Warranty.** The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.

9. **Termination.** Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. **Taxes.** All license fees payable under this agreement are exclusive of tax. Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software. If applicable, valid exemption documentation for each taxing jurisdiction shall be provided to Juniper prior to invoicing, and Customer shall promptly notify Juniper if their exemption is revoked or modified. All payments made by Customer shall be net of any applicable withholding tax. Customer will provide reasonable assistance to Juniper in connection with such withholding taxes by promptly: providing Juniper with valid tax receipts and other required documentation showing Customer's payment of any withholding taxes; completing appropriate applications that would reduce the amount of withholding tax to be paid; and notifying and assisting Juniper in any audit or tax proceeding related to transactions hereunder. Customer shall comply with all applicable tax laws and regulations, and Customer will promptly pay or reimburse Juniper for all costs and damages related to any liability incurred by Juniper as a result of Customer's non-compliance or delay with its responsibilities herein. Customer's obligations under this Section shall survive termination or expiration of this Agreement.

11. **Export.** Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.

12. **Commercial Computer Software.** The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14(ALT III) as applicable.

13. **Interface Information.** To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.

14. **Third Party Software.** Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at http://www.gnu.org/licenses/gpl.html, and a copy of the LGPL at http://www.gnu.org/licenses/lgpl.html .

15. **Miscellaneous.** This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentés confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattaché, soient redigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).

Table of Contents

Policy-Based VPNs Using J Series Routers and SRX Series Devices Overview

Junos OS, the software which runs on J Series and SRX Series devices, provides not only a powerful operating system, but also a rich IP services toolkit. Junos OS ensures an efficient and predictable IP infrastructure through unmatched IP dependability and security. Junos OS has been greatly enhanced with security and VPN capabilities from Juniper Networks Firewall/IPsec VPN Platforms that include the Secure Services Gateway (SSG) product family. This document provides detailed information on IPsec interoperability configuration between a J Series router or SRX Series device (Junos OS device) and an SSG device. This application note also provides troubleshooting information for Junos OS.

The configuration of a Junos OS routing/security device for virtual private network (VPN) support is particularly flexible. You can create route-based and policy-based VPN tunnels. This document focuses on policy-based VPN tunnels.

This document is intended for network design and security engineers, as well as anyone who requires secure connectivity over public networks.

Related Documentation

- Comparing Policy-Based and Route-Based VPNs on page 3
- Example: Configuring Policy-Based VPNs Using J Series Routers and SRX Series Devices on page 5

Comparing Policy-Based and Route-Based VPNs

It is important to understand the differences between policy-based and route-based VPNs and why one may be preferable to the other. With policy-based VPN tunnels, a tunnel is treated as an object that, together with source, destination, application, and action, constitutes a tunnel policy that permits VPN traffic. In a policy-based VPN configuration, a tunnel policy specifically references a VPN tunnel by name. With route-based VPNs, a policy does not specifically reference a VPN tunnel. Instead, the policy references a destination address. When the security device does a route lookup to find the interface through which it must send traffic to reach that address, it finds a route via a secure tunnel (st0) interface, which is bound to a specific VPN tunnel. Thus, with a policy-based VPN tunnel, you can consider a tunnel as an element in the construction of a policy. With a route-based VPN tunnel, you can consider a tunnel as a means for delivering traffic, and the policy as a method for either permitting or denying the delivery of that traffic.

The number of policy-based VPN tunnels that you can create is limited by the number of policies that the device supports. The number of route-based VPN tunnels that you create is limited by the number of route entries or the number of st0 interfaces that the device supports—whichever number is lower. A route-based VPN tunnel configuration is the preferred choice when you want to conserve tunnel resources while setting granular restrictions on VPN traffic. With a policy-based VPN, although you can create numerous tunnel policies referencing the same VPN tunnel, each tunnel policy pair creates an individual IPsec security association (SA) with the remote peer. Each SA counts as an individual VPN tunnel. With a route-based approach to VPNs, the regulation of traffic is not coupled to the means of its delivery. You can configure dozens of policies to regulate traffic flowing through a single VPN tunnel between two sites, and there is just one IPsec SA at work. Also, a route-based VPN configuration allows you to create policies referencing a destination reached through a VPN tunnel in which the action is deny. This is unlike a policy-based VPN configuration, in which the action must be permit and must include tunnel.

Another advantage that route-based VPNs offer is the exchange of dynamic routing information through VPN tunnels. This is not supported with policy-based VPNs. Also for hub-and-spoke topologies, you must use route-based configuration. In addition, you can define static Network Address Translation (NAT) addresses specifically for st0 interfaces. OSPF, hub-and-spoke, and static NAT configurations are beyond the scope of this document.

When a tunnel does not connect large networks running dynamic routing protocols and when you do not need to conserve tunnels or define various policies to filter traffic through the tunnel, a policy-based tunnel is ideal. Also, because there is no network beyond a dial-up VPN client, policy-based VPN tunnels can be a preferred choice for dial-up VPN configurations. Finally, for interoperability with third-party vendors which do not support the concept of route-based VPNs, policy-based VPNs may be absolutely required especially if the third party requires separate SAs for each remote subnet.

For the purposes of this network configuration example, the focus is on policy-based VPN configuration and troubleshooting. Additional Junos OS specific application notes

can be found on Juniper Networks' Knowledge Base at http://kb.juniper.net. In particular, article KB10182 lists several application notes related to VPN configuration and troubleshooting.

Related Documentation

- Policy-Based VPNs Using J Series Routers and SRX Series Devices Overview on page 1
- Example: Configuring Policy-Based VPNs Using J Series Routers and SRX Series Devices on page 5

Example: Configuring Policy-Based VPNs Using J Series Routers and SRX Series Devices

This topic includes the following sections:

Requirements

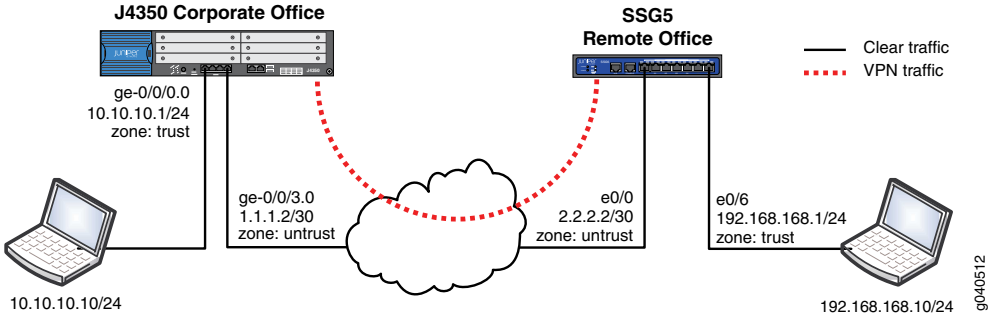
This document applies to the following devices:

- J Series routers running:
  - Junos OS 9.4 and above
  - Junos OS with Enhanced Services 8.5 through 9.3
- SRX Series Services Gateways

Overview and Topology

Refer to Figure 1 on page 5 for the network topology used for this configuration example.

Figure 1: Network Topology



This example assumes the following:

- The Internal LAN interface is **ge-0/0/0** in zone **trust** and has a private IP subnetwork address.
- The Internet interface is **ge-0/0/3** in zone **untrust** and has a public IP subnetwork address.
- All traffic between the local and remote LANs are permitted, and traffic can be initiated from either side.
- The SSG5 is preconfigured with the correct information given in this example.

The basic steps for configuring Junos OS for policy-based VPNs are:

1. Configure the IP addresses for Gigabit Ethernet interfaces: **ge-0/0/0.0** and **ge-0/0/3.0.**
2. Configure the default route to the Internet next hop. Optionally, you can use a dynamic routing protocol such as OSPF instead. Configuring OSPF is beyond the scope of this document.
3. Configure security zones, and bind the interfaces to the appropriate zones. Also ensure that you have enabled the necessary host-inbound services on the interfaces or the zone. For this example, you must enable Internet Key Exchange (IKE) service on either the **ge-0/0/3** interface or the **untrust** zone.
4. Configure address book entries for each zone. This is necessary for the security policies.
5. Configure phase 1 (IKE) gateway settings.

NOTE: For this example, the **Standard** proposal set is used. However, you can create a different proposal if necessary.

6. Configure phase 2 (IP Security [IPsec]) VPN settings. Optionally, you can also configure VPN monitor settings if you wish.

NOTE: For this example, the **Standard** proposal set and **Perfect Forward Secrecy (PFS) group 2** are used. However, you can create a different proposal if necessary.

7. Configure tunnel policies to permit remote office traffic into the corporate LAN and vice versa. Also configure an outgoing **trust** to **untrust** permit-all policy with source NAT for Internet traffic. Ensure that the tunnel policy is above the permit-all policy. Otherwise, the policy lookup never reaches the tunnel policy.
8. Configure the TCP-maximum segment size (tcp-mss) for IPsec traffic to eliminate the possibility of fragmented TCP traffic. This will lessen the resource usage on the device.

Configuration

Configuring Junos OS

Step-by-Step Procedure

To configure Junos OS, perform the following steps:

1. Configure interface IP addresses. Junos OS uses the concept of units for the logical component of an interface. In this example, **unit 0** and **family inet (IPv4)** are used.

```
user@CORPORATE# set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.1/24
user@CORPORATE# set interfaces ge-0/0/3 unit 0 family inet address 1.1.1.2/30
```

2. Configure a default route. When processing the first packet of a new session, the Junos OS device first performs a route lookup. The static route, which happens to be the default route, determines the zone that the VPN traffic needs to egress. In this example, the VPN traffic ingresses on interface **ge-0/0/0.0** with the next hop of **1.1.1.1**. Thus, the traffic egresses out interface **ge-0/0/3.0**. Any tunnel policy needs to take into account the ingress and egress interfaces.

   user@CORPORATE# **set routing-options static route 0.0.0.0/0 next-hop 1.1.1.1**

3. Configure security zones, and assign interfaces to the zones. The ingress and egress zones are determined by the ingress and egress interfaces involved in the route lookup. From steps 1 and 2, you can see that packets ingressing on **ge-0/0/0** and that the ingress zone is the **trust** zone. Following the route lookup, the egress interface is **ge-0/0/3**, which signifies that the egress zone is **untrust** zone. Thus, the tunnel policy needs to be from **from-zone trust to-zone untrust** and vice versa.

   user@CORPORATE# **set security zones security-zone trust interfaces ge-0/0/0.0**
   user@CORPORATE# **set security zones security-zone untrust interfaces ge-0/0/3.0**

4. Configure host-inbound services for each zone. Host-inbound services are for traffic destined for the Junos OS device itself. This includes but is not limited to FTP, HTTP, HTTPS, IKE, ping, rlogin, RSH, SNMP, SSH, Telnet, TFTP, and traceroute. For this example, assume that you want to allow all such services from zone **trust**. For security reasons, allow only IKE on the Internet facing zone **untrust**, which is required for IKE negotiations to occur. However, other services such as management and/or troubleshooting can also be individually enabled if required.

   user@CORPORATE# **set security zones security-zone trust host-inbound-traffic system-services all**
   user@CORPORATE# **set security zones security-zone untrust host-inbound-traffic system-services ike**

5. Configure address book entries for each zone. This example uses the address book object names **local-net** and **remote-net**. There are some limitations with regard to the characters that are supported for address book names. Please refer to the complete Junos OS documentation for more details.

   user@CORPORATE# **set security zones security-zone trust address-book address local-net 10.10.10.0/24**
   user@CORPORATE# **set security zones security-zone untrust address-book address remote-net 192.168.168.0/24**

6. Configure the IKE policy for main mode, Standard proposal set, and preshared key. This application note uses proposal set **Standard**, which includes **preshared-group2-3des-sha1** and **preshared-group2-aes128-sha1** proposals. However, a unique proposal may be created and specified in the IKE policy in accordance with your corporate security policy.

   user@CORPORATE# **set security ike policy ike-policy1 mode main**
   user@CORPORATE# **set security ike policy ike-policy1 proposal-set standard**
   user@CORPORATE# **set security ike policy ike-policy1 pre-shared-key ascii-text "secretkey"**

7. Configure the IKE gateway (phase 1) with a peer IP address, IKE policy, and outgoing interface. A remote IKE peer can be identified by **IP address**, fully qualified domain name/user-fully qualified domain name (**FQDN/u-FQDN**) or **ASN1-DN** (PKI

certificates). For this example, identify the peer by IP address. The gateway address should be the remote peer's public IP address. It is important to specify the correct external interface. If either the peer address or external interface specified is incorrect, then the IKE gateway will not be properly identified during phase 1 negotiations.

```
user@CORPORATE# set security ike gateway ike-gate ike-policy ike-policy1
user@CORPORATE# set security ike gateway ike-gate address 2.2.2.2
user@CORPORATE# set security ike gateway ike-gate external-interface ge-0/0/3.0
```

8. Configure an IPsec policy for the Standard proposal set. As mentioned for phase 1, for the purposes of this application note, the **Standard** proposal set is used, which includes the **esp-group2- 3des-sha1** and **esp-group2-aes128-sha1** proposals. However, a unique proposal may be created and then specified in the IPsec policy if needed.

```
user@CORPORATE# set security ipsec policy vpn-policy1 proposal-set standard
```

9. Configure an IPsec VPN with an IKE gateway and an IPsec policy. For this example, the VPN name **ike-vpn** needs to be referenced in the security policy to be able to create a security association.

```
user@CORPORATE# set security ipsec vpn ike-vpn ike gateway ike-gate
user@CORPORATE# set security ipsec vpn ike-vpn ike ipsec-policy vpn-policy1
```

10. Configure VPN bidirectional security policies for tunnel traffic. For this example, traffic from the corporate LAN to the remote office LAN requires a **from-zone trust to-zone untrust** tunnel policy. However, if a session needs to originate from the remote LAN to the corporate LAN, then a tunnel policy in the opposite direction **from-zone untrust to-zone trust** is also needed. By including the **pair-policy** statement, the VPN becomes bidirectional. Enter the zone **trust** to zone **untrust** hierarchy.

NOTE:

- In addition to the permit action, you need to specify the IPsec profile to be used. Source NAT can be enabled on the policy if desired, but that is beyond the scope of this document.
- For tunnel policies, the action is always permit. If you are configuring a policy with the deny action, you will not see an option for specifying the tunnel.

```
user@CORPORATE# edit security policies from-zone trust to-zone untrust
user@CORPORATE# set policy vpnpolicy-tr-unt match source-address local-net
user@CORPORATE# set policy vpnpolicy-tr-unt match destination-address
  remote-net
user@CORPORATE# set policy vpnpolicy-tr-unt match application any
user@CORPORATE# set policy vpnpolicy-tr-unt then permit tunnel ipsec-vpn ike-vpn
user@CORPORATE# set policy vpnpolicy-tr-unt then permit tunnel pair-policy
  vpnpolicy-unt-tr
```

11. Enter the zone **untrust** to zone **trust** hierarchy.

```
user@CORPORATE# edit security policies from-zone untrust to-zone trust
user@CORPORATE# set policy vpnpolicy-unt-tr match source-address remote-net
user@CORPORATE# set policy vpnpolicy-unt-tr match destination-address local-net
user@CORPORATE# set policy vpnpolicy-unt-tr match application any
```

```
user@CORPORATE# set policy vpnpolicy-unt-tr then permit tunnel ipsec-vpn ike-vpn
user@CORPORATE# set policy vpnpolicy-unt-tr then permit tunnel pair-policy
  vpnpolicy-tr-unt
```

12. Configure a security policy for Internet traffic. Enter the from-zone trust to-zone untrust hierarchy.

```
user@CORPORATE# edit security policies from-zone trust to-zone untrust
user@CORPORATE# set policy any-permit match source-address any
user@CORPORATE# set policy any-permit match destination-address any
user@CORPORATE# set policy any-permit match application any
user@CORPORATE# set policy any-permit then permit source-nat interface
exit
```

This policy permits all traffic from zone **trust** to zone **untrust**. With **source-nat interface** specified, the device translates the source IP and port for outgoing traffic, using the IP address of the egress interface as the source IP address and a random higher port for the source port. If required, more granular policies can be created to permit/deny certain traffic. Note that this policy must be below the VPN policy because the policy list is read from top to bottom. If this policy is above the VPN policy, then the traffic always matches this policy and does not continue to the next policy. Thus, no user traffic is encrypted. To move the VPN policy, use the **insert** command:

```
user@CORPORATE# edit security policies from-zone trust to-zone untrust
[edit security policies from-zone trust to-zone untrust]
user@CORPORATE# insert policy vpnpolicy-tr-unt before policy any-permit
```

13. Configure the tcp-mss to eliminate fragmentation of TCP traffic across the tunnel. Tcp-mss is negotiated as part of the TCP three-way handshake. It limits the maximum size of a TCP segment to better fit the maximum transmission unit (MTU) limits of a network. This is especially important for VPN traffic because the IPsec encapsulation overhead, along with the IP and frame overhead, can cause the resulting Encapsulating Security Payload (ESP) packet to exceed the MTU of the physical interface, thereby causing fragmentation. Fragmentation increases bandwidth and device resource usage and is always best avoided.

```
user@CORPORATE# set security flow tcp-mss ipsec-vpn mss 1350
```

NOTE: The value of 1350 is a recommended starting point for most Ethernet-based networks with an MTU of 1500 or greater. This value may need to be altered if any device in the path has a lower MTU and/or if there is any added overhead such as PPP or Frame Relay. As a general rule, you may need to experiment with different tcp-mss values to obtain optimal performance.

14. This is an SSG configuration example. The focus of this application note is on Junos OS configuration and troubleshooting. For the purpose of completing the network topology diagram (see Figure 1 on page 5), a sample of the relevant configurations is provided for an SSG5 device. However, the concepts for configuration of policy-based VPNs for Juniper Networks Firewall/VPN products are well documented

in the Concepts and Examples (C&E) guides. For reference, the *SSG C&E guides* can be found at: http://www.juniper.net/techpubs/software/screenos/.

```
user@CORPORATE# set interface ethernet0/6 zone "Trust"
user@CORPORATE# set interface ethernet0/0 zone "Untrust"
user@CORPORATE# set interface ethernet0/6 ip 192.168.168.1/24
user@CORPORATE# set interface ethernet0/6 route
user@CORPORATE# set interface ethernet0/0 ip 2.2.2.2/30
user@CORPORATE# set interface ethernet0/0 route
user@CORPORATE# set flow tcp-mss 1350
user@CORPORATE# set address "Trust" "local-net" 192.168.168.0 255.255.255.0
user@CORPORATE# set address "Untrust" "corp-net" 10.10.10.0 255.255.255.0
user@CORPORATE# set ike gateway "corp-ike" address 1.1.1.2 Main
  outgoing-interface ethernet0/0 preshare "secretkey" sec-level standard
user@CORPORATE# set vpn "corp-vpn" gateway "corp-ike" replay tunnel idletime
  0 sec-level standard
user@CORPORATE# set policy id 11 from "Trust" to "Untrust" "local-net" "corp-net"
  "ANY" tunnel vpn "corp-vpn" pair-policy 10
user@CORPORATE# set policy id 10 from "Untrust" to "Trust" "corp-net" "local-net"
  "ANY" tunnel vpn "corp-vpn" pair-policy 11
user@CORPORATE# set policy id 1 from "Trust" to "Untrust" "ANY" "ANY" "ANY"
  nat src permit
user@CORPORATE# set route 0.0.0.0/0 interface ethernet0/0 gateway 2.2.2.1
```

Verifying VPN Connections for Policy-Based VPNs

Step-by-Step Procedure

To verify VPN Connections for Policy-Based VPNs, perform the following steps:

1. Confirm IKE (phase 1) status. The remote peer is **2.2.2.2**. The state shows **UP**. If the state shows **DOWN** or if there are no IKE security associations present, then there is a problem with phase 1 establishment. Confirm that the remote IP address, IKE policy, and external interfaces are all correct. Common errors include incorrect IKE policy parameters such as incorrect mode type (aggressive or main), preshared keys or phase 1 proposals (all must match on the peers). An incorrect external interface is another common misconfiguration. This interface must be the correct interface to receive the IKE packets. If configurations have been checked, then check the kmd log for any errors, or run traceoptions (see “Troubleshooting” on page 14).

   ```
   user@CORPORATE> show security ike security-associations

   Index Remote Address State Initiator cookie Responder cookie Mode
   4 2.2.2.2 UP 5e1db3f9d50b0de6 e50865d9ebf134f8 Main
   ```

2. In the following show command output, note that the Index number is 4. This value is unique for each IKE security association and allows you to get more details from that particular security association. The detail option gives more information that includes the role (initiator or responder). This is useful to know because troubleshooting is usually best done on the peer that has the responder role. Also shown are details regarding the authentication and encryption algorithms used, the phase 1 lifetime, and the traffic statistics. Traffic statistics can be used to verify that traffic is flowing properly in both directions. Also note the number of IPsec security associations created or in progress. This helps to determine the existence of any completed phase 2 negotiations.

   ```
   user@CORPORATE> show security ike security-associations index 4 detail
   ```



```
IKE peer 2.2.2.2, Index 4,
Role: Responder, State: UP
Initiator cookie: 5e1db3f9d50b0de6, Responder cookie: e50865d9ebf134f8
Exchange type: Main, Authentication method: Pre-shared-keys
Local: 1.1.1.2:500, Remote: 2.2.2.2:500
Lifetime: Expires in 28770 seconds
Algorithms:
Authentication : sha1
Encryption : 3des-cbc
Pseudo random function: hmac-sha1
Traffic statistics:
Input bytes : 852
Output bytes : 856
Input packets: 5
Output packets: 4
Flags: Caller notification sent
IPsec security associations: 1 created, 0 deleted
Phase 2 negotiations in progress: 0
```

3. Confirm IPsec (phase 2) status. From steps 1 and 2, you can see that there is one IPsec security association (SA) pair and that the port used is **500**, which means there is no NAT traversal (nat-traversal would show port 4500 or a random high port). Also, you can see the security parameter index (SPI) used for both directions, as well as the lifetime (in seconds) and usage limits or lifesize (in kilobytes). In the following output, you can see **3565/ unlim**, which means that phase 2 lifetime is set to expire in 3565 seconds. There is no lifesize specified; thus, it shows unlimited (unlim). Phase 2 lifetime can differ from phase 1 lifetime because phase 2 is not dependent on phase 1 after the VPN is up. The **Mon** column refers to VPN monitoring status. If VPN monitoring is enabled, then this shows U (up) or D (down). A hyphen (-) means that VPN monitoring is not enabled for this SA. For more details on VPN monitoring, refer to the complete Junos OS documentation. Note that Vsys always shows 0. Note also the ID number 2. This is the Index value and is unique for each IPsec security association.

```
user@CORPORATE> show security ipsec security-associations

total configured sa: 2
ID Gateway Port Algorithm SPI Life:sec/kb Mon vsys
<2 2.2.2.2 500 ESP:3des/sha1 a63eb26f 3565/ unlim - 0
>2 2.2.2.2 500 ESP:3des/sha1 a1024ed9 3565/ unlim - 0
```

4. In the following show command output, you can view more details for a particular security association. The following output shows the **Local Identity** and **Remote Identity**. These elements compose the proxy ID for this SA. Proxy ID mismatch is a very common reason for phase 2 failing to complete. For policy-based VPNs, the proxy ID is derived from the security policy. From the security policy, the local address and remote address are derived from the address book entries, and the service is derived from the application configured for the policy. If phase 2 fails due to a proxy ID mismatch, confirm from the policy which address book entries are configured, and verify the addresses to confirm that they match what is being sent. Also, verify the service to ensure that the ports match what is being sent.

   Note that if multiple objects are configured in a policy for source address, destination address, or application, then the resulting proxy ID for that parameter changes to zeroes. For example, assume the tunnel policy has multiple local addresses of

10.10.10.0/24 and **10.10.20.0/24**, remote address **192.168.168.0/24**, and application junos-http. The resulting proxy ID would be local **0.0.0.0/0**, remote **192.168.168.0/24**, service **80**. This can affect interoperability if the remote peer is not configured for the second subnet.

For certain third-party vendors, you may need to manually enter the proxy ID to match. If IPsec cannot complete, then check the kmd log or set traceoptions as detailed in "Troubleshooting" on page 14.

```
user@CORPORATE> show security ipsec security-associations index 2 detail

Virtual-system: Root
Local Gateway: 1.1.1.2, Remote Gateway: 2.2.2.2
Local Identity: ipv4_subnet(any:0,[0..7]=10.10.10.0/24)
Remote Identity: ipv4_subnet(any:0,[0..7]=192.168.168.0/24)
DF-bit: clear
Policy-name: vpnpolicy-unt-tr
Direction: inbound, SPI: 2789126767, AUX-SPI: 0
Hard lifetime: Expires in 3558 seconds
Lifesize Remaining: Unlimited
Soft lifetime: Expires in 2986 seconds
Mode: tunnel, Type: dynamic, State: installed, VPN Monitoring: -
Protocol: ESP, Authentication: hmac-sha1-96, Encryption: 3des-cbc
Anti-replay service: enabled, Replay window size: 32
Direction: outbound, SPI: 2701283033, AUX-SPI: 0
Hard lifetime: Expires in 3558 seconds
Lifesize Remaining: Unlimited
Soft lifetime: Expires in 2986 seconds
Mode: tunnel, Type: dynamic, State: installed, VPN Monitoring: -
Protocol: ESP, Authentication: hmac-sha1-96, Encryption: 3des-cbc
Anti-replay service: enabled, Replay window size: 32
```

5. In the following show command output, check the statistics and errors for an IPsec SA. This command is used to check Encapsulating Security Payload (ESP) and Authentication Header (AH) counters and to check for any errors with a particular IPsec security association. You normally do not want to see error values other than zero. However if you experience packet loss issues across a VPN, one approach is to use the show command multiple times and confirm that the encrypted and decrypted packet counters are incrementing. Also, verify whether any error counter increments while you are experiencing the issue. It may also be necessary to enable security flow traceoptions (see "Troubleshooting" on page 14) to view which ESP packets are experiencing errors and why.

```
user@CORPORATE> show security ipsec statistics index 2

ESP Statistics:
Encrypted bytes: 920
Decrypted bytes: 6208
Encrypted packets: 5
Decrypted packets: 87
AH Statistics:
Input bytes: 0
Output bytes: 0
Input packets: 0
Output packets: 0
Errors:
AH authentication failures: 0, Replay errors: 0
ESP authentication failures: 0, ESP decryption failures: 0
Bad headers: 0, Bad trailers: 0
```


6. Test the traffic flow across the VPN. After you have confirmed the status of phase 1 and phase 2, the next step is to test the traffic flow across the VPN. One way to test the traffic flow is through the **ping** command. You can ping from a local host PC to a remote host PC. You can also initiate the **ping** command from the Junos OS device itself. The following is an example of testing using the **ping** command from the Junos OS device to the remote PC host. Note that when initiating ping packets from the Junos OS device, the source interface needs to be specified in order to be sure that route lookup is correct and that the appropriate zones can be referenced in policy lookup. In this case, because **ge-0/0/0.0** resides in the same security zone as the local host PC, **ge-0/0/0** needs to be specified in the **ping** commands so that the policy lookup can be from zone **trust** to zone **untrust**. Similarly, you can initiate a **ping** command from the remote host to the local host.

```
user@CORPORATE> ping 192.168.168.10 interface ge-0/0/0 count 5


PING 192.168.168.10 (192.168.168.10): 56 data bytes
64 bytes from 192.168.168.10: icmp_seq=0 ttl=127 time=8.287 ms
64 bytes from 192.168.168.10: icmp_seq=1 ttl=127 time=4.119 ms
64 bytes from 192.168.168.10: icmp_seq=2 ttl=127 time=5.399 ms
64 bytes from 192.168.168.10: icmp_seq=3 ttl=127 time=4.361 ms
64 bytes from 192.168.168.10: icmp_seq=4 ttl=127 time=5.137 ms
--- 192.168.168.10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 4.119/5.461/8.287/1.490 ms
```

7. You can also initiate a **ping** command from the SSG5 device itself, as shown in the following output. If pings fail from either direction, this could indicate an issue with routing, policy, end host, or perhaps an issue with the encryption/decryption of the ESP packets. One way to check is to view the IPsec statistics to see whether any errors are reported. Also, you can confirm end host connectivity by pinging from a host on the same subnet as the end host. Assuming that the end host is reachable by other hosts, the issue probably is not with the end host. For routing and policy issues, you can enable security flow traceoptions.

```
ssg5-> ping 10.10.10.10 from ethernet0/6
Type escape sequence to abort
Sending 5, 100-byte ICMP Echos to 10.10.10.10, timeout is 1 seconds from
 ethernet0/6
!!!!!
Success Rate is 100 percent (5/5), round-trip time min/avg/max=4/4/5 ms
```

Troubleshooting

Step-by-Step Procedure

Basic troubleshooting begins by first isolating the issue and then focusing the debugging efforts on the area where the problem is occurring. One common approach is to start with the lowest layer of the Open System Interconnection (OSI) model and work up the OSI stack to confirm at which layer the failure occurs.

Following this methodology, the first step to troubleshooting is to confirm the physical connectivity of the Internet link at the physical and data link level. Next, by using the **ping** command, confirm that the Junos OS device has connectivity to the Internet next hop, followed by confirming connectivity to the remote IKE peer. If this is confirmed, then confirm that IKE phase 1 can complete by running the verification commands. After phase 1 is confirmed, confirm phase 2. Finally, confirm that traffic is flowing across the VPN. If the VPN is not in the **UP** state, then there is little reason to test any transit traffic across the VPN. Likewise, if phase 1 is not successful, then it is unnecessary to look at phase 2 issues.

To troubleshoot issues further at the different levels, configure traceoptions. Traceoptions are enabled in configuration mode and are a part of the Junos OS operating configuration. This means that a configuration commit is necessary before a traceoption will take effect. Likewise, removing traceoptions requires deleting or deactivating the configuration, followed by committing the configuration. With a traceoption flag enabled, the data from the traceoption will be written to a log file, which may be predetermined or manually configured and stored in flash memory. This means that any trace logs are retained even after a system reboot. Ensure there is sufficient storage available in the flash memory before implementing traceoptions.

To troubleshoot, perform the following steps:

1. You can check the available storage in the following show command output, in which **/dev/ad0s1a** represents the onboard flash memory and is currently at 65% capacity. You can also view available storage on the J-Web homepage under System Storage. The output of all traceoptions is written to logs stored in the **/var/log** directory. To view a list of all the logs in**/var/log**, use the **show log** operational mode command.

```
user@CORPORATE> show system storage

Filesystem Size Used Avail Capacity Mounted on
/dev/ad0s1a 213M 136M 75M 65% /
devfs 1.0K 1.0K 0B 100% /dev
devfs 1.0K 1.0K 0B 100% /dev/
/dev/md0 144M 144M 0B 100% /junos
/cf 213M 136M 75M 65% /junos/cf
devfs 1.0K 1.0K 0B 100% /junos/dev/
procfs 4.0K 4.0K 0B 100% /proc
/dev/bo0s1e 24M 13K 24M 0% /config
/dev/md1 168M 7.3M 147M 5% /mfs
/dev/md2 58M 38K 53M 0% /jail/tmp
/dev/md3 7.7M 108K 7.0M 1% /jail/var
devfs 1.0K 1.0K 0B 100% /jail/dev
/dev/md4 1.9M 6.0K 1.7M 0% /jail/html/oem
```

2. Check the traceoption logs. Enabling traceoptions begins the logging of the output

to the filenames specified or to the default log file for the traceoption. View the appropriate log to see the trace output. The following are the show commands for viewing the appropriate logs.

```
user@CORPORATE> show log kmd
user@CORPORATE> show log security-trace
```

```
user@CORPORATE> show log messages

Logs can also be uploaded to an FTP server with the ‘file copy’ command. The syntax is as follows:
file copy <filename> <destination> as below.

user@CORPORATE> file copy /var/log/kmd ftp://10.10.10.10/kmd.log

ftp://10.10.10.10/kmd.log 100% of 35 kB 12 MBps
```

3. To view success or failure messages in IKE or IPsec, view the kmd log, using the **show log kmd** command. Although the kmd log displays a general reason for any failure, it may be necessary to obtain additional details by enabling IKE traceoptions. As a general rule, it is always best to troubleshoot on the peer that has the role of responder. Enable IKE traceoptions for phase 1 and phase 2 negotiation issues. The following example shows all of the IKE traceoptions.

```
user@CORPORATE> configure
Entering configuration mode[edit]
user@CORPORATE# edit security ike traceoptions
[edit security ike traceoptions]

user@CORPORATE# set file ?


Possible completions:
<filename> Name of file in which to write trace information
files Maximum number of trace files (2..1000)
match Regular expression for lines to be logged
no-world-readable Don't allow any user to read the log file
size Maximum trace file size (10240..1073741824)
world-readable Allow any user to read the log file

[edit security ike traceoptions]

user@CORPORATE# set flag ?


Possible completions:
all Trace everything
certificates Trace certificate events
database Trace security associations database events
general Trace general events
ike Trace IKE module processing
parse Trace configuration processing
policy-manager Trace policy manager processing
routing-socket Trace routing socket messages
timer Trace internal timer events
```

4. By default, if no filename is specified, then all IKE traceoptions are written to the kmd log. However, you can specify a different filename if you wish. To write trace data to the log, you must specify at least one flag option. The **file size** option determines the maximum size of a log file in bytes. For example, 1m or 1000000 generates a maximum file size of 1 MB. The **file files** option determines the maximum

number of log files that are generated and stored in the flash memory. Remember to commit the configuration changes to start the trace. The following example shows recommended traceoptions for troubleshooting most IKE-related issues.

```
[edit]
user@CORPORATE# edit security ike traceoptions
[edit security ike traceoptions]
user@CORPORATE# set file size 1m
user@CORPORATE# set flag policy-manager
user@CORPORATE# set flag ike
user@CORPORATE# set flag routing-socket
user@CORPORATE# commit
```

5. Review the kmd log for success/failure messages. In the following show command output are some excerpts of successful phase 1 and phase 2 completion as well as some instances of failure. Phase 1 and phase 2 successful. The following output shows that the local address is **1.1.1.2** and the remote peer is **2.2.2.2.** The output **udp:500** indicates that no NAT traversal was negotiated. You should see a phase 1 done message, along with the role (initiator or responder). Next you should see a phase 2 done message with proxy ID information. At this point, you can confirm that the IPsec SA is up, using the verification commands in Steps 1 through 4.

```
user@CORPORATE> show log kmd

Oct 8 10:41:40 Phase-1 [responder] done for local=ipv4(udp:500,[0..3]=1.1.1.2)
remote=ipv4(udp:500,[0..3]=2.2.2.2)
Oct 8 10:41:51 Phase-2 [responder] done for p1_local=ipv4(udp:500,[0..3]=1.1.1.2)
p1_remote=ipv4(udp:500,[0..3]=2.2.2.2) p2_local=ipv4_subnet(any:0,[0..7]=10.10.10.0/24)
p2_remote=ipv4_subnet(any:0,[0..7]=192.168.168.0/24)
```

6. Phase 1 failing to complete, example 1. In the following show command output, the local address is **1.1.1.2** and the remote peer is **2.2.2.2**. The role is responder. The reason for failing is **No proposal chosen**. This is likely caused by mismatched phase 1 proposals. To resolve this issue, configure the phase 1 proposals to match on the peers. Also confirm that a tunnel policy exists for the VPN.

```
user@CORPORATE> show log kmd

Oct 8 10:31:10 Phase-1 [responder] failed with error(No proposal chosen) for
local=unknown(any:0,[0..0]=) remote=ipv4(any:0,[0..3]=2.2.2.2)
Oct 8 10:31:10 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { 011359c9 ddef501d - 2216ed2a bfc50f5f [-
1] / 0x00000000 } IP; Error = No proposal chosen (14)
```

7. Phase 1 failing to complete, example 2. In the following show command output, the local address is **1.1.1.2** and the remote peer is **2.2.2.2**. The role is responder. The reason for failing may seem to indicate that no proposal was chosen. However, you also see **peer:2.2.2.2** is not recognized. This message could be caused by an incorrect peer address, a mismatched peer ID type, or an incorrect peer ID, depending on whether this is a dynamic or static VPN. The peer address must be checked first before the phase 1 proposal is checked. To resolve this issue, confirm that the local peer has the correct peer IP address. Also confirm that the peer is configured with IKE ID type as the IP address.

```
user@CORPORATE> show log kmd

Oct 8 10:39:40 Unable to find phase-1 policy as remote peer:2.2.2.2 is not recognized.
Oct 8 10:39:40 KMD_PM_P1_POLICY_LOOKUP_FAILURE: Policy lookup for Phase-1 [responder] failed for
```



```
p1_local=ipv4(any:0,[0..3]=1.1.1.2) p1_remote=ipv4(any:0,[0..3]=2.2.2.2)
Oct 8 10:39:40 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { 18983055 dbe1d0af - a4d6d829 f9ed3bba [-
1] / 0x00000000 } IP; Error = No proposal chosen (14)
```

8. Phase 1 failing to complete, example 3. In the following show command output, the remote peer is **2.2.2.2**. Invalid payload type usually indicates a problem with the decryption of the IKE packet due to a mismatched preshared key. To resolve this issue, configure the preshared keys to match on the peers.

```
user@CORPORATE> show log kmd

Oct 8 10:36:20 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { e9211eb9 b59d543c - 766a826d bd1d5ca1 [-
1] / 0x00000000 } IP; Invalid next payload type = 17
Oct 8 10:36:20 Phase-1 [responder] failed with error(Invalid payload type) for
local=unknown(any:0,[0..0]=) remote=ipv4(any:0,[0..3]=2.2.2.2)
```

9. Phase 1 successful, phase 2 failing to complete, example 1. In the following show command output, the local address is **1.1.1.2** and the remote peer is **2.2.2.2**. Phase 1 was successful, based on the **Phase-1 [responder] done** message. The reason for the failure is due to **No proposal chosen** during phase 2 negotiation. The issue is likely phase 2 proposal mismatch between the two peers. To resolve this issue, configure the phase 2 proposals to match on the peers.

```
user@CORPORATE> show log kmd

Oct 8 10:53:34 Phase-1 [responder] done for local=ipv4(udp:500,[0..3]=1.1.1.2)
remote=ipv4(udp:500,[0..3]=2.2.2.2)
Oct 8 10:53:34 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { cd9dff36 4888d398 - 6b0d3933 f0bc8e26 [0]
/ 0x1747248b } QM; Error = No proposal chosen (14)
```

10. Phase 1 successful, phase 2 failing to complete, example 2. In the following show command output, phase 1 was successful. The reason for failure in phase 2 may seem to be that no proposal was chosen. However, there is also the message **Failed to match the peer proxy ids**, which means that the proxy ID did not match what was expected. Phase 2 proxy ID of remote=192.168.168.0/24, local=10.10.20.0/24, service=any was received. It is clear that this does not match the configurations on the local peer; thus, proxy ID match fails. This results in the error: **No proposal chosen**. To resolve this, configure one peer proxy ID so that it matches the other peer. Note that for a route-based VPN, the proxy ID, by default, is all zeroes (local=0.0.0.0/0, remote=0.0.0.0/0, service=any). If the remote peer specifies a proxy ID other than all zeroes, then you must manually configure the proxy ID within the IPsec profile of the peer.

```
user@CORPORATE> show log kmd

Oct 8 10:56:00 Phase-1 [responder] done for local=ipv4(udp:500,[0..3]=1.1.1.2)
remote=ipv4(udp:500,[0..3]=2.2.2.2)
Oct 8 10:56:00 Failed to match the peer proxy ids
p2_remote=ipv4_subnet(any:0,[0..7]=192.168.168.0/24)
p2_local=ipv4_subnet(any:0,[0..7]=10.10.20.0/24) for the remote peer:ipv4(udp:500,[0..3]=2.2.2.2)
Oct 8 10:56:00 KMD_PM_P2_POLICY_LOOKUP_FAILURE: Policy lookup for Phase-2 [responder] failed for
p1_local=ipv4(udp:500,[0..3]=1.1.1.2) p1_remote=ipv4(udp:500,[0..3]=2.2.2.2)
p2_local=ipv4_subnet(any:0,[0..7]=10.10.20.0/24)
p2_remote=ipv4_subnet(any:0,[0..7]=192.168.168.0/24)
Oct 8 10:56:00 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { 41f638eb cc22bbfe - 43fd0e85 b4f619d5 [0]
/ 0xc77fafcf } QM; Error = No proposal chosen (14)
```

11. The following is a problem scenario using the network diagram. See Figure 1 on page 5.

    a. Remote PC **192.168.168.10** can ping local PC **10.10.10.10**.

    b. Local PC **10.10.10.10** cannot ping **192.168.168.10**.

    c. Based on the output from show commands, IPsec SA is up, and the statistics show no errors.

    Considering that the IPsec tunnel is up, then it is likely that there is a problem with the route lookup, security policy, or some other flow issue. Enable security flow traceoptions to determine why the traffic is successful in one direction but not the other.

    NOTE: Enabling flow traceoptions can increase system CPU and memory usage. Therefore, we do not recommend enabling flow traceoptions during peak traffic load times or when CPU utilization is very high. We recommend enabling packet filters to lower resource usage and to facilitate pinpointing the packets of interest. Be sure to delete or deactivate all flow traceoptions and remove any unnecessary log files from the flash memory after you complete troubleshooting.

12. Enable security flow traceoptions for routing or policy issues. See the following example of output for security flow traceoptions. By default, if no filename is specified, then all flow traceoptions output is written to the security-trace log file. However, you can specify a different filename if you wish. To write trace data to the log, you must specify at least one flag option. The **file size** option determines the maximum size of a log file in bytes. For example, 1m or 1000000 generates a maximum file size of 1 MB. The **file files** option determines the maximum number of log files that are generated and stored in flash memory. Remember to commit the configuration changes to start the trace.

    [edit] user@CORPORATE# **edit security flow traceoptions**

    [edit security flow traceoptions]

    ```
    user@CORPORATE# set file ?

    Possible completions:
    <filename> Name of file in which to write trace information
    files Maximum number of trace files (2..1000)
    match Regular expression for lines to be logged
    no-world-readable Don't allow any user to read the log file
    size Maximum trace file size (10240..1073741824)
    world-readable Allow any user to read the log file
    ```

    [edit security flow traceoptions]

    ```
    user@CORPORATE# set flag ?


    Possible completions:
    ```

```
ager Ager events
all All events
basic-datapath Basic packet flow
cli CLI configuration and commands changes
errors Flow errors
fragmentation Ip fragmentation and reassembly events
high-availability Flow high-availability information
host-traffic Flow host-traffic information
lookup Flow lookup events
multicast Multicast flow information
packet-drops Packet drops
route Route information
session Session creation and deletion events
session-scan Session scan information
tcp-advanced Advanced TCP packet flow
tcp-basic TCP packet flow
tunnel Tunnel information
```

13. Junos OS can configure packet filters to limit the scope of the traffic to be captured. You can filter the output based on source/destination IP address, source/destination port, interface, and IP protocol. Up to 64 filters can be configured. A packet filter also matches the reverse direction to capture the reply traffic, assuming that the source of the original packet matches the filter. The following example shows the packet flow filter options.

```
[edit security flow traceoptions]
user@CORPORATE# set packet-filter filter-name ?

Possible completions:
+ apply-groups Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
destination-port Match TCP/UDP destination port
destination-prefix Destination IPv4 address prefix
interface Logical interface
protocol Match IP protocol type
source-port Match TCP/UDP source port
source-prefix Source IPv4 address prefix
```

14. Terms listed within the same packet filter act as a Boolean logical **AND** statement. This means that all statements within the packet filter need to match in order to write the output to the log. A listing of multiple filter names acts as a logical **OR**. Using packet filters, the following example lists the recommended traceoptions for security flow for the problem scenario given in Step 11.

```
[edit]
user@CORPORATE# edit security flow traceoptions
[edit security flow traceoptions]
user@CORPORATE# set file size 1m files 3
user@CORPORATE# set flag basic-datapath
user@CORPORATE# set packet-filter remote-to-local source-prefix
  192.168.168.10/32
user@CORPORATE# set packet-filter remote-to-local destination-prefix
  10.10.10.10/32
user@CORPORATE# set packet-filter local-to-remote source-prefix 10.10.10.0/32
user@CORPORATE# set packet-filter local-to-remote destination-prefix
  192.168.168.0/32
user@CORPORATE# set packet-filter remote-esp protocol 50
```

```
user@CORPORATE# set packet-filter remote-esp source-prefix 2.2.2.2/32
user@CORPORATE> commit
```

15. The following output details the reasoning behind each flow traceoption setting.

```
[edit security flow traceoptions]
 user@CORPORATE# show


 file flow-trace-log size 1m files 3;
 flag basic-datapath;
```

16. In the following example the **security-trace** log file is set to 1 MB and up to 3 files can be created. The reason for this is that because of the nature of flow traceoptions, a single file could become full very quickly, depending on how much traffic is captured. The **basic-datapath** flag shows details for most flow-related problems.

```
packet-filter remote-to-local {
  source-prefix 192.168.168.10/32;
  destination-prefix 10.10.10.10/32;
}
```

17. The filter used in Step 16 is for capturing the decapsulated or unencrypted traffic from the remote PC to the local PC. Because there are multiple terms, policy execution acts as a Boolean logical **AND**, which means that the source IP address and destination IP address must both match the filter. If the source IP address matches but the destination IP address does not, then the packet is not captured. Because packet filters are bidirectional, it is not necessary to configure a filter for the reply traffic.

```
packet-filter local-to-remote {
  source-prefix 10.10.10.0/32;
  destination-prefix 192.168.168.0/32;
}
```

18. As mentioned in Step 17, no filter is required for capturing the reply traffic. However, a filter captures only the packets which are originally sourced from the specified side. Thus, the **local-to-remote** filter in Step 17 is still required to capture traffic which sources from the local side to the remote side. The filter in the example is optional and depends on whether or not the previous filter captured any packets. This filter captures all ESP (IP protocol 50) or encrypted packets from remote peer **2.2.2.2**. Note that this filter captures **ALL** encrypted traffic from **2.2.2.2**, including packets that perhaps you are not interested in. If the unencrypted traffic is captured, then this last filter may not be necessary.

    With the three problem statements mentioned in the problem scenario in Step 11, you can now begin to look at the flow traceoptions log to isolate the issue. Assume that the third statement is correct, based on IKE and IPsec troubleshooting. Therefore, the next step is to validate the first problem statement to confirm that the remote PC can ping the local PC. Then you can troubleshoot the second problem statement to find out why the traffic fails in the reverse direction.

```
packet-filter remote-esp {
  protocol 50;
  source-prefix 2.2.2.2/32;
}
```


19. Validate the first problem statement. Send a ping packet from **192.168.168.10** to **10.10.10.10** and then view the security-trace log. Because no filename is specified, view the flow traceoptions output using the **show log security-trace** command. The following flow traceoptions output shows successful traffic flow from remote PC to the local PC. The first packet captured is the ESP, or encrypted packet.

```
user@CORPORATE> show log security-trace

******<2.2.2.2/42558->1.1.1.2/45679;50> matched filter remote-esp: <untrust/ge-0/0/3.0> ******
Oct 6 19:20:33 19:20:33.863580:CID-0:RT: packet [184] ipid = 12384, @497afcee ******
Oct 6 19:20:33 19:20:33.863590:CID-0:RT: ge-0/0/3.0:2.2.2.2->1.1.1.2, 50
Oct 6 19:20:33 19:20:33.863597:CID-0:RT: find flow: table 0x4b5265e0, hash 192852(0x3ffff), sa
2.2.2.2, da 1.1.1.2, sp 42558, dp 45679, proto 50, tok 12
Oct 6 19:20:33 19:20:33.863614:CID-0:RT: find flow: table 0x4b59eb00, hash 340(0xfff), sa 2.2.2.2,
da 1.1.1.2, sp 42558, dp 45679, proto 50, tok 12
Oct 6 19:20:33 19:20:33.863630:CID-0:RT: flow session id 257024
Oct 6 19:20:33 19:20:33.863635:CID-0:RT: flow_decrypt: tun 51761360(flag b), iif 68
Oct 6 19:20:33 19:20:33.863682:CID-0:RT:inject tunnel pkt mbuf 0x497afb40
Oct 6 19:20:33 19:20:33.863689:CID-0:RT:injected tunnel pkt mbuf 0x497afb40
```

Based on the top header in the output in Step 19, the packet is from **2.2.2.2** to **1.1.1.2**; the IP protocol is **50**. The ingress interface is **ge-0/0/3.0** in zone **untrust** and matching packet filter **remote-esp**. This is the ESP packet from the remote peer. The port values for IP protocol 50 are not the same as with TCP/UDP. The values are an amalgamation of the SPI value for the tunnel. The **flow session id** is the tunnel session created for the ESP traffic. You can view details about this session using the **show security flow session session-identifier <session id>** command. The **flow_decrypt** message indicates that the decryption process is to take place. The **tun** value is an internal pointer, and **iif** refers to the incoming logical interface index. You can view all the logical interface index numbers using the **show interface extensive** command.

20. The following is the decrypted packet output. Based on the top header in the output for the show log security-trace command, the packet is from **192.168.168.10** to **10.10.10.10**; the IP protocol is **1**. The ingress interface is **ge-0/0/3.0** because the source is from across the VPN. Therefore, the ingress zone is zone **untrust** and matching packet filter **remote-to-local**. This is an ICMP packet. In particular, **icmp, (8/0)** indicates that this is an ICMP type 8, code 0, which is an echo request. The source port is the ICMP sequence value, and the destination port is the ICMP identifier.

```
user@CORPORATE> show log security-trace

******<192.168.168.10/2048->10.10.10.10/1098;1> matched filter remote-to-local: <untrust/
ge-0/0/3.0> ******
Oct 6 19:20:33 19:20:33.863714:CID-0:RT: packet [128] ipid = 41035, @497afd12 ******
Oct 6 19:20:33 19:20:33.863724:CID-0:RT: ge-0/0/3.0:192.168.168.10->10.10.10.10, icmp, (8/0)
Oct 6 19:20:33 19:20:33.863730:CID-0:RT: find flow: table 0x4b5265e0, hash 223505(0x3ffff), sa
192.168.168.10, da 10.10.10.10, sp 21480, dp 1024, proto 1, tok 12
Oct 6 19:20:33 19:20:33.863746:CID-0:RT: flow_first_sanity_check: in <ge-0/0/3.0>, out <N/A>
Oct 6 19:20:33 19:20:33.863754:CID-0:RT: flow_first_in_dst_nat: in <ge-0/0/3.0>, out <N/A
Oct 6 19:20:33 19:20:33.863757:CID-0:RT: flow_first_in_dst_nat: dst_adr 10.10.10.10, sp 21480, dp
1024
Oct 6 19:20:33 19:20:33.863765:CID-0:RT: chose interface N/A as incoming nat if.
Oct 6 19:20:33 19:20:33.863769:CID-0:RT: flow_first_routing: Before route-lookup ifp: in <ge-
0/0/3.0>, out <N/A>
```

```
Oct 6 19:20:33 19:20:33.863772:CID-0:RT:flow_first_routing: call flow_route_lookup(): src_ip
192.168.168.10, x_dst_ip 10.10.10.10, ifp ge-0/0/3.0, sp 21480, dp 1024, ip_proto 1, tos 0
Oct 6 19:20:33 19:20:33.863782:CID-0:RT:Doing DESTINATION addr route-lookup
Oct 6 19:20:33 19:20:33.863790:CID-0:RT:Doing SOURCE addr route-lookup
Oct 6 19:20:33 19:20:33.863802:CID-0:RT: routed (x_dst_ip 10.10.10.10) from ge-0/0/3.0 (ge-
0/0/3.0 in 0) to ge-0/0/0.0, Next-hop: 10.10.10.10
Oct 6 19:20:33 19:20:33.863810:CID-0:RT: policy search from zone (untrust) 7-> zone
(trust) 6
Oct 6 19:20:33 19:20:33.863826:CID-0:RT: policy found 6
Oct 6 19:20:33 19:20:33.863833:CID-0:RT:No src xlate
Oct 6 19:20:33 19:20:33.863836:CID-0:RT: choose interface ge-0/0/0.0 as outgoing phy if
Oct 6 19:20:33 19:20:33.863840:CID-0:RT:is_loop_pak: No loop: on ifp: ge-0/0/0.0, addr:
10.10.10.10, rtt_idx:0
Oct 6 19:20:33 19:20:33.863846:CID-0:RT: Using app_id from service lookup 0
Oct 6 19:20:33 19:20:33.863849:CID-0:RT: session application type 0, name (null), timeout 60sec,
alg 0
Oct 6 19:20:33 19:20:33.863854:CID-0:RT: service lookup identified service 0.
Oct 6 19:20:33 19:20:33.863858:CID-0:RT: flow_first_final_check: in <ge-0/0/3.0>, out <ge-
0/0/0.0>
Oct 6 19:20:33 19:20:33.863866:CID-0:RT: existing vector list 2-59b5c308.
Oct 6 19:20:33 19:20:33.863872:CID-0:RT: existing vector list 2-59b5c308.
Oct 6 19:20:33 19:20:33.863879:CID-0:RT: Session (id:45) created for first pak 2
Oct 6 19:20:33 19:20:33.863883:CID-0:RT: flow_first_install_session======> 0x4c6fecb0
Oct 6 19:20:33 19:20:33.863889:CID-0:RT: nsp 0x4c6fecb0, nsp2 0x4c6fed08
Oct 6 19:20:33 19:20:33.863900:CID-0:RT: 5 tuple sa 192.168.168.10, da 10.10.10.10, sp 21480, dp
1024, proto 1
Oct 6 19:20:33 19:20:33.863909:CID-0:RT: set route old fto 0x59b5c180, new fto 0x59b5c180
Oct 6 19:20:33 19:20:33.863918:CID-0:RT: 5 tuple sa 10.10.10.10, da 192.168.168.10, sp 1024, dp
21480, proto 1
Oct 6 19:20:33 19:20:33.863926:CID-0:RT: set route old fto 0x59b5c1f8, new fto 0x59b5c1f8
Oct 6 19:20:33 19:20:33.863937:CID-0:RT: flow session id 45
Oct 6 19:20:33 19:20:33.863943:CID-0:RT: post addr xlation: 192.168.168.10->10.10.10.10.
Oct 6 19:20:33 19:20:33.863949:CID-0:RT: encap vector
Oct 6 19:20:33 19:20:33.863952:CID-0:RT: no more encapping needed
```

There is no existing session for this flow, so first-packet processing occurs. Next route lookup occurs. Route lookup must occur to determine the ingress and egress zones for security policy lookup. Route lookup determines that the packet needs to egress out **ge-0/0/0.0**. Because interface **ge-0/0/0.0** is associated with zone **trust**, and **ge-0/0/3.0** is associated with zone **untrust**, the policy lookup is from-zone **untrust** to-zone **trust**. Policy 6 was found, which permits the traffic.

21. The details for policy 6 can be viewed with the **show security policies** command.

```
user@CORPORATE> show security policies | find “Index: 6”


Policy: vpnpolicy-unt-tr, action-type: permit, State: enabled, Index: 6
Sequence number: 1
From zone: untrust, To zone: trust
Source addresses:
remote-net: 192.168.168.0/24
Destination addresses:
local-net: 10.10.10.0/24
Application: any
IP protocol: 0, ALG: 0, Inactivity timeout: 0
Source port range: [0-0]
Destination port range: [0-0]
Tunnel: ike-vpn, Type: IPsec, Index: 2
Pair policy: vpnpolicy-tr-unt
```


22. At this point, the session is created; in this case, the session ID is **45**. The reply packet is also captured and shows existing session 45 is found as shown in the following output. Note that **icmp, (0/0)** indicates that this is an ICMP packet type 0, code 0, which is an ICMP echo reply. The packet is shown going into tunnel 4000002. This means that the tunnel is 0x2, which converts to SA index 2 in decimal notation. This confirms that the traffic initiating from remote PC **192.168.168.10** to local PC **10.10.10.10** is successful.

```
user@CORPORATE> show log security-trace

******<10.10.10.10/0->192.168.168.10/3146;1> matched filter local-to-remote: <trust/
ge-0/0/0.0> ******
Oct 6 19:20:33 19:20:33.865626:CID-0:RT: packet [128] ipid = 42775, @498333ce ******
Oct 6 19:20:33 19:20:33.865637:CID-0:RT: ge-0/0/0.0:10.10.10.10->192.168.168.10, icmp, (0/0)
Oct 6 19:20:33 19:20:33.865643:CID-0:RT: find flow: table 0x4b5265e0, hash 221617(0x3ffff), sa
10.10.10.10, da 192.168.168.10, sp 1024, dp 21480, proto 1, tok 10
Oct 6 19:20:33 19:20:33.865660:CID-0:RT: flow session id 45
Oct 6 19:20:33 19:20:33.865668:CID-0:RT:xlate_icmp_pak: set nat invalid 45, timeout 1, reason 3
Oct 6 19:20:33 19:20:33.865673:CID-0:RT: post addr xlation: 10.10.10.10->192.168.168.10.
Oct 6 19:20:33 19:20:33.865681:CID-0:RT: encap vector
Oct 6 19:20:33 19:20:33.865683:CID-0:RT: going into tunnel 40000002.
Oct 6 19:20:33 19:20:33.865689:CID-0:RT: flow_encrypt: 0x51761360
Oct 6 19:20:33 19:20:33.865734:CID-0:RT:inject tunnel pkt mbuf 0x49833220
Oct 6 19:20:33 19:20:33.865741:CID-0:RT:injected tunnel pkt mbuf 0x49833220
```

23. Troubleshoot the second problem statement. In the the second problem statement, the local PC cannot ping the remote PC. You can determine the problem by reviewing the security-trace log while attempting to ping from **10.10.10.10** to **192.168.168.10**. The following is a sample output showing a failure.

```
user@CORPORATE> show log security-trace

******<10.10.10.10/2048->192.168.168.10/18763;1> matched filter local-to-remote: <trust/
ge-0/0/0.0> ******
Oct 6 19:21:30 19:21:30.416831:CID-0:RT: packet [128] ipid = 42795, @49f59b4e ******
Oct 6 19:21:30 19:21:30.416843:CID-0:RT: ge-0/0/0.0:10.10.10.10->192.168.168.10, icmp, (8/0)
Oct 6 19:21:30 19:21:30.416850:CID-0:RT: find flow: table 0x4b5265e0, hash 41820(0x3ffff), sa
10.10.10.10, da 192.168.168.10, sp 43700, dp 1024, proto 1, tok 10
Oct 6 19:21:30 19:21:30.416867:CID-0:RT: flow_first_sanity_check: in <ge-0/0/0.0>,out <N/A>
Oct 6 19:21:30 19:21:30.416877:CID-0:RT: flow_first_in_dst_nat: in <ge-0/0/0.0>, out <N/A>
Oct 6 19:21:30 19:21:30.416880:CID-0:RT: flow_first_in_dst_nat: dst_adr 192.168.168.10, sp 43700,
dp 1024
Oct 6 19:21:30 19:21:30.416887:CID-0:RT: chose interface ge-0/0/0.0 as incoming nat if.
Oct 6 19:21:30 19:21:30.416891:CID-0:RT: flow_first_routing: Before route-lookup ifp: in <ge-
0/0/0.0>, out <N/A>
Oct 6 19:21:30 19:21:30.416895:CID-0:RT:flow_first_routing: call flow_route_lookup(): src_ip
10.10.10.10, x_dst_ip 192.168.168.10, ifp ge-0/0/0.0, sp 43700, dp 1024, ip_proto 1, tos 0
Oct 6 19:21:30 19:21:30.416904:CID-0:RT:Doing DESTINATION addr route-lookup
Oct 6 19:21:30 19:21:30.416914:CID-0:RT:Doing SOURCE addr route-lookup
Oct 6 19:21:30 19:21:30.416918:CID-0:RT: routed (x_dst_ip 192.168.168.10) from ge-0/0/0.0 (ge-
0/0/0.0 in 0) to ge-0/0/3.0, Next-hop: 1.1.1.1
Oct 6 19:21:30 19:21:30.416926:CID-0:RT: policy search from zone (trust) 6->zone (untrust) 7
Oct 6 19:21:30 19:21:30.416943:CID-0:RT: policy found 4
Oct 6 19:21:30 19:21:30.416954:CID-0:RT: dip id = 2/0, 10.10.10.10/43700->1.1.1.2/1039
Oct 6 19:21:30 19:21:30.416964:CID-0:RT: choose interface ge-0/0/3.0 as outgoing phy if
Oct 6 19:21:30 19:21:30.416967:CID-0:RT:is_loop_pak: No loop: on ifp: ge-0/0/3.0, addr:
192.168.168.10, rtt_idx:0
Oct 6 19:21:30 19:21:30.416973:CID-0:RT: Using app_id from service lookup 0
Oct 6 19:21:30 19:21:30.416976:CID-0:RT: session application type 0, name (null), timeout 60sec,
alg 0
```

```
Oct 6 19:21:30 19:21:30.416982:CID-0:RT: service lookup identified service 0.
Oct 6 19:21:30 19:21:30.416986:CID-0:RT: flow_first_final_check: in <ge-0/0/0.0>, out <ge-
0/0/3.0>
Oct 6 19:21:30 19:21:30.416994:CID-0:RT: existing vector list 0-59b5c220.
Oct 6 19:21:30 19:21:30.417000:CID-0:RT: existing vector list 0-59b5c220.
Oct 6 19:21:30 19:21:30.417006:CID-0:RT: Session (id:50) created for first pak 0
Oct 6 19:21:30 19:21:30.417010:CID-0:RT: flow_first_install_session======> 0x4c6ff318
Oct 6 19:21:30 19:21:30.417016:CID-0:RT: nsp 0x4c6ff318, nsp2 0x4c6ff370
Oct 6 19:21:30 19:21:30.417027:CID-0:RT: 5 tuple sa 10.10.10.10, da 192.168.168.10, sp 43700, dp
1024, proto 1
Oct 6 19:21:30 19:21:30.417036:CID-0:RT: set route old fto 0x59b5c1f8, new fto 0x59b5c1f8
Oct 6 19:21:30 19:21:30.417045:CID-0:RT: 5 tuple sa 192.168.168.10, da 1.1.1.2, sp 1024, dp 1039,
proto 1
Oct 6 19:21:30 19:21:30.417070:CID-0:RT: set route old fto 0x59b5c180, new fto 0x59b5c180
Oct 6 19:21:30 19:21:30.417081:CID-0:RT: flow session id 50
Oct 6 19:21:30 19:21:30.417088:CID-0:RT: post addr xlation: 1.1.1.2->192.168.168.10.
```

Based on the top header in the output in Step 23, the packet is from **10.10.10.10** to **192.168.168.10**; the IP protocol is **1**. No session is found, so first packet processing occurs. Next, route-lookup occurs. The route lookup correctly shows that the egress interface is **ge-0/0/3.0**. Therefore, policy lookup is from zone **trust** to zone **untrust**. The packet matches policy index 4.

24. To confirm if policy index 4 is the correct policy, use the **show security policies** command.

```
user@CORPORATE> show security policies


Default policy: deny-all
From zone: trust, To zone: untrust
Policy: any-permit, State: enabled, Index: 4, Sequence number: 1
Source addresses: any
Destination addresses: any
Applications: any
Action: permit
Policy: vpnpolicy-tr-unt, State: enabled, Index: 7, Sequence number: 2
Source addresses: local-net
Destination addresses: remote-net
Applications: any
Action: permit, tunnel
From zone: trust, To zone: trust
Policy: intrazone-permit, State: enabled, Index: 5, Sequence number: 1
Source addresses: any
Destination addresses: any
Applications: any
Action: permit
Customer Service Solution Development Page 26 App Note
9/30/2009 JunOS Policy Based VPN
From zone: untrust, To zone: trust
Policy: vpnpolicy-unt-tr, State: enabled, Index: 6, Sequence number: 1
Source addresses: remote-net
Destination addresses: local-net
Applications: any
Action: permit, tunnel
```

From the output in Step 24, you can see that policy index 4 is the **any-permit** policy. However, in order to be sent across the VPN, the traffic must match tunnel policy **vpnpolicy-tr-unt**, which is policy index 7. But policy index 7 is below policy index 4;

thus, the traffic always matches the **any-permit** policy first. Recall that policy lookup is always from top to bottom.

25. To resolve the order of policy issue, place the tunnel policy above the **any-permit** policy using the insert command as follows.

```
[edit]
user@CORPORATE# edit security policies from-zone trust to-zone untrust

[edit security policies from-zone trust to-zone untrust]
user@CORPORATE# insert policy vpnpolicy-tr-unt before policy any-permit

[edit security policies from-zone trust to-zone untrust]
user@CORPORATE# commit
```

Why did the remote PC to local PC traffic succeed despite that there is no route or policy configured for the reply traffic? The order of packet processing is important to answering this question. Junos OS first inspects the packet to see whether there is already an existing session. If no session exists, then a route lookup is performed. Next, policy lookup is performed. When the first packet reaches the device from the remote PC to the local PC, the session is built for the reply packet. When the reply packet is received, it matches the existing session and is then forwarded. If a session match is found, then no further route or policy lookup occurs.

Results For reference, the configuration of the Corporate Office Router is shown.

Corporate Office Router

```
system {
host-name CORPORATE;
  root-authentication {
    encrypted-password "$1$heGUvm8Y$t4wl4Oc0NR8dZlDNz0No2."; ## SECRET-DATA
    syslog {
      user * {
        any emergency;
      }
      file messages {
        any any;
        authorization info;
      }
      file interactive-commands {
        interactive-commands any;
      }
    }
  }
  interfaces {
    ge-0/0/0 {
      unit 0 {
        family inet {
          address 10.10.10.1/24;
        }
      }
    }
    ge-0/0/3 {
      unit 0 {
        family inet {
```

```
            address 1.1.1.2/30;
        }
      }
    }
}
routing-options {
    static {
      route 0.0.0.0/0 next-hop 1.1.1.1;
    }
}
security {
    ike {
      traceoptions {
        flag ike;
        flag policy-manager;
        flag routing-socket;
      }
      policy ike-policy1 {
        mode main;
        proposal-set standard;
        pre-shared-key ascii-text "$9$dhwoGF39A0IGDPQFnpu8X7"; ## SECRET-DATA
      }
      gateway ike-gate {
        ike-policy ike-policy1;
        address 2.2.2.2;
        external-interface ge-0/0/3.0;
      }
    }
    ipsec {
      policy vpn-policy1 {
        proposal-set standard;
      }
      vpn ike-vpn {
        ike {
          gateway ike-gate;
          ipsec-policy vpn-policy1;
        }
      }
    }
    zones {
      security-zone untrust {
        address-book {
          address remote-net 192.168.168.0/24;
        }
        host-inbound-traffic {
          system-services {
            ike;
          }
        }
        interfaces {
          ge-0/0/3.0;
        }
      }
      security-zone trust {
        address-book {
          address local-net 10.10.10.0/24;
```

```
    }
    host-inbound-traffic {
      system-services {
        all;
      }
    }
    interfaces {
      ge-0/0/0.0 {
      }
    }
  }
  policies {
    from-zone trust to-zone untrust {
      policy vpnpolicy-tr-unt {
        match {
          source-address local-net;
          destination-address remote-net;
          application any;
        }
        then {
          permit {
            tunnel {
              ipsec-vpn ike-vpn;
              pair-policy vpnpolicy-unt-tr;
            }
          }
        }
      }
      policy any-permit {
        match {
          source-address any;
          destination-address any;
          application any;
        }
        then {
          permit {
            source-nat {
              interface;
            }
          }
        }
      }
    }
    from-zone untrust to-zone trust {
      policy vpnpolicy-unt-tr {
        match {
          source-address remote-net;
          destination-address local-net;
          application any;
        }
        then {
          permit {
            tunnel {
              ipsec-vpn ike-vpn;
              pair-policy vpnpolicy-tr-unt;
            }
```

```
            }
          }
        }
      }
    }
    flow {
      traceoptions {
        file size 1m files 3;
        flag basic-datapath;
        packet-filter remote-to-local {
          source-prefix 192.168.168.10/32;
          destination-prefix 10.10.10.10/32;
        }
        packet-filter local-to-remote {
          source-prefix 10.10.10.0/32;
          destination-prefix 192.168.168.0/32;
        }
        packet-filter remote-esp {
          protocol 50;
          source-prefix 2.2.2.2/32;
        }
      }
      tcp-mss {
        ipsec-vpn {
          mss 1350;
        }
      }
    }
  }
```

Related Documentation

- Policy-Based VPNs Using J Series Routers and SRX Series Devices Overview on page 1
- Comparing Policy-Based and Route-Based VPNs on page 3