NETWORK ATTACKS AND MITIGATIONS.../

ABOUT US

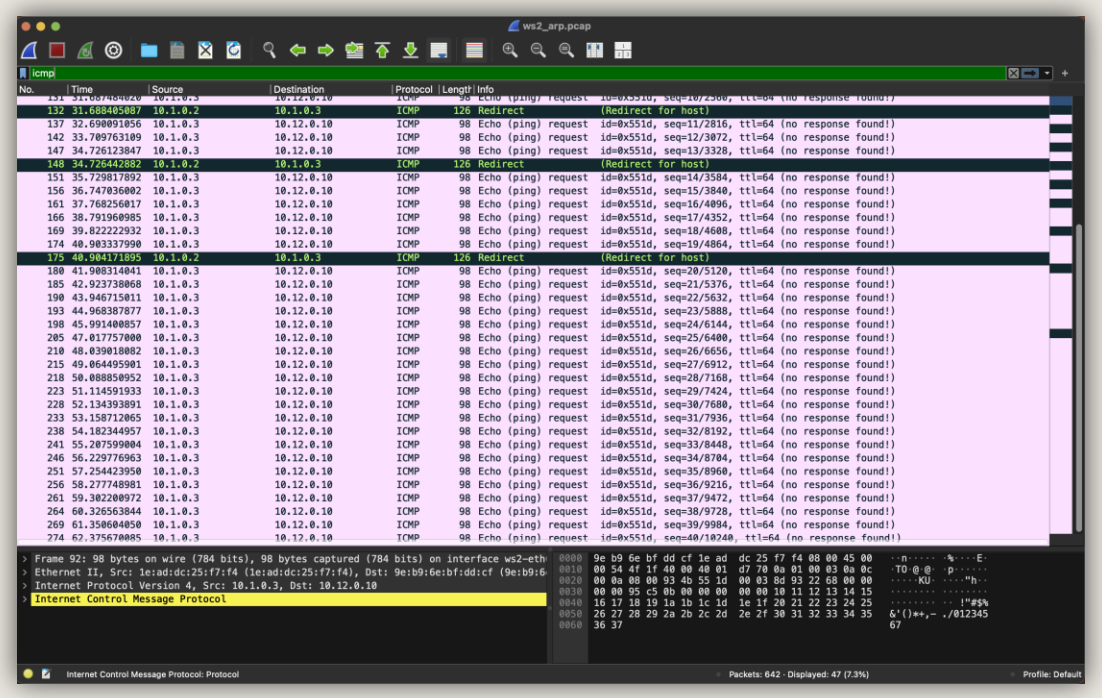
ADEWUYI SAMUEL DAN

NOMA: 25682410

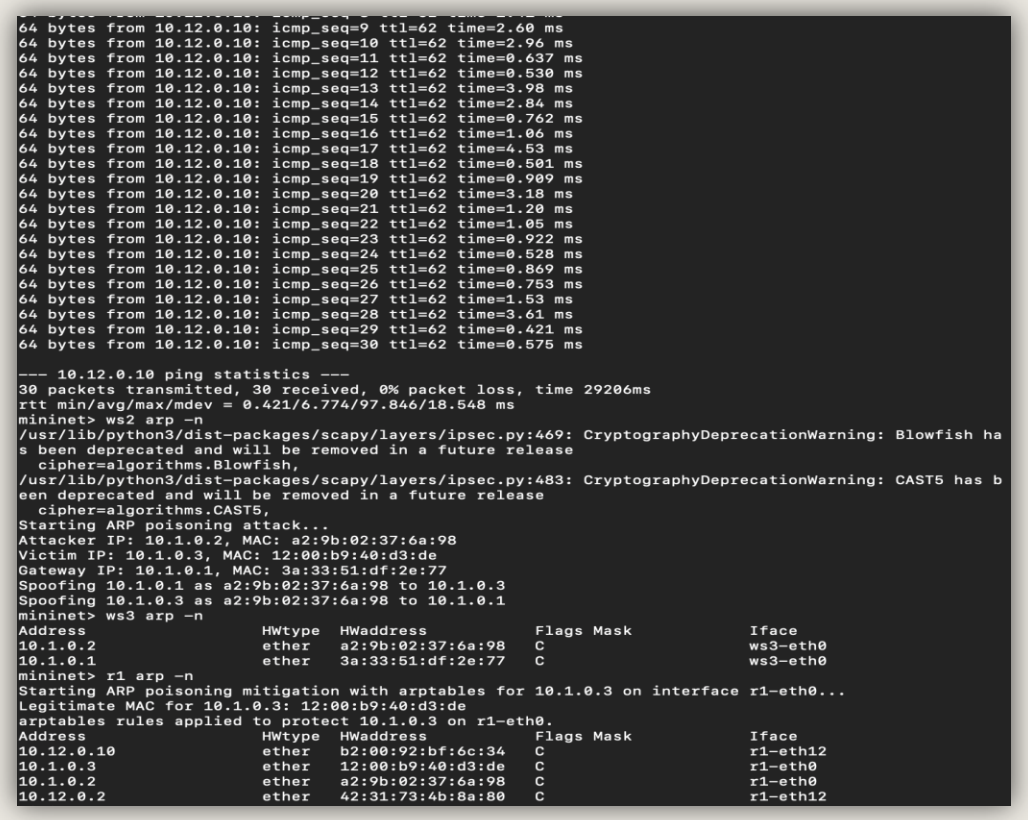
DJONFANG TCHUANGUE VITALY BELVINE

NOMA: 16632410

ARP CACHE POISONING



- This is a (MitM) attack that allows to intercept communication between network devices.
- The forged responses advertise that the correct MAC address for both IP addresses, belonging to the router and workstation, is the attacker's MAC address.

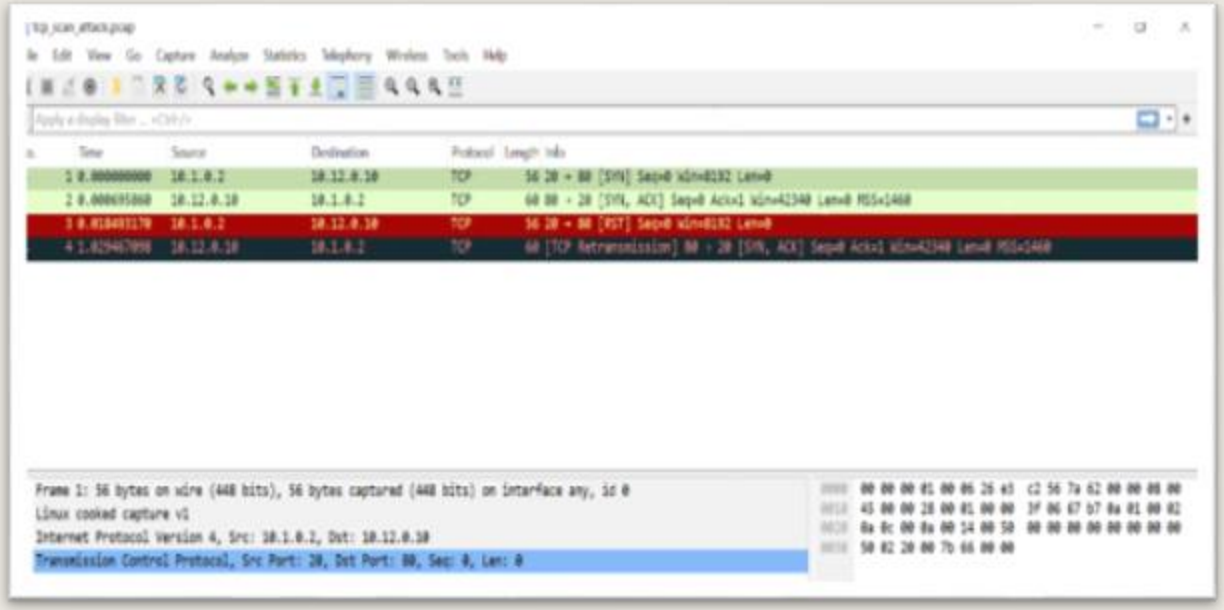


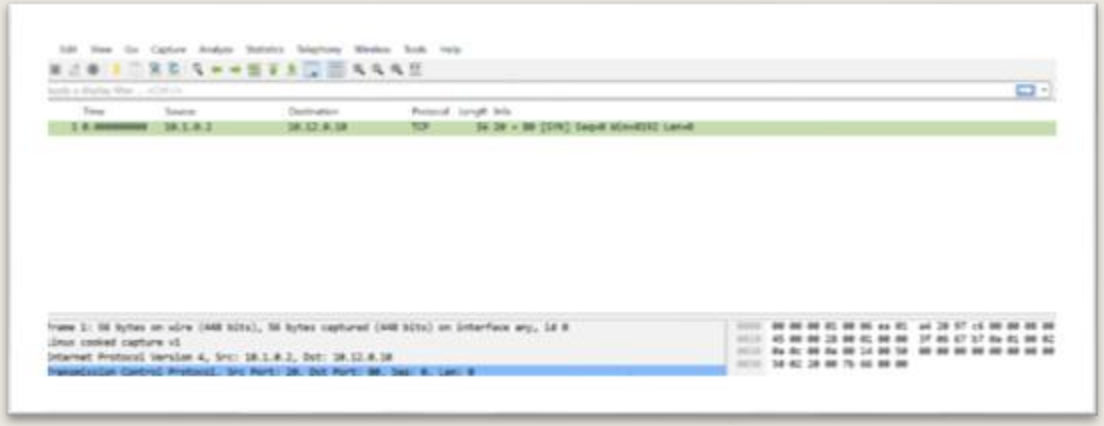
```
64 bytes from 10.12.0.10: icmp_seq=9 ttl=62 time=2.60 ms
64 bytes from 10.12.0.10: icmp_seq=10 ttl=62 time=2.96 ms
64 bytes from 10.12.0.10: icmp_seq=11 ttl=62 time=0.637 ms
64 bytes from 10.12.0.10: icmp_seq=12 ttl=62 time=0.530 ms
64 bytes from 10.12.0.10: icmp_seq=13 ttl=62 time=3.98 ms
64 bytes from 10.12.0.10: icmp_seq=14 ttl=62 time=2.84 ms
64 bytes from 10.12.0.10: icmp_seq=15 ttl=62 time=0.762 ms
64 bytes from 10.12.0.10: icmp_seq=16 ttl=62 time=1.06 ms
64 bytes from 10.12.0.10: icmp_seq=17 ttl=62 time=4.53 ms
64 bytes from 10.12.0.10: icmp_seq=18 ttl=62 time=0.501 ms
64 bytes from 10.12.0.10: icmp_seq=19 ttl=62 time=0.909 ms
64 bytes from 10.12.0.10: icmp_seq=20 ttl=62 time=3.18 ms
64 bytes from 10.12.0.10: icmp_seq=21 ttl=62 time=1.20 ms
64 bytes from 10.12.0.10: icmp_seq=22 ttl=62 time=1.05 ms
64 bytes from 10.12.0.10: icmp_seq=23 ttl=62 time=0.922 ms
64 bytes from 10.12.0.10: icmp_seq=24 ttl=62 time=0.528 ms
64 bytes from 10.12.0.10: icmp_seq=25 ttl=62 time=0.869 ms
64 bytes from 10.12.0.10: icmp_seq=26 ttl=62 time=0.753 ms
64 bytes from 10.12.0.10: icmp_seq=27 ttl=62 time=1.53 ms
64 bytes from 10.12.0.10: icmp_seq=28 ttl=62 time=3.61 ms
64 bytes from 10.12.0.10: icmp_seq=29 ttl=62 time=0.421 ms
64 bytes from 10.12.0.10: icmp_seq=30 ttl=62 time=0.575 ms

--- 10.12.0.10 ping statistics ---
30 packets transmitted, 30 received, 0% packet loss, time 29206ms
rtt min/avg/max/mdev = 0.421/6.774/97.846/18.548 ms
mininet> ws2 arp -n
/usr/lib/python3/dist-packages/scapy/layers/ipsec.py:469: CryptographyDeprecationWarning: Blowfish ha
s been deprecated and will be removed in a future release
  cipher=algorithms.Blowfish,
/usr/lib/python3/dist-packages/scapy/layers/ipsec.py:483: CryptographyDeprecationWarning: CAST5 has b
een deprecated and will be removed in a future release
  cipher=algorithms.CAST5,
Starting ARP poisoning attack...
Attacker IP: 10.1.0.2, MAC: a2:9b:02:37:6a:98
Victim IP: 10.1.0.3, MAC: 12:00:b9:40:d3:de
Gateway IP: 10.1.0.1, MAC: 3a:33:51:df:2e:77
Spoofing 10.1.0.1 as a2:9b:02:37:6a:98 to 10.1.0.3
Spoofing 10.1.0.3 as a2:9b:02:37:6a:98 to 10.1.0.1
mininet> ws3 arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.1.0.2                 ether   a2:9b:02:37:6a:98   C                     ws3-eth0
10.1.0.1                 ether   3a:33:51:df:2e:77   C                     ws3-eth0
mininet> r1 arp -n
Starting ARP poisoning mitigation with arptables for 10.1.0.3 on interface r1-eth0...
Legitimate MAC for 10.1.0.3: 12:00:b9:40:d3:de
arptables rules applied to protect 10.1.0.3 on r1-eth0.
Address                  HWtype  HWaddress           Flags Mask            Iface
10.12.0.10               ether   b2:00:92:bf:6c:34   C                     r1-eth12
10.1.0.3                 ether   12:00:b9:40:d3:de   C                     r1-eth0
10.1.0.2                 ether   a2:9b:02:37:6a:98   C                     r1-eth0
10.12.0.2                ether   42:31:73:4b:8a:80   C                     r1-eth12
```

- ARPtables to drop packets that don't match the legitimate IP-MAC mapping using --source-ip --source-mac

TCP SYN SCAN(HTTP SERVER)





```
mininet> http sudo nft -f ~/LINFO2347/syn_protect.nft
mininet> http tshark -i any -f "tcp port 80" -w /tmp/tcp_protect.pcap &
mininet> ws2 python3 tcp_syn_attack.py
/usr/lib/python3/dist-packages/scapy/layers/ipsec.py:469: CryptographyDeprecationWar
ning: Blowfish has been deprecated and will be removed in a future release
  cipher=algorithms.Blowfish,
/usr/lib/python3/dist-packages/scapy/layers/ipsec.py:483: CryptographyDeprecationWar
ning: CAST5 has been deprecated and will be removed in a future release
  cipher=algorithms.CAST5,
Port 80 is filtered (no response)
```

- **This is used to identify open ports on a target system. It is frequently employed during the pre-attack reconnaissance phase to map a network's attack surface.**
- **This focus is on executing this attack on a specific server in the network (HTTP Server)**
- **we will use NFTables to implement protective measure.**
- **The aim of this is to limit the number of SYN packets per time unit, restrict new connections to 10 per minute, which helps prevent rapid scanning, track IPs that send too many SYN packets, and block them for 60 seconds**

MAC/SWITCH FLOOD (CAM TABLE)

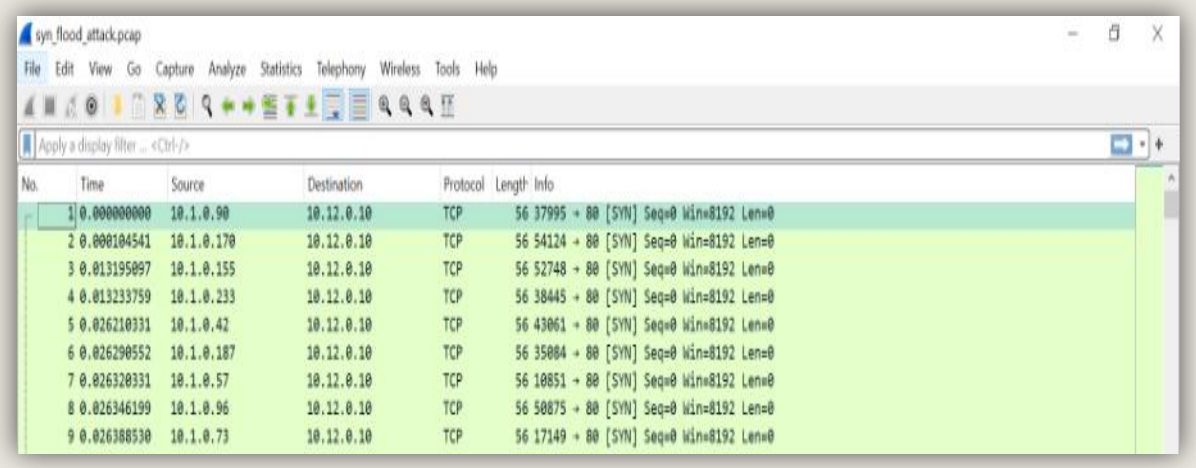
```
mininet> sh ovs-ofctl dump-ports s1
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
mininet> ws2 sudo tshark -i ws2-eth0 -w /var/tmp/ws2_mac.pcap &
mininet> ws3 sudo tshark -i ws3-eth0 -w /var/tmp/ws3_mac.pcap &
mininet> ws2 python3 ~/project/mac_flood.py ws2-eth0 10 &
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ws2-eth0'
mininet> ws3 ping -c 30 10.12.0.10
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ws3-eth0'
3431 PING 10.12.0.10 (10.12.0.10) 56(84) bytes of data.
3464 64 bytes from 10.12.0.10: icmp_seq=1 ttl=62 time=272 ms
3487 64 bytes from 10.12.0.10: icmp_seq=2 ttl=62 time=8.16 ms
3555 64 bytes from 10.12.0.10: icmp_seq=3 ttl=62 time=0.442 ms
4311 64 bytes from 10.12.0.10: icmp_seq=14 ttl=62 time=4.32 ms
4382 64 bytes from 10.12.0.10: icmp_seq=15 ttl=62 time=0.572 ms
5476
--- 10.12.0.10 ping statistics ---
30 packets transmitted, 5 received, 83.3333% packet loss, time 29590ms
rtt min/avg/max/mdev = 0.442/57.004/271.530/107.300 ms
mininet> sh ovs-ofctl dump-ports s1
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=19474, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=10, bytes=812, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=19479, bytes=819798, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=19446, bytes=816732, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=28, bytes=2576, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=33, bytes=3066, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=19456, bytes=817544, drop=0, errs=0, coll=0
mininet> pingall
*** Ping: testing ping reachability
dns -> X X X X X X X X
ftp -> X X X X X X X X
http -> X X X X X X X X
internet -> dns ftp http ntp X X X X
ntp -> X X X X X X X X
r1 -> dns ftp http internet ntp r2 ws2 ws3
r2 -> dns ftp http internet ntp r1 ws2 ws3
ws2 -> dns ftp http internet ntp r1 r2 ws3
ws3 -> dns ftp X X X X r2 ws2
*** Results: 55% dropped (32/72 received)
```

- This is a type of attack that is common in layer 2, The attacker simply fills up the CAM table of a switch with a very large number of ethernet frames.
- This forces the switch in a fail-open mode and acts like a hub.

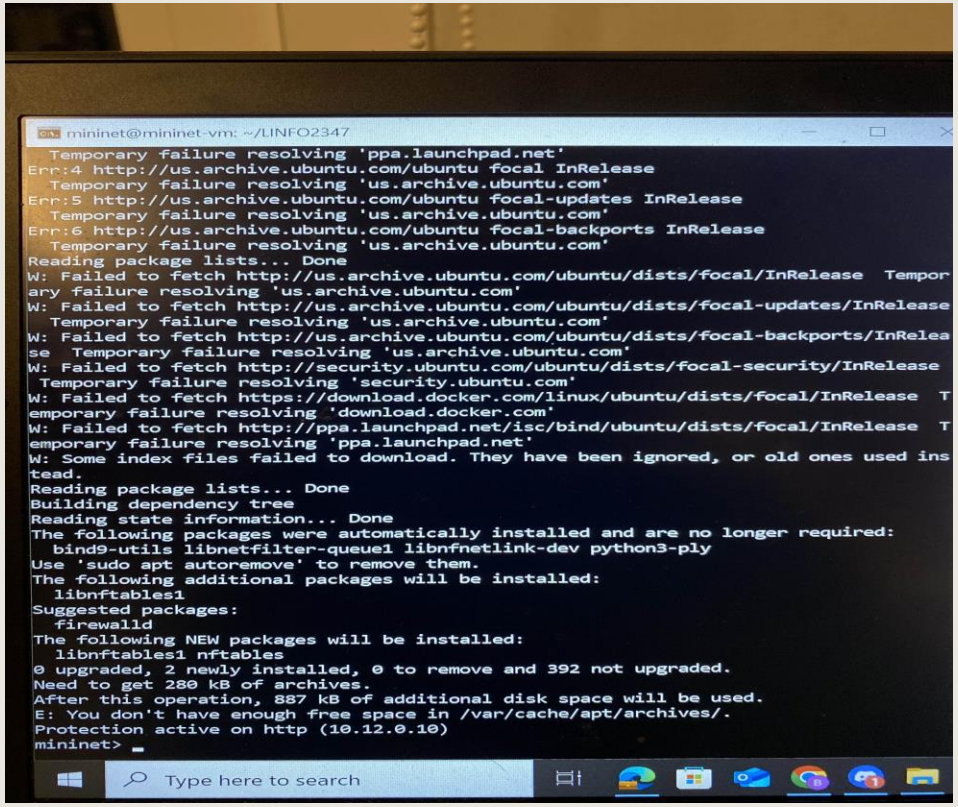
```
mininet> sh ovs-ofctl dump-ports s1
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
mininet> sh bash ~/project/mac_mitigate.sh
[*] Checking for switch s1...
[*] Switch s1 found.
[*] Configuring port security on s1...
[*] Dynamically learning MAC addresses...
[*] Adding flow rules for allowed MACs...
[*] Port security and flow rules configuration applied. Verifying...
{port-security="enabled=true,max-mac=1"}
{port-security="enabled=true,max-mac=1"}
{port-security="enabled=true,max-mac=1"}
[*] Port security mitigation applied successfully on s1.
mininet> ws2 python3 ~/project/mac_flood.py ws2-eth0 10 &
mininet> sh ovs-ofctl dump-ports s1
OFPST_PORT reply (xid=0x2): 4 ports
  port LOCAL: rx pkts=0, bytes=0, drop=952, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth1": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=952, bytes=39984, drop=0, errs=0, coll=0
  port  "s1-eth2": rx pkts=953, bytes=40026, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=0, bytes=0, drop=0, errs=0, coll=0
  port  "s1-eth3": rx pkts=0, bytes=0, drop=0, errs=0, frame=0, over=0, crc=0
           tx pkts=952, bytes=39984, drop=0, errs=0, coll=0
```

- The mitigation intention is to populate the arp_table of the hosts in the subnet of the victim and learn their repsectivc IP-MAC mapping

SYN FLOOD



- **This attack intention is to overwhelm a target system's resources by exploiting the TCP handshake process, rendering it unable to respond to legitimate traffic.**
- **Eventually, the target cannot handle legitimate connections, causing a Denial-of-Service (DoS).**



- **This script mitigates SYN floods and limits HTTP connections while allowing legitimate traffic and logging attacks.**

DHCP_STARVE_SPOOF

278	6.330490658	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x31ef7605
279	6.351315743	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x3bb2290
280	6.372586829	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x2569ce69
281	6.395851923	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x31911c9e
282	6.416541007	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x215be869
283	6.441864110	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0xeb09e6e
284	6.467461214	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x11a974fb
285	6.490124305	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1aacd7de
286	6.511881394	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0xdd426bc
287	6.535274488	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x31396bad
288	6.558397582	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x253a919d
289	6.578808665	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x18ebf683
290	6.605405773	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x6926c24
291	6.625330853	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1f0f5af7
292	6.646928941	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0xee0afd4
293	6.668517028	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x19f348a0
294	6.693953132	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x109ca480
295	6.714801216	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x23f83fc6
296	6.734472296	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0xa60ecf0
297	6.754608377	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x1892199d
298	6.774956460	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0xac21f01
299	6.795789544	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x3612452
300	6.819663641	0.0.0.0	255.255.255.255	DHCP	286	DHCP Discover - Transaction ID 0x3700178
301	7.342200759	8a:21:98:85:c8:2c	Broadcast	ARP	42	Who has 10.1.0.254? Tell 10.1.0.3
302	8.353689859	8a:21:98:85:c8:2c	Broadcast	ARP	42	Who has 10.1.0.254? Tell 10.1.0.3
303	14.388296318	0.0.0.0	255.255.255.255	DHCP	342	DHCP Discover - Transaction ID 0x512ea306
304	14.454220585	10.1.0.254	255.255.255.255	DHCP	316	DHCP Offer - Transaction ID 0x512ea306
305	14.456335593	0.0.0.0	255.255.255.255	DHCP	342	DHCP Request - Transaction ID 0x512ea306
306	14.482658700	10.1.0.254	255.255.255.255	DHCP	316	DHCP ACK - Transaction ID 0x512ea306

- **DHCP is a Network Protocol used to Automatically assign IP Information.**
  - **This attacks aims to flood the dhcp server with bogus DHCP requests and leases all of the available IP addresses.**

```
mininet> ws3 ~/project/mac_mitigate.sh &
mininet> ws2 python3 ~/project/dhcp_starve_spoof.py -i ws2-eth0 &
mininet> ws3 dhclient -r ws3-eth0
[*] Checking for switch s1...
[*] Switch s1 found.
[*] Clearing existing flows on s1...
[*] Dynamically learning MAC addresses...
[*] Configuring port security on s1...
[*] Adding flow rules for IP-MAC binding...
[*] Port security and flow rules configuration applied. Verifying...
{port-security="enabled=true,max-mac=1"}
{port-security="enabled=true,max-mac=1"}
{port-security="enabled=true,max-mac=1"}
 cookie=0x0, duration=0.491s, table=0, n_packets=0, n_bytes=0, priority=1800,dl_dst=ff:ff:ff:ff:ff:ff
 actions=CONTROLLER:10
 cookie=0x0, duration=0.418s, table=0, n_packets=0, n_bytes=0, priority=1700,dl_dst=ff:ff:ff:ff:ff:ff
 actions=drop
 cookie=0x0, duration=0.349s, table=0, n_packets=0, n_bytes=0, priority=1000,arp actions=NORMAL
 cookie=0x0, duration=0.274s, table=0, n_packets=0, n_bytes=0, priority=1000,ip actions=NORMAL
[*] Port security mitigation applied successfully on s1.
Killed old client process
```

This attacks aims to flood the dhcp server with bogus DHCP requests and leases all of the available IP addresses.

THANK YOU

/// 404

THE QUESTION IS: ARE YOU SURE YOU'RE IN THE RIGHT PLACE?

[START OVER]