Adding NAT Policies

The first step we need to do is add in NAT policies so your NetDevOps VM can access the outside world.

Using Ansible to enable NAT

For this step we will use Ansible to create both our firewall policies and the required address book objects. This is one of the most requested automation elements when dealing with a firewall. Typically once a firewall is deployed and integrated into the network there are few changes that are required on the networking side of things. However the majority of heavy lifting is centered around managing security policies and the associated address book objects. Address book objects are typically quite painful due to the need to create, manage, and have the correct association to zones. Luckily using Ansible makes these tasks a snap.

Reviewing the playbook

First let's take a look at the playbook that is used to accomplish this task. We briefly looked at this playbook during the Ansible overview section, but now we will dive deeper into the applied steps.

Playbook Review

1. Define the name of the playbook - Configure basic firewall policies
    - This will be displayed and logged as you start to run the playbook
2. Define the hosts the playbook should be applied to
    - In this case we use the group "**mysrx**" to apply to
        - The host list is picked up from either the default Ansible host list in "/etc/ansible/hosts"
        - Alternatively when the playbook is run you can specify your own custom inventory
3. Connection is defined to as local - Typically when Ansible runs it transports an execution environment over to the host and runs it
    - Because this will not work on Junos hosts we use connection defined to local to run the execution environment

4. Gather facts
    - Ansible will gather local facts about the host such as interfaces and hostnames
    - Because this isn't possible on Junos we disable this feature
5. Vars - These are the variables that we will use to apply to our tasks
    - They can be applied at many different locations for our run
    - But to keep everything together we have included the variables into the playbook
    - address_entires will be used to generate the address book entries
    - nat_policy_info will be used to generate the NAT policies
6. Tasks - These are the tasks that we will use
    - The build phase for the playbook generates the Junos config from the templates
    - The apply phase will apply the configuration to the device
    - This will be run as two separate commits, but in doing so we can simplify the tasks and see which step fails

Playbook

```
---
- name: Configure basic NAT policies
  hosts: mysrx
  connection: local
  gather_facts: no
  vars:
    junos_user: "root"
    junos_password: "Juniper"
    build_dir: "/tmp/"
    address_entries: [ {'name':'LocalNet','prefix':'172.16.0.0/24'},{'name'
    nat_policy_info: [{'rule_set':'fw-nat', 'src_zone':'trust', 'dst_zone':

  tasks:
    - name: Build address book entries
      template: src=templates/fw_address_book_global.set.j2 dest={{build_di
      with_items: address_entries

    - name: Apply address book entries
      junos_install_config: host={{ inventory_hostname }} user={{ junos_use

    - name: Build NAT policies
      template: src=templates/nat_src_policy.set.j2 dest={{build_dir}}/nat_
      with_items: nat_policy_info

    - name: Apply NAT policies
      junos_install_config: host={{ inventory_hostname }} user={{ junos_use
```

Address book template

- The address book template loops through the address entries in the variable
- It generates one "set" configuration line per loop
- Here we are generating one

```
{% for i in address_entries %}
set security address-book global address {{ i.name }} {{ i.prefix }}
{% endfor %}
```

Output after generation

- This is the generated output from the template being applied with variables
- These commands are then committed to Junos
- If one or more of the entries are already created it will recognize this as "OK"

```
set security address-book global address NetDevOpsVM 172.16.0.10/32
```

NAT Template

This template is a bit more complex. We need to loop through source IPs, Destination IPs, and applications. Each loop through these variables will generate a single line of "set" commands. Creating the template this way allows us to reuse it in the future when we have a larger list of addresses and applications to apply.

```
{% for item in nat_policy_info %}
set security nat source rule-set {{ item.rule_set }} from zone {{ item.src_
set security nat source rule-set {{ item.rule_set }} to zone {{ item.dst_zo
    {% for rule in item.rules %}
        {% for src_ip in rule.src_ips %}
        set security nat source rule-set {{ item.rule_set }} rule {{ rule.n
        {% endfor %}
        {% for dst_ip in rule.dst_ips %}
        set security nat source rule-set {{ item.rule_set }} rule {{ rule.n
        {% endfor %}
        {% if rule.interface is defined %}
        set security nat source rule-set {{ item.rule_set }} rule {{ rule.n
        {% endif %}
        {% if rule.pool_name is defined %}
        set security nat source rule-set {{ item.rule_set }} rule {{ rule.n
        {% endif %}
    {% endfor %}
{% endfor %}
```

Output after generation

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. While we could create the applications with a single command using a list of applications this keeps it simple to implement using individual commands.

```
set security nat source rule-set fw-nat from zone trust
set security nat source rule-set fw-nat to zone untrust
set security nat source rule-set fw-nat rule rule1 match source-address 172
set security nat source rule-set fw-nat rule rule1 match destination-addres
set security nat source rule-set fw-nat rule rule1 then source-nat interfac
```

Running the playbook

To run the playbook you must use the "ansible-playbook" command.

Playbook Command

Ensure before running the command you are in the "**ansible**" directory.

```
vagrant@NetDevOps-Student:/vagrant/ansible$ ansible-playbook -i inventory.y
```

Playbook Run Example

Once run the output should look like the following

```
vagrant@NetDevOps-Student:/vagrant/ansible$ ansible-playbook -i inventory.y
sudo password:

PLAY [Configure basic NAT policies] ************************************

TASK: [Build address book entries] *************************************
changed: [172.16.0.1] => (item={'prefix': '172.16.0.0/24', 'name': 'LocalNe
ok: [172.16.0.1] => (item={'prefix': '192.168.10.0/24', 'name': 'PrivateNet
ok: [172.16.0.1] => (item={'prefix': '10.10.0.0/22', 'name': 'PublicNet'})

TASK: [Apply address book entries] *************************************
changed: [172.16.0.1]

TASK: [Build NAT policies] *********************************************
changed: [172.16.0.1] => (item={'rules': [{'interface': True, 'dst_ips': ['

TASK: [Apply NAT policies] *********************************************
changed: [172.16.0.1]

PLAY RECAP *************************************************************
172.16.0.1                 : ok=4    changed=4    unreachable=0    failed=0
```

Validate connectivity

To validate the connectivity you can issue a ping command from your NetDevOps VM. If everything is working correctly you should see a response from the host.

```
vagrant@NetDevOps-Student:~$ ping 10.10.0.5
PING 10.10.0.10 (10.10.0.5) 56(84) bytes of data.
64 bytes from 10.10.0.5: icmp_seq=1 ttl=64 time=0.482 ms
64 bytes from 10.10.0.5: icmp_seq=2 ttl=64 time=0.480 ms
64 bytes from 10.10.0.5: icmp_seq=3 ttl=64 time=0.520 ms
64 bytes from 10.10.0.5: icmp_seq=4 ttl=64 time=0.548 ms

--- 10.10.0.5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.480/0.507/0.548/0.036 ms
vagrant@NetDevOps-Student:~$
```

If the ping request is not working? Then why?