

TCP SYN Flood Exercise

Offensive Technologies (145519)

Francisco Manuel Colmenar Lamas
219757

Trento, October 2020

Contents

| | | |
|----------|--|----------|
| 1 | TCP SYN Flood Exercise's Instructions | 5 |
| 1.1 | Introduction | 5 |
| 1.2 | Set up | 5 |
| 1.3 | Attack and Capture | 8 |
| 1.4 | Analysis instructions | 11 |
| 1.4.1 | Csv file generation | 11 |
| 1.4.2 | Graph creation | 12 |

List of Figures

| | | |
|------|--|----|
| 1.1 | Sftp command for login. | 5 |
| 1.2 | Sftp command for sending the scripts to the nodes. | 5 |
| 1.3 | Ssh command to access the nodes from the exercise. | 6 |
| 1.4 | Command to install Apache at the server node. | 6 |
| 1.5 | Command to install the flooder at the attacker node. | 6 |
| 1.6 | Command to check the SYN cookies at the server node. | 6 |
| 1.7 | Command to disable the SYN cookies at the server node. | 7 |
| 1.8 | Output from the script <code>get_ip.sh</code> | 7 |
| 1.9 | Running the script <code>send_traffic</code> using the <code>-i</code> flag. | 7 |
| 1.10 | Running the script <code>send_traffic</code> without flags. | 8 |
| 1.11 | Call to the flooder belonging to the script <code>scripts/traffic/attack.sh</code> | 9 |
| 1.12 | Curl call used to send normal traffic from the client to the server. | 9 |
| 1.13 | Tcpdump command from the script <code>scripts/stopwatch/capture.sh</code> | 9 |
| 1.14 | Setting to run the attack and to capture the traffic. | 10 |
| 1.15 | Copying the pcap file from the client node to the personal computer. | 10 |

1. TCP SYN FLOOD EXERCISE'S INSTRUCTIONS

1.1. Introduction

In this exercise the concept of *TCP SYN Flood* is going to be analyzed in order to get a better understanding of it.

1.2. Set up

Firstly, to interact with the nodes of the experiment in an efficient way it is recommended to have five different terminal window interfaces. One of them is using *sftp* while the other four terminals are for interacting with the different nodes of the exercise.

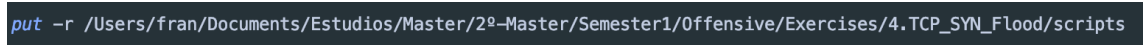
To obtain these five different terminal windows, the commands from the file *guide.sh* are going to be used. Firstly, at the terminal window which is going to use *sftp* the following line of code is going to be executed.

A terminal window with a dark background. The first line is a comment: "##### SFTP login and Sending the Scripts #####". The second line is the command "sftp otech2ae@users.deterlab.net" in blue text.

```
##### SFTP login and Sending the Scripts #####  
sftp otech2ae@users.deterlab.net
```

Figure 1.1: Sftp command for login.

Furthermore, in order to send the scripts which are going to be used in this exercise the next command should be run. Note that the path needs to be changed according to the location of the folder.

A terminal window with a dark background. The command "put -r /Users/fran/Documents/Estudios/Master/2º-Master/Semester1/Offensive/Exercises/4.TCP_SYN_Flood/scripts" is shown in blue text.

```
put -r /Users/fran/Documents/Estudios/Master/2º-Master/Semester1/Offensive/Exercises/4.TCP_SYN_Flood/scripts
```

Figure 1.2: Sftp command for sending the scripts to the nodes.

In addition to this, to use the other four terminals it is needed to be performed a "double ssh" into the server, attacker and client node, in the last one two interfaces are

going to be used. In order to accomplish it, the following commands should be run, one in each terminal window and the last command should be executed into two different windows. Note that the username should be changed to the one with access to.

```
##### SSH Login Commands #####  
ssh -tt otech2ae@users.deterlab.net 'ssh -tt server.TCP.OffTech 'sudo -s'  
ssh -tt otech2ae@users.deterlab.net 'ssh -tt attacker.TCP.OffTech 'sudo -s'  
ssh -tt otech2ae@users.deterlab.net 'ssh -tt client.TCP.OffTech 'sudo -s'
```

Figure 1.3: Ssh command to access the nodes from the exercise.

Once that all the interface terminals are obtained, the files needed for the server and the attacker are going to be installed. According to the server, Apache is going to be installed using the following command.

```
##### Install Apache - on the SERVER #####  
cd /share/education/TCP SYN Flood_USC_ISI/ && ./install-server
```

Figure 1.4: Command to install Apache at the server node.

Moreover, the flooder which is going to produce the TCP SYN Flood attack is also needed to be installed but at the attacker node. The command below is used for this purpose.

```
##### Install the flooder - on the ATTACKER #####  
cd /share/education/TCP SYN Flood_USC_ISI/ && ./install-flooder
```

Figure 1.5: Command to install the flooder at the attacker node.

Then, the SYN cookies should be checked in order to know if they are enabled or not. The following command, which should be executed at the server node, just achieve this result.

```
##### Check if the SYN cookies are enabled - on the SERVER #####  
sudo sysctl net.ipv4.tcp_syncookies
```

Figure 1.6: Command to check the SYN cookies at the server node.

In the case that the cookies are set to 1 and it is desired that they are disabled, the command which is going to be described below should be executed.

```
##### Set the cookies - on the SERVER #####  
sudo sysctl -w net.ipv4.tcp_syncookies=0 && sysctl -w net.ipv4.tcp_max_syn_backlog=10000
```

Figure 1.7: Command to disable the SYN cookies at the server node.

Finally, to check that all the steps of the set up has been performed correctly and that there is no issue with the server, the scripts *scripts/set_up/get_ip.sh* and *scripts/traffic/send_traffic.sh* are going to be used.

The script *scripts/set_up/get_ip.sh* just runs an ip route get command to the server. The output of this script should be similar to the following picture.

```
root@client:~# ./scripts/set_up/get_ip.sh  
5.6.7.8 via 1.1.2.2 dev eth0 src 1.1.2.3 uid 0  
cache
```

Figure 1.8: Output from the script *get_ip.sh*.

As it can be seen at the above picture, the output shows the interface which is used to send the request. This interface is going to be used in the following steps so it is recommended to write it down. Note that the interface could change each time that the experiment is swaped in.

Finally, in order to check that the server is working correctly the script *scripts/traffic/send_traffic.sh* is going to be run at the client node. This script sends each second a curl request for the index page of the server. The script receives as a parameter the interface which is going to be used by writing the flag *-i* and then the interface. If not, the script will ask for the interface interactively to the user.

```
root@client:~# ./scripts/traffic/send_traffic.sh -i eth0
```

Figure 1.9: Running the script *send_traffic* using the *-i* flag.

```
root@client:~# ./scripts/traffic/send_traffic.sh
What interface do you want to use?
eth0
```

Figure 1.10: Running the script `send_traffic` without flags.

Therefore, if the set up has been accomplished correctly the index HTML document should be displayed at the window terminal of the client node which has executed the *send_traffic* script.

Note that all the scripts from this exercises are supposed to be run from the folder */users/otech2ae/*, therefore calling the scripts from outside the *scripts/* folder.

1.3. Attack and Capture

Once that the server is running correctly and that the client is able to connect to it, the TCP SYN flood attack is going to be performed according to the specification of the exercise statement. Also, the traffic generated is going to be captured.

In order to accomplish this step three window terminals are needed: one window for the attacker node and two for the client (one for capturing the traffic and another for sending the normal traffic).

At the attacker node the script *scripts/stopwatch/attack_stopwatch.sh* is going to be executed. The default highrate use for this script is 100. In the case that another rate is desired to be the rate used, the flag *-r* can be used followed by the new highrate. Note that *attack_stopwatch.sh* handles the calls to the script *scripts/traffic/attack.sh*, according to the timing specification from the statement, which is the one containing the *flooder* call.

Furthermore, the main code segment of the script is the call to the *flooder* which is the following one.

```
flooder --dst 5.6.7.8 --highrate "$high_rate" --proto 6 --dportmin 80 --dportmax 80 --src 1.1.2.4 --srcmask 255.255.255.0
```

Figure 1.11: Call to the flooder belonging to the script *scripts/traffic/attack.sh*.

In addition to this, the script which is going to be run at the client node's window which is going to send normal traffic to the server is *scripts/stopwatch/normal_stopwatch.sh*. This script generates normal traffic from the client node for the server using a curl call from the script *scripts/traffic/send_traffic.sh* previously explained. The curl code executed is the following one.

```
while sleep 1; do curl --interface "$Interface" -x 5.6.7.8:80 index.html & done
```

Figure 1.12: Curl call used to send normal traffic from the client to the server.

Note that the script *scripts/stopwatch/normal_stopwatch.sh* uses the same flag for the interface as *scripts/traffic/send_traffic.sh*.

Furthermore, the last script to be explained before the attack and the capture is performed is *scripts/stopwatch/capture.sh*. This script just captures using tcpdump on the given interface. Also, the name of the output file can be modified using the flag *-w* and adding the name of the file as a parameter. The code belonging to the tcpdump call is the following one.

```
cd /tmp/  
sudo timeout "$timeout_num" tcpdump -s 0 -nn -i "$Interface" -w "$file_name" port 80
```

Figure 1.13: Tcpdump command from the script *scripts/stopwatch/capture.sh*.

Once that all the scripts are explained and it is known how to execute them, the order of the calls to run the attack correctly is going to be described. The recommended configuration for the windows is the following one.

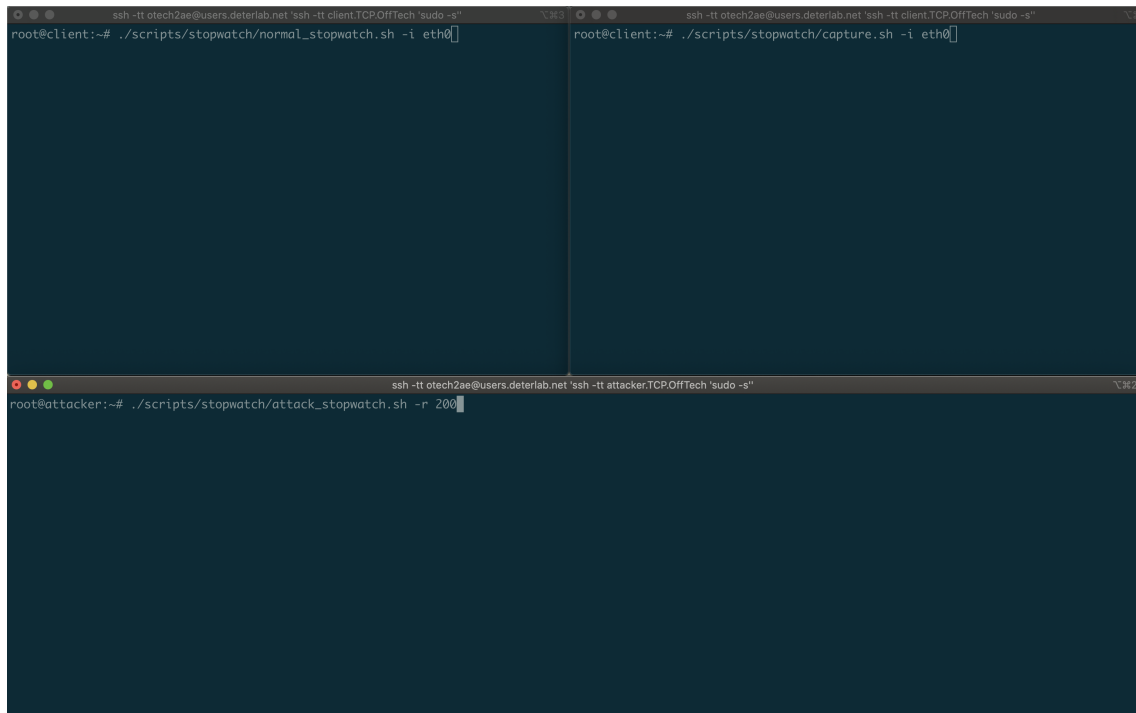


Figure 1.14: Setting to run the attack and to capture the traffic.

The first script to be run is *scripts/stopwatch/capture.sh*, then *scripts/stopwatch/normal_stopwatch.sh* is executed and finally *attack_stopwatch.sh*. Note that all the scripts are needed to be run at the same time following the previous explained order and with the smallest time possible between the start of the execution of one script and the next one.

Finally the pcap file generated is located in the folder */tmp/*. In the case that it is desired to copy the pcap file for treating it with the python code, which is going to be explained at the following section, the next command can be executed at the sftp window.

```
get output.pcap /Users/fran/Documents/Estudios/Master/2º-Master/Semester1/Offensive/Offensive-Technologies/TcpSynFlood/pcap
```

Figure 1.15: Copying the pcap file from the client node to the personal computer.

Note that the path from the above command should be changed to the desired destination folder and the output.pcap file should be copied from the folder */tmp/* to the folder from the user, such as */users/otech2ae/*.

1.4. Analysis instructions

In this last section a brief description on how to use the python code used for the analysis of the pcap files is going to be provided.

The python code is developed as a CLI which receives different flags and parameters. There are two main functionalities which it supports: the creation of the csv files from the pcap files and the generation of the graphs using the csv files.

1.4.1. Csv file generation

In order to generate the csv files from the pcap files the following command should be used.

```
python main csv
```

The above command runs the csv option as its default settings. It searches for a file called *output.pcap* and the folder *pcap/* and stores the file *output.csv* at the folder *csv/*. However, the name of the input pcap file to be searched and processed can be specified using the following command instead. Note that in this case also the csv file takes the input file name.

```
python main csv -pcap <Name>
```

Furthermore, in the case that only the name of the csv file generated is desired to be changed the command below can be used.

```
python main csv -csv <Name>
```

Finally, the last option which can be used with the *csv* command is the possibility to generate the csv files of all the pcap files from the *pcap/* folder. In order to accomplish it the next command can be used.

```
python main csv -a
```

1.4.2. Graph creation

To create the graphs from the csv files previously created the next command can be used.

```
python main graph
```

In this case the CLI does not accept any specific flag or parameter to modify the default behaviour of the graph creation. This command runs the file *src/graph/createGraph.py* which behaves similar to a custom API graph creation, which can be easily modified through the files *src/constant/const.py* and *src/constant/graph_const.py*.

The resulting graphs with the default behaviour are stored into the folder *img/line_disc*. Note that in the current version the graph creation is set to use just the csv files currently located at the csv folders. In the case that it is desired to use a new csv file the files *src/constant/const.py* and *src/constant/graph_const.py* should be changed to introduce the correct file name and the graph names.