

ARP Poisoning and Network Traffic Capturing

- 1. Explain the utility of the following settings / commands used in these labs. Provide clear, concise answers with one example scenario highlighting their significance.**

- a. Promiscuous mode**

Capture/view all traffic capture on a network segment. This is useful for network traffic monitoring as well as identifying attacks. If an attacker was using an alternate MAC address to transmit on a network the promiscuous code will capture that traffic to identify the attack.

- b. IP forward field**

Identifies whether a device enables the transmission of packets across its interfaces which is important for interconnecting multiple networks. This is critical for network routing as well as connectivity. Enabling IP forwarding on a home router will allow connected devices to access internet as well as communicate with other networks.

- c. Arpwatch**

Keeps an eye on the network actions and spots ARP spoofing threats as well as watching over MAC and IP addresses on a network, notifying network admins of any irregularities in the mappings. For example, if there is a MAC address associated with an IP address that has not been seen before on the network, arpwatch will catch this and alert network administrator.

- d. Urlsnarf**

Watches web activity on a network and records HTTP requests/responses in plain text which it then logs. This helps administrators to identify a possible security breach or improper network usage in real-time, such as users accessing phishing sites or unauthorized access to restricted sites at a workplace.

- e. Tcpdump**

Is a packet analyzer that can intercept/display packets in a network. Used for fixing issues as well as investigate security concerns. If a user has trouble accessing a website, tcpdump will capture to then review the network traffic between the website and the user's device. This will then allow the admin to analyze the captured data to pinpoint the issue.

- f. Netstat**

Shows current network status and stats. Helps keep an eye on network behavior as well as security threats and risks. Can be used to check active network connections, for example, an admin can use netstat to check open ports on a device, this then allows them to detect security threats such as unauthorized access attempts which the admin can investigate to act against the connection by blocking it and removing malware if present.

Packet Crafting with Scapy

1. Explain the utility of the following settings/commands used in this lab concisely.

a. TTL

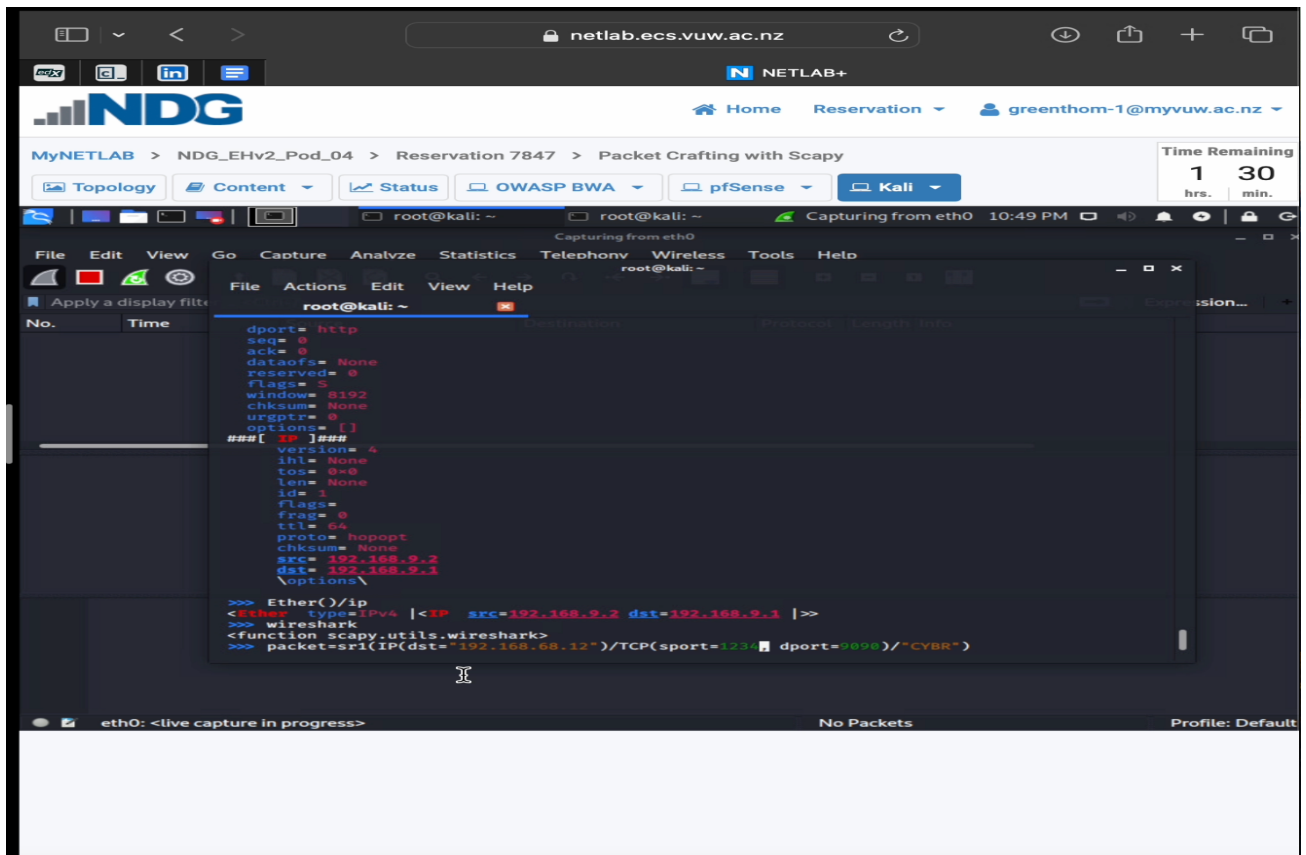
Shows the maximum number of hops a packet can make before it's discarded, preventing packets from indefinitely circulating in the network. It is handy for network diagnostics as well as monitoring, helping spot network congestion, high latency or malicious activity.

b. Operator

Utilized for layer stacking where upper layer information is overlaid to the lower layer's default fields. When sending a TCP packet over an IP packet within a network, promiscuous mode enables a network device to intercept and read each arriving network packet. This enables build of custom packets with control over parameters and protocol headers in each layer.

2. Write the commands in Scapy for the following. Please make sure your Scapy code matches the requirements in each task. Partially correct answers are not accepted.

a. Create and send a TCP packet with the payload "CYBER" with the source port of 1234 from the Kali host, and to OWASP BWA host on destination port of 9090.



Take a screenshot of the tcpdump capture on the destination, confirming your packet was delivered.

The screenshot shows the NDG MyNETLAB interface with the following details:

- Navigation:** MyNETLAB > NDG_EHv2_Pod_03 > Reservation 7904 > Packet Crafting with Scapy
- Tools:** Topology, Content, Status, OWASP BWA, pSense, Kali
- Time Remaining:** 1 30 hrs. min.
- Packet List:**

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-------------|-----------------|-----------------|----------|--------|--|
| 1 | 0.000000000 | Vmware_99:25:99 | Broadcast | ARP | 42 | who has 192.168.9.17 Tell 192.168.9.17 |
| 2 | 0.000472187 | Vmware_9a:63:ac | Vmware_99:25:99 | ARP | 60 | 192.168.9.1 is at 00:50:56:9a:dc:58 |
| 3 | 0.000826269 | 192.168.9.2 | 192.168.68.12 | TCP | 58 | 1234 → 9090 [SYN] Seq=0 Win=0 Len=0 |
| 4 | 0.018842907 | 192.168.68.12 | 192.168.9.2 | TCP | 60 | 9090 → 1234 [RST, ACK] Seq=1 Ack=1 |
- Packet Details:** The selected packet is an ICMP echo request from 192.168.9.2 to 192.168.68.12, ID 0, Seq 0, Length 19.

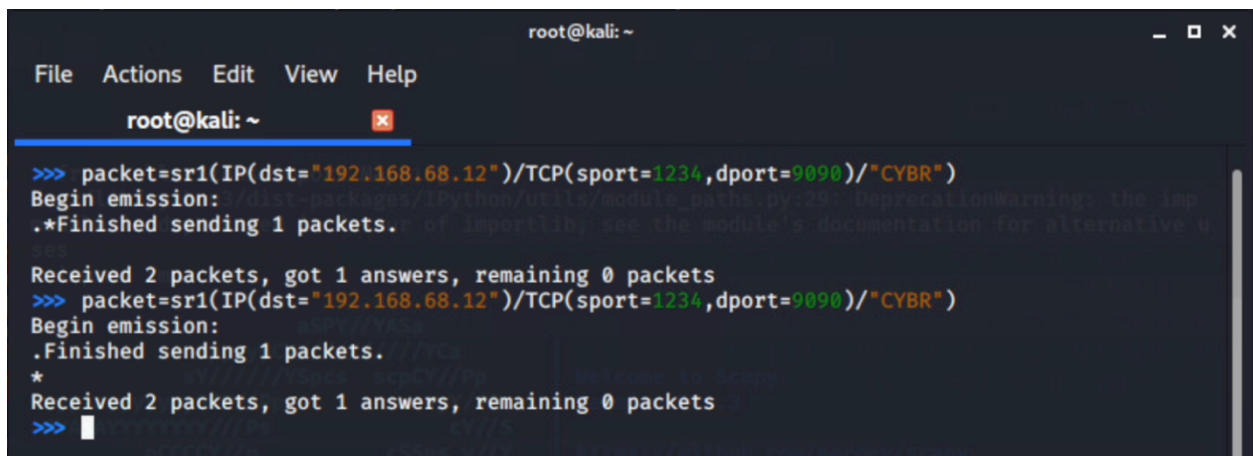
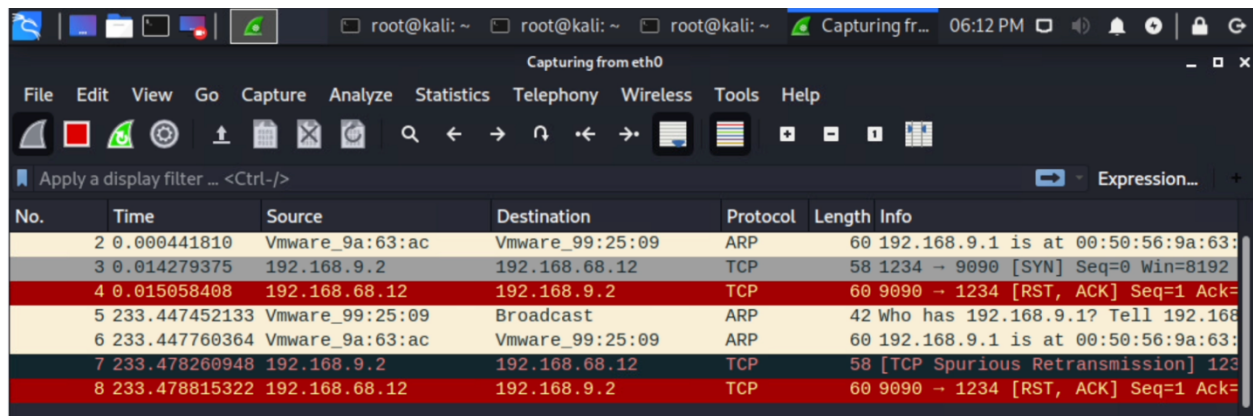
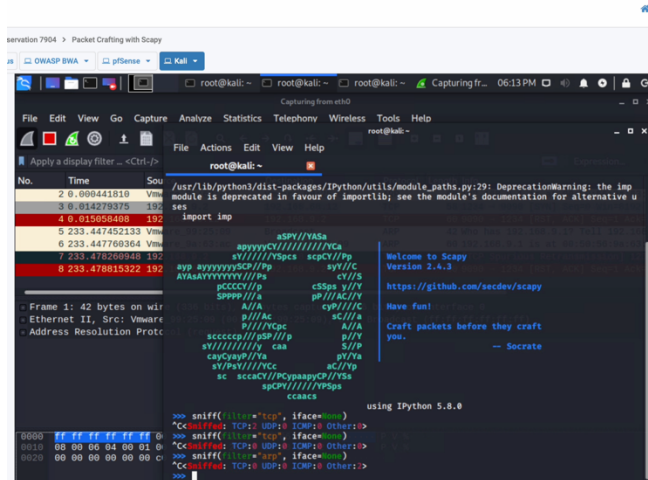
b. Create and send an “ICMP echo” packet from the host (Kali) to the destination OWASP BWA. Take a screenshot of the tcpdump capture on the destination confirming your ICMP packet was delivered.

The screenshot shows the NDG MyNETLAB interface with the following details:

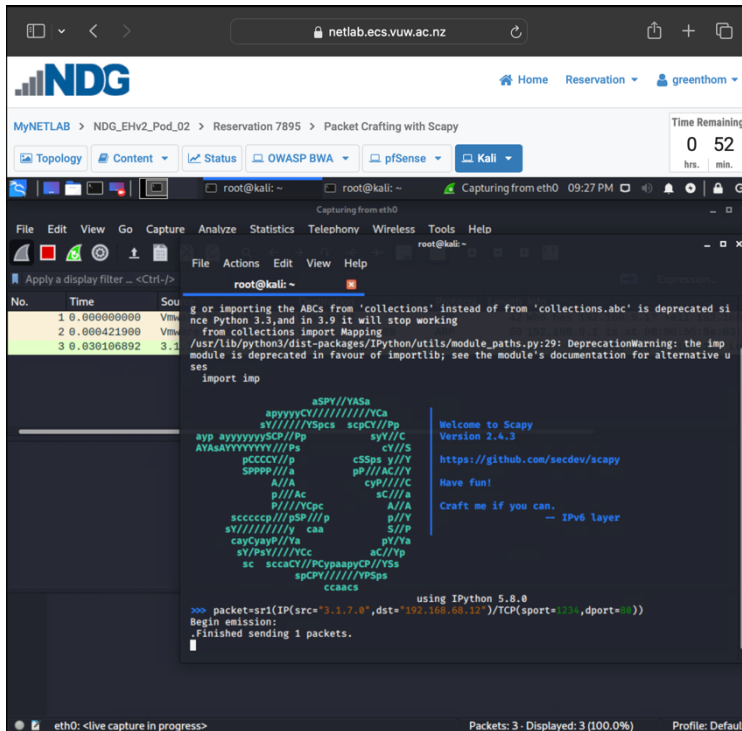
- Navigation:** MyNETLAB > NDG_EHv2_Pod_03 > Reservation 7904 > Packet Crafting with Scapy
- Tools:** Topology, Content, Status, OWASP BWA, pSense, Kali
- Time Remaining:** 1 20 hrs. min.
- Packet Crafting:** The user is crafting an ICMP echo packet using Scapy. The command shown is:


```
ip=IP(ttl=10)
ip
<IP ttl=10 >
ip.dst
'127.0.0.1'
ip.dst='192.168.68.12'
ip
<IP ttl=10 dst=192.168.68.12 >
ip.src='192.168.9.2'
ip
<IP ttl=10 src=192.168.9.2 dst=192.168.68.12 >
ip
<IP src=192.168.9.2 dst=192.168.68.12 >
packet=IP(dst='192.168.68.12')/ICMP()/'XXXXXXXXXX'
Begin emission:
Finished sending 1 packets.
Received 2 packets, got 1 answers, remaining 0 packets
```
- Packet List:** The packet list shows the ICMP echo request packet being sent from 192.168.9.2 to 192.168.68.12, ID 0, Seq 0, Length 19.

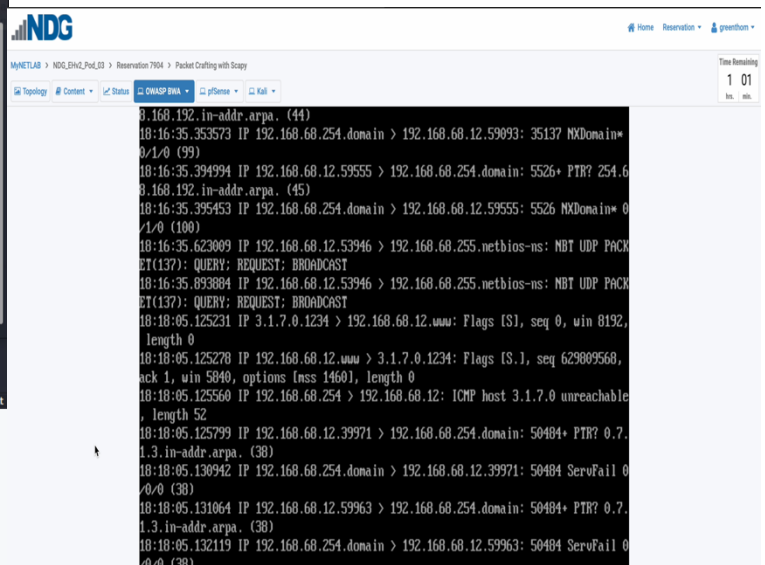
- c. Sniff (i.e. capture) ARP traffic on all the interfaces on the host machine (i.e. Kali). Provide a snippet of the screenshot of its output showing it runs as expected.



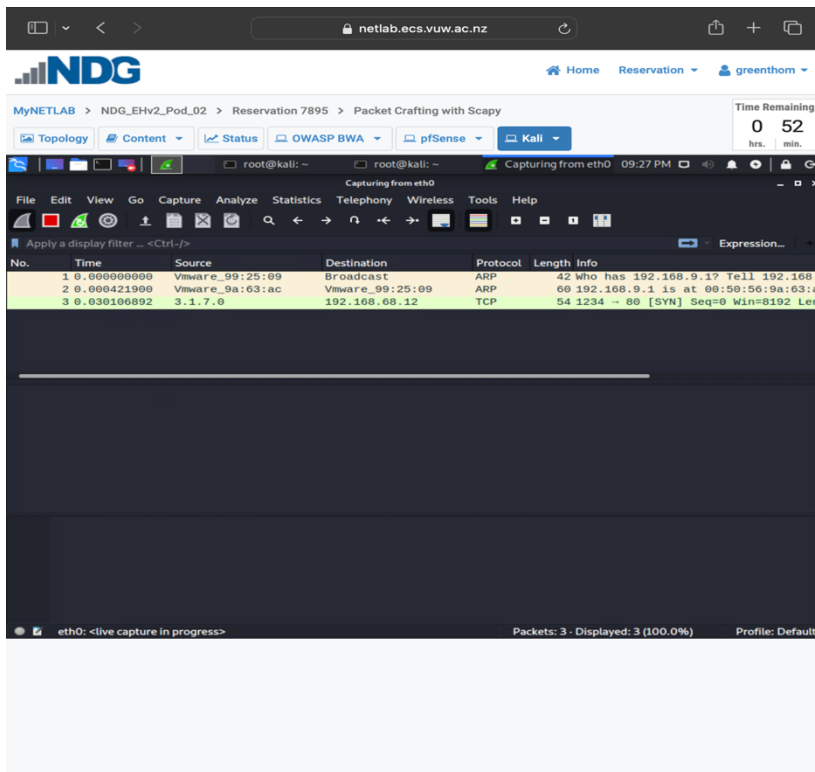
- d. Create and send a spoofed TCP/IP packet (with forged source IP address) from the host (Kali) to the destination machine, OWASP BWA with forged source IP address of 3.1.7.0. Take a screenshot of the tcpdump capture on the destination machine proving your message was delivered. Briefly explain why the message was not dropped in transmission even though the source IP address does not exist.



```
root@kali: ~  
Capturing from eth0 09:27 PM  
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help  
root@kali: ~  
root@kali: ~  
g or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated since Python 3.3, and in 3.9 it will stop working  
from collections import Mapping  
/usr/lib/python3/dist-packages/IPython/utils/module_paths.py:29: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses  
import imp  
aSP//YASa  
apyyyyyC//NCA  
sY//Nspcs scpC//Pp  
ayp ayyyyyySCP//Pp sy//C  
AYAsAYyyyyyy//Ps cy//S  
pCCCC//p cSSps y//Y  
Sppp//a p//AC//Y  
A//A cyp//C  
p//Ac sC//a  
P//NDC A//A  
sccccp//sdp//p p//Y  
sY//N y caa S//P  
cayCyayp//Ya p//Ya  
sY//NCC AC//Pp  
sc sccaCY//PCypaayCP//Ys  
spCP//YPSps  
ccacs  
using IPython 5.8.0  
>>> packet=IP(src='3.1.7.0',dst='192.168.68.12')/TCP(sport=80,dport=80)  
Begin emission:  
Finished sending 1 packets.
```



```
18:16:35.353573 IP 192.168.68.254.domain > 192.168.12.59093: 35137 NXDomain*  
0/1/0 (99)  
18:16:35.394994 IP 192.168.68.12.59555 > 192.168.68.254.domain: 5526* PTR? 254.6  
8.168.192.in-addr.arpa. (45)  
18:16:35.395453 IP 192.168.68.254.domain > 192.168.12.59555: 5526 NXDomain* 0  
/1/0 (100)  
18:16:35.623009 IP 192.168.68.12.53946 > 192.168.68.255.netbios-ns: NBT UDP PACK  
ET(137): QUERY: REQUEST: BROADCAST  
18:16:35.893884 IP 192.168.68.12.53946 > 192.168.68.255.netbios-ns: NBT UDP PACK  
ET(137): QUERY: REQUEST: BROADCAST  
18:18:05.125231 IP 3.1.7.0.1234 > 192.168.68.12.www: Flags [S], seq 0, win 8192,  
length 0  
18:18:05.125278 IP 192.168.68.12.www > 3.1.7.0.1234: Flags [S.], seq 629809568,  
ack 1, win 5840, options (mss 1460), length 0  
18:18:05.125560 IP 192.168.68.254 > 192.168.68.12: ICMP host 3.1.7.0 unreachable  
, length 52  
18:18:05.125799 IP 192.168.68.12.39971 > 192.168.68.254.domain: 50404* PTR? 0.7.  
1.3.in-addr.arpa. (38)  
18:18:05.130942 IP 192.168.68.254.domain > 192.168.12.39971: 50404 ServFail 0  
/0/0 (38)  
18:18:05.131064 IP 192.168.68.12.59963 > 192.168.68.254.domain: 50404* PTR? 0.7.  
1.3.in-addr.arpa. (38)  
18:18:05.132119 IP 192.168.68.254.domain > 192.168.12.59963: 50404 ServFail 0  
/0/0 (38)
```



When the packet is sent from the source to destination, it's recognized by the switches and routers without verifying the legitimacy of the source IP. Packet delivery is based on destination IP address not source which means routers and switches use the destination IP to forward the packet to its destination. Upon arrival at the destination machine, the receiver sends a response to the manipulated IP address. Since this IP address is non-existent, the response packet is discarded, and the sender does not receive the response. If the destination IP is reachable, packets can be forwarded to it without an authentic source IP.