Connecting to the VPN Headend

For this step we will use a simple Jinja2 Template and a pre-built Python script to push a prepared template to the vSRX to configure a VPN tunnel and NAT egress traffic. One of the most common jobs for a network engineer is to make the same set of changes on a regular basis with just a few minor changes such as IP address or VLAN assignments. By using a prepared template, you can drastically reduce the time spent making common network changes.

Reviewing the Template

First let's take a look at the template of changes we are going to push to this device. The configuration below is almost entirely standard Junos set commands with a view variables sprinkled in.

Template Review

1. The first two lines demonstrate using *loops* inside a Jijna template
    - Here we say "For every VPN defined for each student in the YAML file, perform the follow actions"
2. **Line 4** sets the TCP Maximum Segment Size for IPSec traffic to 1350 to take into account the additional overhead of the VPN headers
3. **Line 6 and 7** create an st0 interface that will be used by the Route Based VPN and assigns both an IP address and puts the interface into a new "vpn" zone.
4. **Line 9** opens up the firewall to accept "ike" traffic to come in via the external interface on the vSRX.
5. **Line 11-13** configures the IPSec Phase 1 VPN requirements between the local vSRX and the Headend VPN run by the Proctor
    - Note that in Line 11 we are referencing a variable from "Headend" and not our previous interfaces. This allows us to seperate Local vs remote device configuration variables as needed.
6. **Line 19-22** configures the IPSec Phase 2 VPN requirements between the local vSRX and the Headend VPN run by the Proctor
7. **Line 23-24** close out the Jinja loops from Step 1.
8. **Line 26-30** ensure that all traffic leaving over the st0 tunnel inteface between the Student network and the "Private Network behind the HeadEnd will be NAT'd using the interface egress address

Jinja2 Template

The template used for this example is found located in **"examples/vpn/vpn.j2"**

```
 {% for student in students -%}
  {% for vpn in student.vpns -%}
set security flow tcp-mss ipsec-vpn mss 1350

set interfaces {{ vpn.tunnel_int }} family inet address {{ vpn.vpn_tunnel_i
set security zones security-zone vpn interfaces {{ vpn.tunnel_int }}

set security zones security-zone untrust host-inbound-traffic system-servic

set security ike gateway ike-gate address {{ HeadEnd.vpn_ext_ip }}
set security ike gateway ike-gate external-interface {{ vpn.vpn_ext_interfa
set security ike gateway ike-gate ike-policy ike-policy1

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 "{{ vpn.share

set security ipsec policy vpn-policy1 proposal-set standard
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 {{ vpn.tunnel_int }}
  {% endfor -%}
 {% endfor -%}

set security nat source rule-set vpn_nat_out from zone trust
set security nat source rule-set vpn_nat_out to zone vpn
set security nat source rule-set vpn_nat_out rule interface-nat match sourc
set security nat source rule-set vpn_nat_out rule interface-nat match desti
set security nat source rule-set vpn_nat_out rule interface-nat then source
```

YAML Variables file

Here is a look at the YAMAL structure used by the above example. One of the keys to good templating is to build sensible and scalable variable structures. Note the following elements in this example:

- We seperate "HeadEnd" and "students" into seperate structures to allow us to refer to each seperately.
- Inside each "student" we include a list of "vpns", which could allow us to configure multiple VPNs with this template.

The template used for this example is found located in **"examples/vpn/vpn.yml"**

```
---
---
HeadEnd:
  subnets:
    - "10.10.0.0/18"
    - "10.11.0.0/18"
  vpn_ext_interface: ge-0/0/1.0
  vpn_ext_ip: 10.10.0.5

students:
  - pod-1:
    shortname: "Pod 1"
    desc: "This is Student Pod 1"
    active: true
    vpns:
      - vpn:
        active: true
        tunnel_int: st0.1
        int_descr: "Customer A Primary Link"
        active: true
        vpn_ext_interface: ge-0/0/2.0
        vpn_local_ip: 10.10.0.100
        vpn_tunnel_ip: 10.255.1.2
        vpn_zone: "vpn"
        shared_secret: "AwesomePassword123"
```

Loading the Templates

To allow us start immediately applying templates to devices without writing any Python code, we have prepared a small Python script to handle this for you.

The script is called "template-load.py" and is found in the **examples** direcory. The output below shows the arguments accepted by this script.

```
vagrant@NetDevOps-Student:/vagrant/examples$ python ./template-load.py
usage: template-load.py [-h] -b BUNDLE [-u USER] [-d DEVICE] [-p PASSWORD]
                        [-f {text,set,xml}]
template-load.py: error: argument -b/--bundle is required
vagrant@NetDevOps-Student:/vagrant/examples$
```

Once run here are the set commands that will be loaded onto the device. Again if

additional elements are added they will be generated into individual set commands.

```
vagrant@NetDevOps-Student:/vagrant/examples$ python ./template-load.py -u n
##############################################################################
# 2015-03-30 08:39:28 Connecting to Device (172.16.0.1):
##############################################################################

##############################################################################
# 2015-03-30 08:39:29 Loading Template to Candidate Config:
##############################################################################

##############################################################################
# 2015-03-30 08:39:30 Performing Config Diff:
##############################################################################


[edit interfaces]
+    st0 {
+        unit 1 {
+            family inet {
+                address 10.255.1.2/30;
+            }
+        }
+    }
[edit security]
+    ike {
+        policy ike-policy1 {
+            mode main;
+            proposal-set standard;
+            pre-shared-key ascii-text "$9$7y-dwJGiPfzDi/tu1yrWLx7bYg4ZjkPaZ
+        }
+        gateway ike-gate {
+            ike-policy ike-policy1;
+            address 10.10.0.10;
+            external-interface ge-0/0/2.0;
+        }
+    }
+    ipsec {
+        policy vpn-policy1 {
+            proposal-set standard;
+        }
+        vpn ike-vpn {
+            bind-interface st0.1;
+            ike {
+                gateway ike-gate;
+                ipsec-policy vpn-policy1;
+            }
+        }
```

```
+    }
+    flow {
+        tcp-mss {
+            ipsec-vpn {
+                mss 1350;
+            }
+        }
+    }
+    nat {
+        source {
+            rule-set vpn_nat_out {
+                from zone trust;
+                to zone vpn;
+                rule interface-nat {
+                    match {
+                        source-address-name LocalNet;
+                        destination-address-name PrivateNet;
+                    }
+                    then {
+                        source-nat {
+                            interface;
+                        }
+                    }
+                }
+            }
+        }
+    }
[edit security zones security-zone untrust]
+     host-inbound-traffic {
+         system-services {
+             ike;
+         }
+     }
[edit security zones]
     security-zone untrust { ... }
+    security-zone vpn {
+        interfaces {
+            st0.1;
+        }
+    }

################################################################################
# 2015-03-30 08:39:30 Performing Commit Check:
################################################################################

True
Proceed with Commit? [y | N]: y
################################################################################
# 2015-03-30 08:39:33 Performing Config Commit:
################################################################################
```

```
##############################################################################

##############################################################################
# 2015-03-30 08:39:33 Disconnecting from Device (172.16.0.1):
##############################################################################

vagrant@NetDevOps-Student:/vagrant/examples$
```

In the above example you see from the output logs that the script went through the following steps:

- Establish a NETCONF connection to the vSRX
- Load the template
- Perform a "show | compare" and print the output to screen
- Ensure that the configuration is a valid candidate config
- Ask you if you wish to commit the change
    - If you say "Y", the configuration is applied
    - If you say "N", the configuration is left as a candidate config
- Lastly, close out the NETCONF session to the vSRX and cleanup.

Validating the template

Now connect to your vSRX instance from your NetDevOpsVM and validate the change.

First, lets verify our Phase 1 configuration:

```
vagrant@NetDevOps-Student:~/JNPRAutomateDemo-Class/ansible$ ssh root@172.16
Password:
--- JUNOS 12.1X47-D20.7 built 2015-03-03 21:53:50 UTC
root@NetDevOps-SRX01% cli
root@NetDevOps-SRX01> show configuration security ike
policy ike-policy1 {
    mode main;
    proposal-set standard;
    pre-shared-key ascii-text "$9$7y-dwJGiPfzDi/tu1yrWLx7bYg4ZjkPaZA0IcvMaZ
}
gateway ike-gate {
    ike-policy ike-policy1;
    address 10.10.0.10;
    external-interface ge-0/0/2.0;
}
```

Second, lets verify our Phase 2 configuration:

```
root@NetDevOps-SRX01> show configuration security ipsec
policy vpn-policy1 {
    proposal-set standard;
}
vpn ike-vpn {
    bind-interface st0.1;
    ike {
        gateway ike-gate;
        ipsec-policy vpn-policy1;
    }
}
```

Lastly, lets check that we have the correct NAT configuration:

```
root@NetDevOps-SRX01> show configuration security nat
source {
    rule-set vpn_nat_out {
        from zone trust;
        to zone vpn;
        rule interface-nat {
            match {
                source-address-name LocalNet;
                destination-address-name PrivateNet;
            }
            then {
                source-nat {
                    interface;
                }
            }
        }
    }
}

root@NetDevOps-SRX01>
```