# Network Configuration Example

## Configuring Route-Based VPNs Using J Series and SRX Series Devices

Modified: 2017-01-17

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

## END USER LICENSE AGREEMENT

# Table of Contents

CHAPTER 1

# Configuring Route-Based VPNs Using J Series and SRX Series Devices

## About This Network Configuration Example

This network configuration example describes route-based VPNs and provides a step-by-step example for configuring route-based VPNs on Juniper Networks SRX Series Services Gateways.

## Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device Overview

The Juniper Networks Junos operating system (Junos OS), 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 has been enhanced with security and virtual private network (VPN) features from the Juniper Networks Firewall/IPsec VPN platforms, which include the Secure Services Gateway (SSG) product family. This document provides detailed information about IPsec interoperability between a J Series or SRX Series device and an SSG device using a route-based VPN. This example also provides troubleshooting information for J Series or SRX Series devices.

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

**Related Documentation**

- Comparing Policy-Based VPNs and Route-Based VPNs on page 6
- Example: Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device on page 7

## Comparing Policy-Based VPNs and Route-Based VPNs

It is important to understand the differences between policy-based VPNs and route-based VPNs, and why one might be preferable to the other.

Table 1 on page 6 lists the differences between route-based VPNs and policy-basedVPNs.

### Table 1: Differences Between Route-Based VPNs and Policy-Based VPNs

| Route-Based VPNs | Policy-Based VPNs |
|---|---|
| With route-based VPNs, a policy does not specifically reference a VPN tunnel. | 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. |
| The policy references a destination address. | In a policy-based VPN configuration, a tunnel policy specifically references a VPN tunnel by name. |
| 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. | The number of policy-based VPN tunnels that you can create is limited by the number of policies that the device supports. |
| Route-based VPN tunnel configuration is a good 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 only one IPsec SA is 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. | In a policy-based VPN configuration, the action must be permit and must include a tunnel. |
| Route-based VPNs support the exchange of dynamic routing information through VPN tunnels. You can enable an instance of a dynamic routing protocol, such as OSPF, on an st0 interface that is bound to a VPN tunnel. | The exchange of dynamic routing information is not supported in policy-based VPNs. |
| Route-based configurations are used for hub-and-spoke topologies. | Policy-based VPNs cannot be used for hub-and-spoke topologies. |

Table 1: Differences Between Route-Based VPNs and Policy-Based VPNs *(continued)*

| Route-Based VPNs | Policy-Based VPNs |
|---|---|
| With route-based VPNs, a policy does not specifically reference a VPN tunnel. | When a tunnel does not connect large networks running dynamic routing protocols and you do not need to conserve tunnels or define various policies to filter traffic through the tunnel, a policy-based tunnel is the best choice. |
| Route-based VPNs do not support remote-access (dial-up) VPN configurations. | Policy-based VPN tunnels are required for remote-access (dial-up) VPN configurations. |
| Route-based VPNs might not work correctly with some third-party vendors. | Policy-based VPNs might be required if the third party requires separate SAs for each remote subnet. |
| When the security device does a route lookup to find the interface through which it must send traffic to reach an address, it finds a route via a secure tunnel (st0) interface, which is bound to a specific VPN tunnel.<br><br>With a route-based VPN tunnel, you can consider a tunnel as a means for delivering traffic, and can consider the policy as a method for either permitting or denying the delivery of that traffic. | With a policy-based VPN tunnel, you can consider a tunnel as an element in the construction of a policy. |
| Route-based VPNs support NAT for st0 interfaces. | Policy-based VPNs cannot be used if NAT is required for tunneled traffic. |

This example focuses on route-based VPN configuration and troubleshooting.

**Related Documentation**

- Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device Overview on page 5
- Example: Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device on page 7

## Example: Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device

## Requirements

This example uses the following hardware and software components:

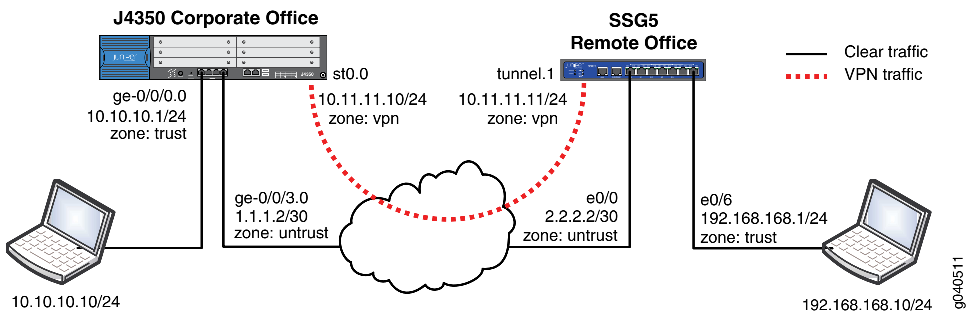
- Junos OS Release 9.5 or later
- Juniper Networks SRX Series Services Gateways or J Series Services Routers

NOTE: This configuration example has been tested using the software release listed and is assumed to work on all later releases

## Overview and Topology

Figure 1 on page 8 shows the network topology used in this configuration example.

### Figure 1: Network Topology



This example assumes the following (refer to Figure 1 on page 8):

- The internal LAN interface is **ge-0/0/0** in zone trust and has a private IP subnet.
- The internet interface is **ge-0/0/3** in zone untrust and has a public IP address.
- The secure tunnel interface st0 is in the vpn zone to allow you to configure unique policies specifically for tunnel (encrypted) traffic while maintaining unique policies for clear (non-encrypted) traffic.
- All traffic between the local and remote LANs is permitted, and traffic can be initiated from either side.
- The Juniper Networks SSG5 Secure Services Gateway has already been configured with the correct information for this example.

## Configuration

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. Though it is not mandatory, for st0 interfaces we recommend that both peers have an IP address within the same logical subnet because the link is logically a point-to-point link.

For static routes you normally specify the gateway IP address as the next hop. Creating a unique zone for tunnel traffic allows you to create a set of policies specifically for VPN traffic while maintaining separation of policies for non-VPN traffic. Also you can create deny policies to exclude specific hosts from accessing the VPN.

Host-inbound services are for traffic destined for the SRX Series or J Series devices itself. This includes but is not limited to FTP, HTTP, HTTPS, Internet Key Exchange (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 IKE only on the Internet-facing zone untrust which, is required for IKE negotiations to occur. However, other services such as management and/or troubleshooting can be individually enabled if required.

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

When you configure a remote Internet Key Exchange (IKE) peer, the IKE peer is identified by IP address, fully qualified domain name/user fully qualified domain name (FQDN/u-FQDN), or ASN1-DN public key infrastructure ([PKI] certificates). In this example, the peer is identified by the IP address. This example uses the standard proposal set for IKE gateway (phase 1) configuration. However, a unique proposal might be created and then specified in the IPsec policy if needed.

After configuring an IPsec VPN with an IKE gateway and an IPsec policy, bind the (st0) interface. This differentiates the VPN as a route-based VPN. For policy-based VPNs, you do not configure an st0 interface. If an st0 interface is not specified, then phase 2 cannot complete negotiations in a route-based VPN.

A security policy permits traffic in one direction but also allows all reply traffic, without the need for a reverse direction policy. However, because traffic might be initiated from either direction, bidirectional policies might be required. Also, you can create more granular policies between zone vpn and zone trust and can permit or deny accordingly. Note that the policies are regular non-tunnel policies; thus, the policies do not specify the IPsec profile. Also note that Network Address Translation (NAT) can be enabled on the policies if required, but that is beyond the scope of this example.

When you configure a security policy, the policy permits all traffic from zone trust to zone untrust. The device translates the source IP and port for outgoing traffic, using the IP address of the egress interface as the source IP and a random higher port for the source port. If required, more granular policies can be created to permit or deny certain traffic entering from zone trust to zone untrust.

The TCP- maximum segment size (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 on 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 ESP packet to exceed the MTU of the physical interface, thus causing fragmentation. Fragmentation increases bandwidth and device resources and is always best avoided.

The basic steps to configure route-based VPNs for SRX Series and J Series devices are:

1. Configure the IP addresses for Gigabit Ethernet (ge-0/0/0.0 and ge-0/0/3.0) and secure tunnel (st0) interfaces. Secure tunnel (st0) interfaces are used in the creation of route-based VPNs.
2. Configure a default route to the Internet next hop and a static route for the remote office LAN. Optionally, you can use a dynamic routing protocol such as OSPF instead, but that is beyond the scope of this application note.
3. Configure security zones, and bind the interfaces to the appropriate zones. Also be sure to enable 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 Gigabit Ethernet (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. For this example, the standard proposal set is used. However you can create a different proposal if necessary.
6. Configure phase 2 (IPsec) VPN settings. Optionally, you can also configure VPN monitor settings, if desired. Note that for this example the standard proposal set is used. However, you can create a different proposal if necessary.
7. Bind secure tunnel (st0) interface to the VPN.
8. Configure security policies to permit remote office traffic into the corporate LAN and vice versa. Also configure the outgoing trust to untrust permit all policy with source NAT for Internet traffic.
9. Configure the TCP- maximum segment size (tcp-mss) for IPsec traffic to eliminate the possibility of fragmented TCP traffic. This lessens the resource usage on the device.

To configure a route-based VPN, perform the following tasks:

- Configuring Junos OS on page 10
- Verifying Route-Based VPN Connections on page 14
- Troubleshooting on page 18
- Results on page 29

## Configuring Junos OS

**Step-by-Step Procedure**

To configure the Junos OS

1. Configure IP addresses for the private LAN, public Internet, and secure tunnel (st0) interfaces.

```
[edit]
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
user@CORPORATE# set interfaces st0 unit 0 family inet address 10.11.11.10/24
```

2. Configure a default route and a route for tunnel traffic for route-based VPNs by specifying the remote peer **st0** interface IP address, or you can simply specify the local **st0** interface itself as the next-hop.

```
[edit]
user@CORPORATE# set routing-options static route 0.0.0.0/0 next-hop 1.1.1.1
user@CORPORATE# set routing-options static route 192.168.168.0/24 next-hop
  st0.0
```

3. Configure security zones and assign interfaces to the zones. If you are terminating the **st0** interface in the same zone as the trusted LAN and if a policy exists to allow intrazone traffic on that zone, then no additional security policies are required.

```
[edit]
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
user@CORPORATE# set security zones security-zone vpn interfaces st0.0
```

4. Configure host-inbound services for each zone.

```
[edit]
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.

```
[edit]
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 vpn address-book address
  remote-net 192.168.168.0/24
```

6. Configure the IKE policy for main mode, predefined standard proposal set and preshared key.

```
[edit]
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 an IKE gateway (phase 1) with a peer IP address, IKE policy, and outgoing interface. The gateway address should be the remote peer's public IP address. It is important also to specify the correct external interface. If either the peer address or external interface is incorrect, then the IKE gateway is not identified during phase 1 negotiation.

```
[edit]
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, which includes the **esp-group2-3des-sha1** and **esp-group2-aes128-sha1** proposals.

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

9. Configure an IPsec VPN with an IKE gateway and an IPsec policy, and then bind it to the **st0** interface.

```
[edit]
```

```
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
user@CORPORATE# set security ipsec vpn ike-vpn bind-interface st0.0
```

10. Configure security policies for tunnel traffic in both directions.

    a. Configure security policies for tunnel traffic entering the zone **trust** to zone **vpn** hierarchy.

    ```
    [edit security policies from-zone trust to-zone vpn]
    user@CORPORATE# set policy vpn-tr-vpn match source-address local-net
    user@CORPORATE# set policy vpn-tr-vpn match destination-address remote-net
    user@CORPORATE# set policy vpn-tr-vpn match application any
    user@CORPORATE# set policy vpn-tr-vpn then permit
    ```

    b. Configure security policies for tunnel traffic in the zone **vpn** to zone **trust** hierarchy.

    ```
    [edit security policies from-zone vpn to-zone trust]
    user@CORPORATE# set policy vpn-vpn-tr match source-address remote-net
    user@CORPORATE# set policy vpn-vpn-tr match destination-address local-net
    user@CORPORATE# set policy vpn-vpn-tr match application any
    user@CORPORATE# set policy vpn-vpn-tr then permit
    ```

11. Configure a source NAT rule and a security policy for Internet traffic.

    ```
    [edit security nat source rule-set nat-out]
    user@CORPORATE# set from zone trust
    user@CORPORATE# set to zone untrust
    user@CORPORATE# set rule interface-nat match source-address 192.168.10.0/24
    user@CORPORATE# set rule interface-nat match destination-address 0.0.0.0/0
    user@CORPORATE# set rule interface-nat then source-nat interface


    [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
    ```

12. Configure the **tcp-mss** to eliminate fragmentation of TCP traffic across the tunnel. Note that the value of 1350 is a recommended starting point for most Ethernet-based networks with an MTU of 1500 or greater. This value might 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, etc. As a general rule, you might need to experiment with different **tcp-mss** values to obtain optimal performance.

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

**User Reference** This is the SSG5 portion of the configuration and is provided for your reference.

The focus of this example is the configuration and troubleshooting of the Junos OS device. For the purpose of completing the network shown in Figure 1 on page 8, a sample of the relevant configurations is provided for an SSG5 device. However the concepts with regard to configuration of route-based VPNs for Juniper Networks Firewall/VPN products are well documented in the Concepts and Examples (C &E) guides. Thus this example does not focus on the SSG configuration. For more information on SSG C&E guides, see: http://www.juniper.net/techpubs/software/screenos/ .

```
set zone name "VPN"
set interface ethernet0/6 zone "Trust"
set interface ethernet0/0 zone "Untrust"
set interface "tunnel.1" zone "VPN"
set interface ethernet0/6 ip 192.168.168.1/24
set interface ethernet0/6 route
set interface ethernet0/0 ip 2.2.2.2/30
set interface ethernet0/0 route
set interface tunnel.1 ip 10.11.11.11/24
set flow tcp-mss 1350
set address "Trust" "192.168.168-net" 192.168.168.0 255.255.255.0
set address "VPN" "10.10.10-net" 10.10.10.0 255.255.255.0
set ike gateway "corp-ike" address 1.1.1.2 Main outgoing-interface ethernet0/0 preshare
   "secretkey" sec-level standard
set vpn "corp-vpn" gateway "corp-ike" replay tunnel idletime 0 sec-level standard
set vpn "corp-vpn" monitor optimized rekey
set vpn "corp-vpn" bind interface tunnel.1
set policy from "Trust" to "Untrust" "ANY" "ANY" "ANY" nat src permit
set policy from "Trust" to "VPN" "192.168.168-net" "10.10.10-net" "ANY" permit
set policy from "VPN" to "Trust" "10.10.10-net" "192.168.168-net" "ANY" permit
set route 10.10.10.0/24 interface tunnel.1
set route 0.0.0.0/0 interface ethernet0/0 gateway 2.2.2.1
```

## Verifying Route-Based VPN Connections

**Step-by-Step Procedure**

To verify route-based VPNs on SRX Series and J Series devices:

1. Confirm VPN status to check the status of any IKE phase 1 security association and Internet Key Exchange (IKE) (phase 1) security associations status using the **show security ike security-associations** command and verifying the following:

   a. In the **show security ike security-associations** command output, notice that the remote address is **2.2.2.2** and the state is **UP**. If the State shows **DOWN** or if there are no IKE security associations present, then there is a problem with phase 1 establishment.

   b. Confirm that the remote IP address, IKE policy, and external interfaces are all correct. Common errors include incorrect IKE policy parameters such as wrong mode type (Aggressive or Main) or mismatched preshared keys or phase 1 proposals (all must match on both peers). An incorrect external interface is another common misconfiguration. This interface must be the correct interface that receives the IKE packets.

   c. If the configurations have been checked, then check the kmd log for any errors or use the **traceoptions** option.

   For information about **traceoptions**, see “Troubleshooting” on page 18.

   ```
   user@CORPORATE> show security ike security-associations
   Index Remote Address State Initiator cookie Responder cookie Mode
   1 2.2.2.2 UP 744a594d957dd513 1e1307db82f58387 Main
   ```

2. Use the **show security ike security-associations index 1 detail** command and verify that the Index number is 1 when you use the **show security ike security-associations index 1 detail** command. This value is unique for each IKE security association and allows you to get more details from that particular security association as shown in this step. 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 and the phase 1 lifetime and traffic statistics. Traffic statistics can be used to verify that traffic flow is proper in both directions.

   Verify also that the number of IPsec security associations created are also in progress. This can help to determine the existence of any completed phase 2 negotiations.

   ```
   user@CORPORATE> show security ike security-associations index 1 detail
   IKE peer 2.2.2.2, Index 1,
   Role: Responder, State: UP
   Initiator cookie: 744a594d957dd513, Responder cookie: 1e1307db82f58387
   Exchange type: Main, Authentication method: Pre-shared-keys
   Local: 1.1.1.2:500, Remote: 2.2.2.2:500
   Lifetime: Expires in 28570 seconds
   Algorithms:
   Authentication : sha1
   Encryption : 3des-cbc
   Pseudo random function: hmac-sha1
   Traffic statistics:
   ```

```
Input bytes : 852
Output bytes : 940
Input packets: 5
Output packets: 5
Flags: Caller notification sent
IPsec security associations: 1 created, 0 deleted
Phase 2 negotiations in progress: 0
```

3. Confirm IPsec (phase 2) status. After IKE phase 1 is confirmed, use the **show security ipsec security-associations** command to view IPsec (phase 2) security associations and verify the following:

   a. The **show security ipsec security-associations** command output verifies that there is one IPsec security association (SA) pair and that the port used is **500**, which means that there is no NAT traversall (nat-traversal would show port 4500 or a random high port).

   b. The security parameter index (SPI) is used for both directions, the lifetime is in seconds, and the usage limits or lifesize is in kilobytes. From the show command output, you can see **3363/ unlim**, which means that the phase 2 lifetime is set to expire in 3363 seconds and that there is no lifesize specified (thus it shows unlimited). The Phase 2 lifetime can differ from the phase 1 lifetime because phase 2 is not dependent on phase 1 after the VPN is up.

   c. The **Mon** column refers to the 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 information about VPN monitoring, refer to the complete Junos OS documentation.

   d. Note that **vsys** always shows **0**, and the ID number is **16384**. 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
<16384 2.2.2.2 500 ESP:3des/sha1 76d64d1d 3363/ unlim - 0
>16384 2.2.2.2 500 ESP:3des/sha1 a1024ee2 3363/ unlim - 0
```

By using the **show security ipsec security-associations index 16384 detail** command you can see **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.

   a. If no IPsec SA is listed, confirm that the phase 2 proposals, including the proxy ID settings, are correct for both peers.

   b. Note that for route-based VPNs, the default local proxy ID is **0.0.0.0/0**, the remote proxy ID is **0.0.0.0/0**, and the service is **any**. This can cause issues if you have multiple route-based VPNs from the same peer IP. In this case, you need to specify unique proxy IDs for each IPsec SA.

   c. Also, for some third-party vendors, you might need to configure the proxy ID to match. Another common reason for phase 2 failing to complete might be failure to specify ST interface binding. If IPsec cannot complete, check the kmd log or set **traceoptions** as detailed in “Troubleshooting” on page 18.

```
user@CORPORATE> show security ipsec security-associations index 16384 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
Direction: inbound, SPI: 1993755933, AUX-SPI: 0
Hard lifetime: Expires in 3352 seconds
Lifesize Remaining: Unlimited
Soft lifetime: Expires in 2775 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: 2701283042, AUX-SPI: 0
Hard lifetime: Expires in 3352 seconds
Lifesize Remaining: Unlimited
Soft lifetime: Expires in 2775 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
```

4. Check statistics and errors for an IPsec SA. Use the **show security ipsec statistics index 16384** command to check Encapsulating Security Payload (ESP) and Authentication Header (AH) counters and 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 security ipsec statistics index 16384** command multiple times and confirm that the encrypted and decrypted packet counters are incrementing. Also, see whether any of the error counters increment while you are experiencing the issue. It might also be necessary to enable security flow traceoptions to see which ESP packets have errors and why. See "Troubleshooting" on page 18.

```
user@CORPORATE> show security ipsec statistics index 16384
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
```

5. Test traffic flow across the VPN. After you confirm 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 to use the **ping** command. You can ping from the local host PC to the remote host PC. You can also initiate ping packets from the SRX Series or J Series devices itself. To send ping packets from the SRX Series or J Series devices to the remote host PC use the **ping** command. Below is an example of ping testing from the SRX Series or J Series devices to the remote PC 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
```

6. Note that when sending ping packets from the SRX Series or J Series devices, the source interface must be specified to make sure that route lookup is correct and that the appropriate zones can be referenced in the policy lookup. In this case because **ge-0/0/0.0** resides in the same security zone as the local host PC, then **ge-0/0/0** needs to be specified in the ping command so that the policy lookup can be from zone **trust** to zone **vpn**. Likewise, you can initiate a ping command from the remote host to the local host. Also, you can initiate a ping from the SSG5 itself as shown.

```
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
```

A ping failure from either direction could indicate an issue with routing, policy or end host, or perhaps an issue with the encryption/decryption of the ESP packets. One way to check is to view IPsec statistics to see whether any errors are reported. You can also 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, then the issue is probably not with the end host. For routing and policy issues, you can enable security flow traceoptions, as detailed in "Troubleshooting" on page 18.

## 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 determine at which layer the failure occurs.

Following this methodology, the first step in troubleshooting is to confirm the physical connectivity of the Internet link at the physical and data link level. Next, using the **ping** command, confirm that the SRX Series or J Series devices has connectivity to the Internet next-hop device, and then confirming connectivity to the remote Internet Key Exchange (IKE) peer. Assuming that there are no problems, confirm that IKE phase 1 can complete by running the verification commands as shown in "Verifying Route-Based VPN Connections" on page 14. Once phase 1 is confirmed, then confirm phase 2. Finally, confirm that traffic is flowing across the VPN. If the VPN is not in the **UP** state, then there is very little reason to test any transit traffic across the VPN. Likewise, if phase 1 was not successful, 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 a Junos OS operating configuration. This means that a configuration commit is necessary before a traceoption takes effect. Likewise, removing traceoptions require deleting or deactivating the configuration, followed by committing the configuration. With a traceoption flag enabled, the data from the traceoption is written to a log file, which might be predetermined or manually configured and stored in persistent memory. Any trace log is retained even after a system reboot. Keep in mind the available storage on the flash memory before you implement traceoptions.

To troubleshoot route-Based VPNs on SRX Series and J Series devices:

1. Check the available storage using the **show system storage** command.

   The **/dev/ad0s1a** directory represents the onboard flash memory and in the following example is at **65%** of 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 directory /var/log. To view a list of all logs in **/var/log**, use the **show log** 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 logging of the output to the filenames specified or to the default log file for the traceoption. View the appropriate log to view the trace output. Execute the following commands to view the appropriate logs:

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

   NOTE: For the Juniper Networks SRX3000 line, SRX5000 line, and SRX1400 devices, the logs are located in the /var/tmp directory, and the SPU ID values are included in the log filename. For example /var/tmp/kmd14.

   Logs can also be uploaded to an FTP server by running the **file copy** command. The syntax is: **file copy <filename> <destination>** as shown.

   ```
   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. Troubleshoot IKE and IPsec issues. To view success or failure messages in IKE or IPsec, display the kmd log using the **show log kmd** command. Although the kmd log gives a general reason for any failure, You might want to obtain additional details by enabling IKE traceoptions.

   NOTE: 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 is an example of all IKE **traceoptions**.

   ```
   user@CORPORATE#  set security ike traceoptions 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]
   root@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
   ```

By default, if no filename is specified, then all IKE traceoptions output is written to the kmd log. However, you can specify a different filename if you wish. If a different filename is specified, then all IKE and IPSec related logs are no longer written to the kmd log.

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 is generated and stored in flash.

NOTE: **Remember to commit the changes to start the trace.**

The following is an example of recommended **traceoptions** for troubleshooting most IKE-related issues.

```
[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
```

4. Review the kmd log for phase 1 and phase 2 success or failure messages. You can view and verify successful phase 1 and phase 2 completions. Some failure instances from the **show log kmd** command are shown.

   a. The local address is **1.1.1.2** and the remote peer is **2.2.2.2**

   b. The output **udp:500** indicates that no NAT-traversal is negotiated.

   c. You should see a phase 1 done message, along with the role (initiator or responder).

   d. You should also 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 mentioned in "Verifying Route-Based VPN Connections" on page 14.

   The following is an example of the **show log kmd** command output.

```
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)
```

5. 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 both peers.

```
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)
```

6. Phase 1 failing to complete, example 2. In the following example, you can see that 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 might seem to indicate that no proposal was chosen. However, in this case, you see a message, **peer:2.2.2.2 is not recognized**. You need to check if this message is due to incorrect peer address, mismatched peer ID type, or incorrect peer ID, depending on whether this is a dynamic or static VPN before the phase 1 proposal is checked. To resolve this issue, configure the local peer with the correct peer IP address. Also confirm that the peer is configured with IP address as the IKE ID type.

```
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)
```

7. Phase 1 failing to complete, example 3. In the following show command output, the remote peer address is **2.2.2.2**. The message **Invalid payload type** usually means that there is a problem with the decryption of the IKE packet due to mismatched preshared keys. 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)
```

8. Phase 1 successful, phase 2 failing to complete, example 1. In Step 7 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 failing is shown in the output as **No proposal chosen** during phase 2 negotiations. The issue is probably 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)
```

9. Phase 1 successful, phase 2 failing to complete, example 2. The following output indicates that phase 1 was successful. The reason for phase 2 failing might seem to be No proposal was chosen. However, you also see 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 (remote=192.168.168.0/24, local=10.10.20.0/24, service=any) was received and because this does not match the configurations on the local peer, 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 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)
```

10. The following is a problem scenario using the network diagram. See Figure 1 on page 8.

    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, 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 security traceoptions can increase system CPU and memory usage. Therefore, enabling security flow traceoptions is not recommended during peak traffic load times or when CPU usage 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 security flow traceoptions and remove any unnecessary log files from the flash memory after you complete troubleshooting.**

11. Enable security flow traceoptions for routing or policy issues. The following example shows the of output for security flow traceoptions.

```
user@CORPORATE# set security flow traceoptions 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
```

```
user@CORPORATE# set security flow traceoptions 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
```

By default if no filename is specified, then all flow traceoptions output is written to the security-trace 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.

Junos OS can configure packet filters to limit the scope of the traffic to be captured by the flow traceoptions. 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. Furthermore 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 flow packet filter options.

```
user@CORPORATE# set security flow traceoptions 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
```

12. Terms listed within the same packet filter act as a Boolean logical AND statement. That 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 shows an example of recommended traceoptions for security flow.

```
[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
```

13. The output in this step helps explain the reasoning behind each flow traceoption setting. In the example the **security-trace** log file is set to 1 MB and up to 3 files are created. The reason for this is that because of the nature of flow traceoptions, a single file could become full fairly quickly, depending on how much traffic is captured. The **basic-datapath** flag shows details for most flow-related problems.

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

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

14. The filter in Step 12 is for capturing the decapsulated or unencrypted traffic from the remote PC to the local PC. Because there are multiple terms, this filter acts as a Boolean logical AND. That 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. Since packet filters are bidirectional, it is not necessary to configure a filter for the reply traffic.

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

15. No filter is required for capturing the reply traffic. However, a filter captures only the packets which were originally sourced from the specified side. Thus, the **local-to-remote filter** in Step 12 is still required to capture traffic which sources from the local side to the remote side.

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

16. The filter in Step 12 is optional and depends on whether or not the previous filter is able to capture 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 are perhaps not of interest. If the unencrypted traffic is captured, this last filter might not be necessary.

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

So with the three problem statements mentioned in the problem scenario using Figure 1 on page 8 in Step 10, 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. The next step is to validate the first problem statement to confirm that the remote PC can ping the local PC. Next, troubleshoot the second problem statement to find out why the traffic fails in the reverse direction.

17. Validate the first problem statement. Begin by sending 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 all flow traceoptions output by running the **show log security-trace** command. Below is the flow traceoptions output showing the successful traffic flow from the remote PC to the local PC. The first packet captured is the ESP, or encrypted packet.

    Based on the top header, 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 by running 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 logical interface index numbers by running the **show interface extensive** command.

```
user@CORPORATE> show log security-trace
******<2.2.2.2/30422->1.1.1.2/19741;50> matched filter remote-esp: <untrust/ge-0/0/3.0> ******
Oct 6 22:58:39 22:58:38.1430964:CID-0:RT: packet [184] ipid = 30440, @498aab8e ******
Oct 6 22:58:39 22:58:38.1430974:CID-0:RT: ge-0/0/3.0:2.2.2.2->1.1.1.2, 50
Oct 6 22:58:39 22:58:38.1430981:CID-0:RT: find flow: table 0x4b5265e0, hash 216892(0x3ffff), sa
2.2.2.2, da 1.1.1.2, sp 30422, dp 19741, proto 50, tok 14
Oct 6 22:58:39 22:58:38.1430998:CID-0:RT: find flow: table 0x4b59eb00, hash 3900(0xfff), sa
2.2.2.2, da 1.1.1.2, sp 30422, dp 19741, proto 50, tok 14
Oct 6 22:58:39 22:58:38.1431014:CID-0:RT: flow session id 257024
Oct 6 22:58:39 22:58:38.1431019:CID-0:RT: flow_decrypt: tun 51761360(flag b), iif 68
Oct 6 22:58:39 22:58:38.1431061:CID-0:RT:inject tunnel pkt mbuf 0x498aa9e0
Oct 6 22:58:39 22:58:38.1431068:CID-0:RT:injected tunnel pkt mbuf 0x498aa9e0
```

18. Based on the top header in the output of the **show log security-trace** command, the packet is from **192.168.168.10** to **10.10.10.10**, and the IP protocol is 1. The ingress interface is **st0.0**, which means that the source was from across the VPN. The ingress zone is the **vpn** zone, and the matching packet filter is **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. Below is the decrypted packet output.

```
user@CORPORATE> show log security-trace
******<192.168.168.10/2048->10.10.10.10/64949;1> matched filter remote-to-local: <vpn/st0.0> ******
Oct 6 22:58:39 22:58:38.1431093:CID-0:RT: packet [128] ipid = 9728, @498aabb2 ******
Oct 6 22:58:39 22:58:38.1431102:CID-0:RT: st0.0:192.168.168.10->10.10.10.10, icmp, (8/0)
Oct 6 22:58:39 22:58:38.1431108:CID-0:RT: find flow: table 0x4b5265e0, hash 59180(0x3ffff), sa
192.168.168.10, da 10.10.10.10, sp 23164, dp 1024, proto 1, tok 10
Oct 6 22:58:39 22:58:38.1431125:CID-0:RT: flow_first_sanity_check: in <st0.0>, out <N/A>
Oct 6 22:58:39 22:58:38.1431133:CID-0:RT: flow_first_in_dst_nat: in <st0.0>, out <N/A>
Oct 6 22:58:39 22:58:38.1431136:CID-0:RT: flow_first_in_dst_nat: dst_adr 10.10.10.10, sp 23164,
dp 1024
Oct 6 22:58:39 22:58:38.1431144:CID-0:RT: chose interface st0.0 as incoming nat if.
Oct 6 22:58:39 22:58:38.1431148:CID-0:RT: flow_first_routing: Before route-lookup ifp: in
<st0.0>, out <N/A>
Oct 6 22:58:39 22:58:38.1431151:CID-0:RT:flow_first_routing: call flow_route_lookup(): src_ip
192.168.168.10, x_dst_ip 10.10.10.10, ifp st0.0, sp 23164, dp 1024, ip_proto 1, tos 0
Oct 6 22:58:39 22:58:38.1431161:CID-0:RT:Doing DESTINATION addr route-lookup
Oct 6 22:58:39 22:58:38.1431170:CID-0:RT:Doing SOURCE addr route-lookup
Oct 6 22:58:39 22:58:38.1431174:CID-0:RT: routed (x_dst_ip 10.10.10.10) from st0.0 (st0.0 in 0)
```

```
to ge-0/0/0.0, Next-hop: 10.10.10.10
Oct 6 22:58:39 22:58:38.1431188:CID-0:RT: policy search from zone (vpn) 8-> zone (trust) 6
Oct 6 22:58:39 22:58:38.1431204:CID-0:RT: policy found 6
Oct 6 22:58:39 22:58:38.1431209:CID-0:RT:No src xlate
Oct 6 22:58:39 22:58:38.1431212:CID-0:RT: choose interface ge-0/0/0.0 as outgoing phy if
Oct 6 22:58:39 22:58:38.1431216: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 22:58:39 22:58:38.1431222:CID-0:RT: Using app_id from service lookup 0
Oct 6 22:58:39 22:58:38.1431226:CID-0:RT: session application type 0, name (null), timeout
60sec, alg 0
Oct 6 22:58:39 22:58:38.1431230:CID-0:RT: service lookup identified service 0.
Oct 6 22:58:39 22:58:38.1431235:CID-0:RT: flow_first_final_check: in <st0.0>, out <ge-0/0/0.0>
Oct 6 22:58:39 22:58:38.1431243:CID-0:RT: install vector flow_ttl_vector
Oct 6 22:58:39 22:58:38.1431246:CID-0:RT: install vector flow_l2prepare_xlate_vector
Oct 6 22:58:39 22:58:38.1431250:CID-0:RT: install vector flow_frag_list_vector
Oct 6 22:58:39 22:58:38.1431253:CID-0:RT: install vector flow_fragging_vector1
Oct 6 22:58:39 22:58:38.1431255:CID-0:RT: install vector flow_encap_vector
Oct 6 22:58:39 22:58:38.1431258:CID-0:RT: install vector flow_send_vector
Oct 6 22:58:39 22:58:38.1431261:CID-0:RT: install vector NULL
Oct 6 22:58:39 22:58:38.1431283:CID-0:RT: create new vector list 2-59b5c330.
Oct 6 22:58:39 22:58:38.1431290:CID-0:RT: existing vector list 2-59b5c330.
Oct 6 22:58:39 22:58:38.1431295:CID-0:RT: Session (id:4) created for first pak 2
Oct 6 22:58:39 22:58:38.1431299:CID-0:RT: flow_first_install_session======> 0x4c6fb828
Oct 6 22:58:39 22:58:38.1431305:CID-0:RT: nsp 0x4c6fb828, nsp2 0x4c6fb880
Oct 6 22:58:39 22:58:38.1431317:CID-0:RT: 5 tuple sa 192.168.168.10, da 10.10.10.10, sp 23164, dp
1024, proto 1
Oct 6 22:58:39 22:58:38.1431327:CID-0:RT: set route old fto 0x59b5c1a8, new fto 0x59b5c1a8
Oct 6 22:58:39 22:58:38.1431336:CID-0:RT: 5 tuple sa 10.10.10.10, da 192.168.168.10, sp 1024, dp
23164, proto 1
Oct 6 22:58:39 22:58:38.1431344:CID-0:RT: set route old fto 0x59b5c130, new fto 0x59b5c130
Oct 6 22:58:39 22:58:38.1431355:CID-0:RT: flow session id 4
Oct 6 22:58:39 22:58:38.1431362:CID-0:RT: post addr xlation: 192.168.168.10->10.10.10.10.
Oct 6 22:58:39 22:58:38.1431368:CID-0:RT: encap vector
Oct 6 22:58:39 22:58:38.1431371:CID-0:RT: no more encapping needed
Oct 6 22:58:39 22:58:38.1431376:CID-0:RT:mbuf 0x498aa9e0, exit nh 0xf0000006
```

In this example, there is no existing session for this flow, so the first thing that happens is packet processing occurs. Next, the route lookup takes place. Route lookup must occur in order 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 **st0.0** is associated with zone **vpn**, the policy lookup is from-zone **vpn** to-zone **trust**. Policy 6 was found, which permits the traffic.

19. To see details for policy 6, use the **show security policies** command.

```
user@CORPORATE> show security policies | find “Index: 6”
Policy: vpn-vpn-tr, State: enabled, Index: 6, Sequence number: 1
Source addresses: remote-net
Destination addresses: local-net
Applications: any
Action: permit, log
```

20. In the following example the session is created; in this case, the session ID is 4. The reply packet should also be captured and shows existing session 4 is found. 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 40004000. This means that the tunnel is 0x4000, which converts to SA index 16384. 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->192.168.168.10/7009;1> matched filter local-to-remote: <trust/
ge-0/0/0.0> ******
Oct 6 22:58:39 22:58:38.1454263:CID-0:RT: packet [128] ipid = 47151, @49797e8e ******
Oct 6 22:58:39 22:58:38.1454274:CID-0:RT: ge-0/0/0.0:10.10.10.10->192.168.168.10, icmp, (0/0)
Oct 6 22:58:39 22:58:38.1454280:CID-0:RT: find flow: table 0x4b5265e0, hash 184363(0x3ffff), sa
10.10.10.10, da 192.168.168.10, sp 1024, dp 23164, proto 1, tok 12
Oct 6 22:58:39 22:58:38.1454297:CID-0:RT: flow session id 4
Oct 6 22:58:39 22:58:38.1454305:CID-0:RT:xlate_icmp_pak: set nat invalid 4, timeout 1, reason 3
Oct 6 22:58:39 22:58:38.1454311:CID-0:RT: post addr xlation: 10.10.10.10->192.168.168.10.
Oct 6 22:58:39 22:58:38.1454319:CID-0:RT: encap vector
Oct 6 22:58:39 22:58:38.1454322:CID-0:RT: going into tunnel 40004000.
Oct 6 22:58:39 22:58:38.1454327:CID-0:RT: flow_encrypt: 0x51761360
Oct 6 22:58:39 22:58:38.1454333:CID-0:RT:mbuf 0x49797d00, exit nh 0x60010
```

21. Troubleshooting the second problem statement.

    Based on 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 sample output showing a ping failure.

    Based on the top header in the output, the packet is from **10.10.10.10** to **192.168.168.10**, and the IP protocol is **1**. Because no session is found, the first thing that happens is packet processing occurs. Next, route lookup occurs. However, instead of finding a route for **192.168.168.10** to **st0.0** in the **vpn** zone, this packet is instead routed to **ge-0/0/0.0** in the **untrust** zone. Because policy lookup is from zone **trust** to zone **untrust**, the packet matches policy 4, which happens to be the **any-permit** policy and the packet never reaches the **trust** to **vpn** policy.

```
user@CORPORATE> show log security-trace
******<10.10.10.10/2048->192.168.168.10/17763;1> matched filter local-to-remote: <trust/
ge-0/0/0.0> ******
Oct 6 23:01:07 23:01:07.697258:CID-0:RT: packet [128] ipid = 47206, @498c03ae ******
Oct 6 23:01:07 23:01:07.697269:CID-0:RT: ge-0/0/0.0:10.10.10.10->192.168.168.10, icmp, (8/0)
Oct 6 23:01:07 23:01:07.697276:CID-0:RT: find flow: table 0x4b5265e0, hash 20039(0x3ffff), sa
10.10.10.10, da 192.168.168.10, sp 44700, dp 1024, proto 1, tok 12
Oct 6 23:01:07 23:01:07.697293:CID-0:RT: flow_first_sanity_check: in <ge-0/0/0.0>, out <N/A>
Oct 6 23:01:07 23:01:07.697303:CID-0:RT: flow_first_in_dst_nat: in <ge-0/0/0.0>, out <N/A>
Oct 6 23:01:07 23:01:07.697306:CID-0:RT: flow_first_in_dst_nat: dst_adr 192.168.168.10, sp 44700,
dp 1024
Oct 6 23:01:07 23:01:07.697313:CID-0:RT: chose interface ge-0/0/0.0 as incoming nat if.
Oct 6 23:01:07 23:01:07.697317:CID-0:RT: flow_first_routing: Before route-lookup ifp: in <ge-
0/0/0.0>, out <N/A>
Oct 6 23:01:07 23:01:07.697321: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 44700, dp 1024, ip_proto 1, tos 0
Oct 6 23:01:07 23:01:07.697331:CID-0:RT:Doing DESTINATION addr route-lookup
Oct 6 23:01:07 23:01:07.697340:CID-0:RT:Doing SOURCE addr route-lookup
Oct 6 23:01:07 23:01:07.697345: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 23:01:07 23:01:07.697353:CID-0:RT: policy search from zone (trust) 6-> zone
(untrust) 7
Oct 6 23:01:07 23:01:07.697368:CID-0:RT: policy found 4
Oct 6 23:01:07 23:01:07.697380:CID-0:RT: dip id = 2/0, 10.10.10.10/44700->1.1.1.2/1024
Oct 6 23:01:07 23:01:07.697391:CID-0:RT: choose interface ge-0/0/3.0 as outgoing phy if
Oct 6 23:01:07 23:01:07.697395: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 23:01:07 23:01:07.697401:CID-0:RT: Using app_id from service lookup 0
Oct 6 23:01:07 23:01:07.697404:CID-0:RT: session application type 0, name (null), timeout 60sec,
alg 0
Oct 6 23:01:07 23:01:07.697409:CID-0:RT: service lookup identified service 0.
Oct 6 23:01:07 23:01:07.697413:CID-0:RT: flow_first_final_check: in <ge-0/0/0.0>, out <ge-
0/0/3.0>
Oct 6 23:01:07 23:01:07.697420:CID-0:RT: existing vector list 0-59b5c2a8.
Oct 6 23:01:07 23:01:07.697427:CID-0:RT: existing vector list 0-59b5c2a8.
Oct 6 23:01:07 23:01:07.697433:CID-0:RT: Session (id:11) created for first pak 0
Oct 6 23:01:07 23:01:07.697436:CID-0:RT: flow_first_install_session======> 0x4c6fc120
Oct 6 23:01:07 23:01:07.697442:CID-0:RT: nsp 0x4c6fc120, nsp2 0x4c6fc178
Oct 6 23:01:07 23:01:07.697453:CID-0:RT: 5 tuple sa 10.10.10.10, da 192.168.168.10, sp 44700, dp
1024, proto 1
Oct 6 23:01:07 23:01:07.697462:CID-0:RT: set route old fto 0x59b5c068, new fto 0x59b5c068
Oct 6 23:01:07 23:01:07.697479:CID-0:RT: 5 tuple sa 192.168.168.10, da 1.1.1.2, sp 1024, dp 1024,
proto 1
Oct 6 23:01:07 23:01:07.697487:CID-0:RT: set route old fto 0x59b5c1a8, new fto 0x59b5c1a8
Oct 6 23:01:07 23:01:07.697498:CID-0:RT: flow session id 11
Oct 6 23:01:07 23:01:07.697506:CID-0:RT: post addr xlation: 1.1.1.2-192.168.168.10.
Oct 6 23:01:07 23:01:07.697512:CID-0:RT:mbuf 0x498c0200, exit nh 0x60010
```

22. To view the route, use the **show route <destination_IP_address>** command.

    ```
    user@CORPORATE> show route 192.168.168.10

    inet.0: 7 destinations, 7 routes (7 active, 0 holddown, 0 hidden)
    + = Active Route, - = Last Active, * = Both
    0.0.0.0/0 *[Static/5] 00:23:56
    > to 1.1.1.1 via ge-0/0/3.0
    ```

    From the output, it is clear that a route does not exist for 192.168.168.0/24. Thus, the default route is used.

23. To create a route for 192.168.168.0/24, configure a route with the next hop as **st0.0**. After the route is in place and the configuration is committed, you might still see traffic failing as shown in the following output. Use the **show log security-trace** command to see the traffic failing.

```
user@CORPORATE> show log security-trace
******<10.10.10.10/2048->192.168.168.10/17163>;1 matched filter local-to-remote: <trust/
ge-0/0/0.0> ******
Oct 6 23:03:12 23:03:11.937590:CID-0:RT: packet [128] ipid = 47252, @497a63ee ******
Oct 6 23:03:12 23:03:11.937602:CID-0:RT: ge-0/0/0.0:10.10.10.10->192.168.168.10, icmp, (8/0)
Oct 6 23:03:12 23:03:11.937609:CID-0:RT: find flow: table 0x4b5265e0, hash 43594(0x3ffff), sa
10.10.10.10, da 192.168.168.10, sp 45300, dp 1024, proto 1, tok 12
Oct 6 23:03:12 23:03:11.937626:CID-0:RT: flow_first_sanity_check: in <ge-0/0/0.0>, out <N/A>
Oct 6 23:03:12 23:03:11.937636:CID-0:RT: flow_first_in_dst_nat: in <ge-0/0/0.0>, out <N/A>
Oct 6 23:03:12 23:03:11.937640:CID-0:RT: flow_first_in_dst_nat: dst_adr 192.168.168.10, sp 45300,
dp 1024
Oct 6 23:03:12 23:03:11.937647:CID-0:RT: chose interface ge-0/0/0.0 as incoming nat if.
Oct 6 23:03:12 23:03:11.937651:CID-0:RT: flow_first_routing: Before route-lookup ifp: in <ge-
0/0/0.0>, out <N/A>
Oct 6 23:03:12 23:03:11.937654: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 45300, dp 1024, ip_proto 1, tos 0
Oct 6 23:03:12 23:03:11.937664:CID-0:RT:Doing DESTINATION addr route-lookup
Oct 6 23:03:12 23:03:11.937674:CID-0:RT:Doing SOURCE addr route-lookup
Oct 6 23:03:12 23:03:11.937678: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 st0.0, Next-hop: 192.168.168.10
Oct 6 23:03:12 23:03:11.937686:CID-0:RT: policy search from zone (trust) 6-> zone (vpn) 8
```

```
Oct 6 23:03:12 23:03:11.937692:CID-0:RT: policy found 2
Oct 6 23:03:12 23:03:11.937695:CID-0:RT: packet dropped, denied by policy
```

In the output you can see that the route lookup is behaving as expected unlike in Step 21. The policy lookup is from zone **trust** to zone **vpn**. However the packet matches policy 2, which is the preconfigured default deny policy.

24. To view all the configured policies, use the **show security policies** command. From the output, you can see that there is no policy from zone **trust** to zone **vpn** to permit the traffic. To resolve this issue, add an appropriate policy. After the policy is added, ping commands from the local PC to the remote PC are successful.

```
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
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
From zone: vpn, To zone: trust
Policy: vpn-vpn-tr, State: enabled, Index: 6, Sequence number: 1
Source addresses: remote-net
Destination addresses: local-net
Applications: any
Action: permit
```

25. Did the remote PC to local PC traffic succeed despite the fact that there is no route or policy configured for the reply traffic?

    The order of packet processing is important to answer the question. Junos OS first inspects the packet to see whether an existing session already exists. If no session exists, then a route lookup is performed. Next the policy lookup is performed. When the first packet reached the device from **st0.0** to **ge-0/0/0.0**, the session was built for the reply packet. When the reply packet was received, it matched the existing session and was 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$t4wI4Oc0NR8dZlDNz0No2."; ## 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 10.10.10.10/24;
            }
        }
    }
    ge-0/0/3 {
        unit 0 {
            family inet {
                address 1.1.1.2/30;
            }
        }
    }
    st0 {
        unit 0 {
            family inet {
                address 10.11.11.10/24;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 1.1.1.1;
        route 192.168.168.0/24 next-hop st0.0;
    }
}
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 {
      bind-interface st0.0;
      ike {
        gateway ike-gate;
        ipsec-policy vpn-policy1;
      }
    }
  }
}
zones {
  security-zone untrust {
    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;
    }
  }
  security-zone vpn {
    address-book {
      address remote-net 192.168.168.0/24;
    }
    interfaces {
      st0.0;
    }
  }
}
policies {
  from-zone trust to-zone untrust {
    policy any-permit {
      match {
        source-address any;
        destination-address any;
        application any;
      }
      then {
        permit {
          source-nat {
```

```
                            interface;
                        }
                    }
                }
            }
        }
        from-zone trust to-zone vpn {
            policy vpn-tr-vpn {
                match {
                    source-address local-net;
                    destination-address remote-net;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
        from-zone vpn to-zone trust {
            policy vpn-vpn-tr {
                match {
                    source-address remote-net;
                    destination-address local-net;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
    }
    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**

- Configuring Route-Based VPN Using an SRX Series or a J Series Device and an SSG Device Overview on page 5
- Comparing Policy-Based VPNs and Route-Based VPNs on page 6