SCAPYを使用したトラフィックジェネレータとしてのNexus 9000の設定

内容

概要

このドキュメントでは、N9Kスイッチでパケットを簡単に作成および操作するためのPythonパケット操作ツールであるScapyについて説明します。

前提条件

Scapyをスイッチのブートフラッシュにダウンロードします。

Scapyをダウンロードするには、GitHub GitHub-SCAPYからのリンクを使用してください。

要件

次の項目に関する知識があることが推奨されます。

- Nexus 9000/3000スイッチ

使用するコンポーネント

- N9K-C9396PX

このドキュメントの情報は、特定のラボ環境にあるデバイスに基づいて作成されました。このド

キュメントで使用するすべてのデバイスは、クリアな（デフォルト）設定で作業を開始しています。本稼働中のネットワークでは、各コマンドによって起こる可能性がある影響を十分確認してください。

設置

Scapyコードをダウンロードしてスイッチのブートフラッシュに抽出します。FTP、SFTP、またはSCPを使用できます。

機能（この場合はSCP）を有効にします。

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

ラップトップからスイッチにファイルをコピーします。

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

イメージがブートフラッシュに保存されたら、圧縮を解除する必要があります。bash機能を有効にし、bashから解凍する必要があります。

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

圧縮解除されたファイルは、ブートフラッシュ上でdirコマンドを使用して検索できます。このコマンドは圧縮および圧縮解除されています。

```
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
```

これでScapyが使用可能になりました。

root権限でプログラムを呼び出す必要があり、Scapyディレクトリに移動する必要があることに注意してください。

```
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
>>>
```

パケットの作成

これは、Scapyを使用してトラフィックを生成する手順を示すために、基本的なIPパケットを作成する方法の例です。

```
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'
```

もう1つの機能は、以前にキャプチャしたpcapファイルからパケットを送信することです。これは、rdpcapコマンドを使用して実行します。

このコマンドの出力は、pcapファイルでキャプチャされたすべてのパケットを含むPythonリストです。この例では、traffic.pcapに10個のパケットが含まれており、これらのパケットはpktsとして作成されたリストに割り当てられています。

```
>>> pkts = rdpcap('bootflash/traffic.pcap')
>>> len(pkts)
```

```
10
>>> type(pkts)
<class 'scapy.plist.PacketList'>
```

注:pcapファイルは、スイッチのブートフラッシュに保存する必要があります。

トラフィックの送信

パケットが作成されたら、コマンドsendpを使用して、指定したインターフェイスを介したパケットの送信を開始します。

```
>>> 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.
```

その後、パケットリストを反復処理して、指定したインターフェイス経由でトラフィックを送信できます。

```
>>> 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.
```

注：使用できるのはスイッチポートモードアクセスだけです。それ以外の場合は、エラーが表示されます。

エラーの例：

```
>>> 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
```

インターフェイスが使用可能であることを確認し、ifconfigコマンドを実行します。インターフェイスがそこにリストされている必要があります。

```
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
...
```

確認

コマンドを使用して、任意のパケットを確認できます。

```
>>> 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'
```

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。