**ARP Poisoning (MITM) Attack**

**Lab Overview**

In this lab, you will implement a Man-in-the-Middle (MitM) attack on our simulated LAN by poisoning the ARP cache and sniffing legitimate traffic on the LAN. We will assume all participants have a general understanding of how the ARP protocol works. If you do not know ARP, now would be a great time to read up on the basics of [ARP Spoofing](https://en.wikipedia.org/wiki/ARP_spoofing) before moving forward.

ARP is a common network communication protocol that resolves Internet-Layer Addresses into Link-Layer Addresses. On a LAN, machines communicate with each other by (Link-Layer) MAC Addresses. The basic idea behind an ARP Spoofing attack will be to change the Attacker’s MAC address to be the same as a Victim’s MAC Address. This will allow the Attacker to masquerade as the Victim on the LAN and possibly receive traffic intended for the Victim. This attack will only work networks that use ARP and require an Attacker to have direct access to the LAN. Lets get started.

**Preparation**

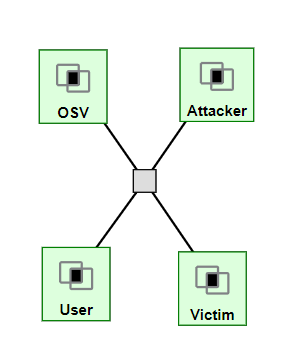
**Before you start, you will need to complete with the following setups (If you have complete some steps before, you can skip them):**

1. A working terminal program (see Setup-VM). In this lab, we suggest you set the memory size of your VM to 4GB, so that you can use wireshark on it.
2. Setup user account on GENI and join a Project. (see Setup-User)
3. Setup Lab in GENI (see Setup-Lab). The RSpec file location is shown as follows.
4. Establish SSH connections from host (VM) to each machine in the Lab (see Setup-Lab)

**RSpec File Location:**

<https://raw.githubusercontent.com/informationcomputerscience/EdGENI/master/Rspec-Files/ARP-Poisoning-Rspec.txt>

**Network Topology**



Here is the network topology for this lab. Even though our machines are real, the MitM attack will be a simulation. We need to thoroughly understand the topology to really understand our attack. In this lab, we use 4 different machines configured as a LAN. All machines are connected together as a fully switched network to mimic a switch on the network with 4 machines connected to it. OSV will function as “gateway” to demonstrate normal network traffic from a neutral machine. Attacker will be the machine performing the ARP Poisoning as well as the MitM in the MitM attack. User and Victim will both function as victims in the attack.

**Task 1 - Show Normal Network Traffic**

To fully understand how ARP is being used on the network, we can examine the ARP cache that display data about the ARP cache. Since this network is brand new, we need to generate some normal network traffic to populate our ARP cache with actual MAC addresses. First examine the empty ARP cache to see the cache format.

Go to the terminal connected to the User machine and type **arp** and enter. Your arp output should look similar to the output below.

calvinmc@user:~$ arp

Address HWtype HWaddress Flags Mask Iface

172.17.253.254 ether fe:ff:ff:ff:ff:ff C eth0

172.16.0.1 ether fe:ff:ff:ff:ff:ff C eth0

172.16.0.3 ether fe:ff:ff:ff:ff:ff C eth0

172.31.248.22 ether fe:ff:ff:ff:ff:ff C eth0

...snip…

Now to populate the ARP cache for the User machine, we need to initiate communication with our Victim machine. Type **ping victim** to send a ICMP packet from the User machine to the Victim. After a few seconds, stop the ping by hitting the **CTRL+C** keys. Now examine the ARP cache on the User machine again. You should see a few changes. The Victim machine will be displayed under the address column. The MAC address of the Victim machine should be displayed under the HWaddress column. The Iface column also shows a change of interface.

calvinmc@user:~$ arp

Address HWtype HWaddress Flags Mask Iface

172.17.253.254 ether fe:ff:ff:ff:ff:ff C eth0

172.16.0.1 ether fe:ff:ff:ff:ff:ff C eth0

172.16.0.3 ether fe:ff:ff:ff:ff:ff C eth0

**Victim-link-0 ether 02:92:45:cf:85:6c C eth1**

172.31.248.22 ether fe:ff:ff:ff:ff:ff C eth0

...snip…

**Screenshot User’s ARP cache and include in documentation.**

Doing this ping from the User machine will also populate the ARP cache on the Victim machine. Type **arp** at the Victim machine’s terminal to show the MAC address of the User machine. These MAC addresses will become important in Step 2.

calvinmc@victim:~$ arp

Address HWtype HWaddress Flags Mask Iface

172.17.253.254 ether fe:ff:ff:ff:ff:ff C eth0

**User-link-0 ether 02:78:50:cc:04:68 C eth1**

172.16.0.1 ether fe:ff:ff:ff:ff:ff C eth0

172.16.0.3 ether fe:ff:ff:ff:ff:ff C eth0

172.31.248.22 ether fe:ff:ff:ff:ff:ff C eth0

...snip…

**Screenshot Victim’s ARP cache and include in documentation.**

When a machine on the LAN wants to talk to a machine for the first time, it first must learn that machine’s MAC address. To find this MAC address, the requesting machine sends out a ARP broadcast message to every node on the LAN. When the correct machine receives the ARP broadcast, they will send out an ARP reply message; telling our requesting machine it’s MAC address and completing the ARP handshake for protocol communication.

To demonstrate this process we will be putting a traffic sniffer on the network and analysing ARP packets. We will be using a python script with the Scapy library on the OSV machine to do this network sniffing. Scapy is a powerful tool for network sniffing/spoofing and you could fill a book with all of its possible uses. For simplicity sake, we will only be using it to show our network traffic packets.

We should have a Scapy installed on the host OSV. If it is the first attempt of using Scapy on OSV, we might have to install it. From the command line, type: **pip install scapy,** and to start the sniffer, type: **sudo python /local/sniffer.py.** A closer look into this python script (sniffer.py) would show we have preconfigured the sniffer to sniff all arp traffic from an **eth1** interface on the LAN.

You should not see any traffic until there is active communication with the OSV machine. From the User or Victim machine, type: **ping osv**

You should now see the sniffer’s output displayed on the OSV machine. These are the ARP broadcast and reply packets. A closer look at these packets will reveal the corresponding machine’s MAC address.

**Screenshot a ARP packet & indicate if its a broadcast or reply packet in doarparcumentation.**

**Task 2 - Poison the Victim**

For the next task, we need to find out the MAC address for the Attacker machine. Instead of using the ping command, we will use a different tool to avoid populating the ARP cache with Attacker’s MAC address. On the Attacker machine’s terminal, type **ifconfig** and then enter. Notice in the above example, the MAC of the User and Victim machines were associated with the eth1 interface. We need to look for the MAC address associated with eth1 on the Attacker machine. Your ifconfig output should look similar to the output below:

...snip…

eth1 Link encap:Ethernet HWaddr **02:78:50:cc:04:68**

inet addr:10.10.1.3 Bcast:10.10.1.255 Mask:255.255.255.0

inet6 addr: fe80::78:50ff:fecc:468/64 Scope:Link

...snip...

**Screenshot Attacker’s MAC address and include in documentation.**

Before we start poisoning, it is important to setup traffic forwarding on our Attacker machine to avoid causing a Denial of Service condition for the Victim. Fortunately, traffic forwarding is turned on by default within the GENI environment. To verify traffic forwarding is turned on, type: **cat /proc/sys/net/ipv4/ip\_forward** and this should print the number 1 if turned on.

Now from the Attacking machine, type **sudo arpspoof -i eth1 -t user victim**

If you meet the error saying “arpspoof: command not found”, type in the command to install it: “sudo apt-get install dfniff”, and type in “Y” if you meet questions during the installation.

The Arpspoof tool is being utilized to perform our poisoning attack. The -i option allows us to specify the interface (eth1), and the -t lets us specify a target to spoof. The last field represents the victim we want to pretend to be. If we ran this command without the -t user target, we would effectively poison the whole network. This would be useful if we wanted to masquerade as a default gateway on the LAN. For now, we will only focus on the user and victim.

**Screenshot the Arpspoof output and include in documentation with an explanation.**

Leave the arpspoof command running. This will ensure that the ARP replies sprayed out onto the network find their intended targets. Now let’s examine what happened. On the User machine, type **arp -a | grep “eth1”** to see User’s new “poisoned” ARP cache:

calvinmc@user:~$ arp -a | grep "eth1"

Attacker-link-0 (10.10.1.1) at 02:a1:c2:ac:a5:c5 [ether] on eth1

Victim-link-0 (10.10.1.4) at 02:a1:c2:ac:a5:c5 [ether] on eth1

OSV-link-0 (10.10.1.2) at <incomplete> on eth1

**Screenshot User’s poisoned ARP cache and include in documentation with an explanation.**

**Task 3 - Show the effect the Man in the Middle Attack**

Design a lab to show that the package sent from user to victim will be intercepted by the attacker. E.g., you can send the ping packets from the user to the victim. Then you can show the message is received by the attacker. (Hint, you can reuse the python script /local/sniffer.py on the osv machine. However, you may need to change the “arp” to “icmp” in order to have the script to sniffer ping packets.)

**Write your steps and take screenshots to show the success of your lab. You may need to paste the code of the modified sniffer.py.**

**Task 4 - Restore ARP Cache**

Type CTRL+C on Attacker’s arpspoof process to stop it. It will restore the ARP cache of the user before exiting. On the User machine, type **arp -a | grep “eth1”** to see User’s restored ARP cache:

**Screenshot User’s restored ARP cache and include in documentation with an explanation.**

**Useful Commands:**

Install the scapy library: **pip install scapy**

Run Scapy Sniffer:  **sudo python /local/sniffer.py**

Clear an ARP Cache: **ip -s -s neigh flush all**

Turn on traffic forwarding: **echo 1 > /proc/sys/net/ipv4/ip\_forward**