Application Note

JUNOS OS LAN-TO-LAN VPN WITH OVERLAPPING SUBNETS

Configuration and Troubleshooting Guidelines for
IPsec VPN Configurations Between
J Series or SRX Series Devices

Version 2.0
June 2011

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

Contents

Introduction

Juniper Networks® Junos® operating system runs on Juniper Networks J Series Services Routers and Juniper Networks SRX Series Services Gateways, providing not only a powerful operating system but also a rich IP services tool kit. Junos OS has enhanced security and VPN capabilities via Juniper's firewall/IPsec VPN platforms, which include the Juniper Networks SSG Series Secure Services Gateways. The purpose of this application note is to discuss IPsec VPN configurations between J Series or SRX Series devices in a scenario where the subnets on both sides overlap. This application note also includes troubleshooting information.

Scope

This document applies to LAN-to-LAN overlapping subnets using J Series Services Routers and SRX Series Services Gateways.

This document is intended for network design and security engineers, as well as implementation partners who support customers requiring secure connectivity over public networks.

Design Considerations

• J Series Services Routers running Junos OS 9.5 and above

• SRX Series Services Gateways

Description and Deployment Scenario

The configuration of a services router running Junos OS for VPN support is quite flexible. You can create route-based and policy-based VPN tunnels. Furthermore, Network Address Translation (NAT) can be incorporated to provide solutions for certain problematic networking scenarios. This application note discusses one such problem scenario.

Problem Scenario

With corporate mergers, branch office consolidations, and partner collaborations being commonplace, often a company must create a VPN to another site with the same private addressing scheme as its IP network. Because both networks use the same internal IP addressing, it is not possible to simply build a tunnel between the two sites. However, if the tunnel endpoints on both sides are Juniper services routers, it is possible to configure a tunnel between these sites with an advanced configuration using NAT.

It is important to understand this basic routing dilemma. If a host is attached to a network, say 10.0.0.0/24, and the other device on the remote end is attached to a network using the same IP address subnet, it is not possible to build a tunnel and route the traffic to the other device without some sort of address translation. This is because all packets are routed based on the destination IP address. Before routing occurs, a determination must be made as to whether the destination IP is on the same (local) network or not. If the destination IP is on the same network, say 10.0.0.10, the destination device is found using Address Resolution Protocol (ARP). However, if the destination IP resides on a different network, the packet is sent to the next-hop router based on the device's routing table.

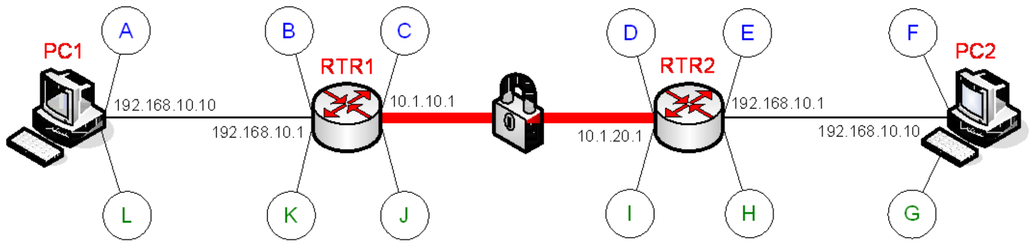
Because both the local and remote networks share the same IP addressing scheme, the packets will be handled locally and never route to the VPN tunnel. To work around this, we can perform static NAT on the source IP and destination IP of all traffic destined for the remote network at the other end of the tunnel. For this reason, a route-based approach to IPsec VPNs makes sense, because the creation of a "virtual" network interface on each services router by way of a "secure tunnel" or "st0" interface is required. It is important to note that when configuring the example in this application note, both source and destination addresses are translated as the packet traverses the VPN tunnel to the end host. Thus the services routers at each end of the tunnel must contact each other using a newly created IP network. This can introduce some administrative issues with certain applications, so keep this in mind when migrating two networks with overlapping subnets.

This application note discusses route-based VPN configuration using static NAT on the st0 interface for both peers. Additional Junos OS application notes can be found on Juniper Networks' Knowledge Base at http://kb.juniper.net. In particular, article KB10182 lists several application notes related to VPN configuration and troubleshooting.

Packet Flow Details with Static NAT

See Figure 1 below for a packet flow example.

Figure 1.



For this example, *PC1* and *RTR1* are at one site while *PC2* and *RTR2* are at a remote site with an IPSec VPN tunnel linking the two sites. Both *PC1* and *PC2* have IP address **192.168.10.10**, and all network masks are /24 (255.255.255.0). A TCP port 80 session is initiated from *PC1* destined for *PC2*. Below outlines the packet flow with static NAT configured on tunnel interfaces on both sites.

A. Packet leaves *PC1* destined for **10.1.20.10** to reach the remote side host *PC2*. Note that devices must attempt to reach devices at the remote end of the tunnel using the IP network owned by the remote device tunnel interface. Based on the default gateway configuration of *PC1*, the next hop would be 192.168.10.1 which is *RTR1*.

Packet Detail

src IP/port	192.168.10.10/1024	dst IP/port	10.1.20.10/80

B. The packet arrives at *RTR1* internal interface with no change to the source or destination IPs or ports. The packet is then routed internally to the tunnel interface on *RTR1*.

Packet Detail

src IP/port	192.168.10.10/1024	dst IP/port	10.1.20.10/80

C. When the packet reaches *RTR1* tunnel interface, a static NAT setting and associated policy is found. The static NAT has been defined for the entire 10.1.10.0 network range. Thus all outgoing traffic from **192.168.10.0** network destined for the tunnel interface will be source translated to a **10.1.10.0** equivalent address. The packet leaves encrypted out *RTR1* external interface, but inner packet now has source IP changed to **10.1.10.10** (port does not change with static NAT). Note that even though the source IP is translated to a **10.1.10.0** address, the security policy still needs to have the original source IP in the match objects.

Packet Detail

src IP/port	10.1.10.10/1024	dst IP/port	10.1.20.10/80

D. The encrypted packet is received by the *RTR2* external interface and decrypted. The inner packet shows source IP as **10.1.10.10** and destination IP as **10.1.20.10**. The packet is sent to the tunnel interface on *RTR2*.

Packet Detail

src IP/port	10.1.10.10/1024	dst IP/port	10.1.20.10/80

E. *RTR2* also has static NAT defined on the tunnel interface which covers the entire network range for 10.1.20.0 network. Thus all traffic destined for **10.1.20.0** network will be destination translated to an internal **192.168.10.0** equivalent address. The route lookup will determine that 192.168.10.0 network will be routed to *RTR2* internal interface. Packet leaves *RTR2* internal interface with destination IP changed to **192.168.10.10**.

Packet Detail

src IP/port	10.1.10.10/1024	dst IP/port	192.168.10.10/80

F. Since *PC2* has IP address **192.168.10.10**, the packet will be received by *PC2* with the same source and destination IP and ports as in step E.

As shown above, the original packet was both source and destination NAT translated by the time it reached *PC2*. Note that the NAT translations do not both occur on the same device. Rather one device performs the source address translation and the remote device performs the destination address translation. The reply from *PC2* back to *PC1* will follow similar steps only reversed as below.

G. Reply packet leaves *PC2* destined for **10.1.10.10** to reach host *PC1*. Based on the default gateway configuration of *PC2*, the next hop would be 192.168.10.1 which is *RTR2*.

Packet Detail

src IP/port	192.168.10.10/80	dst IP/port	10.1.10.10/1024

H. The packet arrives at *RTR2* internal interface with no change to the source or destination IPs or ports.

Packet Detail

src IP/port	192.168.10.10/80	dst IP/port	10.1.10.10/1024

I. Packet matches existing session in *RTR2* in which static NAT was defined. Thus the traffic from **192.168.10.0** network destined for the tunnel interface will be source translated to a **10.1.20.0** equivalent address. Packet leaves encrypted out *RTR2* external interface, but inner packet now has source IP changed to **10.1.20.10**.

Packet Detail

src IP/port	10.1.20.10/80	dst IP/port	10.1.10.10/1024

J. Packet arrives at *RTR1* external interface and decrypted. The inner packet shows source IP as **10.1.20.10** and destination IP as **10.1.10.10**.

Packet Detail

src IP/port	10.1.20.10/80	dst IP/port	10.1.10.10/1024

K. Packet matches existing session on *RTR1* in which static NAT was defined. Thus the traffic destined for **10.1.10.0** network will destination translate to an internal **192.168.10.0** equivalent address. Packet leaves *RTR1* internal interface with destination IP changed to **192.168.10.10**.

Packet Detail

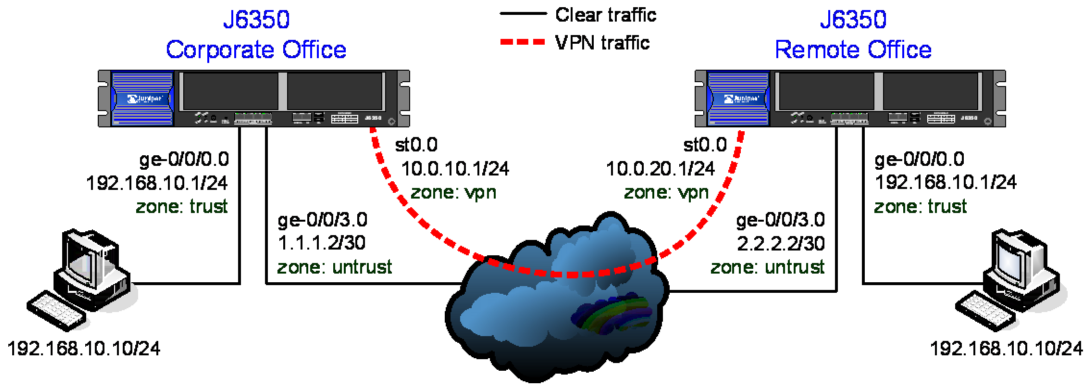
src IP/port	10.1.20.10/80	dst IP/port	192.168.10.10/1024

L. Packet arrives at *PC1* with the same source and destination IP and ports as in step K.

Network Diagram

Refer to Figure 2 below for Network Topology used for this configuration example.

Figure 2.



Configuration Steps

This example assumes the following (refer to figure 2 above):

- Internal LAN interface for both sites will be ge-0/0/0 in zone “trust” and will have private IP 192.168.10.1/24.
- Internet interface for both sites will be ge-0/0/3 in zone “untrust” and will each have a unique public IP.
- The secure tunnel interface st0.0 will be in zone “vpn” on both sites to allow for configuring unique policies specifically for tunnel (encrypted) traffic while maintaining unique policies for clear (non-encrypted) traffic.
- The address range to reach Remote side hosts from Corporate side is 10.1.20.0/24.
- The address range to reach Corporate side hosts from Remote side is 10.1.10.0/24
- All traffic between the Corporate and Remote LANs are to be permitted, and traffic may be initiated from either side.
- Basic non-VPN settings such as system settings, user login, and default security settings are already pre-configured on both devices.

Basic Steps to Configure

Note: Both Corporate and Remote sides have similar configuration steps. Thus the below information applies to both peers.

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.
4. 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.
5. Configure address book entries for each zone. This will be necessary for the security policies.
6. Configure phase 1 (IKE) proposal.
7. Configure IKE policy referencing the phase 1 proposal from step 6.
8. Configure IKE gateway profile referencing the IKE policy from step 7.
9. Configure phase 2 (IPSec) proposal.
10. Configure IPSec policy referencing the phase 2 proposal from step 9.
11. Configure VPN profile referencing IPSec policy from step 10 and IKE gateway from step 8. Then bind interface st0.0 to the VPN. This determines that the VPN is a route-based VPN.
12. Configure static NAT rule-set for st0 traffic.
13. Configure static NAT rules for each side.
14. Configure security policies to permit remote VPN traffic into the corporate LAN and vice versa.
15. Configure outgoing "trust" to "untrust" permit all policy with interface source NAT for Internet traffic.
16. Configure tcp-mss for IPSec traffic to eliminate the possibility of fragmented TCP traffic. This will lessen the resource utilization on the device.

Corporate Site Configuration Example

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

Configure IP addresses for private LAN, Internet and st0 interfaces

```
user@CORPORATE# set interfaces ge-0/0/0 unit 0 family inet address
192.168.10.1/24
user@CORPORATE# set interfaces ge-0/0/3 unit 0 family inet address 1.1.1.2/24
user@CORPORATE# set interfaces st0 unit 0 family inet address 10.0.10.1/24
```

JUNOS uses the concept of units for the logical component of an interface. In this example unit 0 and family inet (IPv4) are used. It is not mandatory for st0 interfaces on both sides to reside on the same IP subnet since the link is logically a point-to-point link.

Configure default route and route for tunnel traffic

```
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 10.1.20.0/24 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 st0 interface IP, or to simplify you can specify just the local st0 interface itself as the next-hop.

Configure security zones and assign interfaces to the zones

```
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
```

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 st0 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 interface in the zones

```
user@CORPORATE# set security zones security-zone trust interfaces ge-0/0/0 host-
inbound-traffic system-services all
user@CORPORATE# set security zones security-zone untrust interfaces ge-0/0/3
host-inbound-traffic system-services ike
user@CORPORATE# set security zones security-zone vpn interfaces st0.0 host-
inbound-traffic system-services all
```

Host-inbound services are for traffic destined for the JUNOS 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

```
user@CORPORATE# set security zones security-zone trust address-book address
local-net 192.168.10.0/24
user@CORPORATE# set security zones security-zone vpn address-book address remote-
net 10.1.20.0/24
```

For this example we are using address-book object names “local-net” and “remote-net” to define the Corporate to Remote LAN segments. However, in order to permit all traffic from the reverse direction, the address book object needs to reflect the static NAT address range for st0.0. The reason for this is because the traffic egressing out from Remote side to reach the Corporate side will have destination address as 10.1.10.0/24 and not 192.168.10.0/24.

Configure IKE phase 1 proposal

```
user@CORPORATE# set security ike proposal p1-prop1 authentication-method pre-
shared-keys
user@CORPORATE# set security ike proposal p1-prop1 dh-group group2
user@CORPORATE# set security ike proposal p1-prop1 authentication-algorithm sha1
user@CORPORATE# set security ike proposal p1-prop1 encryption-algorithm 3des-cbc
```

For the purposes of this application note we are using a proposal which includes presharedkeys, group2, 3des encryption and sha1 algorithm. Define your proposal in accordance with your corporate security policy.

Configure IKE policy

```
user@CORPORATE# set security ike policy ike-policy1 mode main
user@CORPORATE# set security ike policy ike-policy1 proposals p1-prop1
user@CORPORATE# set security ike policy ike-policy1 pre-shared-key ascii-text
"secretkey"
```

The IKE policy will specify main mode which is most commonly used for site-to-site VPNs in which both peers have static IP addresses. Aggressive mode is typically used when one peer is a dynamic peer. For both peers, the mode must match. In the IKE policy, the phase 1 proposal is defined as well as the preshared-key.

Configure IKE gateway (phase 1) with peer IP address peer ID type

```
user@CORPORATE# set security ike gateway remote-ike ike-policy ike-policy1
user@CORPORATE# set security ike gateway remote-ike address 2.2.2.2
user@CORPORATE# set security ike gateway remote-ike 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 phase 2 proposal

```
user@CORPORATE# set security ipsec proposal p2-prop1 protocol esp
user@CORPORATE# set security ipsec proposal p2-prop1 authentication-algorithm
hmac-sha1-96
user@CORPORATE# set security ipsec proposal p2-prop1 encryption-algorithm 3des-
cbc
user@CORPORATE# set security ipsec proposal p2-prop1 lifetime-seconds 3600
```

For the purposes of this application note we are using a proposal which includes ESP, 3des encryption, sha1 algorithm and lifetime of 3600 seconds (1 hour). Define your proposal in accordance with your corporate security policy.

Configure IPSec policy

```
user@CORPORATE# set security ipsec policy vpn-policy1 perfect-forward-secrecy
keys group2
user@CORPORATE# set security ipsec policy vpn-policy1 proposals p2-prop1
```

The VPN policy must specify a phase 2 proposal. Perfect-forward-secrecy is optional, though recommended for increased security.

Configure IPSec VPN with IKE gateway and IPSec policy, then bind to st0 interface

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

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

Binding an st0 interface differentiates this VPN as a route-based VPN. For policy-based VPNs you do not configure an st0 interface. If st0 interface is not specified, then phase 2 cannot complete negotiations if this is a route-based VPN.

Configure static NAT rule-set for st0 interface

```
user@CORPORATE# set security nat static rule-set snat1 from interface st0.0
```

For this example we are mapping the entire 10.1.10.0/24 subnet with the 192.168.10.0/24 subnet. This is a one-to-one mapping for each host on the subnet to the other. For example, 10.1.10.101 would map to 192.168.10.101, 102 to 102, etc. For this reason we cannot use the same IP subnet as the st0 interface itself. The reason for this is because the IP range would overlap the interface IP itself and commit check would fail. No port translations are performed in this instance.

Configure static NAT rule

```
user@CORPORATE# edit security nat static rule-set snat1
user@CORPORATE# set rule remote1 match destination-address 10.1.10.0/24
user@CORPORATE# set rule remote1 then static-nat prefix 192.168.10.0/24
user@CORPORATE# exit
```

Static NAT policies must have destination zone as predefined zone “Junos-global”. Configure security policy from zone “vpn” to zone “Junos-global” since the static NAT will be between these two zones. This will allow the static NAT to be bi-directional. Note that the destination address must be in the format “*static_nat_x.x.x.x_y*” where *x.x.x.x* is the IP prefix defined in static NAT and *y* is the subnet mask slash notation value. This destination object MUST match exactly in this format or commit check will fail.

Configure security policies for tunnel traffic in both directions

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

user@CORPORATE# edit security policies from-zone vpn to-zone trust
user@CORPORATE# set policy remote-vpn-incoming match source-address remote-net
user@CORPORATE# set policy remote-vpn-incoming match destination-address local-
net
user@CORPORATE# set policy remote-vpn-incoming match application any
user@CORPORATE# set policy remote-vpn-incoming then permit
user@CORPORATE# 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 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. Although the static NAT policy is bi-directional in nature, security policies are still required to actually permit the traffic in both directions.

Configure source-NAT and security policy for Internet traffic

```
user@CORPORATE# 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
user@CORPORATE# exit

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

This policy will permit all traffic from zone "trust" to zone "untrust". Additionally, the NAT rule with "source-nat interface" will translate the source IP and port for outgoing Internet 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 traffic.

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

```
user@CORPORATE# 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.

Remote Site Configuration Example

Much of the details from the Corporate Site configuration applies here as well. Therefore the same information will not be repeated unless some specific details for the Remote Site need to be pointed out. To begin, enter configuration mode with either command: `configure` or `edit`.

Configure IP addresses for private LAN, Internet and st0 interfaces

```
user@REMOTE# set interfaces ge-0/0/0 unit 0 family inet address 192.168.10.1/24
user@REMOTE# set interfaces ge-0/0/3 unit 0 family inet address 2.2.2.2/24
user@REMOTE# set interfaces st0 unit 0 family inet address 10.0.20.1/24
```

See Corporate Site Configuration Example for details.

Configure default route and route for tunnel traffic

```
user@REMOTE# set routing-options static route 0.0.0.0/0 next-hop 2.2.2.1
user@REMOTE# set routing-options static route 10.1.10.0/24 next-hop st0.0
```

See Corporate Site Configuration Example for details.

Configure security zones and assign interfaces to the zones

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

See Corporate Site Configuration Example for details.

Configure host-inbound services for each interface in the zones

```
user@REMOTE# set security zones security-zone trust interfaces ge-0/0/0 host-
inbound-traffic system-services all
user@REMOTE# set security zones security-zone untrust interfaces ge-0/0/3 host-
inbound-traffic system-services ike
user@REMOTE# set security zones security-zone vpn interfaces st0.0 host-inbound-
traffic system-services all
```

See Corporate Site Configuration Example for details.

Configure address book entries for each zone

```
user@REMOTE# set security zones security-zone trust address-book address local-
net 192.168.10.0/24
user@REMOTE# set security zones security-zone vpn address-book address corp-net
10.1.10.0/24
```

See Corporate Site Configuration Example for details. However, for the Remote site, the static NAT network would 10.1.20.0/24.

Configure IKE phase 1 proposal

```
user@REMOTE# set security ike proposal p1-prop1 authentication-method pre-shared-
keys
user@REMOTE# set security ike proposal p1-prop1 dh-group group2
user@REMOTE# set security ike proposal p1-prop1 authentication-algorithm sha1
user@REMOTE# set security ike proposal p1-prop1 encryption-algorithm 3des-cbc
```

The proposal must match the peer side exactly or phase 1 would fail.

Configure IKE policy

```
user@REMOTE# set security ike policy ike-policy1 mode main
user@REMOTE# set security ike policy ike-policy1 proposals p1-prop1
user@REMOTE# set security ike policy ike-policy1 pre-shared-key ascii-text
"secretkey"
```

Be sure that the preshared-key matches exactly with the peer side.

Configure IKE gateway (phase 1) with peer IP address peer ID type

```
user@REMOTE# set security ike gateway corp-ike ike-policy ike-policy1
user@REMOTE# set security ike gateway corp-ike address 1.1.1.2
user@REMOTE# set security ike gateway corp-ike external-interface ge-0/0/3.0
```

Since both peers have static IP addresses, you can use the IP address as the peer ID type. If one peer had a dynamic IP address then the IKE peer would need to be identified by either domain name (FQDN), email address (u-FQDN) or distinguished name (ASN1-DN).

Configure IPSec phase 2 proposal

```
user@REMOTE# set security ipsec proposal p2-prop1 protocol esp
user@REMOTE# set security ipsec proposal p2-prop1 authentication-algorithm hmac-
sha1-96
user@REMOTE# set security ipsec proposal p2-prop1 encryption-algorithm 3des-cbc
user@REMOTE# set security ipsec proposal p2-prop1 lifetime-seconds 3600
```

The proposal must match the peer side exactly or phase 2 would fail.

Configure IPSec policy

```
user@REMOTE# set security ipsec policy vpn-policy1 perfect-forward-secrecy keys
group2
user@REMOTE# set security ipsec policy vpn-policy1 proposals p2-prop1
```

If the remote peer is configured for perfect-forward-secrecy, then this would also need to be configured here.

Configure IPSec VPN with IKE gateway and IPSec policy, then bind to st0 interface

```
user@REMOTE# set security ipsec vpn corp-vpn ike gateway corp-ike
user@REMOTE# set security ipsec vpn corp-vpn ike ipsec-policy vpn-policy1
user@REMOTE# set security ipsec vpn corp-vpn bind-interface st0.0
user@REMOTE# set security ipsec vpn corp-vpn establish-tunnels immediately
```

See Corporate Site Configuration Example for details.

Configure static NAT rule-set for st0 interface

```
user@REMOTE# set security nat static rule-set snat1 from interface st0.0
```

Like the Corporate site, we are mapping the entire 10.1.20.0/24 subnet with the Remote LAN 192.168.10.0/24 subnet. This is again a one-to-one mapping for each host on the subnet to the other. Again, no port translations are performed in this instance.

Configure static NAT rule

```
user@REMOTE# edit security nat static rule-set snat1
user@REMOTE# set rule remote1 match destination-address 10.1.10.0/24
user@REMOTE# set rule remote1 then static-nat prefix 192.168.10.0/24
user@REMOTE# exit
```

See Corporate Site Configuration Example for details.

Configure security policies for tunnel traffic in both directions

```
user@REMOTE# edit security policies from-zone trust to-zone vpn
user@REMOTE# set policy corp-vpn-outgoing match source-address local-net
user@REMOTE# set policy corp-vpn-outgoing match destination-address corp-net
user@REMOTE# set policy corp-vpn-outgoing match application any
user@REMOTE# set policy corp-vpn-outgoing then permit
user@REMOTE# exit

user@REMOTE# edit security policies from-zone vpn to-zone trust
user@REMOTE# set policy corp-vpn-incoming match source-address corp-net
user@REMOTE# set policy corp-vpn-incoming match destination-address local-net
user@REMOTE# set policy corp-vpn-incoming match application any
user@REMOTE# set policy corp-vpn-incoming then permit
user@REMOTE# exit
```

See Corporate Site Configuration Example for details.

Configure interface source-nat. Configure a security policy for Internet traffic.

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

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

See Corporate Site Configuration Example for details.

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

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

See Corporate Site Configuration Example for details.

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 from Corporate Site services router.

```
root@CORPORATE> show security ike security-associations
Index   Remote Address  State  Initiator cookie  Responder cookie  Mode
```

```
  3       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 (Aggressive 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 3. 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 3 detail
IKE peer 2.2.2.2, Index 3,
  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 active tunnels: 1
  ID    Gateway          Port  Algorithm       SPI      Life:sec/kb  Mon vsys
  <131073 2.2.2.2        500   ESP:3des/sha1   33377b8c 1997/ unlim   -   0
  >131073 2.2.2.2        500   ESP:3des/sha1   4f99c1df 1997/ 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 '1997/ unlim' which means phase 2 lifetime is set to expire in 1997 seconds and there is no lifesize specified thus it shows unlimited. Phase 2 life time can differ from phase 1 life time 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 JUNOS. Note that Vsys will always show 0.

Note also the ID number 131073 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 131073 detail
  Virtual-system: Root
  Local Gateway: 1.1.1.2, Remote Gateway: 2.2.2.2
  Local Identity: ipv4_subnet(any:0,[0..7]=0.0.0.0/0)
  Remote Identity: ipv4_subnet(any:0,[0..7]=0.0.0.0/0)
    DF-bit: clear
    Direction: inbound, SPI: 33377b8c, AUX-SPI: 0
    Hard lifetime: Expires in 1909 seconds
    Lifesize Remaining:  Unlimited
    Soft lifetime: Expires in 1332 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: 64

    Direction: outbound, SPI: 4f99c1df, AUX-SPI: 0
    Hard lifetime: Expires in 1909 seconds
    Lifesize Remaining:  Unlimited
    Soft lifetime: Expires in 1332 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: 64
```

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 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 131073
ESP Statistics:
  Encrypted bytes:            18328
  Decrypted bytes:             5604
  Encrypted packets:            157
  Decrypted packets:             90
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 JUNOS device itself. Below is an example of ping testing from the Corporate side JUNOS device to the Remote side PC host.

```
root@CORPORATE> ping 10.1.20.10 interface ge-0/0/0.0 count 5
PING 10.1.20.10 (10.1.20.10): 56 data bytes
64 bytes from 192.168.10.1: icmp_seq=0 ttl=65 time=3.552 ms
64 bytes from 192.168.10.1: icmp_seq=1 ttl=65 time=2.683 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=65 time=2.666 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=65 time=2.667 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=65 time=3.109 ms

--- 10.1.20.10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.666/2.935/3.552/0.352 ms
```

Recall that to reach the Remote side network, the destination address must be the remote peer static NAT address. Note also that when initiating pings from the JUNOS device the source interface or source address 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 ge-0/0/0.0 resides in the same security zone as the Corporate host PC. Therefore interface ge-0/0/0.0 or the IP address of the interface will need to be specified in pings so that the policy lookup can be from zone “trust” to zone “vpn”.

Likewise, to confirm bi-directional operation, we can initiate a ping from the Remote side SRX device to the Corporate network host PC.

```
root@REMOTE> ping 10.1.10.10 interface ge-0/0/0.0 count 5
PING 10.1.10.10 (10.1.10.10): 56 data bytes
64 bytes from 192.168.10.1: icmp_seq=0 ttl=65 time=2.194 ms
64 bytes from 192.168.10.1: icmp_seq=1 ttl=65 time=2.619 ms
64 bytes from 192.168.10.1: icmp_seq=2 ttl=65 time=4.138 ms
64 bytes from 192.168.10.1: icmp_seq=3 ttl=65 time=2.316 ms
64 bytes from 192.168.10.1: icmp_seq=4 ttl=65 time=2.597 ms

--- 10.1.10.1 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.194/2.773/4.138/0.702 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 any issues are not with the end host. For routing and policy issues, 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 Junos 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 testing transit traffic across the VPN makes little sense. 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 JUNOS 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 persistent 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 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 J-Web homepage under System Storage. The output of all traceoptions writes to logs stored in directory /var/log. To view a list of all logs in /var/log, run operational mode command: **`show log`**.

Checking 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
```

NOTE: For the Juniper Networks SRX3000 line, SRX5000 line, and SRX1400 devices, the kmd logs are located in the /var/tmp directory and the SPU ID value is included in the log filename (i.e. /var/tmp/kmd14).

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://192.168.10.10/kmd.log
ftp://192.168.10.10/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.

Enabling IKE Traceoptions

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. If the filename is specified then all IKE and IPSec related logs would no longer write to kmd log. 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.

```
root@CORPORATE# 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# exit
root@CORPORATE# commit and-quit
```

Reviewing IKE Traceoption Output Log

Below are some excerpts of successful phase 1 and phase 2 completion and some failure instances from '**`show log kmd`**'.

Phase 1 and phase 2 successful.

```
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]=0.0.0.0/0)
p2_remote=ipv4_subnet(any:0,[0..7]=0.0.0.0/0)
```

From above we can see that our local address is 1.1.1.2 and the remote peer is 2.2.2.2. The output "udp:500" indicates that nat-traversal was not 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. Note that by default the proxy ID for a route-based VPN will show as all zeros. At this point you can confirm that the IPSec SA is up using the verification commands mentioned previously.

Phase 1 failing to complete, example 1.

```
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)
```

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.

Phase 1 failing to complete, example 2.

```
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)
```

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 may seem to indicate No proposal was chosen. However in this case we also see a message peer:2.2.2.2 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 IKE id type as IP address.

Phase 1 failing to complete, example 3.

```
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)
```

From above we can see that the remote peer is 2.2.2.2. Invalid next 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.

Phase 1 successful, phase 2 failing to complete, example 1.

```
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)
```

From above 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 during phase 2 negotiations. 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.

Phase 1 successful, phase 2 failing to complete, example 2.

```
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)
```

From above we can see that phase 1 was successful. The reason for failing may seem to indicate No proposal was chosen. However in this case we also see the message Failed to match the peer proxy ids which means 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). 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

Assuming that IKE phase 1 and 2 are successful, the next step is to confirm that traffic flow is working properly. If the Security Association is up yet traffic does not seem to traverse the tunnel, then the next step would be to troubleshoot at the flow level.

One option to confirm traffic is matching a particular policy is to enable logging on the policy itself and then check the messages log with command: `show log messages`. However, there may be instances where the traffic for some reason is not matching the expected policy. In those instances you would enable flow traceoptions to help determine how the Services Router is handling the traffic.

This section describes enabling of flow traceoptions and interpreting the output. An example of successful output will illustrate the various messages to look for in the log. In addition, some output of failed traffic flow will also be illustrated to highlight some common flow problems.

Enabling Security Flow Traceoptions

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, JUNOS has the ability to 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, 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.

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. Enable packet-filters 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.

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 the use of traceoptions for security flow issues from the Corporate site router.

```
root@CORPORATE# edit security flow traceoptions
root@CORPORATE# set file size 1m files 3
root@CORPORATE# set flag basic-datapath
root@CORPORATE# set packet-filter local-host source-prefix 192.168.10.10/32
root@CORPORATE# set packet-filter local-host destination-prefix 10.1.20.0/24
root@CORPORATE# set packet-filter vpn-incoming interface st0.0
root@CORPORATE# exit
root@CORPORATE# commit and-quit
```

The below output details the reasoning behind each of the above 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 local-host {
    source-prefix 192.168.10.10/32;
    destination-prefix 10.1.20.0/24;
}
The above filter is for capturing the traffic from the local PC host to any remote
PC destination. 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 vpn-incoming {
    interface st0.0 ;
}
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 "vpn-incoming" filter may be desired to capture traffic
which sources from remote side of the VPN to the local side. By specifying the
st0.0 interface, this will capture ALL traffic incoming from the VPN.
```

Reviewing Flow Traceoption Output Log

As mentioned previously, if no file name was specified in flow traceoptions, then the log would write to security-trace log. View all flow traceoptions output with command: **`show log security-trace`**. To illustrate the trace output from a successful traffic flow, a ping echo from host 192.168.10.10 from the Corporate side was transmitted with destination 10.1.20.10 which is a Remote side host.

First packet processing, creating a new session.

Below is the first packet captured.

```
May 19 13:53:32 13:53:31.1561520:CID-0:RT:<192.168.10.10/0->10.1.20.10/27656;1> matched
filter local-host:
```

```
May 19 13:53:32 13:53:31.1561530:CID-0:RT:packet [84] ipid = 25922, @4b260c8e

May 19 13:53:32 13:53:31.1561534:CID-0:RT:---- flow_process_pkt: (thd 0): flow_ctxt type 0,
common flag 0x0, mbuf 0x4b260a80, rtbl_idx = 7872

May 19 13:53:32 13:53:31.1561539:CID-0:RT: flow process pak fast ifl 68 in_ifp ge-0/0/0.0

May 19 13:53:32 13:53:31.1561545:CID-0:RT:  ge-0/0/0.0:192.168.10.10->10.1.20.10, icmp, (8/0)

May 19 13:53:32 13:53:31.1561551:CID-0:RT: find flow: table 0x581e8088, hash 15912(0xffff),
sa 192.168.10.10, da 10.1.20.10, sp 0, dp 27656, proto 1, tok 384

May 19 13:53:32 13:53:31.1561559:CID-0:RT:  no session found, start first path. in_tunnel -
0, from_cp_flag - 0

May 19 13:53:32 13:53:31.1561567:CID-0:RT:  flow_first_create_session

May 19 13:53:32 13:53:31.1561575:CID-0:RT:  flow_first_in_dst_nat: in <ge-0/0/0.0>, out <N/A>
dst_adr 10.1.20.10, sp 0, dp 27656

May 19 13:53:32 13:53:31.1561581:CID-0:RT:  chose interface ge-0/0/0.0 as incoming nat if.

May 19 13:53:32 13:53:31.1561588:CID-0:RT:flow_first_rule_dst_xlate: DST no-xlate: 0.0.0.0(0)
to 10.1.20.10(27656)

May 19 13:53:32 13:53:31.1561594:CID-0:RT:flow_first_routing: call flow_route_lookup():
src_ip 192.168.10.10, x_dst_ip 10.1.20.10, in ifp ge-0/0/0.0, out ifp N/A sp 0, dp 27656,
ip_proto 1, tos 0

May 19 13:53:32 13:53:31.1561600:CID-0:RT:Doing DESTINATION addr route-lookup

May 19 13:53:32 13:53:31.1561611:CID-0:RT:  routed (x_dst_ip 10.1.20.10) from trust (ge-
0/0/0.0 in 0) to st0.0, Next-hop: 10.1.20.10

May 19 13:53:32 13:53:31.1561618:CID-0:RT:  policy search from zone trust-> zone vpn
(0x0,0x6c08,0x6c08)

May 19 13:53:32 13:53:31.1561635:CID-0:RT:  app 0, timeout 60s, curr ageout 60s

May 19 13:53:32 13:53:31.1561642:CID-0:RT:  found reversed mip 10.1.10.10 for 192.168.10.10
(on st0.0)

May 19 13:53:32 13:53:31.1561648:CID-0:RT:flow_first_src_xlate: 192.168.10.10/0 ->
10.1.20.10/27656 | 10.1.20.10/27656 -> 10.1.10.10/0: nat_src_xlated: True,
nat_src_xlate_failed: False

May 19 13:53:32 13:53:31.1561657:CID-0:RT:flow_first_src_xlate: hip xlate: 192.168.10.10-
>10.1.10.10 at st0.0 (vs. st0.0)

May 19 13:53:32 13:53:31.1561662:CID-0:RT:  dip id = 0/0, 192.168.10.10/0->10.1.10.10/0

May 19 13:53:32 13:53:31.1561667:CID-0:RT:flow_first_get_out_ifp: 1000 -> cone nat test

May 19 13:53:32 13:53:31.1561670:CID-0:RT: tunnel if vpn st0.0

May 19 13:53:32 13:53:31.1561673:CID-0:RT:tunnel out 0x4fdcaf68

May 19 13:53:32 13:53:31.1561676:CID-0:RT:  choose interface ge-0/0/3.0 as outgoing phy if

May 19 13:53:32 13:53:31.1561681:CID-0:RT:is_loop_pak: No loop: on ifp: ge-0/0/3.0, addr:
10.1.20.10, rtt_idx:0

May 19 13:53:32 13:53:31.1561687:CID-0:RT:policy is NULL (wx/pim scenario)

May 19 13:53:32 13:53:31.1561694:CID-0:RT:sm_flow_interest_check: app_id 0, policy 4,
app_svc_en 0, flags 0x2. not interested

May 19 13:53:32 13:53:31.1561700:CID-0:RT:sm_flow_interest_check: app_id 1, policy 4,
app_svc_en 0, flags 0x2. not interested

May 19 13:53:32 13:53:31.1561704:CID-0:RT:flow_first_service_lookup(): natp(0x4fdd8bd8):
app_id, 0(0).

May 19 13:53:32 13:53:31.1561708:CID-0:RT:  service lookup identified service 0.

May 19 13:53:32 13:53:31.1561712:CID-0:RT:  flow_first_final_check: in <ge-0/0/0.0>, out <ge-
0/0/3.0>

May 19 13:53:32 13:53:31.1561716:CID-0:RT:flow_first_final_check: flow_set_xlate_vector.

May 19 13:53:32 13:53:31.1561720:CID-0:RT:  existing vector list 1204-605130a8.

May 19 13:53:32 13:53:31.1561724:CID-0:RT:  existing vector list 4-604fcba8.
May 19 13:53:32 13:53:31.1561728:CID-0:RT:  Session (id:4119) created for first pak 1204

May 19 13:53:32 13:53:31.1561732:CID-0:RT:  flow_first_install_session======> 0x4fdd8bd8

May 19 13:53:32 13:53:31.1561735:CID-0:RT: nsp 0x4fdd8bd8, nsp2 0x4fdd8c48

May 19 13:53:32 13:53:31.1561740:CID-0:RT:  make_nsp_ready_no_resolve()

May 19 13:53:32 13:53:31.1561745:CID-0:RT:  route lookup: dest-ip 192.168.10.10 orig ifp ge-
0/0/0.0 output_ifp ge-0/0/0.0 orig-zone 6 out-zone 6 vsd 0
May 19 13:53:32 13:53:31.1561751:CID-0:RT:  route to 192.168.10.10

May 19 13:53:32 13:53:31.1561764:CID-0:RT:Installing c2s NP session wing

May 19 13:53:32 13:53:31.1561770:CID-0:RT:  flow got session.

May 19 13:53:32 13:53:31.1561772:CID-0:RT:  flow session id 4119

May 19 13:53:32 13:53:31.1561778:CID-0:RT:  post addr xlation: 10.1.10.10->10.1.20.10.

May 19 13:53:32 13:53:31.1561784:CID-0:RT:  encap vector

May 19 13:53:32 13:53:31.1561787:CID-0:RT:  going into tunnel 131073 (nsp_tunnel=0x4fdcaf68).

May 19 13:53:32 13:53:31.1561791:CID-0:RT:  flow_encrypt: 0x4fdcaf68

May 19 13:53:32 13:53:31.1561796:CID-0:RT:mbuf 0x4b260a80, exit nh 0xa0010

May 19 13:53:32 13:53:31.1561799:CID-0:RT: ----- flow_process_pkt rc 0x0 (fp rc 0)
```

Based on the top header, the packet is from 192.168.10.10 to 10.1.20.10; IP protocol 1. The ingress interface is ge-0/0/0.0 0 in zone "trust" and matching packet-filter "local-host". 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 st0.0 interface. 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 "trust" to-zone "vpn". Policy 4 was found which permits the traffic. To determine which policy has index 4, use command: **`show security policies`**.

```
root@CORPORATE> show security policies
Default policy: deny-all
From zone: trust, To zone: vpn
  Policy: remote-vpn-outgoing, State: enabled, Index: 4, Sequence number: 1
    Source addresses: local-net
    Destination addresses: remote-net
    Applications: any
    Action: permit
From zone: trust, To zone: untrust
  Policy: any-permit, State: enabled, Index: 6, Sequence number: 1
    Source addresses: any
    Destination addresses: any
    Applications: any
    Action: permit
From zone: vpn, To zone: trust
  Policy: remote-vpn-incoming, State: enabled, Index: 5, Sequence number: 1
    Source addresses: remote-net
    Destination addresses: local-net
    Applications: any
    Action: permit
```

From above we can see that the policy name for index 4 is remote-vpn-outgoing. You can then view details for this policy with command: **`show security policies policy-name <name> detail`**.

```
root@CORPORATE> show security policies policy-name remote-vpn-outgoing detail
Policy: remote-vpn-outgoing, action-type: permit, State: enabled, Index: 4
  Sequence number: 1
  From zone: trust, To zone: vpn
  Source addresses:
    local-net: 192.168.10.0/24
  Destination addresses:
    remote-net: 10.1.20.0/24
  Application: any
    IP protocol: 0, ALG: 0, Inactivity timeout: 0
      Source port range: [0-0]
      Destination port range: [0-0]
```

In addition to finding policy 4, the next line of the trace output shows that a reversed mip 10.1.10.10 for 192.168.10.10 on st0.0 was found. A reverse mip refers to the static NAT configured and means that the source address will be translated for host 192.168.10.10 to 10.1.10.10. This confirms the static NAT functionality.

Now that the route lookup has completed, a policy was found to permit the traffic and finally static NAT was applied, the next step is to create a new session for this traffic to allow the reply packet through. Session id 4119 is created for the Corporate to Remote direction. However, note the reverse direction for session id 4119 shows 10.1.20.10 to 10.1.10.10 which reflects the static NAT address. This is also reflected in the **post address xlation: 10.1.10.10->10.1.20.10** output.

The final element of the flow trace output shows **going into tunnel 131073**. The SA index should correlate to the SA index shown in “show security ipsec security-association” output. This confirms that the flow has processed the first packet correctly and the packet has successfully passed the flow module. Assuming that traffic flow is working, you should expect to see a reply match the existing session 4119.

Reply packet processing, matching the existing session.

The reply is considerable simpler since a session already exists.

```
May 19 13:53:32 13:53:31.1563882:CID-0:RT:<10.1.20.10/27656->10.1.10.10/0;1> matched filter
vpn-incoming:

May 19 13:53:32 13:53:31.1563887:CID-0:RT:packet [84] ipid = 25922, @4b88fcb2

May 19 13:53:32 13:53:31.1563891:CID-0:RT:---- flow_process_pkt: (thd 0): flow_ctxt type 1,
common flag 0x0, mbuf 0x4b88fa80, rtbl_idx = 7872

May 19 13:53:32 13:53:31.1563897:CID-0:RT: in_ifp <vpn:st0.0>

May 19 13:53:32 13:53:31.1563900:CID-0:RT:flow_process_pkt_exception: setting rtt in lpak to
5cab09d0

May 19 13:53:32 13:53:31.1563907:CID-0:RT:pkt out of tunnel.Proceed normally

May 19 13:53:32 13:53:31.1563910:CID-0:RT:  st0.0:10.1.20.10->10.1.10.10, icmp, (0/0)

May 19 13:53:32 13:53:31.1563914:CID-0:RT: find flow: table 0x581e8088, hash 48327(0xffff),
sa 10.1.20.10, da 10.1.10.10, sp 27656, dp 0, proto 1, tok 576

May 19 13:53:32 13:53:31.1563921:CID-0:RT:  flow got session.

May 19 13:53:32 13:53:31.1563924:CID-0:RT:  flow session id 4119
May 19 13:53:32 13:53:31.1563929:CID-0:RT:  post addr xlation: 10.1.20.10->192.168.10.10.

May 19 13:53:32 13:53:31.1563934:CID-0:RT:  encap vector

May 19 13:53:32 13:53:31.1563937:CID-0:RT:  no more encapping needed

May 19 13:53:32 13:53:31.1563940:CID-0:RT:mbuf 0x4b88fa80, exit nh 0xb0010

May 19 13:53:32 13:53:31.1563944:CID-0:RT:flow_process_pkt_exception: Freeing lpak 4939e3f4
associated with mbuf 0x4b88fa80

May 19 13:53:32 13:53:31.1563948:CID-0:RT: ----- flow_process_pkt rc 0x0 (fp rc 0)
```

Based on the top header, the packet is from 10.1.20.10 to 10.1.10.10; IP protocol 1. The ingress interface is st0.0 which means the source was from across the VPN. Therefore the ingress zone is "vpn" zone. This is an ICMP packet. In particular "icmp, (0/0)" indicates that this is an ICMP type 0, code 0, which is an echo reply. Note that existing **session id 4119** was found, thus no first-packet processing is required. Also due to the session details, the **post addr xlation: 10.1.20.10->192.168.10.10** indicates that the destination IP is translated from 10.1.10.10 back to 192.168.10.10 and then forwarded to the host PC. This completes the successful session flow.

Common Reasons for Traffic Flow Failing

Some common failures which can be resolved with examination of flow traceoptions are listed below.

Route lookup shows incorrect next-hop.

During first-packet processing, the route-lookup is performed first to determine the ingress and egress interfaces and therefore the ingress and egress security zones for the policy lookup. In our example, we expect that the next-hop would be interface st0.0. The flow traceoptions output in the successful example above reflects this. However if the route lookup determines a different next-hop, then the policy lookup outcome will change as well. Below is an example output of an incorrect route lookup result.

```
Dec  7 20:36:01 20:36:00.1196827:CID-0:RT:flow_first_routing: call flow_route_lookup():
src_ip 192.168.10.10, x_dst_ip 10.1.20.10, ifp ge-0/0/0.0, sp 44700, dp 1024, ip_proto 1, tos
0

Dec  7 20:36:01 20:36:00.1196837:CID-0:RT:Doing DESTINATION addr route-lookup

Dec  7 20:36:01 20:36:00.1196841:CID-0:RT:Doing SOURCE addr route-lookup

Dec  7 20:36:01 20:36:00.1196846:CID-0:RT:  routed (x_dst_ip 10.1.20.10) 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

Dec  7 20:36:01 20:36:00.1196850:CID-0:RT:  policy search from zone (trust) 6-> zone
(untrust) 7

Dec  7 20:36:01 20:36:00.1196854:CID-0:RT:   policy found 5
```

Note that the route lookup for destination 10.1.20.10 returned next-hop as 1.1.1.1 and ge-0/0/3.0. Since ge-0/0/0.0 is in zone "trust" and ge-0/0/3.0 is in zone "untrust", the policy lookup will be from-zone "trust" to-zone "untrust" rather than to-zone "vpn".

To resolve this, confirm that a route for 10.1.20.0/24 network exists and that the next-hop is st0.0. Also, assuming a route exists and if VPN monitoring is utilized, the route for 10.1.20.0/24 could be marked as down if the SA is down. If that is the case, then follow the IKE troubleshooting steps from previous section.

Policy lookup shows incorrect policy.

Once route lookup is complete and the ingress and egress zones are determined, the next phase is the policy lookup. Below is an example output of an incorrect policy lookup result.

```
Dec  7 20:36:01 20:36:11.937660:CID-0:RT:flow_first_routing: call flow_route_lookup():
src_ip 192.168.10.10, x_dst_ip 10.1.20.10, ifp ge-0/0/0.0, sp 45300, dp 1024, ip_proto 1, tos
0

Dec  7 20:36:01 20:36:11.937664:CID-0:RT:Doing DESTINATION addr route-lookup

Dec  7 20:36:01 20:36:11.937674:CID-0:RT:Doing SOURCE addr route-lookup

Dec  7 20:36:01 20:36:11.937678:CID-0:RT:  routed (x_dst_ip 10.1.20.10) from ge-0/0/0.0 (ge-
0/0/0.0 in 0) to st0.0, Next-hop: 10.1.20.10

Dec  7 20:36:01 20:36:11.937686:CID-0:RT:  policy search from zone (trust) 6-> zone (vpn) 8

Dec  7 20:36:01 20:36:11.937692:CID-0:RT:   policy found 2

Dec  7 20:36:01 20:36:11.937695: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.

To resolve this, confirm that an active security policy exists from-zone "trust" to-zone "vpn". View all configured policies with command: **`show security policies`**. Note also that the ordering of policies is important. The policy lookup is always from top to bottom. So be sure that a deny policy which can match the given traffic is below the permit policy.

Policy found, but no NAT translation occurs.

Policy lookup can succeed, but if the static NAT translation does not occur, then the reply traffic would not be able to properly traverse the tunnel. The flow traceoption output will also show any NAT translations if properly configured. Below is an example where no NAT translation was found.

```
Dec  7 20:36:01 20:36:38.1431151:CID-0:RT:flow_first_routing: call flow_route_lookup():
src_ip 192.168.10.10, x_dst_ip 10.1.20.10, ifp ge-0/0/0.0, sp 23164, dp 1024, ip_proto 1, tos
0

Dec  7 20:36:01 20:36:38.1431161:CID-0:RT:Doing DESTINATION addr route-lookup

Dec  7 20:36:01 20:36:38.1431170:CID-0:RT:Doing SOURCE addr route-lookup

Dec  7 20:36:01 20:36:38.1431174:CID-0:RT:  routed (x_dst_ip 10.1.20.10) from ge-0/0/0.0 (ge-
0/0/0.0 in 0) to st0.0, Next-hop: 10.1.20.10

Dec  7 20:36:01 20:36:38.1431188:CID-0:RT:  policy search from zone (trust) 6 -> zone (vpn) 8

Dec  7 20:36:01 20:36:38.1431204:CID-0:RT:   policy found 7

Dec  7 20:36:01 20:36:38.1431209:CID-0:RT:No src xlate


Dec  7 20:36:01 20:36:38.1431362:CID-0:RT:  post addr xlation: 192.168.10.10->10.1.20.10.
```

From above we can see that route lookup is correct. We also see that the correct policy is found to permit the traffic. However, rather than finding the reversed mip, no NAT translations were found. Thus later in the trace output we see post addr xlation: 192.168.10.10->10.1.20.10 which means that no address translation occurred.

To resolve this, confirm that the static NAT is configured correctly with the expected prefix and mask. You can check static NAT settings with operational-mode command: **`show security nat static rule <rule-name>`** as below.

```
root@CORPORATE> show security nat static rule remote1

Static NAT rule: remote1                Rule-set: snat1
  Rule-Id                     : 1
```

```
Rule position              : 1
From interface             : st0.0
Destination addresses      : 10.1.10.0
Host addresses             : 192.168.10.0
Netmask                    : 255.255.255.0
Host routing-instance      : N/A
Translation hits           : 53
```

Appendix A: Show Configuration

Below are the outputs of show configuration for both Corporate and Remote sites.

Corporate Site Configuration

```
root@CORPORATE> show configuration | no-more
## Last commit: 2011-05-19 14:06:19 PDT by root
version 10.4R3.4;
system {
    host-name CORPORATE;
    time-zone America/Los_Angeles;
    root-authentication {
        encrypted-password "$1$h4IhupLs$i6ZcJwYzOcxCVneaeKhm5/"; ## SECRET-DATA
    }
    login {
        user lab {
            uid 2001;
            class super-user;
            authentication {
                encrypted-password "$1$RbK4eXuQ$TdfPYuDwDzb5iJ3mRVtfy0"; ## SECRET-DATA
            }
        }
    }
    services {
        ftp;
        ssh;
        telnet;
        web-management {
            http;
        }
    }
    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 192.168.10.1/24;
            }
        }
    }
    ge-0/0/3 {
        unit 0 {
            family inet {
                address 1.1.1.2/24;
            }
        }
    }
    st0 {
        unit 0 {
            family inet {
                address 10.0.10.1/24;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 1.1.1.1;
        route 10.1.20.0/24 next-hop st0.0;
    }
}
security {
    ike {
        proposal p1-prop1 {
            authentication-method pre-shared-keys;
            dh-group group2;
            authentication-algorithm sha1;
            encryption-algorithm 3des-cbc;
        }
        policy ike-policy1 {
            mode main;
            proposals p1-prop1;
            pre-shared-key ascii-text "$9$fz/t1IcleW1RrvM8VbUjHqmTApB"; ## SECRET-DATA
        }
       gateway remote-ike {
            ike-policy ike-policy1;
            address 2.2.2.2;
            external-interface ge-0/0/3.0;
        }
    }
    ipsec {
        proposal p2-prop1 {
            protocol esp;
            authentication-algorithm hmac-sha1-96;
            encryption-algorithm 3des-cbc;
            lifetime-seconds 3600;
        }
        policy vpn-policy1 {
            perfect-forward-secrecy {
                keys group2;
            }
            proposals p2-prop1;
        }
        vpn remote-vpn {
            bind-interface st0.0;
            ike {
                gateway remote-ike;
                ipsec-policy vpn-policy1;
            }
        }
    }
    nat {
        source {
            rule-set nat-out {
                from zone trust;
                to zone untrust;
                rule interface-nat {
                    match {
                        source-address 192.168.10.0/24;
                        destination-address 0.0.0.0/0;
                    }
                    then {
                        source-nat {
                            interface;
                        }
                    }
                }
            }
        }
        static {
            rule-set snat1 {
                from interface st0.0;
                rule remote1 {
                    match {
                        destination-address 10.1.10.0/24;
                    }
                    then {
                        static-nat prefix 192.168.10.0/24;
                    }
                }
            }
        }
    }
    zones {
        security-zone trust {
            address-book {
                address local-net 192.168.10.0/24;
            }
            interfaces {
                ge-0/0/0.0 {
                    host-inbound-traffic {
                        system-services {
                            all;
                        }
                    }
                }
            }
        }
        security-zone untrust {
            interfaces {
                ge-0/0/3.0 {
                    host-inbound-traffic {
                        system-services {
                            ike;
                        }
                    }
                }
            }
        }
        security-zone vpn {
            address-book {
                address remote-net 10.1.20.0/24;
            }
            interfaces {
                st0.0 {
                    host-inbound-traffic {
                        system-services {
                            all;
                        }
                    }
                }
            }
        }
    }
    policies {
        from-zone trust to-zone vpn {
            policy remote-vpn-outgoing {
                match {
                    source-address local-net;
                    destination-address remote-net;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
        from-zone vpn to-zone trust {
            policy remote-vpn-incoming {
                match {
                    source-address remote-net;
                    destination-address local-net;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
        from-zone trust to-zone untrust {
            policy any-permit {
                match {
                    source-address any;
                    destination-address any;
                    application any;
                }
                then {
                    permit;
                }
            }
        }
    }
    flow {
        tcp-mss {
            ipsec-vpn {
                mss 1350;
            }
        }
    }
}
```

Remote Site Configuration

```
root@REMOTE> show configuration | no-more
## Last commit: 2011-05-19 22:51:01 PDT by root
version 11.1R1.10;
system {
    host-name REMOTE;
    time-zone America/Los_Angeles;
    root-authentication {
        encrypted-password "$1$h4IhupLs$i6ZcJwYzOcxCVneaeKhm5/"; ## SECRET-DATA
    }
    login {
        user lab {
            uid 2001;
            class super-user;
            authentication {
                encrypted-password "$1$RbK4eXuQ$TdfPYuDwDzb5iJ3mRVtfy0"; ## SECRET-DATA
            }
        }
    }
    services {
        ftp;
        ssh;
        telnet;
        web-management {
            http;
        }
    }
    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 192.168.10.1/24;
            }
        }
    }
    ge-0/0/3 {
        unit 0 {
            family inet {
                address 2.2.2.2/24;
            }
        }
    }
    st0 {
        unit 0 {
            family inet {
                address 10.0.20.1/24;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 2.2.2.1;
        route 10.1.10.0/24 next-hop st0.0;
    }
}
security {
    ike {
        proposal p1-prop1 {
            authentication-method pre-shared-keys;
            dh-group group2;
            authentication-algorithm sha1;
            encryption-algorithm 3des-cbc;
        }
        policy ike-policy1 {
            mode main;
            proposals p1-prop1;
            pre-shared-key ascii-text "$9$3.2f9u1SyK8LNSrWx7-g4qmfTz6BIc"; ## SECRET-DATA
        }
        gateway corp-ike {
            ike-policy ike-policy1;
            address 1.1.1.2;
            external-interface ge-0/0/3.0;
        }
```

```
}
ipsec {
    proposal p2-prop1 {
        protocol esp;
        authentication-algorithm hmac-sha1-96;
        encryption-algorithm 3des-cbc;
        lifetime-seconds 3600;
    }
    policy vpn-policy1 {
        perfect-forward-secrecy {
            keys group2;
        }
        proposals p2-prop1;
    }
    vpn corp-vpn {
        bind-interface st0.0;
        ike {
            gateway corp-ike;
            ipsec-policy vpn-policy1;
        }
        establish-tunnels immediately;
    }
}
nat {
    source {
        rule-set nat-out {
            from zone trust;
            to zone untrust;
            rule interface-nat {
                match {
                    source-address 192.168.10.0/24;
                    destination-address 0.0.0.0/0;
                }
                then {
                    source-nat {
                        interface;
                    }
                }
            }
        }
    }
    static {
        rule-set snat1 {
            from interface st0.0;
            rule remote1 {
                match {
                    destination-address 10.1.20.0/24;
                }
                then {
                    static-nat prefix 192.168.10.0/24;
                }
            }
        }
    }
```

```
}
zones {
   security-zone trust {
      address-book {
         address local-net 192.168.10.0/24;
      }
      interfaces {
         ge-0/0/0.0 {
            host-inbound-traffic {
               system-services {
                  all;
               }
            }
         }
      }
   }
   security-zone untrust {
      interfaces {
         ge-0/0/3.0 {
            host-inbound-traffic {
               system-services {
                  ike;
               }
            }
         }
      }
   }
   security-zone vpn {
      address-book {
         address corp-net 10.1.10.0/24;
      }
      interfaces {
         st0.0 {
            host-inbound-traffic {
               system-services {
                  all;
               }
            }
         }
      }
   }
}
policies {
   from-zone trust to-zone vpn {
      policy corp-vpn-outgoing {
         match {
            source-address local-net;
            destination-address corp-net;
            application any;
         }
         then {
            permit;
         }
      }
```

```
            }
            from-zone vpn to-zone trust {
                policy corp-vpn-incoming {
                    match {
                        source-address corp-net;
                        destination-address local-net;
                        application any;
                    }
                    then {
                        permit;
                    }
                }
            }
            from-zone trust to-zone untrust {
                policy any-permit {
                    match {
                        source-address any;
                        destination-address any;
                        application any;
                    }
                    then {
                        permit;
                    }
                }
            }
        }
        flow {
            tcp-mss {
                ipsec-vpn {
                    mss 1350;
                }
            }
        }
    }
```