Total: 100 points

1 Overview

The learning objective of this lab is for students to gain first-hand experience on vulnerabilities, as well as on attacks against these vulnerabilities. Wise people learn from mistakes. In security education, we study mistakes that lead to software vulnerabilities. Studying mistakes from the past not only help students understand why systems are vulnerable, why a seemly-benign mistake can turn into a disaster, and why many security mechanisms are needed. More importantly, it also helps students learn the common patterns of vulnerabilities, so they can avoid making similar mistakes in the future. Moreover, using vulnerabilities as case studies, students can learn the principles of secure design, secure programming, and security testing.

The vulnerabilities in the TCP/IP protocols represent a special genre of vulnerabilities in protocol designs and implementations; they provide an invaluable lesson as to why security should be designed in from the beginning, rather than being added as an afterthought. Moreover, studying these vulnerabilities help students understand the challenges of network security and why many network security measures are needed. In this lab, students will conduct several attacks on TCP. This lab covers the following topics:

1. TCP SYN flood attack, and SYN cookies
2. TCP reset attack
3. TCP session hijacking attack

Readings and videos. More details of the TCP attacks can also be found in the following:

- Chapter 16 of the SEED Book, Computer & Internet Security: A Hands-on Approach, 2nd Edition, by Wenliang Du. See details at https://www.handsonsecurity.net.
- Section 6 of the SEED Lecture, Internet Security: A Hands-on Approach, byWenliang Du. See details at https://www.handsonsecurity.net/video.html.

Lab environment. This lab has been tested on our pre-built Ubuntu 16.04 VM, which can be downloaded from the SEED website.

2 Lab Environment

Students will use one VM but different terminals to simulate the 1) Attacker, 2) Victim, and 3) Sniffer. You can change the terminal prompt name using the `export` command, e.g.,

export PS1="XXXXX$ ", to differentiate different terminals on the same host. (better to keep a space after the $ symbol)

Netwox Tools. We need tools to send out network packets of different types and with different contents. We can use Netwag to do that. However, the GUI interface of Netwag makes it difficult for us to automate the process. Therefore, we strongly suggest students to use its command-line version, the Netwox command, which is the underlying command invoked by Netwag.

Netwox consists of a suite of tools, each having a specific number. You can run a command like following (the parameters depend on which tool you are using). For some of the tool, you have to run it with the root privilege:

```
$ sudo netwox number [parameters ...  ]
```

If you are not sure how to set the parameters, you can look at the manual by issuing "netwox number –help". You can also learn the parameter settings by running Netwag: for each command you execute from the graphic interface, Netwag actually invokes a corresponding Netwox command, and it displays the parameter settings. Therefore, you can simply copy and paste the displayed command.

Scapy Tool. Some of the tasks in this lab can also be conducted using Scapy, which is a powerful interactive packet manipulation program. Scapy is very well maintained and is widely used; while Netwox is not being maintained any more. There are many online tutorials on Scapy; we expect students to learn how to use Scapy from those tutorials.

3 Lab Tasks

In this lab, students need to conduct attacks on the TCP/IP protocols. They can use the Netwox tools and/or other tools in the attacks. All the attacks are performed on Linux operating systems. However, instructors can require students to also conduct the same attacks on other operating systems and compare the observations.

To simplify the "guess" of TCP sequence numbers and source port numbers, we assume that attackers are on the same physical network as the victims. Therefore, you can use sniffer tools to get that information. The following is the list of attacks that need to be implemented.

3.1 Task 1: SYN Flooding Attack (30 points)

SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure. Attackers either use spoofed IP address or do not continue the procedure. Through this attack, attackers can flood the victim's queue that is used for half-opened connections, i.e. the connections that has finished SYN, SYN-ACK, but has not yet gotten a final ACK back. When this queue is full, the victim cannot take any more connection. Figure 2 illustrates the

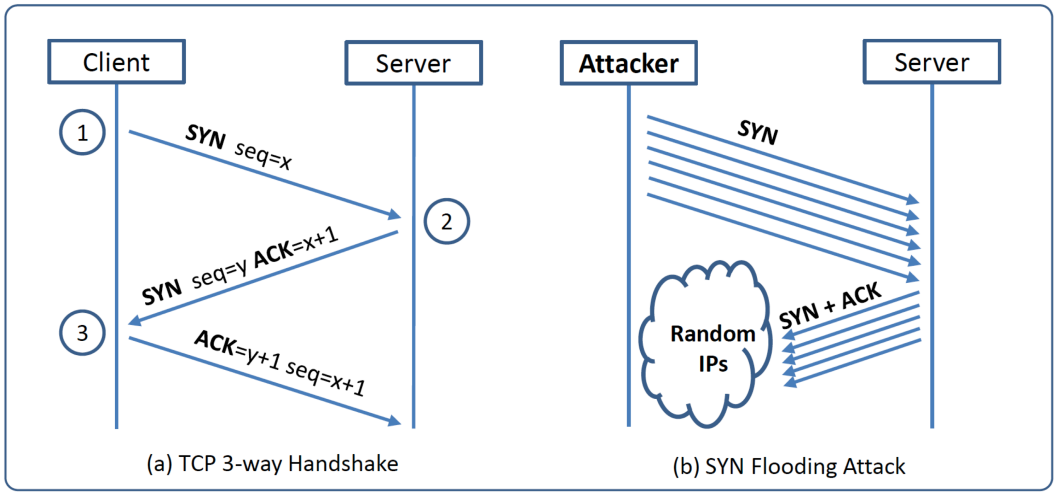

Figure 1: SYN Flooding Attack

attack.

The size of the queue has a system-wide setting. In Linux, we can check the setting using the following command:

```
$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
```

We can use command "netstat -na" to check the usage of the queue, i.e., the number of half-opened connection associated with a listening port. The state for such connections is SYN-RECV. If the 3-way handshake is finished, the state of the connections will be ESTABLISHED.

In this task, you need to demonstrate the SYN flooding attack. You can use the Netwox tool to conduct the attack, and then use a sniffer tool to capture the attacking packets. While the attack is going on, run the "netstat -na" command on the victim machine, and compare the result with that before the attack. **Also try telnet/ssh to the victim machine while the attack is going on.** Please also describe how you know whether the attack is successful or not.

The corresponding Netwox tool for this task is numbered 76. Here is a simple help screen for this tool. You can also type "netwox 76 –help" to get the help information.

```
Title:  Synflood
    Usage: netwox 76 -i ip -p port [-s spoofip]
    Parameters:
    -i|--dst-ip ip              destination IP address
    -p|--dst-port port          destination port number
    -s|--spoofip spoofip        IP spoof initialzation type
```

Figure 2: The usage of the Netwox Tool 76

SYN Cookie Countermeasure: If your attack seems unsuccessful, one thing that you can investigate is whether the SYN cookie mechanism is turned on. SYN cookie is a defense mechanism to counter the SYN flooding attack. The mechanism will kick in if the machine

detects that it is under the SYN flooding attack.

You can use the sysctl command to turn on/off the SYN cookie mechanism:

$ sudo sysctl -a | grep cookie (Display the SYN cookie flag)
$ sudo sysctl -w net.ipv4.tcp_syncookies=0 (turn off SYN cookie)
$ sudo sysctl -w net.ipv4.tcp_syncookies=1 (turn on SYN cookie)

Please run your attacks with the SYN cookie mechanism on and off, and compare the results (with screenshots). In your report, please describe why the SYN cookie can effectively protect the machine against the SYN flooding attack. If your instructor does not cover the mechanism in the lecture, you can find out how the SYN cookie mechanism works from the Internet.

Note on Scapy: Although theoretically, we can use Scapy for this task, we have observed that the number of packets sent out by Scapy per second is much smaller than that by Netwox. This low rate makes it difficult for the attack to be successful. We were not able to succeed in SYN flooding attacks using Scapy.

3.2 Task 2: TCP RST Attacks on telnet and ssh Connections (40 points)

The TCP RST Attack can terminate an established TCP connection between two victims. For example, if there is an established telnet connection (TCP) between two users A and B, attackers can spoof a RST packet from A to B, breaking this existing connection. To succeed in this attack, attackers need to correctly construct the TCP RST packet. In this task, you need to launch an TCP RST attack to break an existing telnet connection between A and B. After that, try the same attack on an ssh connection. Please describe your observations. To simplify the lab, we assume that the attacker and the victim are on the same LAN, i.e., the attacker can observe the TCP traffic between A and B.

```
Title:  Spoof Ip4Tcp packet
    Usage: netwox 40 [parameters ...]
    Parameters:
    -l|--ip4-src ip                Source IP
    -m|--ip4-dst ip                Destination IP
    -j|--ip4-ttl uint32            Time to live
    -o|--tcp-src port              TCP Source port number
    -p|--tcp-dst port              TCP Destination port number
    -q|--tcp-seqnum uint32         TCP sequence number
    -E|--tcp-window uint32         TCP window size
    -r|--tcp-acknum uint32         TCP acknowledge number
    -z|--tcp-ack|+z|--no-tcp-ack   TCP ack bit
    -H|--tcp-data data             TCP data
```

Figure 3: The usage of the Netwox Tool 40

Using Netwox. The corresponding Netwox tool for this task is numbered 40. Here is part of the manual for this tool, as shown in Figure 3. You can also type "netwox 40 –help" to

get the full help information. You may also need to use Wireshark to find out the correct parameters for building the spoofed TCP packet. (**Also see our lecture slides**)

Using Scapy. Please also use Scapy to conduct the TCP RST attack. A skeleton code is provided in the following (you need to replace each @@@@ with an actual value):

Code 1: Scapy code.

```
#!/usr/bin/python
import sys
from scapy.all import *

print("SENDING RESET PACKET.........")
IPLayer = IP(src="@@@@", dst="@@@@")
TCPLayer = TCP(sport=@@@@, dport=@@@@, flags="R", seq=@@@@)
pkt = IPLayer/TCPLayer
ls(pkt)
send(pkt, verbose=0)
```

In your report, please describe your experiment results (with screenshots) using the Netwox tool and Scapy script, respectively. (hint: if the attack is successful, the telnet connection will be closed by the attacker, and you will see "Connection closed by foreign host." on the telnet client side.)

3.3 Task 3: TCP Session Hijacking on Telnet Connection (40 points)

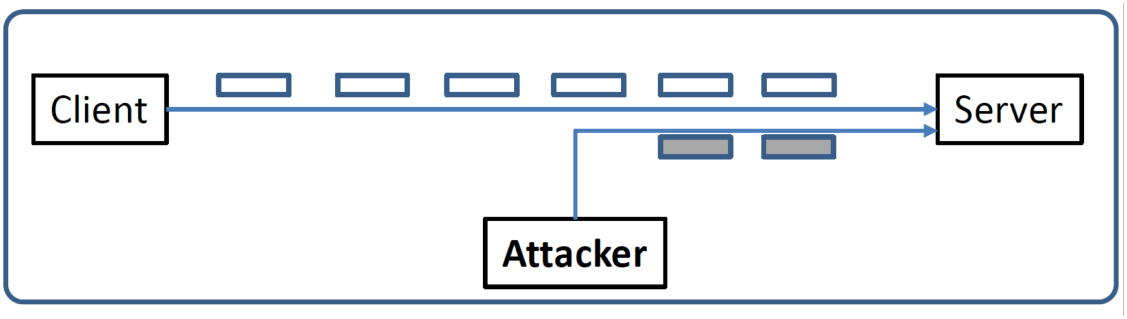

Figure 4: TCP Session Hijacking Attack

The objective of the TCP Session Hijacking attack is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into this session. If this connection is a telnet session, attackers can inject malicious commands into this session, causing the victims to execute the malicious commands. In this task, attackers want to delete an important file by executing "rm /home/seed/myfile.txt" on the victim machine. You first create the file with command "touch /home/seed/myfile.txt". Then, after the attack, when you "ll /home/seed/myfile.txt", it should show "no such file" on the victim machine.

Figure 4 depicts how the attack works. In this task, you need to demonstrate how you can hijack a telnet session between two computers. Your goal is to get the the telnet server to run a malicious command from you. For the simplicity of the task, we assume that the attacker and the victim are on the same LAN.

Using Netwox. The corresponding Netwox tool for this task is numbered 40. (see Figure 3 and our lecture slides.)

You can use Wireshark to figure out what value you should put into each field of the spoofed TCP packets. All the fields that need to be set are listed in Figure 3.

In the netwox command above, the tcp-data part only takes hex data. If we want to inject a command string, which is typically represented as a human-readable ASCII string, we need to convert it into a hex string. There are many ways to do that, but we will just use a very simple command in Python. In the following, we convert an ASCII string "rm /home/seed/myfile.txt" to a hex string (the quotation marks are not included).

$ python
>>> "rm /home/seed/myfile.txt".encode("hex")
'0a20726d202f686f6d652f736565642f6d7966696c652e7478740a'

Using Scapy. Please also use Scapy to conduct the TCP RST attack. A skeleton code is provided in the following (you need to replace each @@@@ with an actual value):

Code 2: Scapy code for TCP Session Hijacking Attack.

```
#!/usr/bin/python
import sys
from scapy.all import *

print("SENDING SESSION HIJACKING PACKET.....")
IPLayer = IP(src="@@@@", dst="@@@@")
TCPLayer = TCP(sport=@@@@, dport=@@@@, flags="A",seq=@@@@,ack=@@@@)
Data = "\n rm /home/seed/myfile.txt\n"
pkt = IPLayer/TCPLayer/Data
ls(pkt)
send(pkt,verbose=0)
```

In your report, please describe your experiment results (with screenshots) using the Netwox tool and Scapy script, respectively.