Nexus 9000을 SCAPY를 통해 트래픽 생성기로 구성

목차

소개

이 문서에서는 N9K 스위치용 Python 패킷 조작 도구인 Scapy를 사용하여 손쉽게 패킷을 생성하고 조작할 수 있습니다.

사전 요구 사항

스위치 부트플래시에 Scapy를 다운로드합니다.

Scapy를 다운로드하려면 GitHub GitHub -SCAPY의 링크를 사용합니다

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Nexus 9000/3000 스위치.

사용되는 구성 요소

- N9K-C9396PX

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바

이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

설치

스위치 부트 플래시에 Scapy 코드를 다운로드하고 압축을 풉니다. FTP, SFTP 또는 SCP를 사용할 수 있습니다.

이 경우 SCP 기능을 활성화합니다.

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

랩톱에서 스위치로 파일을 복사합니다.

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

이미지가 부트 플래시에 저장되면 압축을 풀어야 합니다. 기능 배쉬를 활성화하고 배쉬에서 압축을 풀어야 합니다.

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

압축을 풀면 파일은 부팅 플래시에서 dir 명령을 사용하여 압축되고 압축되지 않은 상태로 찾을 수 있습니다.

```
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
```

이제 Scapy가 제공됩니다.

루트 권한으로 프로그램을 호출해야 하며 Scapy 디렉토리로도 이동해야 합니다.

```
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
>>>
```

패킷 생성

다음은 Scapy를 사용하여 트래픽을 생성하는 절차를 설명하기 위해 기본 IP 패킷을 생성하는 방법의 예입니다.

```
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'
```

또 다른 기능은 이전에 캡처된 pcap 파일에서 패킷을 전송하는 것입니다. 이 작업은 rdpcap 명령으로 수행합니다.

이 명령의 출력은 pcap 파일에서 캡처한 모든 패킷을 포함하는 Python 목록입니다. 이 예에서 traffic.pcap는 10개의 패킷을 포함하며 이러한 패킷은 pkt로 생성된 목록에 할당됩니다.

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

참고: pcap 파일은 스위치의 부트 플래시에 저장해야 합니다.

전송 트래픽

패킷이 생성되면 sendp 명령을 사용하여 지정된 인터페이스를 통해 패킷 전송을 시작합니다.

```
>>> 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.
```

그런 다음 지정한 인터페이스를 통해 트래픽을 전송하도록 패킷 목록을 반복할 수 있습니다.

```
>>> 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.
```

참고: 스위치 포트 모드 액세스만 사용할 수 있습니다. 그렇지 않으면 오류가 표시됩니다.

오류의 예:

```
>>> 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
```

인터페이스를 사용할 수 있는지 확인하고 ifconfig 명령을 실행합니다. 인터페이스가 목록에 있어야 합니다.

```
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
...
```

다음을 확인합니다.

명령을 사용하여 지정된 패킷을 확인할 수 있습니다.

```
>>> 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'
```

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.