Lab Exercise 5: Security Attacks

Objectives:

- Learn about the DHCP masquerade attack
- Learn about the switch table overflow attack (or switch poisoning)
- Learn how to emulate network security attacks in Mininet

Marks:

This exercise is worth 8 marks. All answers will be marked.

Deadline:

Before your scheduled lab next week. So you get one week to work on this lab. For example, if you go to the Monday 18:00 lab, then your submission is due at 17:59 on the following Monday. You can submit as many times as you wish before the deadline. A later submission will override the earlier submission, so make sure you submit the correct file. Do not leave until the last moment to submit, as there may be technical or communications error and you will not have time to rectify it.

Late Penalty:

Late penalty will be applied as follows:

- 1 day after deadline: 20% reduction
- 2 days after deadline: 40% reduction
- 3 days after deadline: 60% reduction
- 4 or more days late: NOT accepted

Note that the above penalty is applied to your final mark. For example, if you submit your lab work 2 days late and your score on the lab is 8, then your final mark will be 8-3.2 (40% penalty) = 4.8.

Submission Instructions:

Submit a PDF document **lab5.pdf** with answers to all questions for all exercises. Create a tar archive of all files called **lab5.tar**. Submit the archive using give. Click on the submission link at the top of the page. Max file size for submission is 3MB.

Original Work Only:

You are strongly encouraged to discuss the questions with other students in your lab. However, each student must submit his or her own work. You may refer to the reference material and also conduct your own research to answer the questions.

Notation: In the examples below, we have used the `$` sign to represent Linux commands that should be typed at the shell prompt, `mininet>` to show Mininet commands that should be typed at Mininet's CLI (command line interface), and `#` to show Linux commands that are typed at a root shell prompt. The actual prompt may look quite different on your computer (e.g. it may contain the computer's hostname, or your username, or the current directory name). The commands that **you** are supposed to type are in **this bold font**.

Overview

In this lab you will gain hands-on experience in conducting two security attacks, which exploit certain network protocol vulnerabilities. In the first exercise we will launch a DHCP masquerade attack whereby an attacker can trick a host into visiting a fake web server by launching a fast-response DHCP server. In the second exercise we will execute a switch table overflow attack to compromise a L2 switch with ultimate goal of eavesdropping traffic on the network. Executing such attacks on operational networks is often not practical. However, we can readily do this on virtual topologies using Mininet.

Exercise 1: DHCP Masquerade Attack

IMPORTANT IF RUNNING ON YOUR OWN VM: Note that, for proper execution of this exercise, it is essential that the network settings of your VM be correctly configured. Your Mininet VM must have two network interfaces, the first one should be a NAT interface that it can use to access the Internet, and the other should be a host-only interface that enables it to communicate with the host machine. Moreover, eth0 (or adapter 1) on the VM must be the host-only interface, while eth1 must be the NAT interface. If the order of interfaces is different (e.g. if eth1 is the host-only interface and vice-versa) then the provided scripts may not work. You may need to change this. You can find some information about properly configuring network settings in VirtualBox here - https://github.com/mininet/openflow-tutorial/wiki/Set-up-Virtual-Machine#VirtualBox and https://github.com/mininet/openflow-tutorial/wiki/VirtualBox-specific-Instructions (setting up host-only network part). If the script does not work on your machine, we recommend that you try it on a CSE lab machine. We have tested that the script works without issues on the lab VMs.

In this exercise, we use a simple network topology to demonstrate a DHCP attack.

The test network consists of three hosts connected to a single switch, as shown in Figure 1:

- *h1* (10.0.0.10 – may be a different address in your topology since it is assigned by the DHCP server) is the "victim" host - a computer which has just connected to the network (for example just joined a Wi-Fi network) and is going to make a DHCP request.
- *dhcp* (10.0.0.50) is the "good" (benign) DHCP server which provides correct information, but which is connected to the switch by a slower link (500 ms delay in this example.)
- *evil* (10.0.0.66) is a malicious host which is connected to the switch by a fast link; it implements a malicious DHCP server and also hosts a malicious DNS server and malicious DHCP server. It is pure evil !!

This network is connected to the global Internet via a NAT router (this is handled by using some functions from the NAT example in mininet mentioned earlier)

When the victim host *h1* broadcasts a DHCP request, it is forwarded to both dhcp and *evil*, but *evil* responds first and the victim host accepts its DHCP offer. The evil DHCP server provides its address as the DNS server address, so the victim sends its DNS requests to the evil DNS server, which provides its own IP address rather than the correct IP address for the DNS lookup. The victim then connects to the address of the evil web server, which serves its own malicious content, asking the victim for an e-mail address and password.

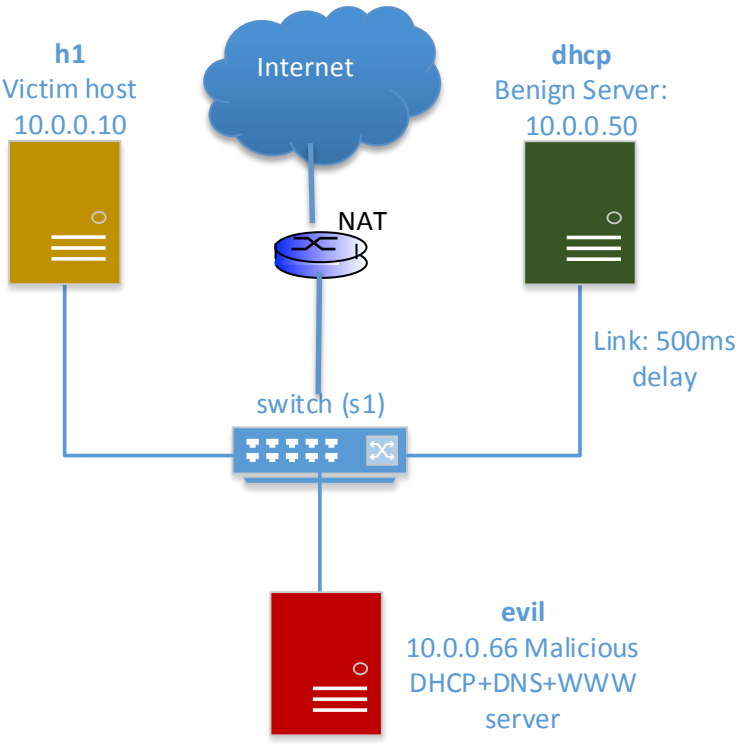

Figure 1 Network topology for running DHCP masquerade attack.

Demo Instructions

Make sure Mininet's X11 support (e.g. sudo -E mn -x) is working correctly. Log in to your mounted home directory, download the exercise code and open the archive by typing:

```
# wget http://www.cse.unsw.edu.au/~cs3331/16s1/Labs/lab5exercise1.tgz
# tar -xvf lab5exercise1.tgz
```

Change the directory to lab5exercise1. The provided code is complete and ready to run. However, we strongly encourage you to read and understand the code. You will notice that the script uses some functions from the NAT example (mininet.example.nat), which allow the Mininet network topology to connect to the global Internet. The first time you run the demo, it may try to install required support software, such as dnsmasq curl and firefox. You may consider running "sudo apt-get update" to update Ubuntu in your VM (if using your personal machine) before you execute the code.

When the script is executed, it will first create the illustrated topology, assign IP addresses, setup NAT (which allows the topology to connect with the Internet), etc. The script also invokes the good DHCP server but does not start the malicious servers on the evil host in the first instance.

Run the provided script by typing (we also illustrate the output that you should see):

```
# cd ~/lab5exercise1
# sudo -E ./dhcp.py
*** Creating network
*** Adding controller
*** Adding hosts:
dhcp evil h1
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 500ms delay) (10.00Mbit 500ms delay) (dhcp, s1) (evil, s1) (h1, s1)
*** Configuring hosts
dhcp evil h1
*** Starting controller
*** Starting 1 switches
s1 (10.00Mbit 500ms delay)
* Starting DHCP server on dhcp at 10.0.0.50
* h1 waiting for IP address...
* h1 is now using nameserver 8.8.8.8
* Fetching google.com:
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
*** You may want to do some DNS lookups using dig
*** Please go to amazon.com in Firefox
*** You may also wish to start up wireshark and look at bootp and/or dns
*** Press return to start up evil DHCP server:
```

If everything has worked correctly, you will see an xterm window for h1 as well as a Firefox window,, which should have loaded www.cse.unsw.edu.au. You can load other websites such as amazon.com if you like.

Next, in the terminal window for host h1, run wireshark by typing “wireshark”. Ignore any warnings that you may encounter. Select h1-eth0 in the interface list and press start (green button). Filter the traffic so that you only observe the bootp traffic (type “bootp” in the filter window at the top and press enter). You should observe a number of DHCP packets.

Question 1: Select one of the DHCP ACK packets received by host h1 and report all the important DHCP related details (e.g. yiaddr, subnet mask, lease time, router, etc.)? Which IP address is responding to the DHCP request generated by host h1?

Question 2: Now change the filter in wireshark to DNS. Examine the captured traffic. Which DNS server is responding to the DNS requests for www.cse.unsw.edu.au and www.amazon.com (assuming you loaded that page in firefox)? How did you determine this?

Now press return (in the terminal running the script) to start up the malicious DHCP server. You should observe the following messages on your terminal (you may need to wait for a few seconds):

```
Starting DHCP server on evil at 10.0.0.66
* Starting fake DNS server evil at 10.0.0.66
* Starting web server evil at 10.0.0.66
* h1 waiting for IP address.....
* h1 is now using nameserver 10.0.0.66
* New DNS result:
google.com has address 10.0.0.66
*** You may wish to look at DHCP and DNS results in wireshark
*** You may also wish to do some DNS lookups using dig
*** Please go to google.com in Firefox
*** You may also want to try going back to amazon.com and hitting shift-refresh
*** Press return to shut down evil DHCP/DNS/Web servers:
```

Next, you can try some DNS lookups (e.g. using dig in the terminal for h1) or try to load different websites in Firefox. You may have to press shift-refresh to cause Firefox to make a new DNS request. Observe that the malicious DNS server replies with the address of the malicious web server, causing Firefox to load the malicious web page instead of the desired page

Question 3: By analysing the captured traffic in Wireshark, explain the details of the attack by precisely outlining the sequence of packets exchanged between all the relevant entities (i.e. who sends what to who and who responds and what happens as a result). Be as elaborate as you can.

Finally, press return to shut down the evil DHCP/DNS/web servers. You may see the following output on your terminal:

```
* Stopping fake DNS server evil at 10.0.0.66
* Stopping web server evil at 10.0.0.66
* Stopping DHCP server on evil at 10.0.0.66
*** Try going to some other web sites if you like
*** Press return to exit:
```

After the malicious servers are shut down, things return to normal - DHCP requests go to the good DHCP server, which responds with the good DNS server address, which responds with correct DNS lookups. You can try navigating to different websites in Firefox.

Press return to exit the demo.

Question 4: Can you provide a solution to prevent the DHCP masquerade attack? Explain your solution in maximum one paragraph.

Remember to clean up mininet before proceeding to the next exercise.

Exercise 2: Switch table overflow attack

IMPORTANT NOTE: I HAD TROUBLE GETTING THIS TO WORK ON THE VM ON MY PERSONAL MACHINE. HOWEVER, THIS EXERCISE WORKS PERFECTLY FINE ON LAB MACHINES. THUS, WE ENCOURAGE THAT YOU COMPLETE THIS ON A LAB MACHINE IF YOU ENCOUNTER PROBLEMS.

In this demo you will observe how a switch table overflow attack (or switch poisoning) can be used to eavesdrop on network traffic. We will use the topology illustrated in Figure 2. Eve wishes to eavesdrop on the messages exchanged between Alice and Bob are exchanging messages. Eve will achieve this by overflowing the MAC address tables in the two switches (s0 and s1). Note that, this attack would even if Eve is not connected to the same switch used by Alice and Bob.

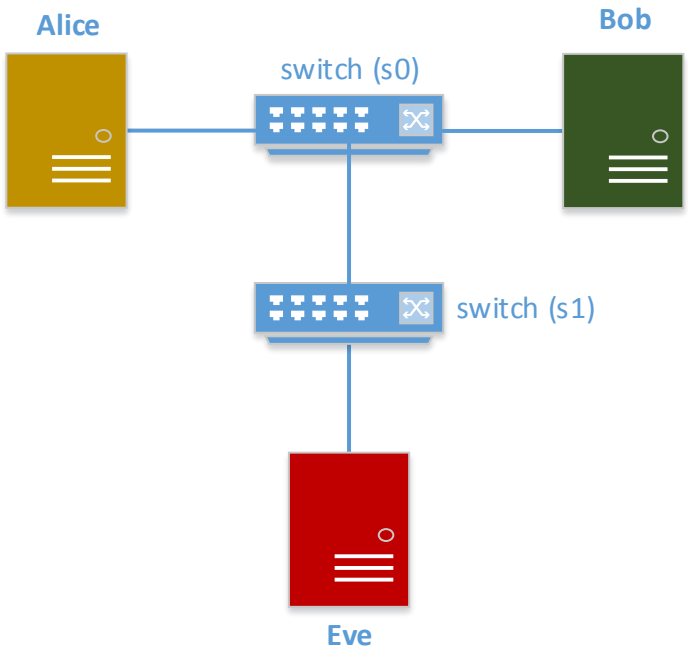

Figure 2 Network topology for MAC address table overflow attack.

Environment Setup

This exercises uses Impacket (https://www.coresecurity.com/corelabs-research/open-source-tools/impacket), which is a collection of Python classes for working with network protocols. This package is installed on the lab machine mininet VM. However, if you are working on your personal machine, you may need to download and install this package.

```
# wget http://www.cse.unsw.edu.au/~cs3331/16s1/Labs/impacket-0.9.14.tar.gz
# tar -xvf impacket-0.9.14.tar.gz
# cd impacket-0.9.14
# sudo python setup.py install
```

First make sure that you have cleaned up mininet

```
# sudo mn-c
```

We have provided scripts to set up the Mininet topology in Figure 2. This exercises use the POX controller (http://www.noxrepo.org/pox/about-pox/) for emulating the behaviour of a L2 (layer 2) learning switch. We also provided the scripts to set up POX and interface it with the topology. Log in to your mounted home directory and download the exercise code and open the archive by typing:

```
# wget http://www.cse.unsw.edu.au/~cs3331/16s1/Labs/lab5exercise2.tgz
# tar -xvf lab5exercise2.tgz
# cd lab5exercise2
```

Symlink POX and configure the POX modules by typing:

```
# ln -s ~/pox/
# ./config.sh
```

If you encounter any errors try updating the following and retry:

```
# sudo apt-get update
# sudo apt-get install python-dev
```

Attack Demonstration

Use the following commands to run the POX network controller:

```
# ./run_pox.sh
```

Run the following command in another terminal (make sure that X-forwarding is enabled, i.e. ssh –Y mininet@mininet):

```
# cd ~/lab5exercise2
# ./run.sh
```

This will start the Mininet network emulator and a number of terminals will be opened – one for each node in the network. Close the terminals for switches (s0, s1) and controllers (c0), but keep the terminals for Alice, Bob and Eve open.

In Alice's Terminal

Alice will now create some traffic by pinging Bob:

```
alice@10.0.0.1:~/lab5exercise2$ ping 10.0.0.2
```

You should be able to see some output like the following:

```
alice@10.0.0.1:~/lab5exercise2$ ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=100 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=98.2 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=96.7 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=58.5 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=56.3 ms
...
```

In Eve's Terminal

We will now run tcpdump to eavesdrop the traffic between Alice (10.0.0.1) and Bob (10.0.0.2). tcpdump is a command line packet analyser that allows the user to display TCP/IP and other packets being received over a network. More information at: https://en.wikipedia.org/wiki/Tcpdump

```
eve@10.0.0.3:~/lab5exercise2$ sudo tcpdump -n host 10.0.0.1
```

Question 5: Based on what you observe in response to the above command, do you think that Eve can eavesdrop on the ping traffic between Alice and Bob?

Now, we will let Eve generate some Ethernet packets with randomly generated source MAC address to overflow the switch tables of the switches. To do so, let's create another for Eve by typing “xterm eve” in the Mininet CLI. Then run the following command in the new terminal:

```
eve@10.0.0.3:~/lab5exercise2$ python attack.py
```

You should be able to see that Eve starts sending a lot of packets into the network.

Question 6: Investigate these packets by running tcpdump on another terminal and explain the packets that are being generated by Eve to lauch the attack.

Go Back to Eve's first terminal. After a while you should be able to see the packets sent by Alice to Bob in Eve's tcpdump trace:

```
eve@10.0.0.3:~/lab5exercise2$ sudo tcpdump -n host 10.0.0.1
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eve-eth0, link-type EN10MB (Ethernet), capture size
65535 bytes
19:25:27.244766 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length
28
19:25:29.241754 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 3, length 64
19:25:30.247182 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 4, length 64
19:25:31.245341 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 5, length 64
19:25:32.244768 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 6, length 64
19:25:33.250965 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 7, length 64
19:25:34.248849 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 8, length 64
19:25:35.249185 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 9, length 64
19:25:36.254024 IP 10.0.0.1 > 10.0.0.2: ICMP echo request, id 5031,
seq 10, length 64
```

Question 7: Explain the above observation. Why does the switch broadcast the ping packets sent by Alice to Bob?

Question 8: Can you provide a solution to prevent this attack? Explain your solution in maximum one paragraph. It may be useful to examine the “pox_module/comp9331/l2_learning.py” script which implements the L2 switch on POX.