ARP Cache Poisoning Attack Lab

Contents

Lab Setup

Please download the Labsetup.zip file from the below link to your VM, unzip it, enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment.

https://seedsecuritylabs.org/Labs_20.04/Networking/ARP_Attack/

In this lab, we need to have at least three machines. We use containers to set up the lab environment.

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.

Students can also use three virtual machines for this lab, but it will be much more convenient to use containers.

Note: When we use the attacker container to launch attacks, we need to put the attacking code inside the attacker container. Code editing is more convenient inside the VM than in containers, because we can use our favorite editors. Hence it is advisable for you to place your respective codes in the "volumes" folder directly (using gedit for example).

Lab Overview

The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link-layer address, such as the MAC address, given an IP address. The ARP protocol is a very simple protocol, and it does not implement any security measure. The ARP cache poisoning attack is a common attack against the ARP protocol. Using such an attack, attackers can fool the victim into accepting forged IP-to-MAC mappings. This can cause the victim's packets to be redirected to the computer with the forged MAC address, leading to potential man-in-the-middle attacks.

The objective of this lab is for students to gain first-hand experience on the ARP cache poisoning attack, and learn what damages can be caused by such an attack. In particular, students will use the ARP attack to launch a man-in-the-middle attack, where the attacker can intercept and modify the packets between the two victims A and B. Another objective of this lab is for students to practice packet sniffing and spoofing skills, as these are essential skills in network security, and they are the building blocks for many network attack and defence tools. Students will use Scapy to conduct lab tasks.

CHANGE THE NAME OF YOUR TERMINAL!!!

ARP Cache Poisoning

OBJECTIVE: The objective of this task is to use packet spoofing to launch an ARP cache poisoning attack on a target. The attack should insert a fake entry into the target machine's cache. The following code skeleton shows how to construct an ARP packet using Scapy -

```
#!/usr/bin/python3
from scapy.all import *
E = Ether()
A = ARP()
pkt = E/A
pkt.show()
```

```
sendp(pkt)
```

In this task, we have three machines (containers), A, B, and M. We use M as the attacker machine. We would like you to launch an attack on A such that a fake entry is added to its ARP cache, in which B’s IP address is mapped to M’s MAC address.
There are many ways to conduct the ARP cache poisoning attack. Students need to try the following three methods and report whether each method works or not.

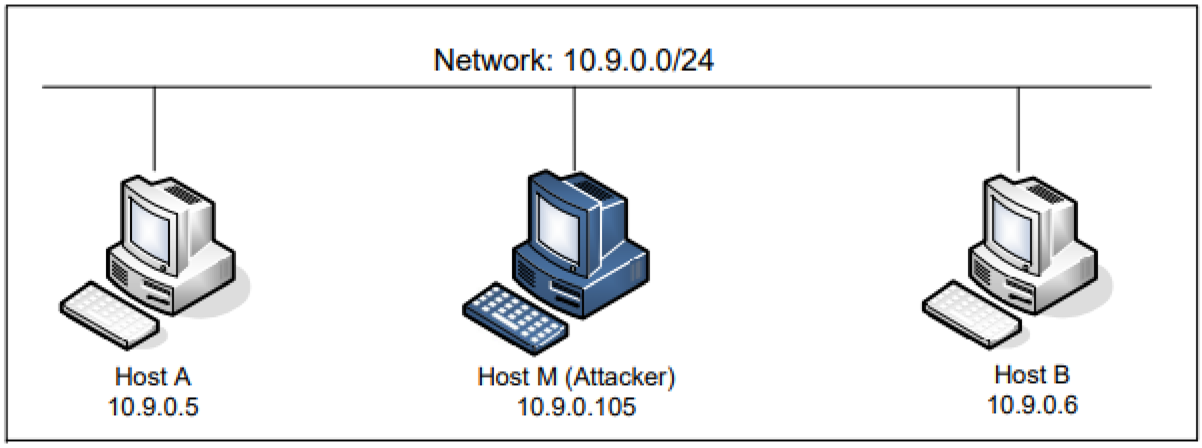


Figure 1: Lab environment setup

There are three ways in which you can launch the cache poisoning attack.

Task 1: Using ARP request

- On host M, construct an ARP request packet and send it to host A. Check whether M's MAC address is mapped to B's IP address in A's ARP cache. We can do this in two ways:

1. <u>Without passing parameters to Ether()</u>

- To view the arp table run the following on both Host’s A and B:
  Command: # arp

Take a screenshot of ARP cache output on both the hosts

- Run tcpdump on both the hosts A and B using the below command:

Command: **# tcpdump -i eth0 -n**

This will show you all the packets captured on the interface eth0

- Finally, execute the below command on the attacker machine M

  Command: **# python3 task1A.py**

- Close the tcpdump **(Ctrl+C)** on A and B and run the following to view the updated arp cache, take a screenshot of the same -

  Command: **# arp**

Take a screenshot of ARP cache output on both the hosts after the attack.

Also take a screenshot of the attacker terminal

- Now delete the ARP Cache entries of the attacker and Host B by executing the below commands on Host A.

  Commands:

  ```
  # arp -d 10.9.0.6
  # arp -d 10.9.0.105
  ```

Take a screenshot showing that the cache entries are deleted

2. <u>Ether() with parameters</u>

- Perform the same steps as mentioned previously, but this time we provide parameters for Ether() layer.

  Take a screenshot of ARP cache output on both the hosts before the attack

- View the arp table on both the hosts and then run tcpdump on them before executing the below code on the attacker machine, M.

  Command: **# python3 task11A.py**

  Take a screenshot of ARP cache output on both the hosts after the attack

  Take a screenshot of the attacker terminal as well.

- Delete ARP cache entries of Attacker and Host B and provide a screenshot.

  Commands:

  ```
  # arp -d 10.9.0.6
  # arp -d 10.9.0.105
  ```

Take a screenshot showing that the cache entries are deleted

Answer the following questions:

1. What does the 'op' in the screenshot of the attacker machine signify? What is its default value?
2. What was the difference between the ARP cache results after the attack in the above 2 approaches? Why did you observe this difference?

Task 2: Using ARP Reply

OBJECTIVE: In this task you need to attack A using ARP reply packet. Try the attack under the following two scenarios, and report the results of your attack:

Scenario 1: B's IP is already in A's cache.

- In order to place B's IP in A's cache -Execute the following on the Attacker Machine M.

  Command: # python3 task11A.py

Take a screenshot showing the cache entry in A

- Then run tcpdump on Host A to sniff packets -

  Command: # tcpdump -i eth0 -n

- Finally, we run the following on the Attacker M

  Command: # python3 task1B.py

Take a screenshot showing showing the cache entries of A after the attack

Take a screenshot of the attacker terminal as well

Scenario 2: B's IP is not in A's cache.

- Delete the ARP Cache entry of Host B in A. By running the below command on host A.

Take a screenshot showing that the cache entries are deleted

- Now run tcpdump on Host A to sniff the packets

  Command: # tcpdump -i eth0 -n

- Now run the following on the Attacker Machine M

  Command: # python3 task1B.py

Take a screenshot showing showing the cache entries of A after the attack

Take a screenshot of the packets captured using tcpdump

Take a screenshot of the attacker terminal as well

EXPLAIN YOUR OBSERVATIONS.

Answer the following question:

1. What does op=2 mean?

Task 3: Using ARP Gratuitous Message

- **OBJECTIVE**: 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: B's IP is already in A's cache.

- Execute the following on the Attacker Machine M to add B's ip address.

  Command: # python3 task1A.py

  Take a screenshot showing A's cache

- Then we run tcpdump on **Host A and Host B** to sniff packets.

  Command: # tcpdump -i eth0 -n

- Finally, we run the following on the Attacker M

  Command: # python3 task1C.py

Take a screenshot of the packets captured using tcpdump

Take a screenshot showing A's cache after the attack

Take screenshots of terminals B and M after the attack

For Scenario 2: B's IP is not in A's cache.

- Now we delete the ARP Cache entries on both Host A and Host B.

Take a screenshot showing that the cache entries are deleted

- Now run tcpdump to sniff the packets on Host A and Host B

  Command: # tcpdump -i eth0 -n

- Now run the following on the Attacker Machine M

  Command: # python3 task1C.py

Take a screenshot of the packets captured using tcpdump

Take a screenshot showing A's cache after the attack

Take screenshots of terminals B and M after the attack

EXPLAIN YOUR OBSERVATIONS.

Answer the following question:

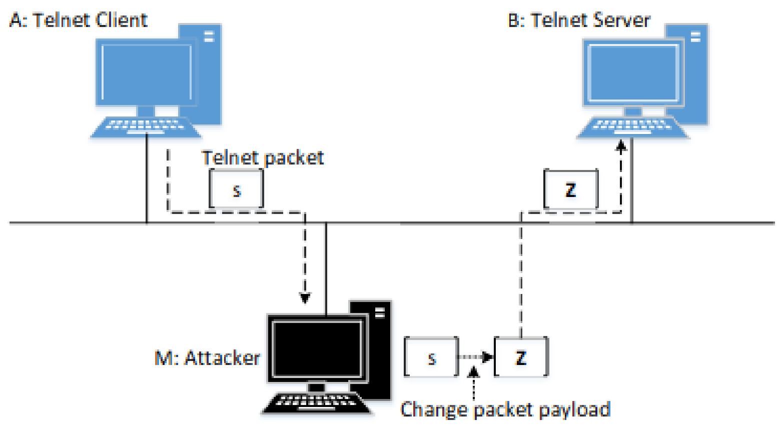
1. Why does VM B's ARP cache remain unchanged in this approach even though the packet was broadcasted on the network?

- Make sure to delete your ARP cache entries of Host A, and Host B before proceeding to the next Task.

Task 4: MITM Attack on Telnet using ARP Cache Poisoning

OBJECTIVE: 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. The setup is depicted in

Figure 1. We have already created an account called "seed" inside the container, the password is "dees". You can telnet into this account.



Step 1 - Launch the ARP cache poisoning attack

First, Host M conducts an ARP cache poisoning attack on both A and B. After this step, packets sent between A and B will all be sent to M.

Answer the following question:

What will be the mappings in A and B's cache after the cache poisoning attack by M?

We will use the ARP cache poisoning attack from Task 1 to achieve this goal.

- First, check the ARP caches of Host A and Host B

    Take a screenshot of A and B's caches

- Now for this step, we execute the code and commands as discussed in Task 1A (with Ether() parameters) mapping B's IP address to M's MAC address in A's ARP Cache.

    Command: **# python3 taskl1A.py**

    Take a screenshot of A's cache and M's terminal

- Then execute the below code to map A's IP address to M's MAC address in B's ARP Cache

    Command: **# python3 task2.py**

    Take a screenshot of B's cache and M's terminal

- Finally check the updated ARP caches of Host A and Host B

  Command: **#arp**

Please Note: From now on, at times you won't be getting the desired output, this is due to the fact that the ARP caches are being made redundant on Both Host Machines (A and B), so you will have to execute **task11A.py** and **task2.py** in order to update the ARP entries.

Step 2 - Testing

- **You will need Wireshark from now - open the container interface 'br-***' in order to capture the required packets.**

- On Attacker M, disable IP forwarding by executing the following

  Command: **# sysctl net.ipv4.ip_forward=0**

- Update the ARP Caches by running the following commands on the attacker machine.

  Commands:

  ```
  # python3 task11A.py
  # python3 task2.py
  ```

- Then we ping from Host A to Host B using the following command:

  Command: **# ping 10.9.0.6**

Take a screenshot of A's terminal

Take a screenshot of the Wireshark packet capture

Answer the following question:

1. What do you observe? Explain

In case the desired output does not occur, then you will have to update the ARP Cache by executing task11A.py and task2.py on Attacker M.

Restart the Wireshark packet capture before this step

Step 3 - Turn on IP Forwarding

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

  Command: **# sysctl net.ipv4.ip_forward=1**

- Now ping Host B from Host A -

  Command: **# ping 10.9.0.6**

Take a screenshot of A's terminal

Take a screenshot of the Wireshark packet capture

Answer the following question:

1. Compare the results between the above two steps.

Step 4 - Launch the MITM Attack

We are ready to make changes to the Telnet data between A and B.

OBJECTIVE: From the previous steps, you are able to redirect the TCP packets to Host M, but instead of forwarding them, you should replace them with a spoofed packet. write a sniff-and-spoof program to accomplish this goal. In particular, we would like to do the following:

Reminder - Make sure to execute Step 1 to update the ARP tables and open Wireshark on the given interface (br-**).**

On Host M

Command :

```
# python3 task11A.py
# python3 task2.py
```

- We first keep the IP forwarding on, so we can successfully **create a Telnet connection between A to B.**

  Command : **# sysctl net.ipv4.ip_forward=1**

- To establish a Telnet connection between Host A and B

  Command: **# telnet 10.9.0.6**

- Once the connection is established, we turn off the IP forwarding. **Back On Host M**

  Command: **# sysctl net.ipv4.ip_forward=0**

Please type something on Host A's Telnet window, and **see the packets captured on Wireshark, take a screenshot of the same.**

Now on Host M, we run the following to accomplish our Attack:

```
# python3 mitm.py
```

- **Now type anything on the Telnet Window (Host A), only 'Z' should be displayed.**

Take screenshots of A's terminal and the Wireshark packet capture.

Take a screenshot of the attacker terminal as well.

EXPLAIN YOUR OBSERVATIONS.

Task 5: 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. Once the connection is made, you can type messages on A.

Commands:

- The sequence of commands to be run:

  On Attacker M -

  ```
  # python3 task11A.py
  # python3 task2.py
  # sysctl net.ipv4.ip_forward=1
  ```

Take a screenshot of the caches of A and B to show they have been poisoned

- You can use the following commands to establish a netcat TCP connection between A and B. B will act as the server and A will be the client.

  On Host B run:

  ```
  # nc -lp 9090
  ```

  On Host A run:

  ```
  # nc 10.9.0.6 9090
  ```

Take a screenshot of the established netcat connection

- To launch the Attack

  On Attacker M -

  ```
  # sysctl net.ipv4.ip_forward=0
  # python3 mitm1.py
  ```

- **Type a 6-character sequence (maximum) or word on Host A's netcat connection, preferably the first 6 characters of your name.**

- **The length of the sequence should be 6, or you will mess up the TCP sequence number,** and hence the entire TCP connection.

Take a screenshot of the terminals on A and B showing that the attack worked.

Explain your observations.

Submission

You need to submit a detailed lab report to describe what you have done and what you have observed; you also need to explain the observations that are interesting or surprising. Please also list the important code snippets followed by an explanation. Simply attaching code without any explanation will not receive credits.