# Experiment 3: DoS and MITM attack

Eshan
K027

**Aim:** To simulate SYN Flooding attack and ARP poisoning attack

**Learning Outcomes:**
After completion of this experiment, student should be able to

1. Launch SYN flooding attack against a target host 2.
Launch MITM attack using ARP poisoning attack.
3. Understand countermeasures to SYN flooding attack and ARP poisoning attack

**Theory:**

Denial of Service (DoS) is an attack on a computer or network that *prevents* legitimate use of its resources. In a DoS attack, attackers *flood* a victim's system with illegitimate service requests or *traffic* to *overload* its resources and prevent it from performing *intended* tasks. Perpetrators of DoS attacks typically target sites or services hosted on high-profile web servers such as banks, creditcard payment gateways, and even root name servers. One common method of attack involves saturating the target machine with external communications requests, so that it cannot respond to legitimate traffic, or it responds so slowly as to be rendered essentially unavailable. Such attacks usually lead to a server overload. DoS attacks can essentially disable your computer or your network. DoS attacks can be lucrative for criminals; recent attacks have shown that DoS attacks are a way for cyber criminals to profit.

A SYN flood is a form of denial-of-service attack in which an attacker sends a succession of SYN requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic. A SYN flood attack works by not responding to the server with the expected ACK code. The malicious client can either simply not send the expected ACK, or by spoofing the source IP address in the SYN, causing the server to send the SYN-ACK to a falsified IP address—which will not send an ACK because it "knows" that it never sent a SYN. The server will wait for the acknowledgement for some time, as simple network congestion could also be the cause of the missing ACK, but in an attack increasingly large numbers of half-open connections will bind resources on the server until no new connections can be made, resulting in a denial of service to legitimate traffic. Some systems may also malfunction badly or even crash if other operating system functions are starved of resources in this way.

ARP poisoning involves causing a target to associate an IP address with an incorrect MAC address. This involves sending an unprompted ARP message indicating an IP address and the supposed MAC address. This can be used as a DoS attack to cause the target to associate the gateway with the incorrect MAC. Poisoning of the cache can also be done to two targets so each associates the other IP address with the MAC address of the attacker. This can be used in MITM or other session hijacking attacks.

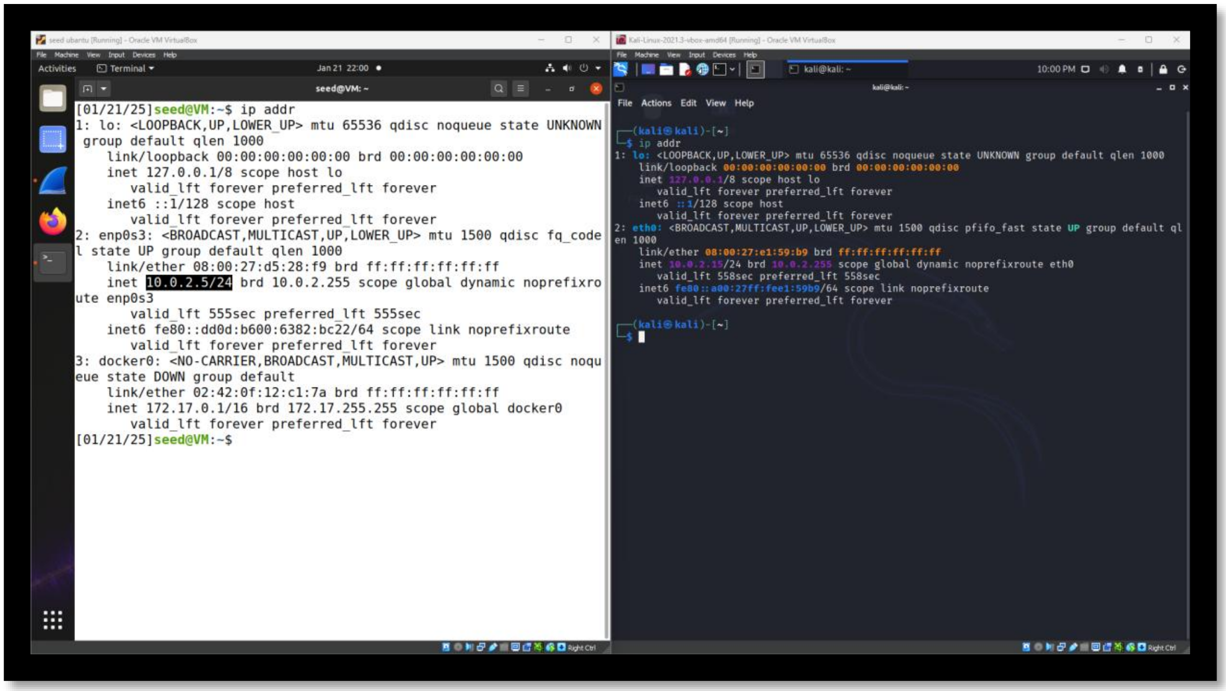
**Procedure:**
**For SYN flooding you need twoVM.**
**For MITM attack you need three VM**

**1. SYN Flooding Attack**
The images included in this section depict the following:

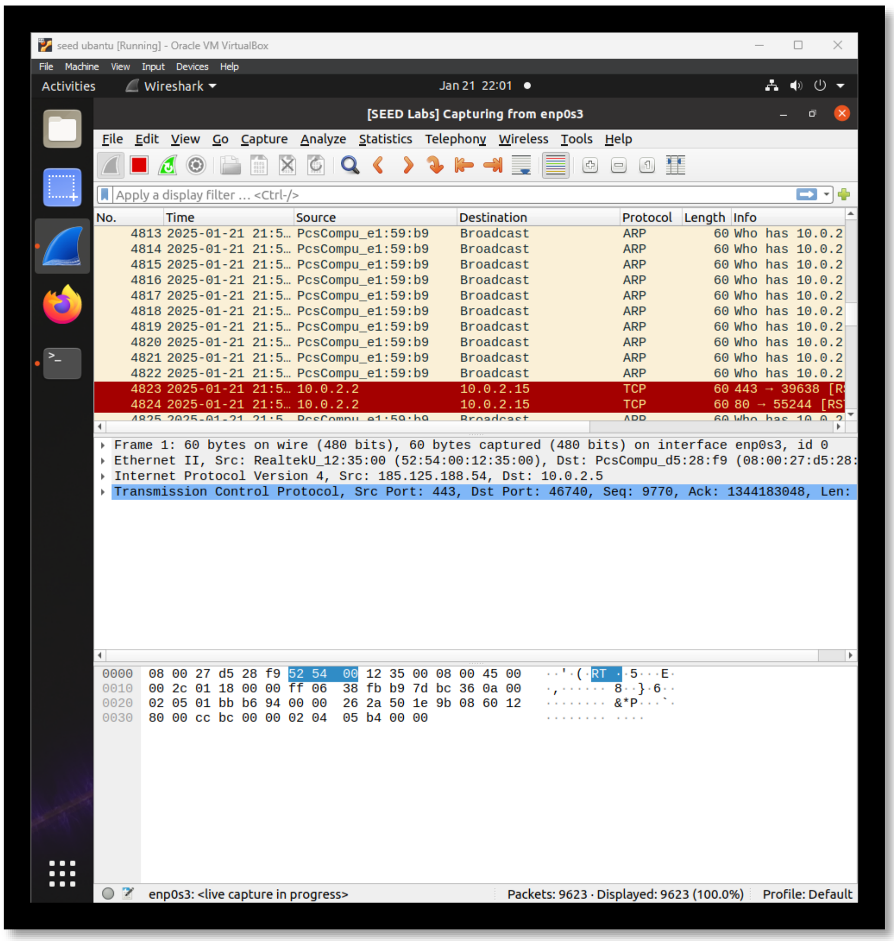
1. **Kali VM Terminal Output:**
   - The nmap command is executed to scan for open ports on the SEEDUbuntu1 VM. The output lists available services and their associated ports.
   - The hping3 command is used to generate a high volume of SYN packets targeting the identified open port.
2. **Wireshark Captures on SEEDUbuntu1:**
   - The network traffic visualization in Wireshark shows an abnormal increase in SYN packets, indicating the impact of the attack.
   - Filtering the traffic by the IP address of the Kali VM highlights the repeated SYN requests without corresponding ACK responses, confirming the flood attack.
3. **SYN Cookie Defense Mechanism:**
   - A terminal screenshot displaying the execution of the sysctl command verifies the status of the SYN cookie mechanism.
   - Comparisons of traffic logs with SYN cookies enabled and disabled demonstrate its effectiveness in mitigating the attack by preventing excessive half-open connections.

**Task 1:** SYN flooding attack

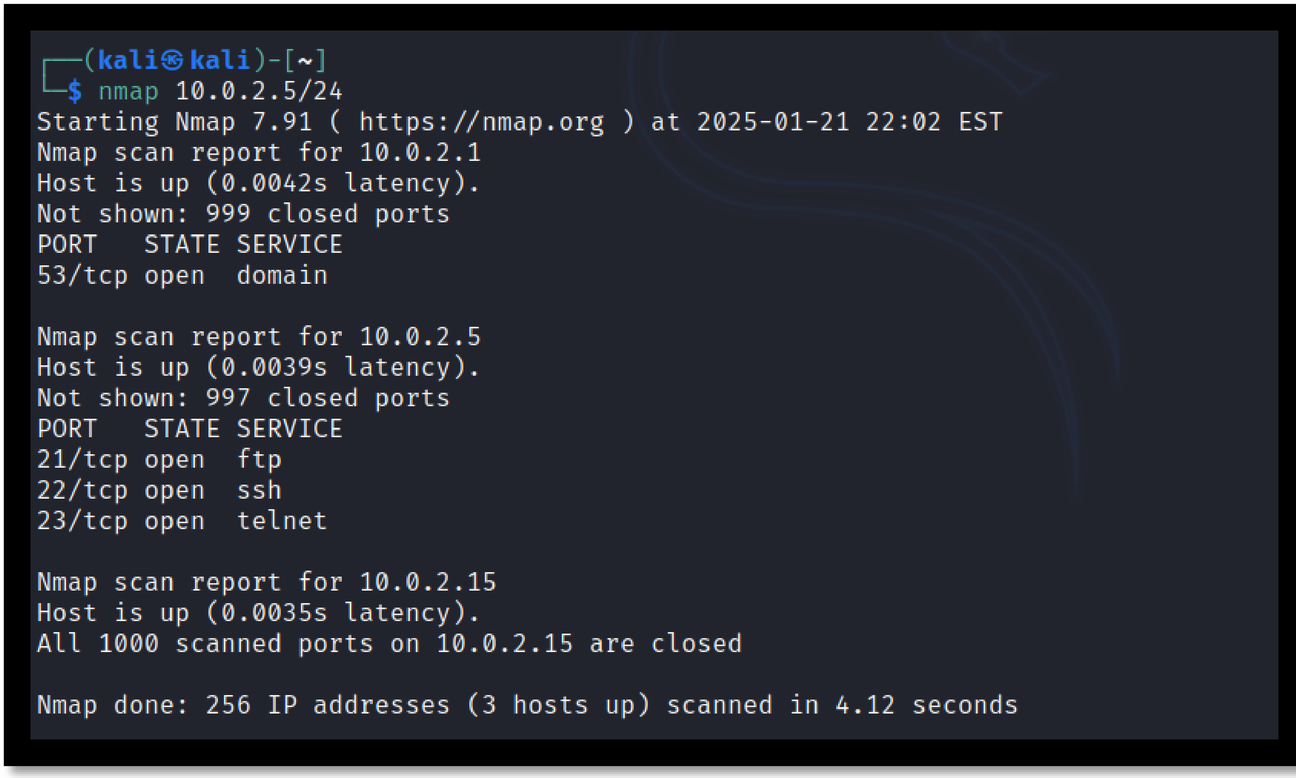
1. Start your kali VM.
2. Login using UN:kali and PW:kali
3. Note IP address of kali VM.
4. Start SEEDUbuntu1 VM.
5. Note IP address of SEEDUbuntu1 VM.




6. Start wire shark on SEEDUbuntu1 VM.



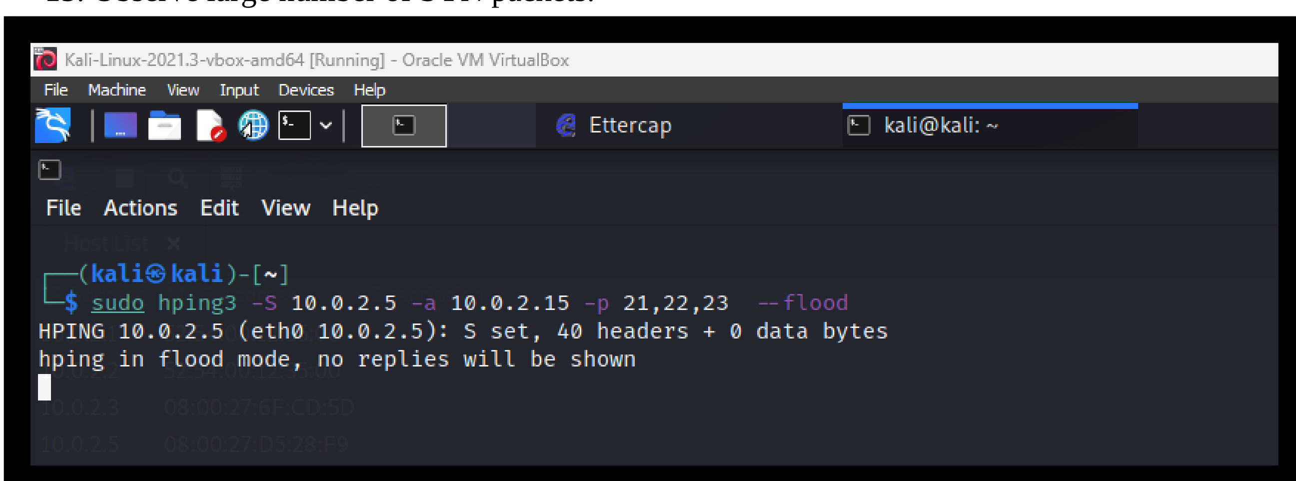
7. Switch to kali VM.

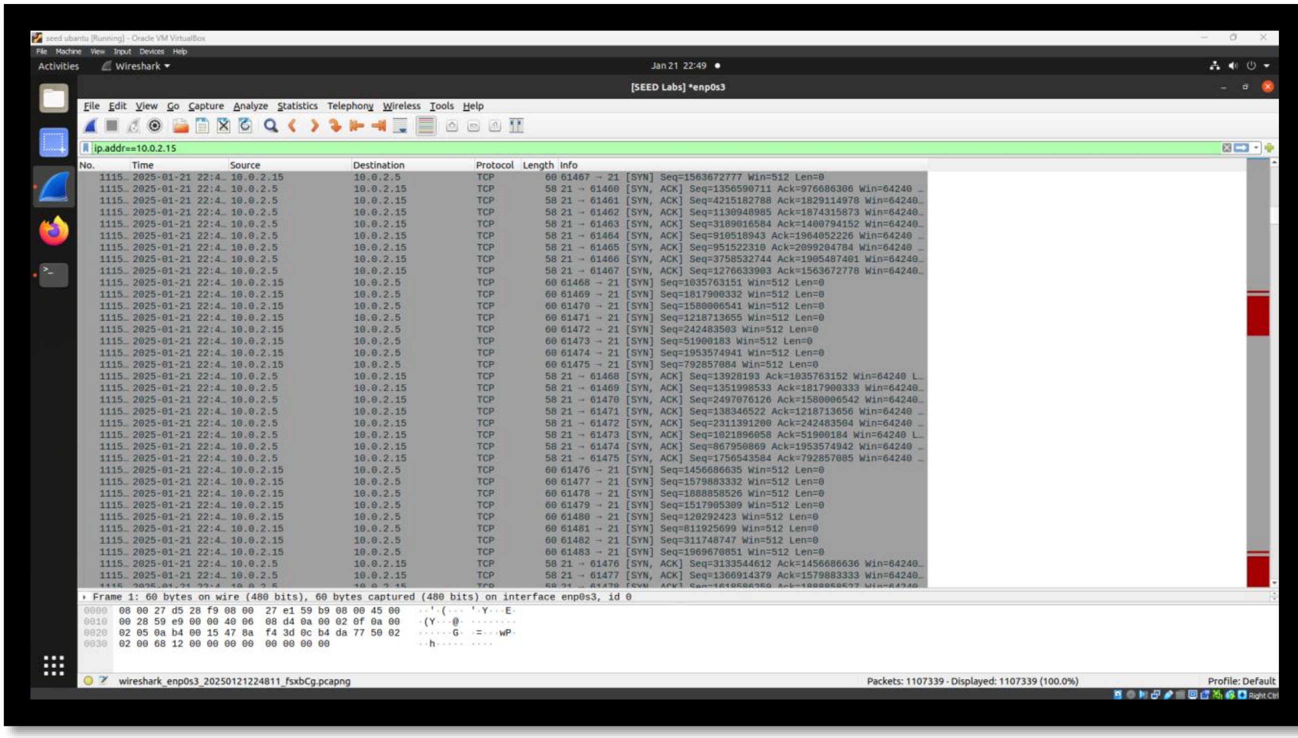



8. On command prompt type following command: nmap [IP of SEEDUbuntu1]
9. Note open port of SEEDUbuntu1
10. On command prompt type the following command hping3 –S [target IP Addr] –a [src IP Addr] –p [open port no.] --flood
11. After sometime stop SYN flooding.
12. Switch to SEEDUbuntu1VM 13.

Stop wire shark.

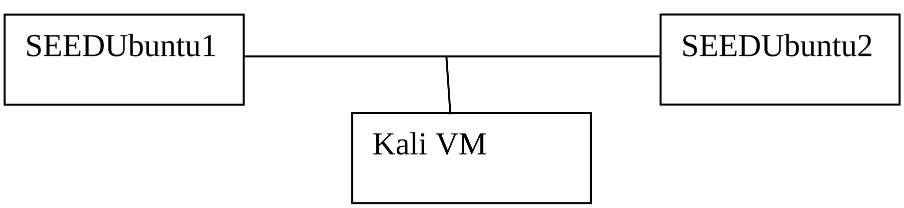
14. Set filter in wire shark to IP addr of kali VM.
15. Observe large number of SYN packets.






16. If your attack seems unsuccessful, one thing that you can investigate is whether the SYN cookie mechanism is turned on. SYN cookie is a defense mechanism to counter the SYN flooding attack. The mechanism will kick in if the machine detects that it is under the SYN flooding attack. You can use the sysctl command to turn on/off the SYN cookie mechanism:
    a. # sysctl -a | grep cookie (Display the SYN cookie flag)
    b. # sysctl -w net.ipv4.tcp_syncookies=0 (turn off SYN cookie)
    c. # sysctl -w net.ipv4.tcp_syncookies=1 (turn on SYN cookie)
    d. Please run your attacks with the SYN cookie mechanism on and off, and compare the results. In your report, please describe why the SYN cookie can effectively protect the machine against the SYN flooding attack.
17. Document all information you collected and comment on your answer.

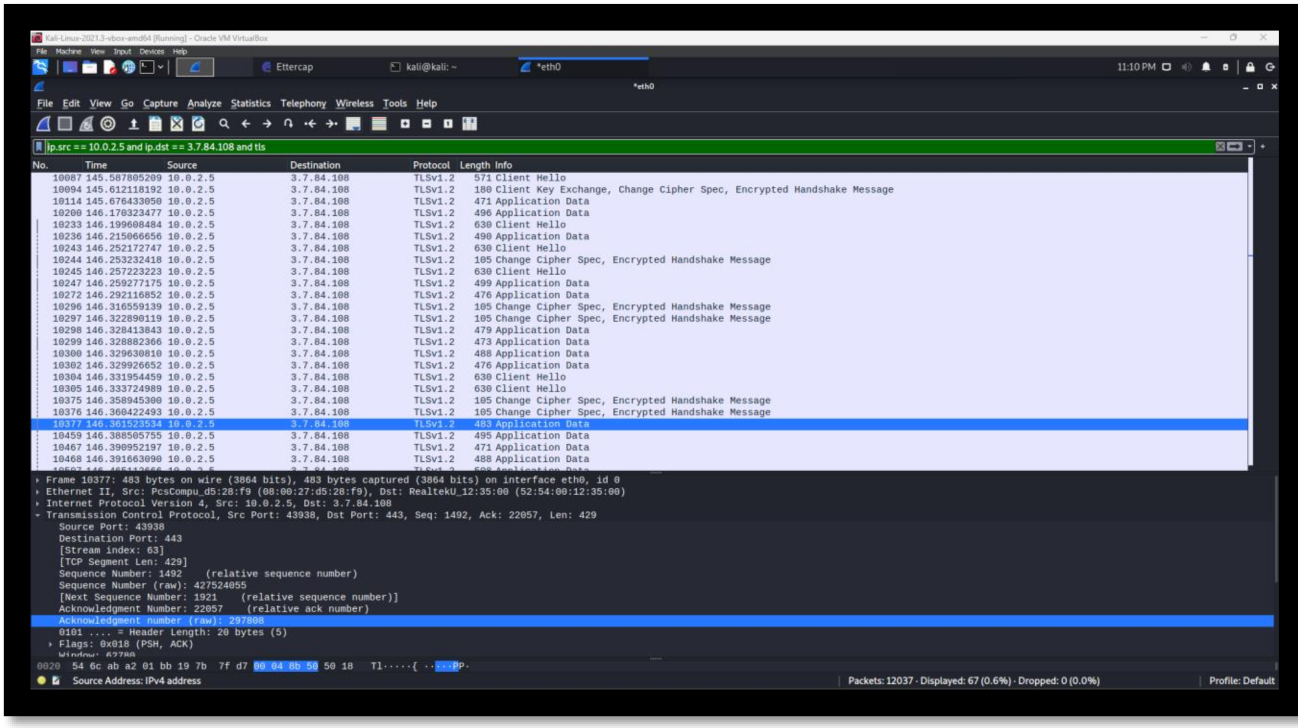
**Task 2: MITM attack**
**Scenario for MITM attack**

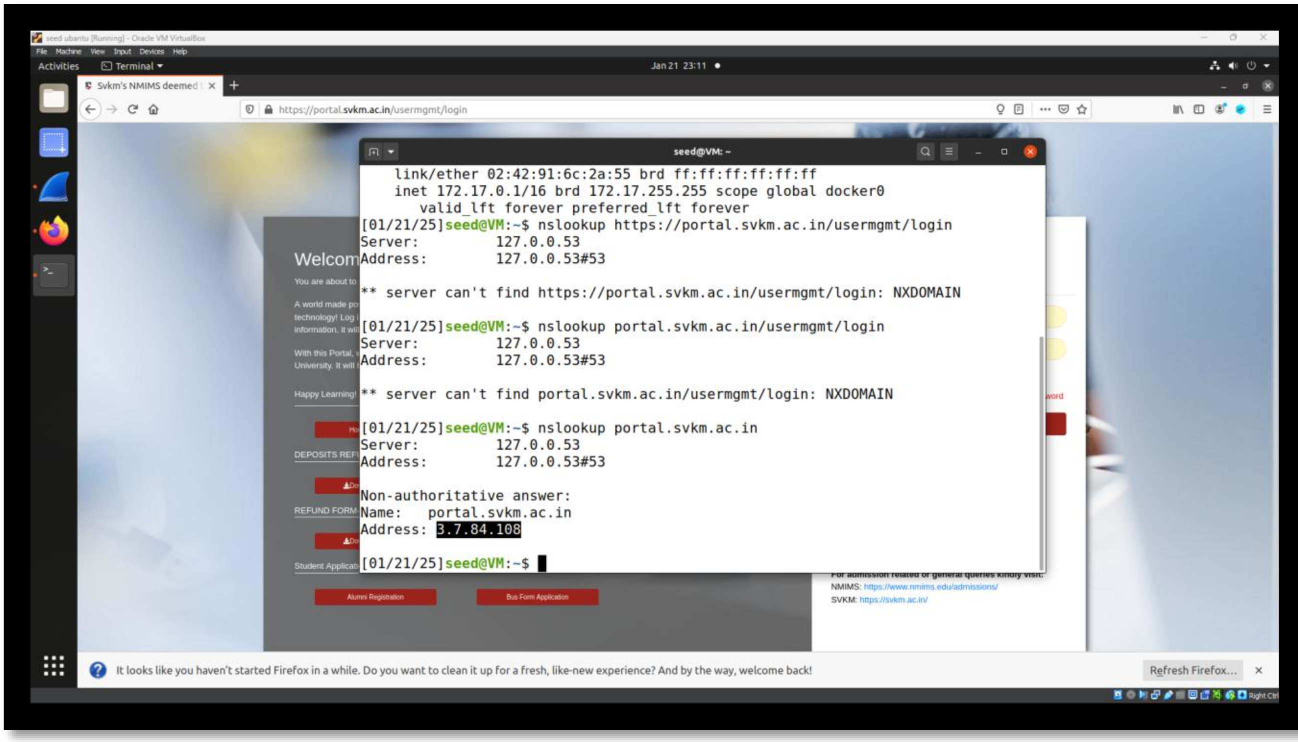



2. Start and login in Kali VM, SEEDUbuntu1 VM and SEEDUbuntu2 VM.
3. Note IP address of each VM and test connectivity using ping.
4. Switch to Kali VM

a. Note connected interface (eth0, eth1 or wlan0 etc.)
b. Go to application →sniffing and spoofing → ettercap.
c. Select unified sniffing and interface.
d. Click host → scan for hosts.
e. Click host → Hosts list
f. Select SEEDUbuntu1 and click Add to target 1
g. Select SEEDUbuntu2 and click Add to target 2
h. Start wireshark on all three VM
i. On ettercap, Click Mitm →ARP poisoning

5. Switch to SEEDUbuntu1 VM
   a. Connect to any website which requires login.






b. Type UN and PW.
c. telnet to SEEDUbuntu 2 VM

6. Switch to Kali VM and view the capture in wireshark
7. Stop live capture on all three VM.
8. Document all your findings.

## 2. MITM Attack Using ARP Poisoning

### Network Configuration and Host Discovery

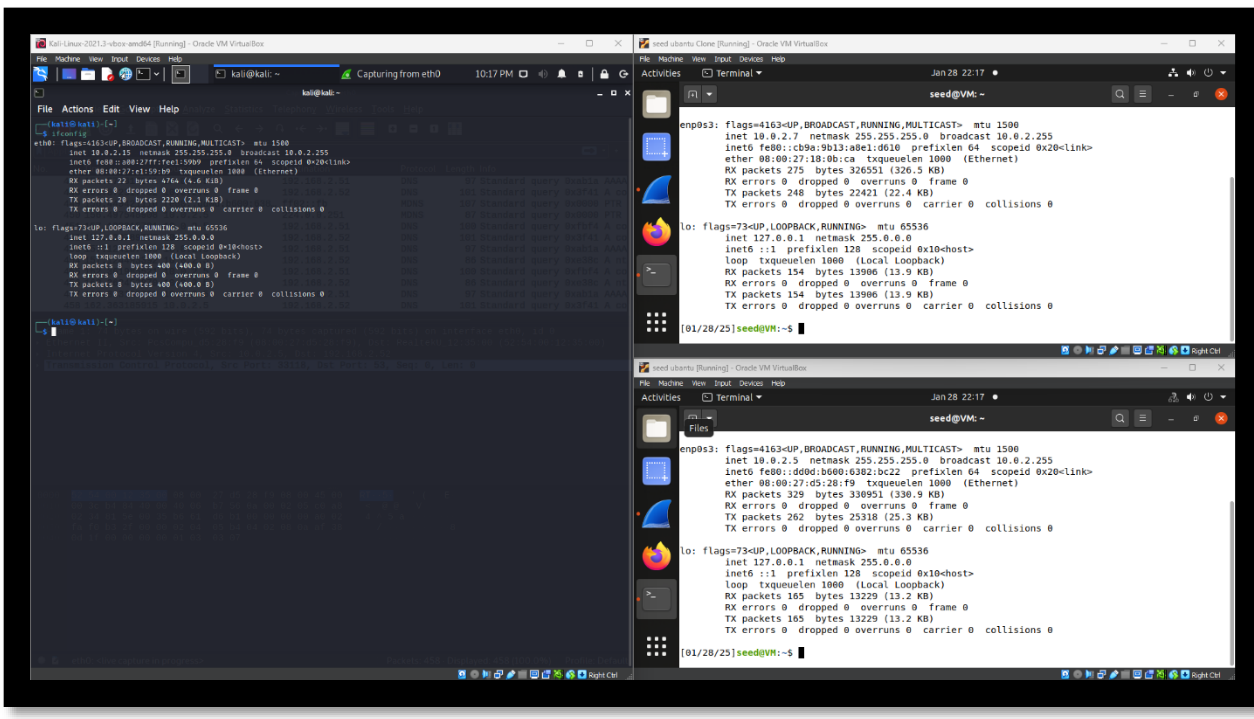
- The **terminal output** from the Kali VM displays the IP addresses of SEEDUbuntu1, SEEDUbuntu2, and Kali VM. This step ensures proper network configuration and connectivity verification before executing the attack.
- The **Ettercap interface** captures the network interfaces available on the attacking machine (e.g., eth0, eth1, or wlan0). Selecting the correct interface ensures proper sniffing and poisoning.
- The **host scanning process in Ettercap** discovers live hosts within the network. The **host list** in Ettercap confirms the presence of SEEDUbuntu1 and SEEDUbuntu2, verifying that the attacker (Kali VM) can reach them.

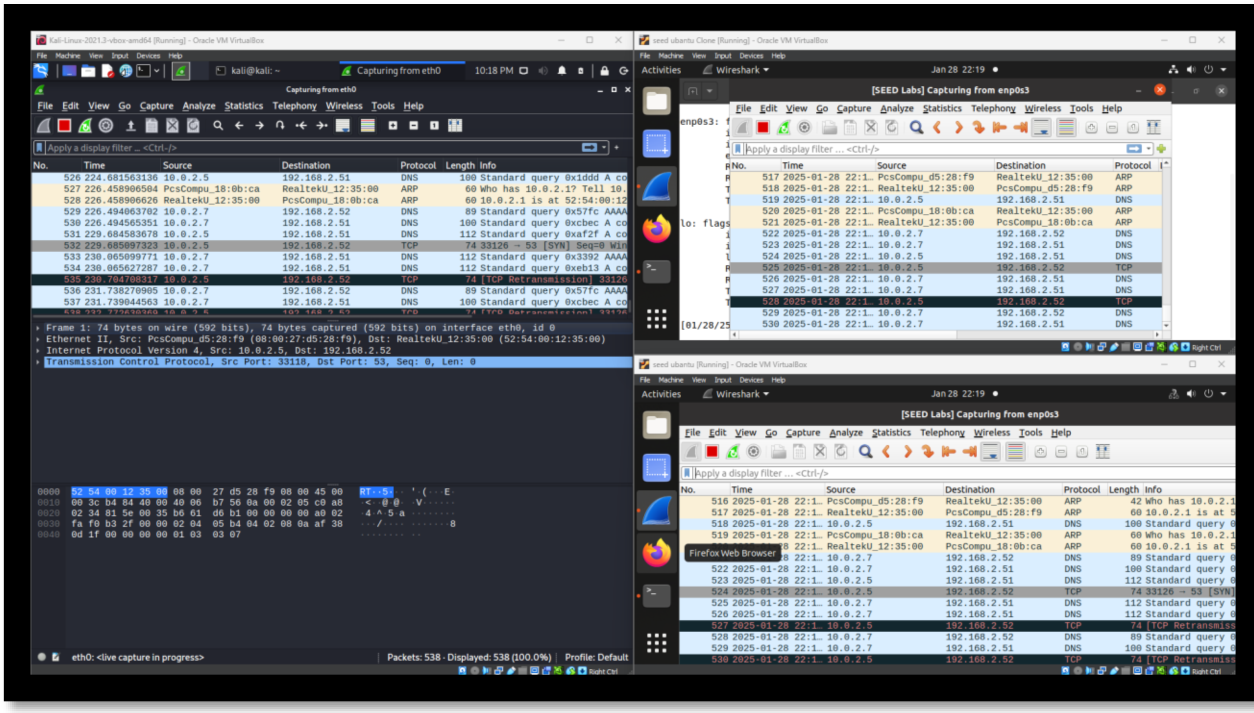
### ARP Poisoning Execution

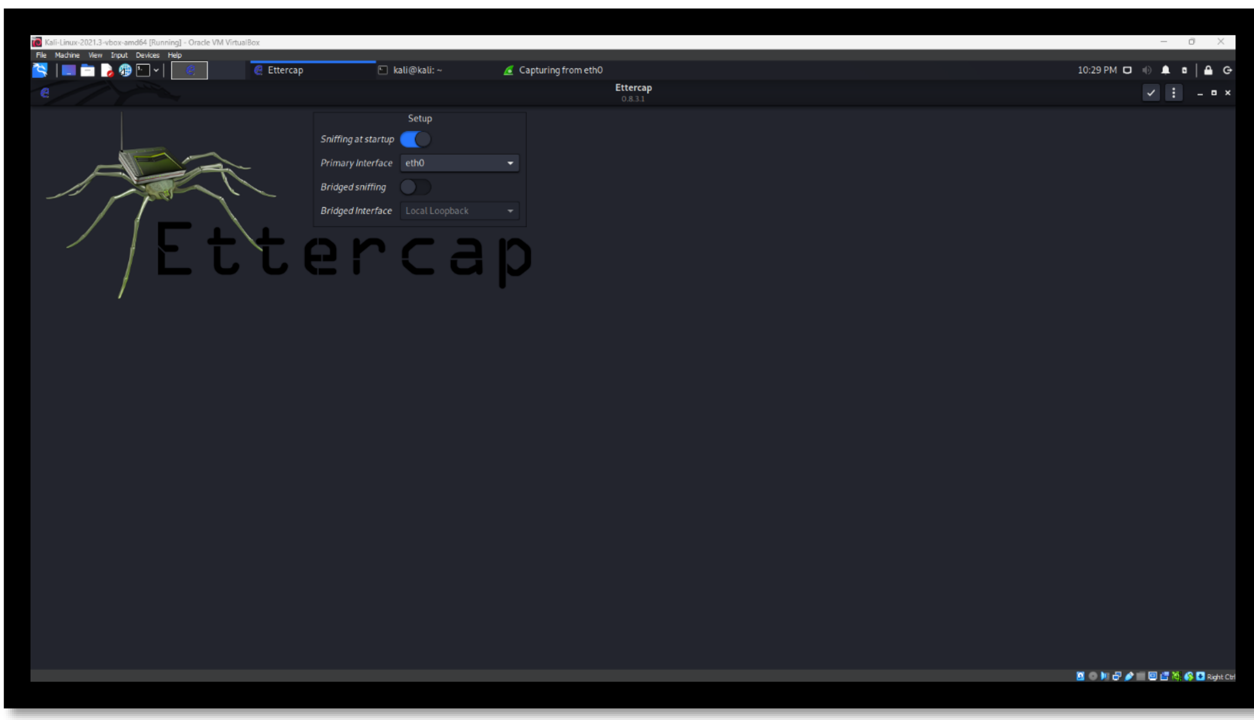
- The **Ettercap interface** allows the attacker to assign SEEDUbuntu1 as **Target 1** and SEEDUbuntu2 as **Target 2**. This setup positions the attacker between the two hosts, making it possible to intercept and manipulate their communication.
- The **logs in Ettercap** confirm the execution of the ARP poisoning attack by showing continuous ARP reply messages being sent to SEEDUbuntu1 and SEEDUbuntu2. These forged ARP packets convince both machines to associate the attacker's MAC address with the IP address of the legitimate machine.

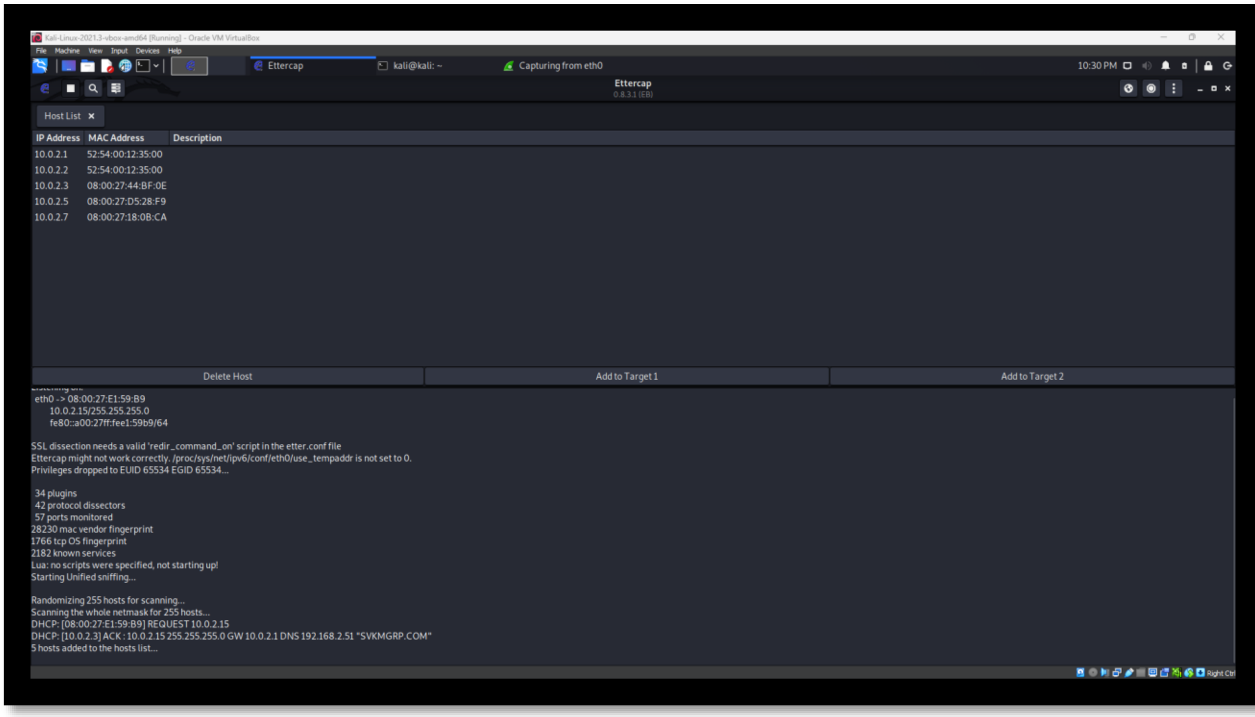
**Traffic Interception and Credential Sniffing**

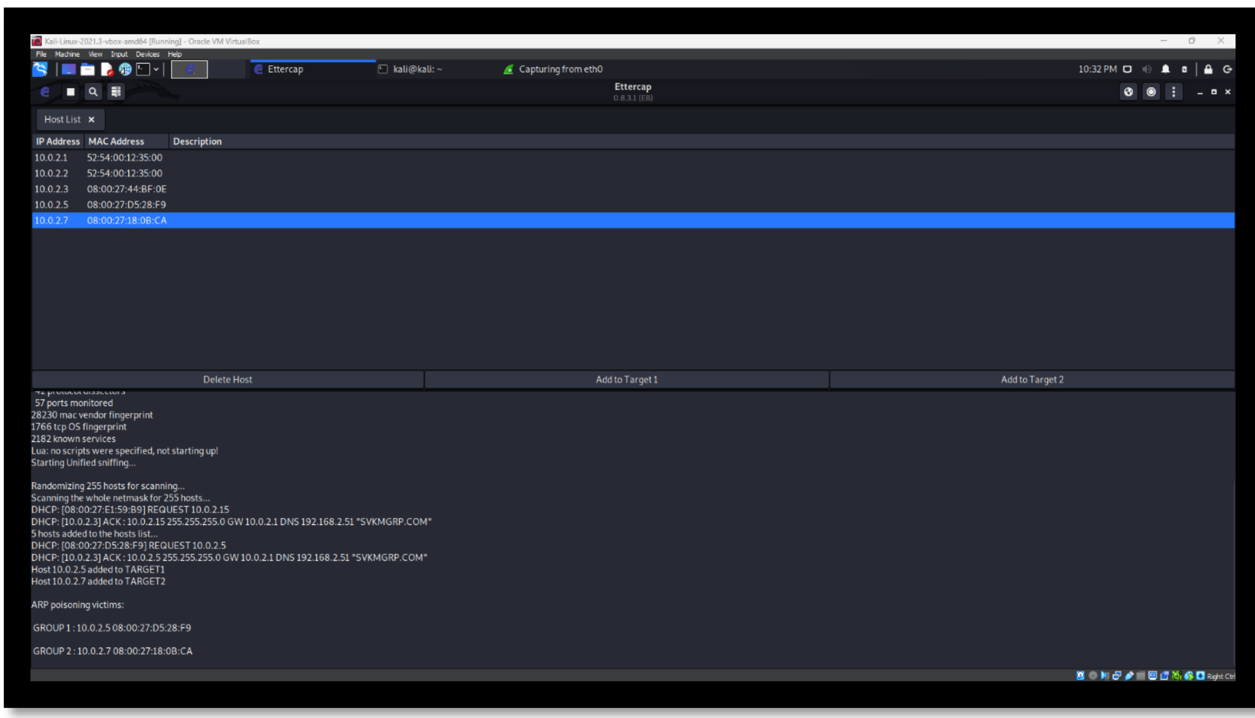
- **Wireshark captures on Kali VM** reveal the interception of credentials when SEEDUbuntu1 attempts to log in to a website. The captured packets contain HTTP POST requests, exposing usernames and passwords in plaintext when no encryption is used.
- The **Telnet session between SEEDUbuntu1 and SEEDUbuntu2** shows cleartext communication, making it vulnerable to sniffing. The intercepted traffic demonstrates the attacker's ability to capture authentication credentials and other sensitive information in real-time.

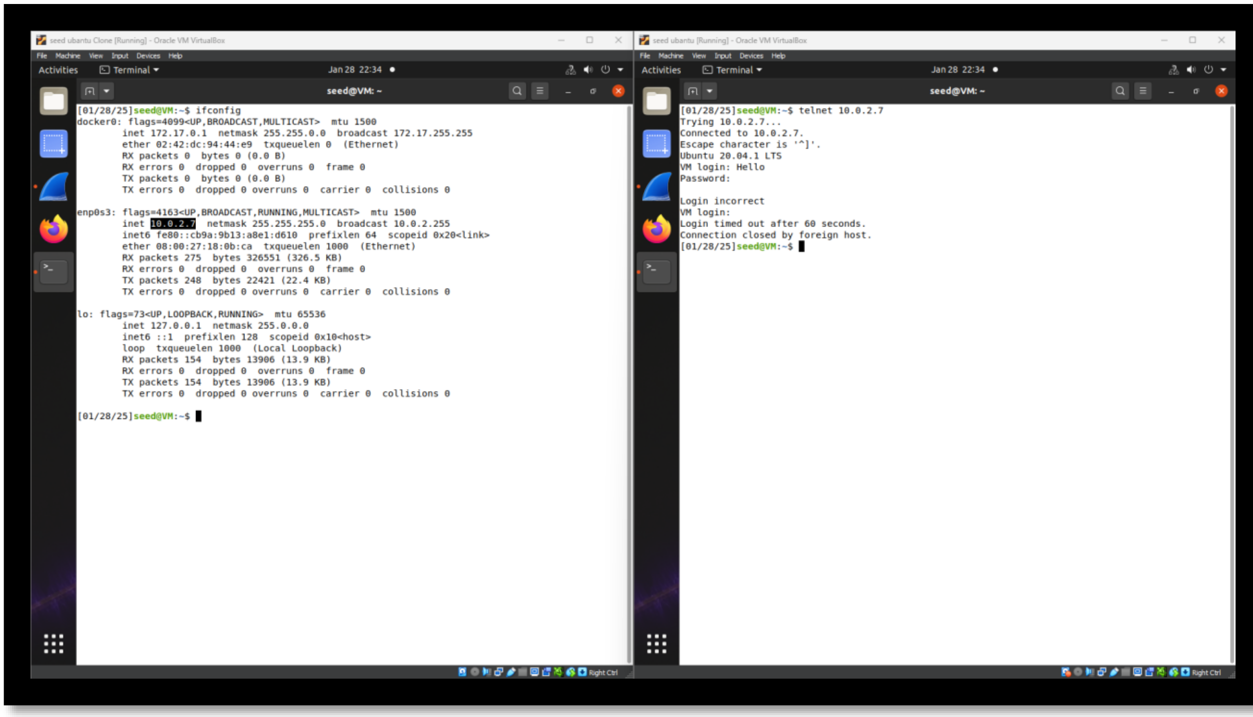



Capturing from eth0
[SEED Labs] Capturing from enp0s3
Apply a display filter ... <Ctrl-/>
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_d5:28:f9 (08:00:27:d5:28:f9), Dst: RealtekU_12:35:00 (52:54:00:12:35:00)
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 192.168.2.52
Transmission Control Protocol, Src Port: 33118, Dst Port: 53, Seq: 0, Len: 0
eth0: <live capture in progress>
Packets: 538 · Displayed: 538 (100.0%)
Profile: Default
Firefox Web Browser

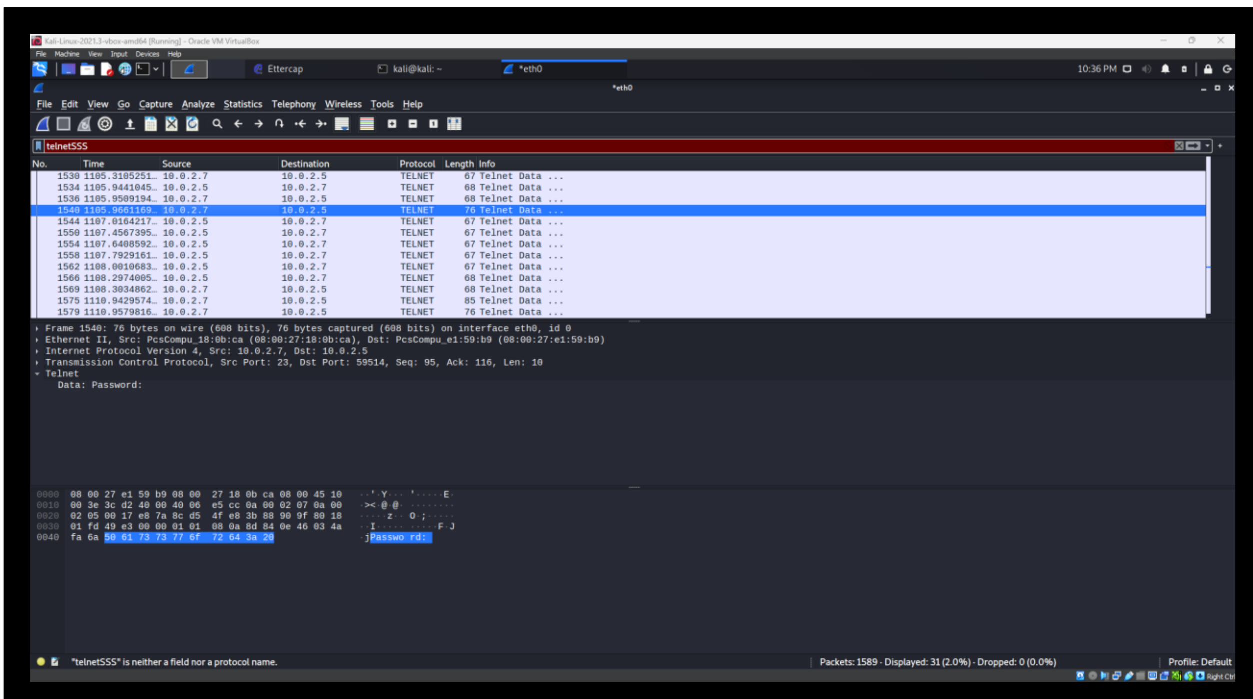

Ettercap
Setup
Sniffing at startup
Primary Interface
eth0
Bridged sniffing
Bridged Interface
Local Loopback
Capturing from eth0
kali@kali: ~
10:29 PM


Ettercap
kali@kali: ~
Capturing from eth0
10:30 PM
Host List
IP Address MAC Address Description
10.0.2.1 52:54:00:12:35:00
10.0.2.2 52:54:00:12:35:00
10.0.2.3 08:00:27:44:BF:0E
10.0.2.5 08:00:27:D5:28:F9
10.0.2.7 08:00:27:18:0B:CA
Delete Host
Add to Target 1
Add to Target 2
eth0 -> 08:00:27:E1:59:B9
10.0.2.15/255.255.255.0
fe80::a00:27ff:fee1:59b9/64
SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Ettercap might not work correctly. /proc/sys/net/ipv6/conf/eth0/use_tempaddr is not set to 0.
Privileges dropped to EUID 65534 EGID 65534...
34 plugins
42 protocol dissectors
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
DHCP: [08:00:27:E1:59:B9] REQUEST 10.0.2.15
DHCP: [10.0.2.3] ACK : 10.0.2.15 255.255.255.0 GW 10.0.2.1 DNS 192.168.2.51 "SVKMGRP.COM"
5 hosts added to the hosts list...

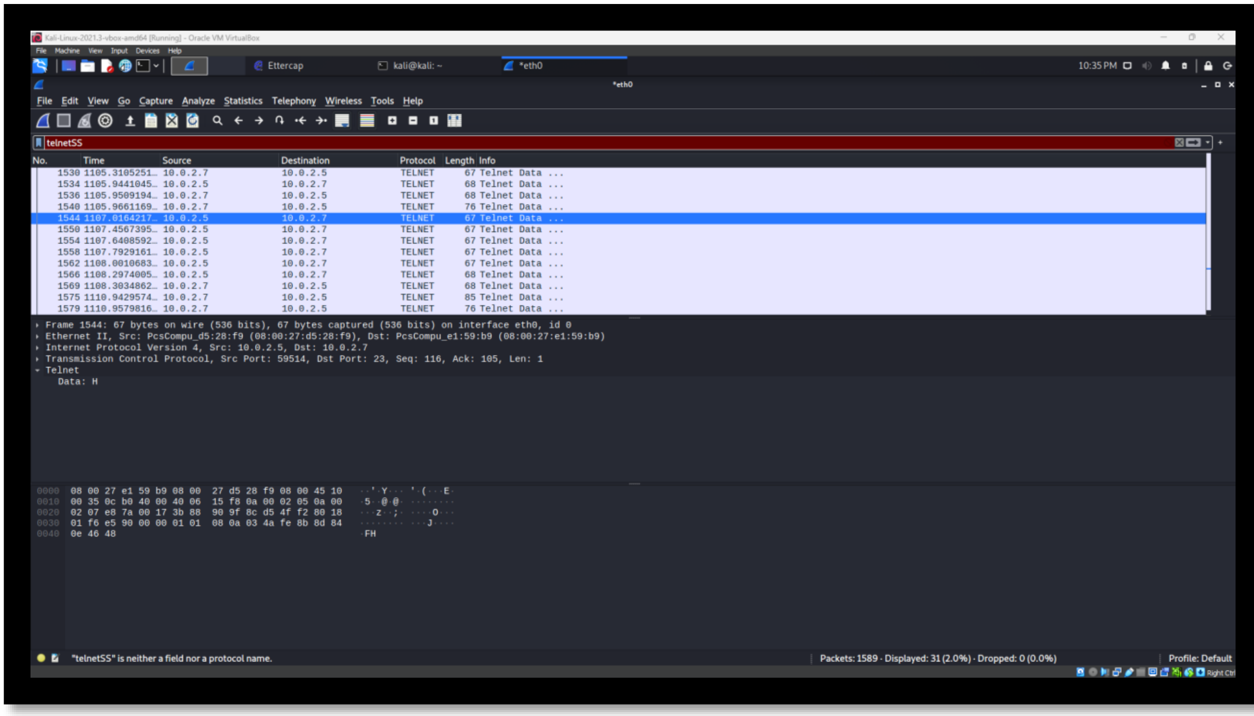

Ettercap
kali@kali: ~
Capturing from eth0
10:32 PM
Host List
IP Address MAC Address Description
10.0.2.1 52:54:00:12:35:00
10.0.2.2 52:54:00:12:35:00
10.0.2.3 08:00:27:44:BF:0E
10.0.2.5 08:00:27:D5:28:F9
10.0.2.7 08:00:27:18:0B:CA
Delete Host
Add to Target 1
Add to Target 2
57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!
Starting Unified sniffing...
Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
DHCP: [08:00:27:E1:59:B9] REQUEST 10.0.2.15
DHCP: [10.0.2.3] ACK : 10.0.2.15 255.255.255.0 GW 10.0.2.1 DNS 192.168.2.51 "SVKMGRP.COM"
5 hosts added to the hosts list...
DHCP: [08:00:27:D5:28:F9] REQUEST 10.0.2.5
DHCP: [10.0.2.3] ACK : 10.0.2.5 255.255.255.0 GW 10.0.2.1 DNS 192.168.2.51 "SVKMGRP.COM"
Host 10.0.2.5 added to TARGET1
Host 10.0.2.7 added to TARGET2
ARP poisoning victims:
GROUP 1 : 10.0.2.5 08:00:27:D5:28:F9
GROUP 2 : 10.0.2.7 08:00:27:18:0B:CA

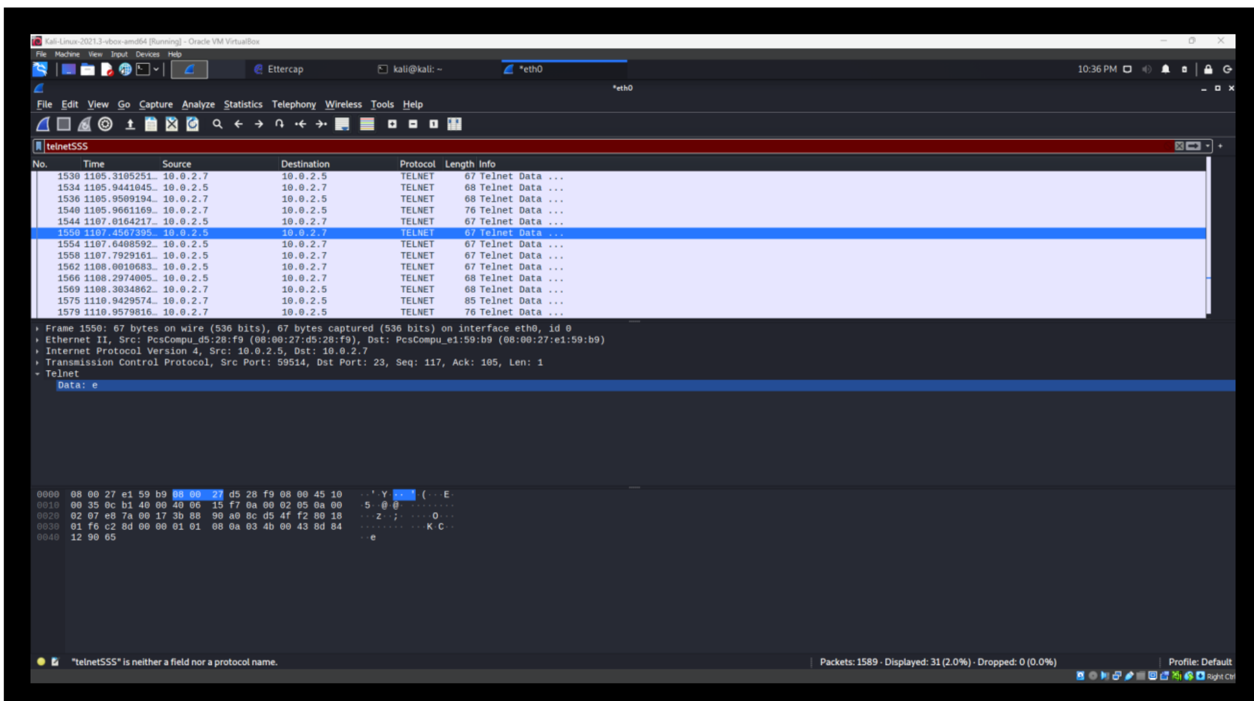


[01/28/25]seed@VM:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
ether 02:42:dc:94:44:e9  txqueuelen 0  (Ethernet)
RX packets 0  bytes 0 (0.0 B)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 0  bytes 0 (0.0 B)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 10.0.2.7  netmask 255.255.255.0  broadcast 10.0.2.255
inet6 fe80::cb9a:9b13:a8e1:d610  prefixlen 64  scopeid 0x20<link>
ether 08:00:27:18:0b:ca  txqueuelen 1000  (Ethernet)
RX packets 275  bytes 326551 (326.5 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 248  bytes 22421 (22.4 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
inet 127.0.0.1  netmask 255.0.0.0
inet6 ::1  prefixlen 128  scopeid 0x10<host>
loop  txqueuelen 1000  (Local Loopback)
RX packets 154  bytes 13906 (13.9 KB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 154  bytes 13906 (13.9 KB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
[01/28/25]seed@VM:~$
[01/28/25]seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: Hello
Password:
Login incorrect
VM login:
Login timed out after 60 seconds.
Connection closed by foreign host.
[01/28/25]seed@VM:~$

Ettercap
kali@kali: ~
*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
telnetSSS
No. Time Source Destination Protocol Length Info
1530 1105.3105251… 10.0.2.7 10.0.2.5 TELNET 67 Telnet Data ...
1534 1105.9441045… 10.0.2.5 10.0.2.7 TELNET 68 Telnet Data ...
1536 1105.9509194… 10.0.2.7 10.0.2.5 TELNET 68 Telnet Data ...
1540 1105.9661169… 10.0.2.7 10.0.2.5 TELNET 76 Telnet Data ...
1544 1107.0164217… 10.0.2.5 10.0.2.7 TELNET 67 Telnet Data ...
1550 1107.4567395… 10.0.2.5 10.0.2.7 TELNET 67 Telnet Data ...
1554 1107.6408592… 10.0.2.5 10.0.2.7 TELNET 67 Telnet Data ...
1558 1107.7929161… 10.0.2.5 10.0.2.7 TELNET 67 Telnet Data ...
1562 1108.0010683… 10.0.2.5 10.0.2.7 TELNET 67 Telnet Data ...
1566 1108.2974005… 10.0.2.5 10.0.2.7 TELNET 68 Telnet Data ...
1569 1108.3034862… 10.0.2.7 10.0.2.5 TELNET 68 Telnet Data ...
1575 1110.9429574… 10.0.2.7 10.0.2.5 TELNET 85 Telnet Data ...
1579 1110.9579816… 10.0.2.7 10.0.2.5 TELNET 76 Telnet Data ...
Frame 1540: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_18:0b:ca (08:00:27:18:0b:ca), Dst: PcsCompu_e1:59:b9 (08:00:27:e1:59:b9)
Internet Protocol Version 4, Src: 10.0.2.7, Dst: 10.0.2.5
Transmission Control Protocol, Src Port: 23, Dst Port: 59514, Seq: 95, Ack: 116, Len: 10
Telnet
Data: Password:
"telnetSSS" is neither a field nor a protocol name.
Packets: 1589 · Displayed: 31 (2.0%) · Dropped: 0 (0.0%)
Profile: Default


Ettercap
kali@kali: ~
*eth0
10:35 PM
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
telnetSS
No. Time Source Destination Protocol Length Info
Frame 1544: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_d5:28:f9 (08:00:27:d5:28:f9), Dst: PcsCompu_e1:59:b9 (08:00:27:e1:59:b9)
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.7
Transmission Control Protocol, Src Port: 59514, Dst Port: 23, Seq: 116, Ack: 105, Len: 1
Telnet
Data: H
"telnetSS" is neither a field nor a protocol name.
Packets: 1589 · Displayed: 31 (2.0%) · Dropped: 0 (0.0%)
Profile: Default


Ettercap
kali@kali: ~
*eth0
10:36 PM
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
telnetSSS
No. Time Source Destination Protocol Length Info
Frame 1550: 67 bytes on wire (536 bits), 67 bytes captured (536 bits) on interface eth0, id 0
Ethernet II, Src: PcsCompu_d5:28:f9 (08:00:27:d5:28:f9), Dst: PcsCompu_e1:59:b9 (08:00:27:e1:59:b9)
Internet Protocol Version 4, Src: 10.0.2.5, Dst: 10.0.2.7
Transmission Control Protocol, Src Port: 59514, Dst Port: 23, Seq: 117, Ack: 105, Len: 1
Telnet
Data: e
"telnetSSS" is neither a field nor a protocol name.
Packets: 1589 · Displayed: 31 (2.0%) · Dropped: 0 (0.0%)
Profile: Default

**Review question:**

1. **<u>Explain countermeasures for SYN flooding attack.</u>**

1. **SYN Cookies:**
   - The SYN cookie mechanism encodes the TCP connection state within the **SYN-ACK response**, preventing excessive resource allocation for half-open connections.
   - When the legitimate client replies with an ACK, the server reconstructs the missing state and establishes the connection. This mechanism ensures that the system remains responsive even under attack.
2. **Rate Limiting:**
   - Implementing rate limiting controls the number of incoming SYN requests per second.
   - Firewalls and intrusion prevention systems (IPS) can **throttle or block excessive SYN requests** originating from a single source.
3. **Firewall Rules:**
   - Configuring firewall rules to **drop or delay SYN packets** from suspicious or repetitive sources helps reduce the impact of the attack.
   - Stateful firewalls can **detect abnormal connection attempts** and block attack traffic dynamically.
4. **Intrusion Detection and Prevention Systems (IDS/IPS):**
   - **Signature-based IDS solutions like Snort** can detect SYN flooding patterns and alert administrators.
   - **Anomaly-based IDS solutions** identify deviations from normal traffic behavior and mitigate attacks in real time.
5. **Increasing Backlog Queue and Timeout Values:**
   - Adjusting the system's TCP backlog queue to **temporarily accommodate more half-open connections** helps mitigate the impact of the attack.
   - Lowering the timeout for half-open connections ensures that malicious requests are dropped faster, freeing resources for legitimate users.

2. **<u>Explain countermeasures for ARP poisoning attack.</u>**

- **Static ARP Entries:**

  - Configuring **static ARP entries** on critical systems ensures that ARP mappings are not altered dynamically.
  - This is particularly effective for securing **servers, routers, and gateway devices** where MAC-IP bindings should remain constant.

- **ARP Spoofing Detection Tools:**
    - Tools such as **Arpwatch, XArp, and Wireshark** continuously monitor the ARP cache and detect suspicious modifications.
    - These tools can generate alerts when duplicate MAC addresses or inconsistent ARP replies are observed.
- **Packet Filtering and Dynamic ARP Inspection (DAI):**
    - Layer 2 security mechanisms such as **Dynamic ARP Inspection (DAI)** on switches validate ARP requests and responses before forwarding them.
    - DAI enforces MAC-IP bindings, ensuring that only authorized ARP replies are processed.
- **Port Security on Switches:**
    - Enabling **port security** on network switches restricts the number of allowed MAC addresses per port.
    - If an attacker attempts to use ARP poisoning to impersonate another device, the switch can block or shut down the compromised port.
- **Encryption (TLS/SSL):**
    - Even if ARP poisoning is successful, using **end-to-end encryption** prevents attackers from reading intercepted data.
    - Implementing **HTTPS instead of HTTP** ensures that credentials are encrypted, making them useless if captured by a Man-in-the-Middle attacker.

3. **Explain other MITM attacks possible using etttercap.**

- **DNS Spoofing Attack:**
    - Ettercap can modify **DNS responses** by intercepting queries and returning false IP addresses.
    - This technique redirects victims to **malicious websites** controlled by the attacker, often used for phishing attacks.
- **SSL Stripping Attack:**
    - By forcing **downgrades from HTTPS to HTTP**, attackers can intercept **plaintext credentials** during login attempts.
    - Ettercap can modify HTTP headers and prevent redirection to secure HTTPS connections.

- **ICMP Redirect Attack:**
  - The attacker sends **false ICMP redirect messages**, tricking the victim into using the attacker's system as the preferred gateway.
  - This method enables long-term traffic interception and manipulation.
- **Packet Injection and Session Hijacking:**
  - Ettercap allows attackers to **inject malicious packets** into an active session.
  - If a user is logged into a session (e.g., a bank or email account), attackers can hijack and execute unauthorized actions on behalf of the victim.
- **VoIP Eavesdropping:**
  - Attackers can intercept and decode **VoIP (Voice over IP) calls**, allowing real-time conversation monitoring.
  - Ettercap plugins enable capturing **SIP and RTP packets**, reconstructing voice data for playback.