Configure Nexus 9000 as a Traffic Generator with SCAPY

Contents

Introduction

This document describes Scapy, a Python packet manipulation tool for N9K switches to create and manipulate packets with ease.

Prerequisites

Download Scapy to the switch bootflash.

To download Scapy, use the link from GitHub GitHub-SCAPY

Requirements

Cisco recommends that you have knowledge of these topics:

- Nexus 9000/3000 Switch.

Components Used

- N9K-C9396PX

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Installation

Download and extract the Scapy code to your switch boot flash; FTP, SFTP, or SCP are available.

Enable the feature, in this case, SCP.

```
switch(config)# feature scp-server
switch(config)# sh feature | i scp
scpServer              1          enabled
```

Copy the file to the switch from the laptop.

```
scp scapy-vxlan-master.zip admin@10.88.164.13:/
```

Once the image is in the boot flash, it needs to be decompressed. It needs to enable feature bash and unzip it from bash.

```
switch(config)# feature bash
switch(config)# run bash
bash-4.3$ sudo su -
root@switch#cd /bootflash
root@switch#unzip scapy-vxlan-master.zip
```

Once decompressed, the files can be located with the **dir** command on the boot flash, the compressed and uncompressed.

```
switch# dir bootflash: | i i scapy
       4096    Jul 09 18:00:01 2019  scapy-vxlan-master/
    1134096    Jul 19 23:35:26 2023  scapy-vxlan-master.zip
```

Now Scapy is available.

Notice that you need to call the program with root privileges and you also need to navigate to the Scapy directory.

```
switch(config)# run bash
Enter configuration commands, one per line. End with CNTL/Z.
bash-4.2$ sudo su -
root@switch#cd /
root@switch#cd bootflash/scapy-vxlan-master       <<< Move to the scapy folder scapy-vxlan-master
root@switch#python                                <<< Run python once located inside the folder
```

```
Python 2.7.2 (default, Mar  9 2015, 15:52:40)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *                       <<< Import libraries from scapy
>>>
```

Create a Packet

This is an example of how to create a basic IP packet to illustrate the procedure to generate traffic using Scapy.

```
Create l2 source and destination mac addresses.
>>> l2=Ether()
>>> l2.src='00:aa:12:34:12:34'
>>> l2.src='00:ff:aa:bb:cc:11'

Create l3 source and destination IP addresses.
>>> l3=IP()
>>> l3.src='10.1.1.1'
>>> l3.dst='10.2.2.2'
```

Another capability is to send a packet from a pcap file previously captured. This is achieved with the command **rdpcap**.

The output of that command is a Python list containing all the packets captured in your pcap file. In this example, **traffic.pcap** contains 10 packets and those packets are being assigned to the list created as pkts.

```
>>> pkts = rdpcap('bootflash/traffic.pcap')
>>> len(pkts)
10
>>> type(pkts)
<class 'scapy.plist.PacketList'>
```

> **Note**: The pcap file needs to be stored in the boot flash of the switch.

Send Traffic

Once the packet is created, we use the command **sendp** to start sending our packet over the specified interface.

```
>>> packet = l2/l3.                  << packet now have the values for source and destination declared
>>> sendp(packet, iface='Eth1-1').   << Sending the packet through interface eth1/1
.
Sent 1 packets.
```

You can then iterate through the list of packets to send the traffic over the interface you specify.

```
>>> while True:
...  for i in range(len(pkts)):         <<< It goes through the list pkts with 10 packets and send 1 by
...   sendp(pkts[i], iface='Eth1-1')
...
.
Sent 1 packets.
.
Sent 1 packets.
```

Note: Only switch ports mode access are available to be used. Otherwise, it displays an error.

Example of the error:

```
>>> sendp(l2/l3, iface='Eth1-6')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "scapy/sendrecv.py", line 335, in sendp
socket = socket or conf.L2socket(iface=iface, *args, **kargs)
File "scapy/arch/linux.py", line 477, in __init__
set_promisc(self.ins, self.iface)
File "scapy/arch/linux.py", line 165, in set_promisc
mreq = struct.pack("IHH8s", get_if_index(iff), PACKET_MR_PROMISC, 0, b"")
File "scapy/arch/linux.py", line 380, in get_if_index
return int(struct.unpack("I", get_if(iff, SIOCGIFINDEX)[16:20])[0])
File "scapy/arch/common.py", line 59, in get_if
ifreq = ioctl(sck, cmd, struct.pack("16s16x", iff.encode("utf8")))
IOError: [Errno 19] No such device
```

Ensure the interface is usable, run the **ifconfig** command, the interface must be listed in there.

```
bash-4.3$ ifconfig | grep Eth
Eth1-1 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:88
Eth1-2 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:89
Eth1-5 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:8c
Eth1-6 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:8d
Eth1-8 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:8f
Eth1-11 Link encap:Ethernet HWaddr 00:a2:ee:74:4b:c1
...
```

Verify

You can use the command to check any given packet.

```
>>> pkts[5].show()
###[ Ethernet ]###
  dst       = 01:00:0c:cc:cc:cd
  src=58:97:bd:00:a4:f2
  type      = 0x8100
###[ 802.1Q ]###
     prio      = 6
     id        = 0
     vlan      = 104
     type      = 0x32
###[ LLC ]###
        dsap      = 0xaa
        ssap      = 0xaa
        ctrl      = 3
###[ SNAP ]###
           OUI       = 0xc
           code      = 0x10b
###[ Spanning Tree Protocol ]###
              proto     = 0
              version   = 2
              bpdutype  = 2
              bpduflags = 60
              rootid    = 32872
              rootmac   = 58:97:bd:00:a4:f1
              pathcost  = 0
              bridgeid  = 32872
              bridgemac = 58:97:bd:00:a4:f1
              portid    = 32769
              age       = 0.0
              maxage    = 20.0
              hellotime = 2.0
              fwddelay  = 15.0
###[ Raw ]###
                 load      = '\x00\x00\x00\x00\x02\x00h'
```