

Application Note

J Series / SRX Series Route-Based VPN Configuration and Troubleshooting

Version 1.3

Richard Kim
Technical Support Engineer
Advanced JTAC

June 2009



Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
408 745 2000 or 888 JUNIPER
www.juniper.net

Introduction

JUNOS, 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 has been greatly enhanced with security and VPN capabilities from Juniper Networks Firewall/IPSec VPN Platforms, which includes the SSG product family. The purpose of this application note is to detail IPSec interoperability configuration between a J Series or SRX Series device and an SSG device using a route-based VPN. This application note also includes troubleshooting information on the JUNOS side.

Included Platforms and Software Versions

This document applies to the following devices:

- J Series devices running:
 - JUNOS 9.4 and above
 - JUNOS with Enhanced Services 8.5 through 9.3
- SRX Series devices

Overview

The configuration of the J Series or SRX Series device for VPN support is particularly flexible. You can create route-based and policy-based VPN tunnels. This application note will focus on route-based VPN tunnels, but first it is important to understand the differences between the two and why one method may be recommended over the other.

With policy-based VPN tunnels, a tunnel is treated as an object that together with source, destination, application, and action, comprises 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 (ST) 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 ST interfaces that the device supports—whichever number is lower. A 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 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

include tunnel.

Another advantage that route-based VPNs offer is the exchange of dynamic routing information through VPN tunnels. You can enable an instance of a dynamic routing protocol, such as OSPF, on an ST interface that is bound to a VPN tunnel. This is not supported with policy-based VPNs. Also for hub-and-spoke topologies you must use route-based configuration. Lastly you can define static NAT addresses specifically for ST interfaces. OSPF, hub-and-spoke and static NAT configurations are beyond the scope of this application note.

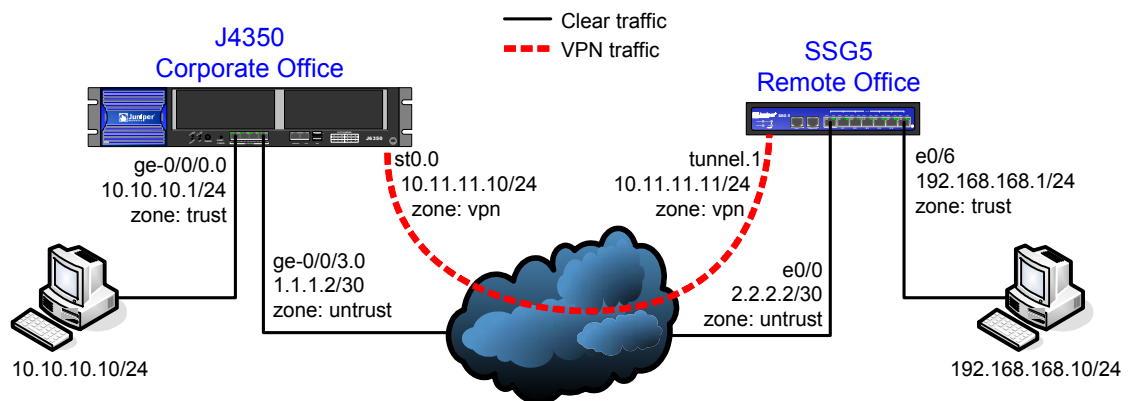
When a tunnel does not connect large networks running dynamic routing protocols and you do not need to conserve tunnels or define various policies to filter traffic through the tunnel, a policy-based tunnel makes sense. Also, because there is no network beyond a dialup VPN client, policy-based VPN tunnels can be a good choice for dialup 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 that third party requires separate SAs for each remote subnet.

For the purposes of this application note we will focus on route-based VPN configuration and troubleshooting.

Network Diagram

Refer to Figure 1 below for Network Topology used for this configuration example. The J4350 is the J Series device documented in this Application Note.

Figure 1.



Configuration Steps

This example assumes the following (refer to figure 1 above):

- Internal LAN interface is `ge-0/0/0` in zone “trust” and will have a private IP subnet.
- Internet interface is `ge-0/0/3` in zone “untrust” and will have a public IP.
- The secure tunnel interface `st0` will be in “vpn” zone to allow for configuring unique policies specifically for tunnel (encrypted) traffic while maintaining unique policies for clear (non-encrypted) traffic.
- All traffic between the local and remote LANs are to be permitted, and traffic may be initiated from either side.

- The SSG5 has already been preconfigured with the correct information from this example.

Basic Steps to Configure:

1. Configure the IP addresses for ge-0/0/0.0, ge-0/0/3.0 and st0.0 interfaces.
2. Configure default route to Internet next-hop and also 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 necessary host-inbound services on the interfaces or the zone. For this example you must enable ike service on either ge-0/0/3 interface or the “untrust” zone.
4. Configure address book entries for each zone. This will be necessary for the security policies.
5. Configure phase 1 (IKE) gateway settings. For this example, “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 we are using “Standard” proposal set. However you can create a different proposal if necessary.
7. Bind interface st0.0 to the VPN.
8. Configure security policies to permit remote office traffic into the corporate LAN and vice versa. Also configure outgoing “trust” to “untrust” permit all policy with source NAT for Internet traffic.
9. Configure tcp-mss for IPSec traffic to eliminate the possibility of fragmented TCP traffic. This will lessen the resource utilization on the device.

J Series/SRX Series Configuration Example

To begin, enter configuration mode with either command: `configure` or `edit`.

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

```
set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.1/24
set interfaces ge-0/0/3 unit 0 family inet address 1.1.1.2/30
set interfaces st0 unit 0 family inet address 10.11.11.10/24
```

JUNOS uses the concept of units for the logical component of an interface. In this example unit 0 and family inet (IPv4) is used. Though not mandatory, for ST interfaces it is recommended that both peers have an IP address within the same logical subnet since the link is logically a point-to-point link.

Configure default route and route for tunnel traffic.

```
set routing-options static route 0.0.0.0/0 next-hop 1.1.1.1
set routing-options static route 192.168.168.0/24 next-hop st0.0
```

For static routes you would normally specify the gateway IP address as the next-hop. For route-based VPNs you can specify the remote peer ST interface IP, or to simplify you can specify the local ST interface itself as the next-hop.

Configure security zones and assign interfaces to the zones.

```
set security zones security-zone trust interfaces ge-0/0/0.0
set security zones security-zone untrust interfaces ge-0/0/3.0
set security zones security-zone vpn interfaces st0.0
```

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 to access the VPN. Note also if terminating the ST interface in the same zone as the trusted LAN and if a policy exists to allow intra-zone traffic on that zone, then no additional security policies would be required.

Configure host-inbound services for each zone.

```
set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic system-services ike
```

Host-inbound services are for traffic destined for the J Series or SRX Series 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 we are assuming that we want to allow all such services from zone “trust”. For security reasons we are only allowing ike on the Internet facing zone “untrust” which is required for IKE negotiations to occur. However other services such as for management and/or troubleshooting can also be individually enabled if required.

Configure address book entries for each zone.

```
set security zones security-zone trust address-book address local-net
10.10.10.0/24
set security zones security-zone vpn address-book address remote-net
192.168.168.0/24
```

For this example we are using address-book object names “local-net” and “remote-net”. There are some limitations with regards to which characters are supported for address-book names. Please refer to complete J Series or SRX Series documentation for more details.

Configure IKE policy for main mode, predefined Standard proposal-set and pre-shared key.

```
set security ike policy ike-policy1 mode main
set security ike policy ike-policy1 proposal-set standard
set security ike policy ike-policy1 pre-shared-key ascii-text "secretkey"
```

For the purposes of this application note we are using proposal set “Standard” which includes preshared-group2-3des-sha1 and preshared-group2-aes128-sha1 proposals. However a unique proposal may be created and then specified in the IKE policy in accordance with your corporate security policy.

Configure IKE gateway (phase 1) with peer IP address, IKE policy and outgoing interface.

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

A remote IKE peer can be identified by either IP address, FQDN/u-FQDN or ASN1-DN (PKI certificates). For this example we are identifying the peer by IP address. Therefore 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 specified is incorrect then the IKE gateway will not be properly identified during phase 1 negotiations.

Configure IPsec policy for Standard proposal set.

```
set security ipsec policy vpn-policy1 proposal-set standard
```

As mentioned for phase 1, for the purposes of this application note we are using "Standard" proposal set which includes 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.

Configure IPsec VPN with IKE gateway and IPsec policy, then bind to ST interface.

```
set security ipsec vpn ike-vpn ike gateway ike-gate
set security ipsec vpn ike-vpn ike ipsec-policy vpn-policy1
set security ipsec vpn ike-vpn bind-interface st0.0
```

Binding an ST interface differentiates this VPN as a route-based VPN. For policy-based VPNs you do not configure an ST interface. If ST interface is not specified, then phase 2 cannot complete negotiations if this is a route-based VPN.

Configure security policy for Internet traffic.

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

This policy will permit all traffic from zone "trust" to zone "untrust". By specifying "source-nat interface" the device will translate the source IP and port for outgoing traffic using the IP address of the egress interface as the source IP and random higher port for the source port. If required more granular policies can be created to permit/deny certain.

Configure security policies for tunnel traffic in either directions.

```
edit security policies from-zone trust to-zone vpn
## Entering zone "trust" to zone "vpn" hierarchy
set policy vpn-tr-vpn match source-address local-net
set policy vpn-tr-vpn match destination-address remote-net
set policy vpn-tr-vpn match application any
set policy vpn-tr-vpn then permit
exit

edit security policies from-zone vpn to-zone trust
## Enter zone "vpn" to zone "trust" hierarchy
set policy vpn-vpn-tr match source-address remote-net
set policy vpn-vpn-tr match destination-address local-net
set policy vpn-vpn-tr match application any
set policy vpn-vpn-tr then permit
exit
```

A security policy permits traffic in one direction but also allows all reply traffic without the need for a reverse direction policy. However since in this example traffic may be initiated from either direction, bi-directional policies are 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 NAT can be enabled on the policies if required, but that is beyond the scope of this application note.

Configure tcp-mss to eliminate fragmentation of TCP traffic across tunnel.

```
set security flow tcp-mss ipsec-vpn mss 1350
```

Tcp-mss is negotiated as part of the TCP 3-way handshake. It limits the maximum size of a TCP segment to better fit the MTU limits on a network. This is especially important for VPN traffic as 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 causing fragmentation. Fragmentation increases bandwidth and device resources and is always best avoided. Note the value of 1350 is a recommended starting point for most ethernet-based networks with MTU of 1500 or greater. This value may need to be altered if any device in the path has lower MTU and/or if there is any added overhead such as PPP, frame relay, etc. As a general rule you may need to experiment with different tcp-mss values to obtain optimal performance.

SSG Configuration Example

The focus of this application note is on J Series and SRX Series configuration and troubleshooting. For the purpose of completing the diagram above, a sample of relevant configurations is provided from 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 we will not focus on the SSG configuration here. For reference the SSG C&E guides can be found here:

<http://www.juniper.net/techpubs/software/screensos/>.

Configuration example for SSG5.

```
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 VPN connection

Confirm IKE (phase 1) status.

The first step to confirm VPN status is to check the status of any IKE phase 1 security associations. Below is the CLI command.

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

We can see that the remote peer is **2.2.2.2**. The State shows **UP**. If the State shows **DOWN** or if there is 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 wrong Mode type (**Aggr** or **Main**), pre-shared keys or phase 1 proposals (all must match on both peers). Incorrect external interface is another common mis-configuration. This interface must be the correct interface that would receive the IKE packets. If configurations have been checked then check kmd log for any errors or run traceoptions (see troubleshooting section later in this application note). Note also Index number **1**. This value is unique for each IKE security association and allows you to get more details from that particular security association as below.

```
root@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
```

The detail command gives much more information which includes the Role (Initiator or Responder). This is useful to know because troubleshooting is usually always best done on the peer which has Responder role. Also shown are details regarding the authentication and encryption algorithms used, the phase 1 lifetime and traffic statistics. Traffic statistics can be used to verify that traffic is flowing properly in both directions. Finally note also the number of IPSec security associations created or in progress. This can help to determine the existence of any completed phase 2 negotiations.

Confirm IPSec (phase 2) status.

Once IKE phase 1 is confirmed then run the command below to view IPSec (phase 2) security associations.

```
root@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
```

From above we can see that there is one IPSec SA pair and that the Port used is 500 which means no nat-traversal (nat-traversal would show port 4500 or random high port). Also we can see the SPI used for both directions as well as the lifetime (in seconds) and usage limits or lifesize (in Kilobytes). So from above, we see '3363/ unlim' which means phase 2 lifetime is set to expire in 3363 seconds and there is no lifesize specified thus it shows unlimited. Phase 2 lifetime can differ from phase 1 lifetime since phase 2 is not dependent on phase 1 once the VPN is up. The 'Mon' column refers to VPN monitoring status. If VPN monitoring was enabled, then this would show U (up) or D (down). A hyphen (-) means VPN monitoring is not enabled for this SA. For more details regarding VPN monitoring, refer to the complete documentation for the J Series or SRX Series. Note that Vsys will always show 0.

Note also the ID number 16384 above. This is the Index value and is unique for each IPSec security association. You can view more details for a particular security association as below.

```
root@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-shal-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-shal-96, Encryption: 3des-cbc
Anti-replay service: enabled, Replay window size: 32
```

From above we can see Local Identity and Remote Identity. These elements comprise the proxy ID for this SA. Proxy ID mismatch is a very most common reason for phase 2 failing to complete. If no IPSec SA is listed then confirm the phase 2 proposals including the proxy ID settings are correct for both peers. Note that for route-based VPNs the default proxy ID is local=0.0.0.0/0, remote=0.0.0.0/0, service=any. This can cause issues if you have multiple route-based VPNs from the same peer IP in which case you will need to specify unique proxy IDs for each IPSec SA. Also for some third-party vendors you may need to manually enter the proxy ID to match. Another common reason for phase 2 failing to complete may be failure to specify ST interface binding. If IPSec cannot complete then check the kmd log or set traceoptions as detailed in the troubleshooting section of this application note.

Check statistics and errors for an IPSec SA.

The command below is used to check ESP and AH counters and for any errors with a particular IPSec security associations.

```
root@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
```

You normally do not want to see error values other than zero. However if you are experiencing packet loss issues across a VPN, then one approach is to run the above command multiple times and confirm that the Encrypted and Decrypted packet counters are incrementing. Also see if any of the error counters increment while you are experiencing the issue. It may also be necessary to enable security flow traceoptions (see troubleshooting section) to see which ESP packets are experiencing errors and why.

Test traffic flow across the VPN.

Once you have confirmed status of phase 1 and phase 2, then the next step is to test traffic flow across the VPN. One way to test traffic flow is through pings. We can ping from local host PC to remote host PC. We can also initiate pings from the J Series or SRX Series device itself. Below is an example of ping testing from the J Series or SRX Series device to the remote PC host.

```
root@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
```

Note that when initiating pings from the J Series or SRX Series device the source interface needs to be specified in order to be sure that route lookup will be correct and the appropriate zones can be referenced in policy lookup. In this case since ge-0/0/0.0 resides in the same security zone as the local host PC then ge-0/0/0 will need to be specified in pings so that the policy lookup can be from zone “trust” to zone “vpn”. Likewise we can initiate a ping from the remote host to the local host. Also we can initiate a ping from the SSG5 itself as below.

```
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
```

If pings fail from either direction then 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 IPsec statistics as mentioned above to see if 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 then likely the issue is not with the end host. For routing and policy issues we can enable security flow traceoptions which will be detailed below.

Troubleshooting Basics

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 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, using ping, confirm that the J Series or SRX Series device has connectivity to the Internet next-hop followed by confirming connectivity to the remote IKE peer. Assuming that has all been confirmed then confirm that IKE phase 1 can complete by running the verification commands as shown above. Once phase 1 is confirmed then confirm phase 2. Finally confirm traffic is flowing across the VPN. If the VPN is not in UP state then there is very little reason to test any transit traffic across the VPN. Likewise if phase 1 was not successful, then looking at phase 2 issues is pointless.

To troubleshoot issues further at the different levels, configure traceoptions. Traceoptions are enabled in configuration mode and are a part of a J Series or SRX Series operating configuration. This means that a configuration commit is necessary before a traceoption will take affect. Likewise, removing traceoptions require deleting or deactivating the configuration followed by a commit. By enabling a traceoption flag, 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 will be retained even after a system reboot. Keep in mind the available storage on flash before implementing traceoptions. You can check your available storage as below.

```
root@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
```

As shown above, `/dev/ad0s1a` represents the onboard flash memory and is currently at 65% capacity. You can also view available storage on the JWeb homepage under System Storage. The output of all traceoptions write to logs stored in directory `/var/log`. To view a list of all logs in `/var/log`, run operational mode command: `show log`.

Check traceoption logs.

As noted earlier, 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 view the trace output. Below are the commands to view the appropriate logs.

```
root@CORPORATE> show log kmd
root@CORPORATE> show log security-trace
root@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.

```
root@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
```

Troubleshooting IKE and IPSec Issues

To view success or failure messages in IKE or IPSec, view the kmd log with command: `show log kmd`. Although the kmd log will give a general reason for any failure, it may be necessary to obtain additional details. For this we can enable IKE traceoptions. Note as a general rule, it is always best to troubleshoot on the peer which has the role of Responder.

Enable IKE traceoptions for phase 1 and phase 2 negotiation issues.

Below is an example of all IKE traceoptions.

```
root@CORPORATE> configure
Entering configuration mode

[edit]
root@CORPORATE# edit security ike traceoptions

[edit security ike traceoptions]
root@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]
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 file name is specified then all IKE traceoptions write to kmd log. However you can specify a different filename if desired. To write trace data to the log you must specify at least one flag option. Option 'file size' determines the maximum size of a log file in bytes. For example 1m or 1000000 will generate a maximum file size of 1 MB. Option 'file files' determines the maximum number of log files that will be generated and stored in flash. Remember to commit the changes to start the trace. Below is an example of recommended traceoptions for troubleshooting most IKE related issues.

```
[edit]
root@CORPORATE# edit security ike traceoptions

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

Review kmd log for success/failure messages.

Below is an excerpt of successful phase 1 and phase 2 completion from '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)
```

So from above we can see that our local address is 1.1.1.2 and the remote peer is 2.2.2.2. UDP port 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 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 previously.

Below is an excerpt of phase 1 failing to complete.

```
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)
```

So from above we can see that our 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 due to no proposal chosen. This is likely mismatched phase 1 proposals. To resolve this issue, confirm that phase 1 proposals match on both peers.

Below is another excerpt of phase 1 failing to complete.

```
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 dbel0daf -
a4d6d829 f9ed3bba [-1] / 0x00000000 } IP; Error = No proposal chosen (14)
```

So from above again we can see that our 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 in this case we also see a message that the peer is not recognized. Peer not recognized could be incorrect peer address, mismatch peer ID type or incorrect peer ID depending on whether this is a dynamic or static VPN. This needs to 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 ID type IP address (as opposed to dynamic).

Below is another excerpt of phase 1 failing to complete.

```
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)
```

So from above we can see that the remote peer is 2.2.2.2. Invalid payload type usually means there was a problem with the decryption of the IKE packet due to mismatch pre-shared key. To resolve this issue confirm that pre-shared keys match on both peers.

Below is an excerpt of phase 1 success, but phase 2 failing to complete.

```
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)
```

So from above again we can see that our local address is 1.1.1.2 and the remote peer is 2.2.2.2. We can clearly see that phase 1 was successful based on the "Phase-1 [responder] done" message. The reason for failing is due to no proposal chosen. The issue is likely phase 2 proposal mismatch between the two peers. To resolve this issue, confirm that phase 2 proposals match on both peers.

Below is another excerpt of phase 1 success, but phase 2 failing to complete.

```
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_INTERNAL_ERROR: Phase2 finish: No sa_cfg found!

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)
```

From above we can see that phase 1 was successful. The reason for failing may seem to indicate that no proposal was chosen. However in this case we also see a message that the proxy ID did not match what was expected. We can see that we received phase 2 proxy ID of remote=192.168.168.0/24, local=10.10.20.0/24, service=any. So it is clear that this does not match the configurations on the local peer thus proxy ID match fails. This results in error: no proposal chosen. To resolve this configure either 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 is specifying a proxy ID other than all zeroes then you must manually configure the proxy ID within the IPsec profile of the peer.

Troubleshooting Flow Issues

Here is a problem scenario using the network diagram shown on page 3:

1. Remote PC 192.168.168.10 can ping local PC 10.10.10.10.
2. Local PC 10.10.10.10 cannot ping 192.168.168.10.
3. Based on show commands, IPsec SA is up and statistics show no errors.

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

Note: Enabling flow traceoptions can cause an increase in system CPU and memory utilization. Therefore enabling flow traceoptions is not recommended during peak traffic load times or if CPU utilization is very high. Enabling packet-filters is also highly recommended to lower resource utilizations and to facilitate pinpointing the packets of interest. Finally be sure to delete or deactivate all flow traceoptions and remove any unnecessary log files from flash after completing troubleshooting.

Enable security flow traceoptions for routing or policy issues.

See the below example of flow traceoptions.

```
[edit]
root@CORPORATE# edit security flow traceoptions

[edit security flow traceoptions]
root@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]
root@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
```

By default if no file name is specified then all flow traceoptions output writes to security-trace log. However you can specify a different filename if desired. To write trace data to the log you must specify at least one flag option. Option `file size` determines the maximum size of a log file in bytes. For example 1m or 1000000 will generate a maximum file size of 1 MB. Option `file files` determines the maximum number of log files that will be generated and stored in flash. Remember to commit the changes to start the trace.

In addition to the above, the J Series or SRX Series device has the ability to configure packet filters to limit the scope of the traffic to be captured. You can filter the output based on source/destination IP, source/destination port, interface and IP protocol. Up to 64 filters can be configured. Furthermore a packet-filter will also match the reverse direction to capture the

reply traffic assuming the source of the original packet matches the filter. See below example of flow packet-filter options.

```
[edit security flow traceoptions]
root@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
```

Terms listed within the same packet-filter act as a Boolean logical AND statement. That means 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, below is an example of recommended traceoptions for security flow for the above problem scenario.

```
[edit]
root@CORPORATE# edit security flow traceoptions

[edit security flow traceoptions]
root@CORPORATE# set file size 1m files 3
root@CORPORATE# set flag basic-datapath
root@CORPORATE# set packet-filter remote-to-local source-prefix 192.168.168.10/32
root@CORPORATE# set packet-filter remote-to-local destination-prefix 10.10.10.10/32
root@CORPORATE# set packet-filter local-to-remote source-prefix 10.10.10.0/32
root@CORPORATE# set packet-filter local-to-remote destination-prefix 192.168.168.0/32
root@CORPORATE# set packet-filter remote-esp protocol 50
root@CORPORATE# set packet-filter remote-esp source-prefix 2.2.2.2/32
root@CORPORATE# commit
```

The below output details the reasoning behind each flow traceoption setting.

```
[edit security flow traceoptions]
root@CORPORATE# show
file flow-trace-log size 1m files 3;
flag basic-datapath;
The log file "security-trace" has been set to 1 MB and up to 3 files will be
created. The reason for this is due to the nature of flow traceoptions a single
file could become full fairly quickly depending on how much traffic is captured.
Flag "basic-datapath" will show 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;
}
The above filter is for capturing the de-capsulated or unencrypted traffic from
remote PC to local PC. Since there are multiple terms this acts as a Boolean
logical AND. That means the source IP and destination IP must both match the
filter. If the source IP matched but the destination IP did not, then the packet
will not be captured. Since packet-filters are bi-directional, 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;
}
As mentioned above, no filter is required for capturing the reply traffic.
However a filter will only capture packets which were originally sourced from
the specified side. Thus the "local-to-remote" filter above is still required to
capture traffic which sources from local to remote side.

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

The above filter is optional and depends on whether or not the previous filter is able to capture any packets. This filter will capture all ESP (IP protocol 50) or encrypted packets from remote peer 2.2.2.2. Note, however, that this last filter will capture ALL encrypted traffic from 2.2.2.2 including packets that perhaps we are not interested in. If the unencrypted traffic is captured then this last filter may not be necessary.

So with the three problem statements mentioned in the problem scenario we can now begin to look at the flow traceoptions log to isolate the issue. We can assume that the third statement is correct based on IKE and IPsec troubleshooting. So the next step is to first validate the first problem statement to confirm that the remote PC can ping the local PC. Next we can troubleshoot the second problem statement to find out why the traffic fails in the reverse direction.

Validate first problem statement.

Begin by sourcing a ping from 192.168.168.10 to 10.10.10.10 and then view the security-trace log. Since no file name was specified, view all flow traceoptions output with command: `show log security-trace`. Below is the flow traceoptions output showing the successful traffic flow from remote PC to local PC. The first packet captured is the ESP or encrypted packet as below.

```
*****<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(0x3fffff), 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(0xffff), 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
```

Based on the top header, the packet is from 2.2.2.2 to 1.1.1.2; IP protocol 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 with command: `show security flow session session-identifier <session id>`. 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 with command: `show interface extensive`.

Below is the decrypted packet output.

```
*****<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(0x3fffff),
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_addr 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: chose 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
```

Based on the top header, the packet is from 192.168.168.10 to 10.10.10.10; IP protocol 1. The ingress interface is st0.0 which means the source was from across the VPN. The ingress zone is "vpn" zone 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.

There is not an existing session for this flow so first-packet processing occurs. Next we see the route lookup take place. Route lookup needs to take place 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. Since 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. The details for policy 6 can be viewed with the below command.

```
root@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
```

At this point the session is created, in this case session id 4. The reply packet should also be captured and will show existing session 4 is found as below.

```
*****<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:xlata_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
```

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.

Troubleshoot second problem statement.

Based on the second problem statement, the local PC cannot ping the remote PC. We can determine the problem by reviewing the security-trace log while attempting to ping from 10.10.10.10 to 192.168.168.10. Below is a sample output showing a failure.

```
*****<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
```

Based on the top header, the packet is from 10.10.10.10 to 192.168.168.10; IP protocol 1. No session is found so first packet processing takes place. Next, the route-lookup takes place. However instead of finding a route for 192.168.168.10 to st0.0 in "vpn" zone, this packet is instead routed to ge-0/0/0.0 in "untrust" zone. Since policy lookup is from zone "trust" to

zone “untrust”, the packet matches the “any-permit” policy and never reaches the “trust” to “vpn” policy. You can view the route by running command: `show route <destination IP>`.

```
root@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
```

It seems clear that a route does not exist for 192.168.168.0/24. Therefore to resolve this issue configure a route for 192.168.168.0/24 with next-hop as st0.0. Once the route is in place and committed, you may still see traffic failing as below.

```
*****<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
```

This time we can see that the route lookup is behaving as expected. The policy lookup is from zone “trust” to zone “vpn”. However the packet matched policy 2 which is the preconfigured default deny policy. View all configured policies with command : `show security policies`.

```
root@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
```

Based on above output we can see that there is no policy from zone “trust” to zone “vpn” to permit the traffic. Adding an appropriate policy can resolve this issue. At this point we would see that pings from local PC to remote PC succeed.

Question: Why did remote PC to local PC traffic succeed despite no route or policy configured for the reply traffic?

Answer: The order of packet processing is important to know to be able to answer the question. The J Series or SRX Series device will first inspect the packet to see if 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 then forwarded. If a session match is found, then no further route or policy lookup occurs.

Show Configuration

Below is the output of show configuration. For reference, highlighted are traceoption configurations for troubleshooting purposes. Always remember to delete or deactivate the traceoptions once troubleshooting is complete.

```
root@CORPORATE> show configuration | no-more
system {
  host-name CORPORATE;
  root-authentication {
    encrypted-password "$1$heGUvm8Y$t4wi40c0NR8dZ1DNz0No2."; ## 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;
        }
    }
}
}
```