EXP #7

ARP CACHE POISONING ATTACK LAB

SLIDES BY: MOHAMAD BALAWI

جامعة بيرزيت BIRZEIT UNIVERSITY

OUTLINE

Intoduction

Task 1: ARP Cache Poisoning

- Task 1.A (using ARP request)
- Task 1.B (using ARP reply)
- Task 1.C (using ARP gratuitous message)

Task 2: MITM Attack on Telnet using ARP Cache Poisoning

Task 3: MITM Attack on Netcat using ARP Cache Poisoning

Address Resolution Protocol (ARP)

- The Address Resolution Protocol (ARP) is a networking protocol used to map a known IP address to a corresponding physical MAC (Media Access Control) address.
- It enables devices within a local network to communicate with each other by translating IP addresses into MAC addresses, which are necessary for data transmission at the data link layer of the OSI model.
- ARP operates by broadcasting a request message containing the IP address for which the MAC address is sought, and the device with the corresponding IP address responds with its MAC address. This mapping is then stored in an ARP cache to expedite future communications within the network.

ARP Cache Poisoning

- ARP cache poisoning is a cyberattack where an attacker sends false Address Resolution Protocol (ARP) messages over a local area network.
- The attacker sends ARP messages with falsified MAC address information, associating their own MAC address with the IP address of another device on the network.
- This causes the ARP cache of the victim device or devices on the network to be corrupted, leading them to send their network traffic to the attacker's MAC address instead of the legitimate destination.

Packet Sniffing

- Packet sniffing is the practice of intercepting and logging network traffic passing through a computer network. It allows users to analyze packets to capture data.
- In this lab, we will use `tcpdump`, which is a command-line packet analyzer.
- It should be noted that inside containers, due to the isolation created by Docker, when we run `tcpdump` inside a container, we can only sniff the packets going in and out of this container. We won't be able to sniff the packets between other containers. However, if a container uses the host mode in its network setup, it can sniff other containers' packets.
- If we run `tcpdump` on the VM, we do not have the restriction on the containers, and we can sniff all the packets going among containers.

tcpdump

- In this lab, we will use `tcpdump`, which is a command-line packet analyzer tool available on Unix-like operating systems. It allows users to capture and display TCP/IP packets transmitted or received over a network interface.

```
ip address                # used to get network interfaces details.
tcpdump -i eth0 -n        # used for packet sniffing.
```

Argument	Description
-i eth0	Specifies the network interface to capture packets from. On containers: each interface name usually starts with eth. On the VM: the interface name for the network created by Docker starts with br-, followed by the ID of the network.
-n	Displays numerical IP addresses and port numbers instead of attempting to resolve them to hostnames or service names.

scapy

- Scapy is a powerful interactive packet manipulation tool and library in Python, allowing for the creation, sending, sniffing, and analyzing of network packets at a low level. Scapy example:

```
#!/usr/bin/env python3
from scapy.all import *

E = Ether()       # Create an Ethernet frame
A = ARP()         # Create an ARP packet

pkt = E/A         # ARP packet is encapsulated within the Ethernet frame.
# pkt.show()      # Uncomment to print the packet to the terminal.
sendp(pkt)        # Send the packet
```

scapy

- The provided code snippet constructs both an Ethernet Frame and an ARP packet. It then encapsulates the ARP packet within the Ethernet Frame using the overloaded "/" operator.
- Subsequently, it sends the composed packet to its intended destination.
- Below are tables displaying the attributes of the Ethernet Frame and ARP Packet classes in Scapy:

Scapy Ethernet Frame "Ether" Attriburtes	
Attribute	Description
dst	Destination MAC Address (device to which the Ethernet frame is being sent).
src	Source MAC Address (the source device that is sending the Ethernet frame).
type	This field indicates the type of the payload encapsulated within the Ethernet frame. The default value is 0x9000 which is “Ethernet Configuration Testing Protocol”

scapy

Scapy ARP Packet “ARP" Attriburtes	
Attribute	Description
hwtype	The type of hardware being used for communication. set to 1 for Ethernet.
ptype	The type of protocol addresses being used. Typically, set to 2048 for IPv4.
hwlen	The length (in bytes) of the hardware (MAC) addresses. Usually, this is 6 for Ethernet.
plen	The length (in bytes) of the protocol (IP) addresses. Typically, this is 4 for IPv4.
op	Defines the ARP operation, 1 represents an ARP request, and 2 represents an ARP reply.
hwsrc	The MAC address of the sender in the ARP packet.
psrc	The IP address of the sender in the ARP packet.
hwdst	The MAC address of the target device in the ARP packet.
pdst	The IP address of the target device in the ARP packet.

Important Commands

- We can check a computer's ARP cache using the following command. If you want to look at the ARP cache associated with a specific interface, you can use the -i option.

```
arp –n
```

- An ARP record can be removed using the `-d` flag, as follows:

```
arp -d a.b.c.d
```

- We can use ping command to update a target's ARP record with our correct IP / MAC pair:

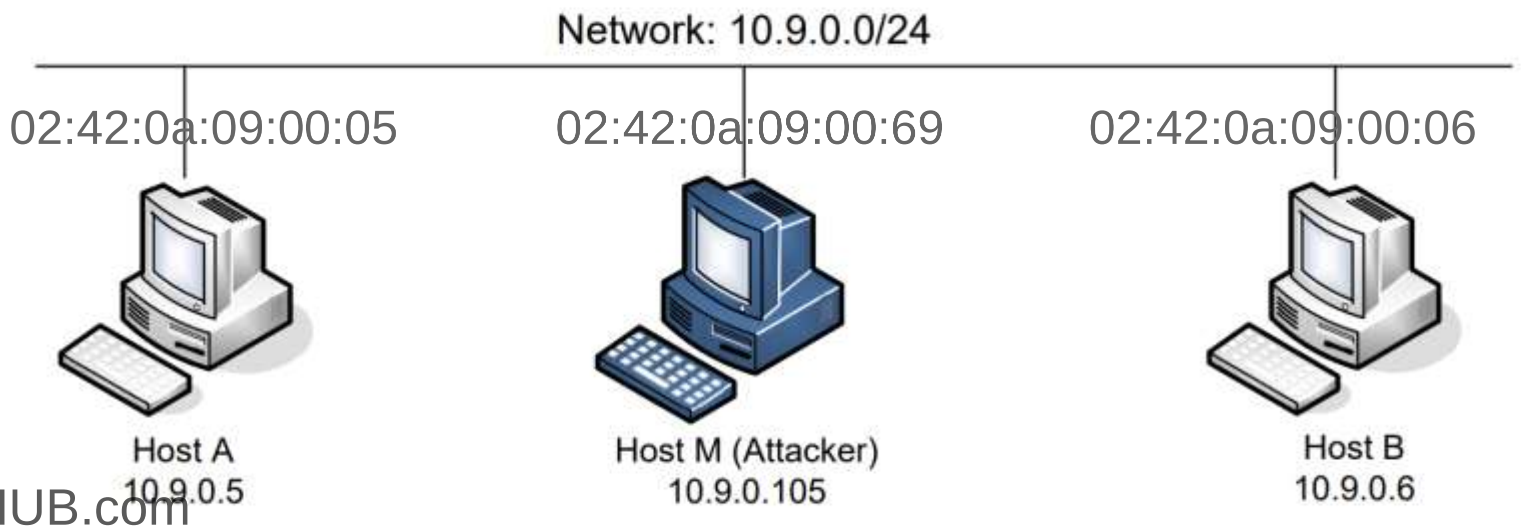
```
ping a.b.c.d
```

- We find the MAC address for the containers, we can do it in the VM using `docker inspect`:

```
docker inspect -f '{{.Name}} - {{range
.NetworkSettings.Networks}}{{.MacAddress}}{{end}}' $(docker ps -aq)
```

Lab setup

- In this lab, we need three machines. We use containers to set up the lab environment, which is depicted in Figure 1. In this setup, we have an attacker machine (Host M), which is used to launch attacks against the other two machines, Host A and Host B. These three machines must be on the same LAN, because the ARP cache poisoning attack is limited to LAN. We use containers to set up the lab environment.



TASK 1

ARP Cache Poisoning

BIRZEIT UNIVERSITY

Task 1: ARP Cache Poisoning

- The objective of this task is to use packet spoofing to launch an ARP cache poisoning attack on a target, such that when two victim machines A and B try to communicate with each other, their packets will be intercepted by the attacker M, who can make changes to the packets, and can thus become the man in the middle between A and B.
- In this task, we focus on the ARP cache poisoning part.
- There are 2 main ARP operations:
  - **ARP request**: Broadcast message, sent to discover the MAC address associated with a specific IP address on a local network.
  - **ARP reply**: Response message, containing the MAC address requested in an ARP request.

Task 1.A (using ARP request)

- On host M, construct an ARP request packet to map B’s IP address to M’s MAC address. Send the packet to A and check whether the attack is successful or not.
- Inside M, write Python Scapy code that does that sends ARP request to A:
  - My MAC address is: (M MAC address).
  - My IP address is: (B IP address).
  - I’m asking for: A MAC address.

Task 1.B (using ARP reply)

- First we need to clean what M has done in Task 1.A, and that is by sending B’s correct IP/MAC pair to A (we can acheive that pinging A from inside B).
- Secondly will replicate what we did in Task 1.A, with a modification to the ARP operation, which will now be **reply** instead of **request**. This will be tested in two distinct scenarios:
  - Scenario 1: Correct B’s IP is already in A’s cache.
  - Scenario 2: B’s IP is not in A’s cache.

ARP gratuitous message

- ARP gratuitous packet is a special ARP request packet.
- It is used when a host machine needs to update outdated information on all the other machine's ARP cache. The gratuitous ARP packet has the following characteristics:
  - The source and destination IP addresses are the same, and they are the IP address of the host issuing the gratuitous ARP.
  - The destination MAC addresses in both ARP header and Ethernet header are the broadcast MAC address (ff:ff:ff:ff:ff:ff).
  - No reply is expected.

Task 1.C (using ARP gratuitous message)

- On host M, construct an ARP gratuitous packet, and use it to map B’s IP address to M’s MAC address.
- Please launch the attack under the same two scenarios as those described in Task 1.B:
  - Scenario 1: Correct B’s IP is already in A’s cache.
  - Scenario 2: B’s IP is not in A’s cache.

TASK 2

MITM Attack on Telnet using ARP Cache Poisoning

Telnet

- Telnet is a protocol that allows you to connect to remote computers (called hosts) over a **TCP**/IP network (such as the internet). Using telnet client software on your computer, you can make a connection to a telnet server (that is, the remote host).

- In Telnet, typically, every character we type in the Telnet window triggers an individual TCP packet, but if you type very fast, some characters may be sent together in the same packet. That is why in a typical Telnet packet from client to server, the payload only contains one character. The character sent to the server will be echoed back by the server, and the client will then display the character in its window. Therefore, what we see in the client window is not the direct result of the typing; whatever we type in the client window takes a round trip before it is displayed.

MITM Attack on Telnet using ARP Cache Poisoning

- Hosts A and B are communicating using Telnet, and Host M wants to intercept their communication, so it can make changes to the data sent between A and B.
- We have already created an account called "seed" inside the container, the password is "dees". You can telnet into this account.
- We will do the attack in 4 steps:
  - Step 1 (Launch the ARP cache poisoning attack).
  - Step 2 (Testing).
  - Step 3 (Turn on IP forwarding).
  - Step 4 (Launch the MITM attack).

Step 1 (Launch the ARP cache poisoning attack).

- First, Host M conducts an ARP cache poisoning attack on both A and B, such that in A’s ARP cache, B’s IP address maps to M’s MAC address, and in B’s ARP cache, A’s IP address also maps to M’s MAC address. After this step, packets sent between A and B will all be sent to M.
- We will use the ARP cache poisoning attack from Task 1 to achieve this goal. You should modify it to send out the spoofed packets constantly (e.g. every 5 seconds); otherwise, the fake entries may be replaced by the real ones.

IP forwarding on Hosts

- IP forwarding on hosts refers to the capability of a computer to forward network packets between its network interfaces. In typical networking setups, hosts (computers) have multiple network interfaces, such as Ethernet, Wi-Fi, or virtual interfaces. When IP forwarding is enabled on a host, it allows the host to act as a simple router, forwarding packets between these interfaces.

Step 2 (Testing).

- Before doing this step, please make sure that the IP forwarding on Host M is turned off. You can do that with the following command:

```
sysctl net.ipv4.ip_forward=0
```

- **Preventing unintended routing**: By disabling IP forwarding, the attacker ensures that their machine doesn't act as a router inadvertently. If IP forwarding were enabled, the attacker's machine might start forwarding packets between different network segments, potentially disrupting network connectivity or exposing the attack.

- After the attack is successful, let Hosts A and B ping each other, and report your observation.

Step 3 (Turn on IP forwarding).

- Now we turn on the IP forwarding on Host M, so it will forward the packets between A and B.

```
sysctl net.ipv4.ip_forward=1
```

- Let Hosts A and B ping each other, and report your observation.

Step 4 (Launch the MITM attack).

- We are ready to make changes to the Telnet data between A and B.
- Assume that A is the Telnet client and B is the Telnet server.
- After A has connected to the Telnet server on B, for every key stroke typed in A's Telnet window, a TCP packet is generated and sent to B. We would like to intercept the TCP packet, and replace each typed character with a fixed character (say Z). This way, it does not matter what the user types on A, Telnet will always display Z.
- From the previous steps, we are able to redirect the TCP packets to Host M, but instead of forwarding them, we would like to replace them with a spoofed packet.

Step 4 (Launch the MITM attack).

- We will write a sniff-and-spoof program to accomplish this goal. In particular, we would like to do the following:
  - We first keep the IP forwarding on, so we can successfully create a Telnet connection between A to B. Once the connection is established, we turn off the IP forwarding. Please type something on A’s Telnet window, and report your observation
  - We run our sniff-and-spoof program on Host M, such that for the captured packets sent from A to B, we convert every letter to “Z”. For packets from B to A (Telnet response), we do not make any change, so the spoofed packet is exactly the same as the original one.

Step 4 (Launch the MITM attack).

- To help students get started, we provide a skeleton sniff-and-spoof program in the manual.
- Here are some important information about the functions used in the program.

`sniff(iface='eth0', filter='tcp', prn=spoof_pkt)`

sniff() function is used to capture packets from the network. It allows you to intercept, analyze, and even modify packets as they traverse the network interface.

Argument	Description
iface	specifies the network interface to capture packets from
filter	allows you to specify a BPF (Berkeley Packet Filter) syntax filter
prn	specifies the callback function to be called for each captured packet
count	specifies the number of packets to capture (use -1 for unlimited)

Step 4 (Launch the MITM attack).

```
newpkt = IP(bytes(pkt[IP]))
```

The IP() function in Scapy is used to create or manipulate IP packets. It can be used to create new IP packets, modify existing ones, or extract information from them. In this context, it is used to construct a new IP packet based on the IP layer extracted from the original packet (pkt).

TASK 3

MITM Attack on Netcat using ARP Cache Poisoning

netcat

- Netcat, often abbreviated as "nc", is a networking utility used for reading from and writing to network connections using TCP or UDP protocols. It's sometimes referred to as the "Swiss Army knife of networking" due to its flexibility and range of capabilities.

MITM Attack on Netcat using ARP Cache Poisoning

- This task is similar to Task 2, except that Hosts A and B are communicating using `netcat`, instead of `telnet`. Host M wants to intercept their communication, so it can make changes to the data sent between A and B. You can use the following commands to establish a netcat TCP connection between A and B:
- On Host B (server, IP address is 10.9.0.6), run the following:

```
nc -lp 9090
```

- On Host A (client), run the following:

```
nc 10.9.0.6 9090
```

Attribute	Description
-lp	listening mode, it tells netcat to bind to the specified address and port and listen for incoming connections.

MITM Attack on Netcat using ARP Cache Poisoning

- Once the connection is made, you can type messages on A. Each line of messages will be put into a TCP packet sent to B, which simply displays the message. Your task is to replace every occurrence of your first name in the message with a sequence of A's. The length of the sequence should be the same as that of your first name, or you will mess up the TCP sequence number, and hence the entire TCP connection. You need to use your real first name, so we know the work is done by you.