Skriptovanie v kontexte PC sietí

Knižnica Scapy

Práca s knižnicou Netmiko

Knižnica Netmiko predstavuje jednoduchého SSH klienta, ktorým sa je možné pripojiť na sieťové zariadenie, odosielať na neho príkazy a prijímať odpovede. Vďaka tomu, že je to knižnica pre programovací jazyk python, tak je možné vytvoriť rôzne scenáre na automatizáciu správy konfigurácie. Nasledujúce úlohy Vás prevedú základnou prácou s knižnicou Netmiko.

1. Nainštalujte knižnicu Netmiko:

```
pip3 install netmiko
```

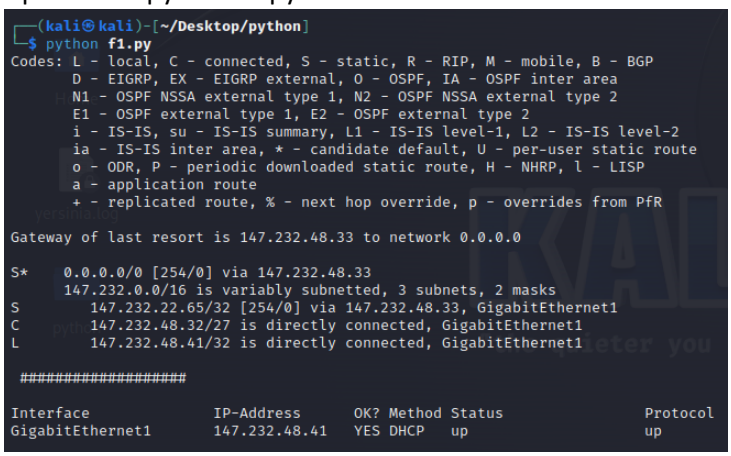
2. Vytvorte jednoduchý python script pre pripojenie sa na službu SSH a odoslanie jednoduchého show príkazu pre zobrazenie smerovacej tabuľky a následne výpis rozhraní:

nano f1.py

```
from netmiko import ConnectHandler

def connect(ip_add):
  ssh_conn = ConnectHandler(
    host=ip_add,
    port='22',
    username='cisco',
    password='cisco123!',
    device_type='cisco_ios'
    )
  return ssh_conn


if __name__ == "__main__":
  router_connection = connect("147.232.48.41")
  route_table = router_connection.send_command('show ip
route')
  print(route_table)
  print('\n ###################\n')

  interfaces = router_connection.send_command('show ip int
br')
  print(interfaces)
  router_connection.disconnect()
```

Ako je možné vidieť, zdrojový kód obsahuje import objektu reprezentujúceho vytvárané spojenie, v ktorom stačí definovať cieľovú adresu, číslo portu, prihlasovacie údaje a typ cieľového zariadenia. Využitím uvedeného objektu je následne možné na zariadenie odosielať príkazy a následne pracovať s návratovou hodnotou, ktorou je reťazec totožný s štandardnými výpismi pri interakcii so sieťovým zariadením.

Spustite program príkazom: python f1.py:

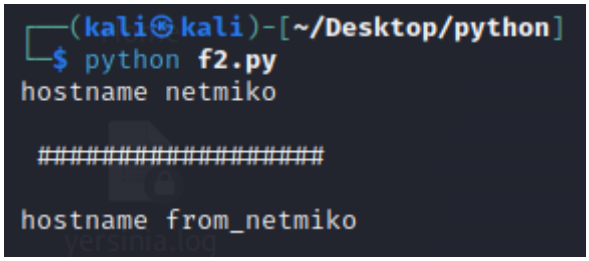


3. Vytvorte script, ktorým zmeníte názov zariadenia a zobrazíte jeho názov pred a po zmene:

nano f2.py

```
from netmiko import ConnectHandler

def connect(ip_add):
  ssh_conn = ConnectHandler(
    host=ip_add,
    port='22',
    username='cisco',
    password='cisco123!',
    device_type='cisco_ios'
    )
  return ssh_conn


if __name__ == "__main__":
  router_connection = connect("147.232.48.41")
  router_connection.enable()

  hostname = router_connection.send_command('show run | inc
hostname')
  print(hostname)
  print('\n ################## \n')

  router_connection.config_mode()

  router_connection.send_command('hostname
from_netmiko_final')

  router_connection.exit_config_mode()

  show_hostname = router_connection.send_command('show run |
inc hostname', expect_string=r'#', read_timeout=90)
  print(show_hostname)

  router_connection.disconnect()
```

Spustite program príkazom python f2.py:

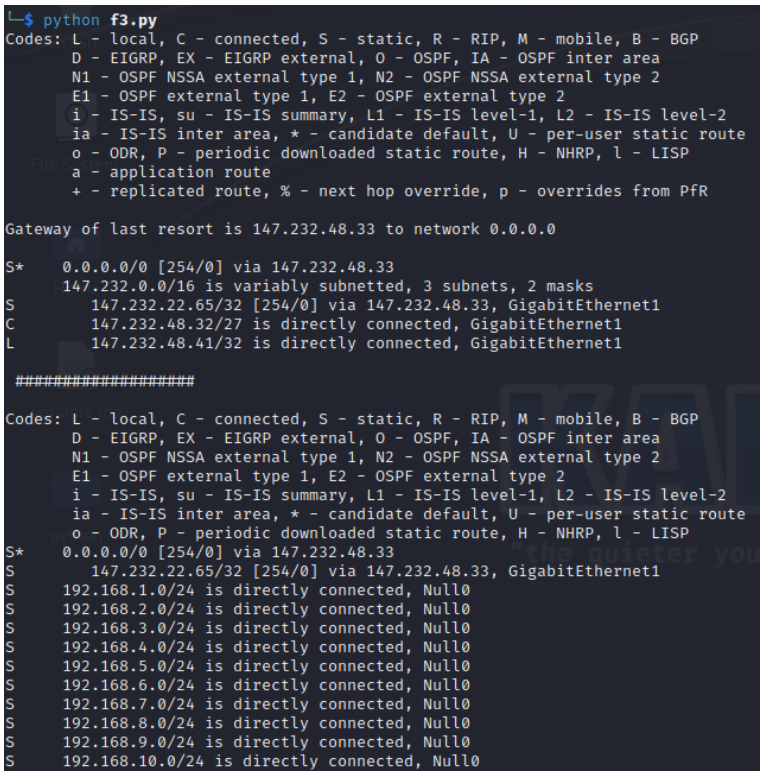


4. Vytvorte skript, ktorý vloží na smerovač 10 statických ciest v rozsahu 192.168.1.0/24-192.168.10.0/24. Správnosť konfigurácie overte zobrazením smerovacej tabuľky.

nano f3.py

```
from netmiko import ConnectHandler

def connect(ip_add):
  ssh_conn = ConnectHandler(
    host=ip_add,
    port='22',
    username='cisco',
    password='cisco123!',
    device_type='cisco_ios'
    )
  return ssh_conn


if __name__ == "__main__":
  router_connection = connect("147.232.48.41")
  route_table = router_connection.send_command('show ip
route')
  print(route_table)
  print('\n ###################\n')

  router_connection.config_mode()

  for i in range(1, 11):
    route = "ip route 192.168.{}.0 255.255.255.0
null0".format(i)
    router_connection.send_command(route)

  route_table = router_connection.send_command('do show ip
route | inc S')
  print(route_table)
  router_connection.disconnect()
```

Spusťte program príkazom python f3.py:



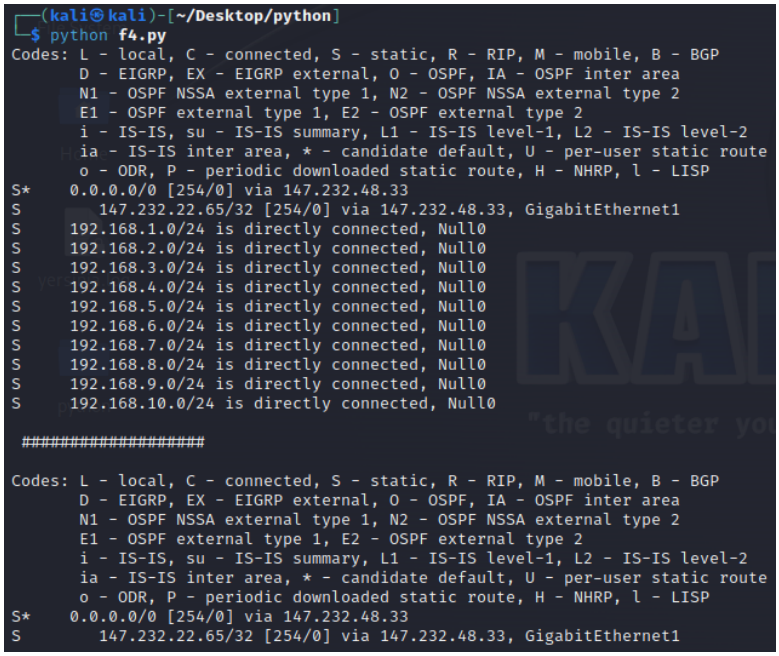
5. Napíšte skript, ktorý vymaže vytvorené statické cesty:

nano f4.py

```
from netmiko import ConnectHandler

def connect(ip_add):
  ssh_conn = ConnectHandler(
    host=ip_add,
    port='22',
    username='cisco',
    password='cisco123!',
    device_type='cisco_ios'
    )
  return ssh_conn


if __name__ == "__main__":
  router_connection = connect("147.232.48.41")
  route_table = router_connection.send_command('show ip route
| inc S')
  print(route_table)
  print('\n ###################\n')
  router_connection.config_mode()

  for i in range(1, 11):
    route = "no ip route 192.168.{}.0 255.255.255.0
null0".format(i)
    router_connection.send_command(route)
```

```
  route_table = router_connection.send_command('do show ip
route | inc S')
  print(route_table)

  router_connection.disconnect()
```

Program spusťte príkazom python f4.py:



Generovanie dátových jednotiek:

Knižnica Scapy poskytuje množstvo funkcií na prácu s dátovými jednotkami. Umožňuje vytvárať a odosielať dátové jednotky, ale aj odchytávať dáta prechádzajúce cez sieťovú kartu. Interagovať s touto knižnicou je možné využitím programovacieho jazyka Python, ale aj využitím terminálovej aplikácie. Knižnica obsahuje objekty opisujúce základné dátové jednotky akými sú hlavičky Ether, IP, ICMP, TCP, a ďalšie, ktoré obsahujú všetky potrebné polia nastavené na defaultnú hodnotu umožňujúce odoslať vytvorené PDU do siete. Hlavičky je možné reťaziť znakom '/'. Knižnica tiez obsahuje funkcie na odosielanie vytvorených dát, ale aj na odchytávanie všetkých dát prenášaných sieťovou kartou. Nasledujúce úlohy ukážu oba prístupy.

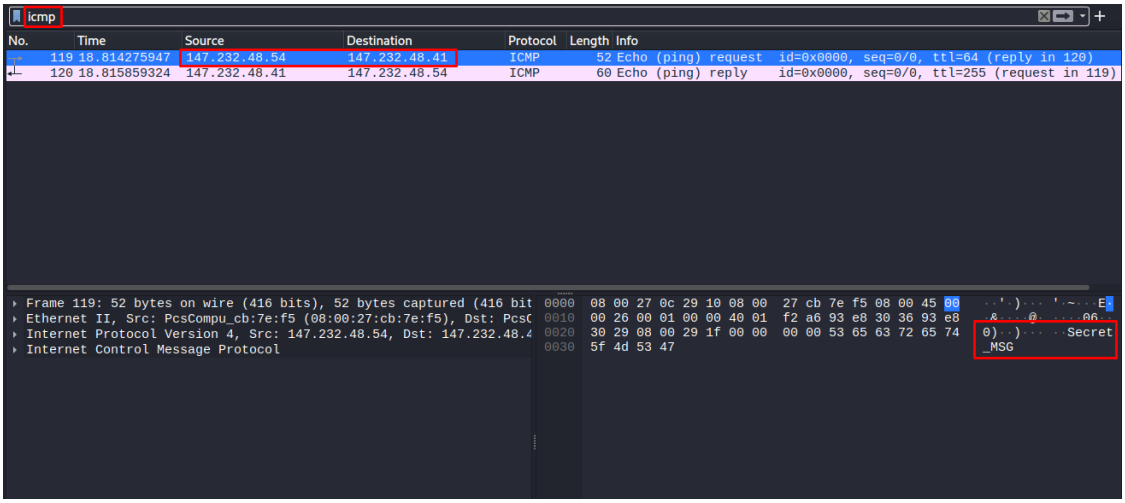
Interakcia s knižnicou Scapy využitím programovacieho jazyka

1. Vytvorte skript, ktorý vygeneruje ICMP echo request správu s Vami zvoleným obsahom v tele:

nano s1.py

```
from scapy.layers.inet import ICMP, IP
from scapy.sendrecv import sr1

#vytvorenie dátovej jednotky = default IP  + default ICMP +
payload
icmp = IP()/ICMP()/"Secret_MSG"

#úprava atribútov IP hlavičky
icmp[IP].src = "147.232.48.54"
icmp[IP].dst = "147.232.48.41"

#odoslanie požiadaviek
resp = sr1(icmp, timeout=2)

if resp:
  print("Sprava uspesne odoslana!!!")
```

Otvorte nástroj wireshark a odchyťte ICMP echo request správu, pričom skontrolujte obsah prenášaných dát.



Program spusťte príkazom: **sudo python s1.py** (tým, že knižnica generuje dáta a má prístup k sieťovej karte, tak je skripty potrebné spúšťať s vyššími právami).

2. Vytvorte skript, ktorý umožní odchytávať ICMP správy a zobrazí o nich informácie v termináli:

nano s2.py

```
from scapy.layers.inet import IP
from scapy.packet import Raw
from scapy.sendrecv import sniff

# odchytenie 5 ICMP správ a vytvorenie poľa pre ukladanie
kľúčových charakteristík
packets = sniff(filter="icmp", count= 5)
rcv_data = []

#cyklus, ktorý prejde každou odchytenou ICMP správou
for packet in range(0, 5):
    #do JSON podoby uloží hľadané charakteristiky
(zdrojová/cieľová adresa a payload)
    packet_info = {
        "src_address": packets[packet][IP].src,
        "dst_address": packets[packet][IP].dst,
        "payload": packets[packet][Raw].load.decode('utf-
8','ignore')
    }
    #uloženie dát do vytvoreného poľa
    rcv_data.append(packet_info)
#výpis obsahu odchytených dát
for i in range(0, len(rcv_data)):
  print(str(rcv_data[i])+"\n")
```

Program spusťte príkazom: **sudo python s2.py**
Následne otvorte druhý terminál, z ktorého vygenerujte ICMP komunikáciu – napr. ping na cnl.sk

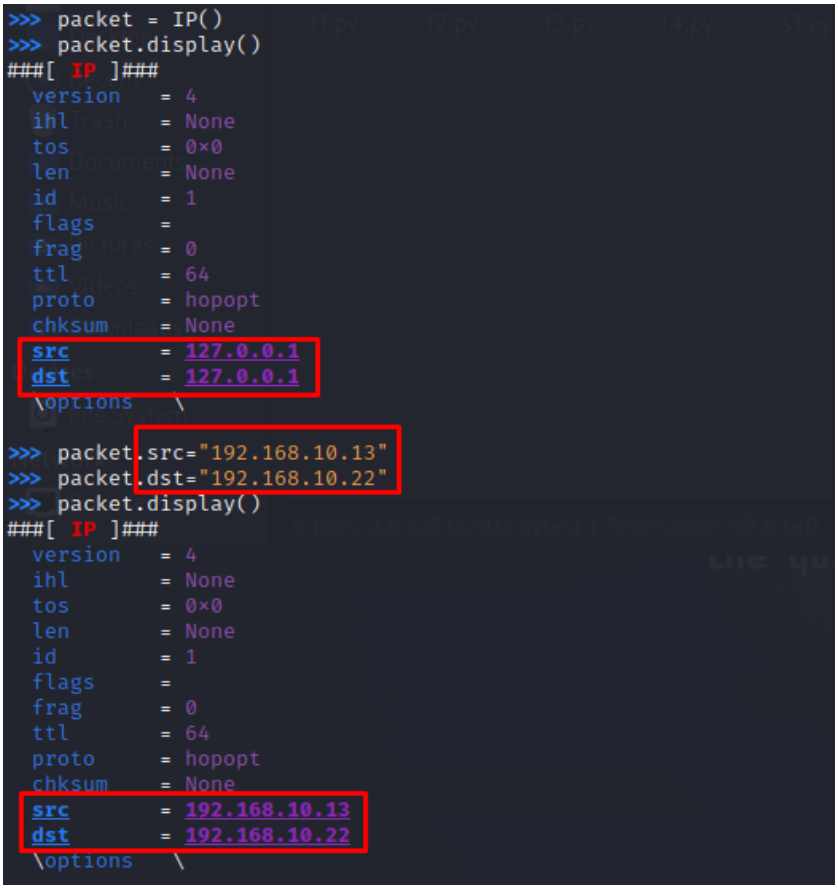
Interakcia s knižnicou Scapy využitím terminálovej aplikácie

1. Interagujte priamo s knižnicou Scapy, pričom skúste vytvoriť rôzne dátové jednotky, ktoré následne odošlite do siete a overte funkčnosť využitím nástroja wireshark:
    a. Vytvorenie IP hlavičky, úprava polí v hlavičke a ich zobrazenie:

```
sudo scapy

packet = IP()
```

```
packet.display()


packet.src="192.168.10.13"
packet.dst="192.168.10.22"
packet.display()
```

Pozn.: Knižnica scapy generuje dátové jednotky s prázdnymi hlavičkami (polia nastavené na hodnoty tak, aby bolo možné poslať defaultnú dátovú jednotku). Obsah hlavičky je možné zmeniť.



b. Vytvorenie a odoslanie ICMP správy:

```
ping_data = IP()/ICMP()/"fromScapy"
ping_data['IP'].dst="147.232.48.41"
ping_data.display()
send(ping_data)
```

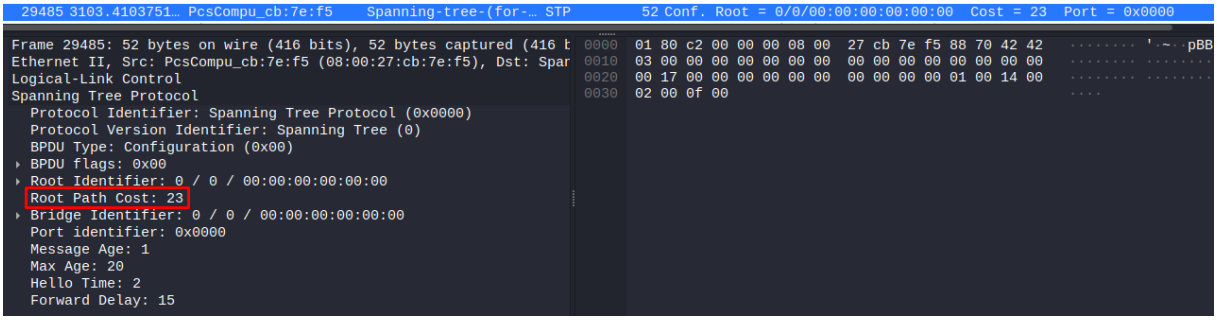
```
>>> ping_data = IP()/ICMP()/"fromScapy"
>>> ping_data['IP'].dst="147.232.48.41"
>>> ping_data.display()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = None
  src       = 147.232.48.54
  dst       = 147.232.48.41
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = None
     id        = 0x0
     seq       = 0x0
     unused    = ''
###[ Raw ]###
        load      = 'fromScapy'

>>> send(ping_data)
.
Sent 1 packets.
```

c. Vytvorenie, úprava a odoslanie STP správy:

```
bpdu = Ether(dst="01:80:c2:00:00:00")/LLC()/STP()
bpdu.pathcost=23
bpdu.display()
sendp()
```

```
>>> bpdu = Ether(dst="01:80:c2:00:00:00")/LLC()/STP()
>>> bpdu.pathcost=23
>>> bpdu.display()
###[ Ethernet ]###
  dst       = 01:80:c2:00:00:00
  src       = 08:00:27:cb:7e:f5
  type      = 0x8870
###[ LLC ]###
     dsap      = 0x42
     ssap      = 0x42
     ctrl      = 3
###[ Spanning Tree Protocol ]###
        proto     = 0
        version   = 0
        bpdutype  = 0
        bpduflags = 0
        rootid    = 0
        rootmac   = 00:00:00:00:00:00
        pathcost  = 23
        bridgeid  = 0
        bridgemac = 00:00:00:00:00:00
        portid    = 0
        age       = 1
        maxage    = 20
        hellotime = 2
        fwddelay  = 15

>>> sendp(bpdu, iface="eth0")
.
Sent 1 packets.
```

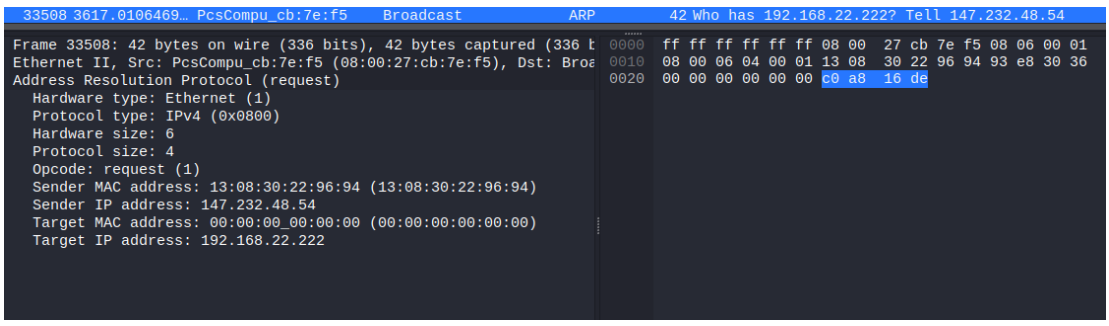
Overiť správnosť je možné nástrojom Wireshark:



3. Vytvorenie, zobrazenie a odoslanie ARP správy + zobrazenie jej hexa reprezentácie:

```
arp = pkt=Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(pdst="192.168.22.222",
hwsrc="13:08:30:22:96:94")
arp.display()
sendp(arp, iface="eth0")
hexdump(arp)
```

Overiť správnosť vygenerovanej ARP správy je možné využitím nástroja wireshark:



Interakcia s nástrojom Yersinia

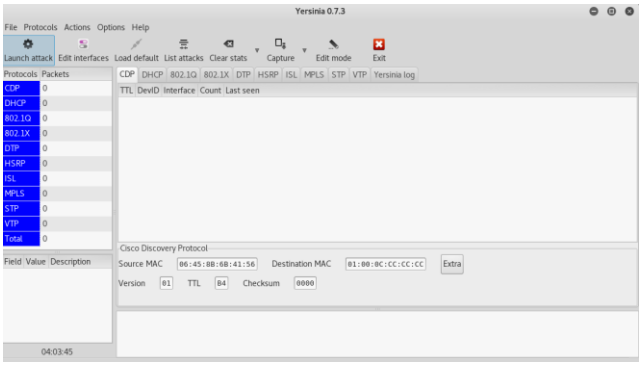
Jedným z ďalších nástrojov, ktoré je možné využiť na vykonanie útokov cielených na LAN sieť je nástroj Yersinia, ktorý umožňuje generovať dátové jednotky pre nasledujúce protokoly:

- Spanning Tree Protocol
- Cisco Discovery Protocol
- Dynamic Trunking Protocol
- Dynamic Host Configuration Protocol
- Host Standby Router Protocol
- 802.1q
- 802.1x
- Inter-Switch Link Protocol
- VLAN Trunking Protocol

Inštalácia a spustenie nástroja je možná nasledujúcimi príkazmi:

```
sudo apt install yersinia
sudo yersinia -G
sudo yersinia -I
```

Používateľské rozhranie nástroja Yersinia je možné spustiť príkazom **sudo yersinia -G**:



Tento nástroj je možné používať aj využitím terminálovej interakcie – spustenie možné príkazom:

```
sudo yersinia -I
```

Užitočné klávesové skratky:

- h (pomoc)
- g (prepínanie medzi typmi útokov)
- e (editovanie posielaných správ)
- x (spustenie útoku)
- K (zastavenie všetkých útokov)

Pohrajte sa s týmto nástrojom a jeho funkčnosť si overte show výpismi na prepínači, prípadne využitím nástroja Wireshark.