



ADDIS ABABA UNIVERSITY

AAiT - SITE

Cyber Security Stream

The Mitnick Attack Lab

PROJECT PROGRESS

GROUP MEMBERS

NAOL KECHAUGR/6903/12

ZENAMARKOS MOLLA.....UGR/4176/12

DINKA EFAUGR/3476/12

Project progress up to Jan 16, 2023

Submitted to Mr. Daniel

The Mitnick Attack Lab

Introduction

Kevin Mitnick is probably one of the most well-known hackers in USA. He was on FBI's wanted list of criminals.

In 1994, Mitnick successfully launched an attack on Shimomura's computer, by exploiting the vulnerabilities in the TCP protocol and the trusted relationship between two of Shimomura's computers. The attack triggered a dramatic showdown between the two people, and it eventually led to the arrest of Mitnick. The showdown was turned into books and Hollywood movies later. The attack is now known as the Mitnick attack, which is a special type of TCP session hijacking.

In this lab we will cover the following topics:

- TCP session hijacking attack
- TCP three-way handshake protocol
- The Mitnick attack
- Remote shell rsh
- Packet sniffing and spoofing

How Mitnick Attack works

In the actual Mitnick attack, host A was called X-Terminal, which was the target. Mitnick wanted to log into X-Terminal and run his commands on it. Host B was a trusted server, which was allowed to log into X-Terminal without a password. In order to log into X-Terminal, Mitnick had to impersonate the trusted server, so he did not need to provide any password. Figure below depicts the high-level picture of the attack. There are four primary steps in this attack.

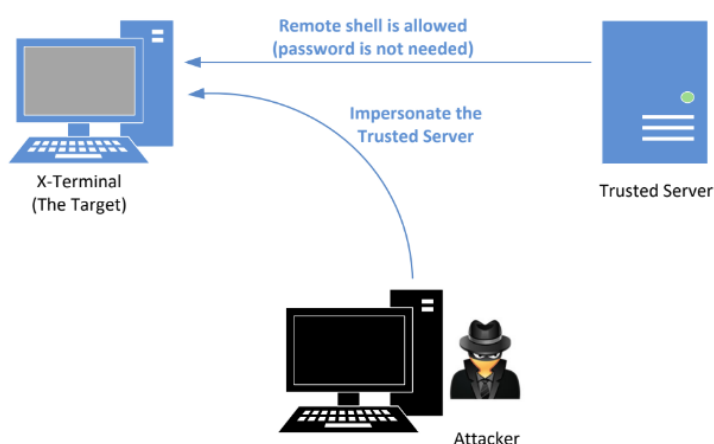


Fig 1: The illustration of the mitnick attack

- Step 1: Sequence number prediction.
- Step 2: SYN flooding attack on the trusted server.
- Step 3: Spoofing a TCP connection.
- Step 4: Running a remote shell.

Lab Environment Setup

In this lab, we need three machines, one for X-Terminal, one for Trusted Server, and the other for the attacker. The reason why we choose three virtual machines rather than using a docker container in one machine is that when we use a separate virtual machine it is more convenient to use python script codes. So, We used three virtual machines as shown in figure below.

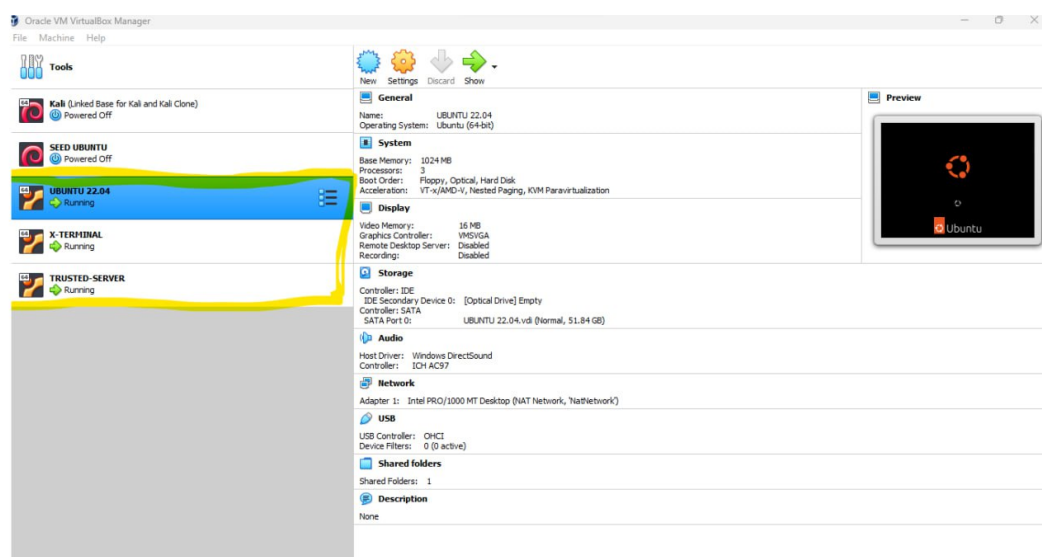
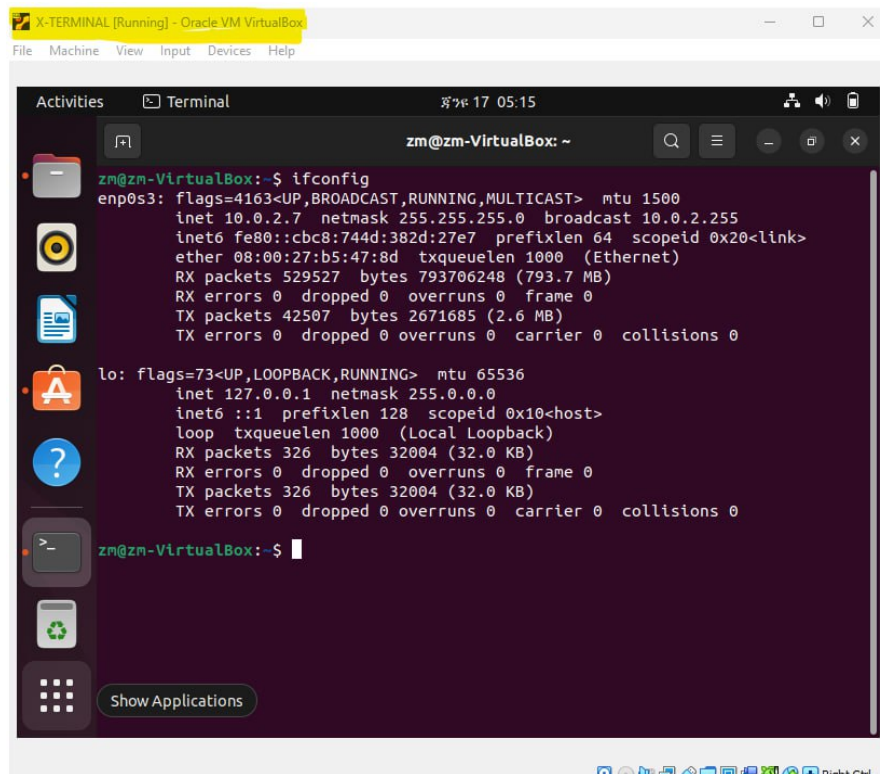


Fig 2: Our virtual machine setup



```

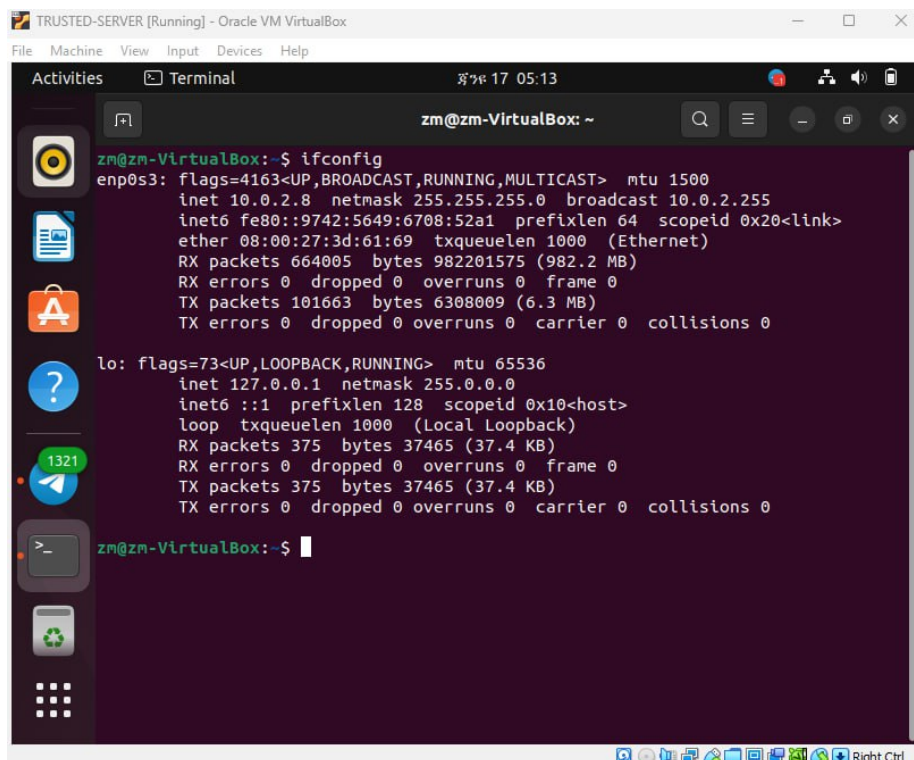
zm@zm-VirtualBox: ~
zm@zm-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.7 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::cbc8:744d:382d:27e7 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:b5:47:8d txqueuelen 1000 (Ethernet)
        RX packets 529527 bytes 793706248 (793.7 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 42507 bytes 2671685 (2.6 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 326 bytes 32004 (32.0 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 326 bytes 32004 (32.0 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zm@zm-VirtualBox:~$

```

Fig 3: X-terminal Virtual machine



```

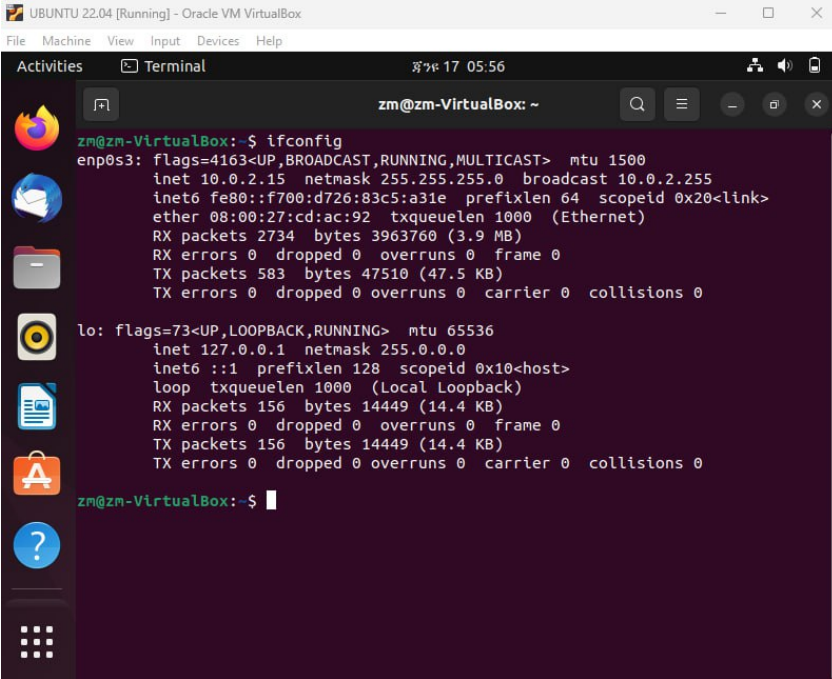
zm@zm-VirtualBox: ~
zm@zm-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.8 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::9742:5649:6708:52a1 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:3d:61:69 txqueuelen 1000 (Ethernet)
        RX packets 664005 bytes 982201575 (982.2 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 101663 bytes 6308009 (6.3 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 375 bytes 37465 (37.4 KB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 375 bytes 37465 (37.4 KB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

zm@zm-VirtualBox:~$

```

Fig 4: Trusted-server Virtual machine



```

UBUNTU 22.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal 17 05:56
zm@zm-VirtualBox: ~
zm@zm-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::f700:d726:83c5:a31e prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:cd:ac:92 txqueuelen 1000 (Ethernet)
    RX packets 2734 bytes 3963760 (3.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 583 bytes 47510 (47.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 156 bytes 14449 (14.4 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 156 bytes 14449 (14.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

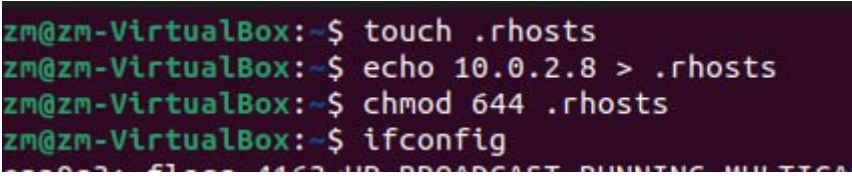
zm@zm-VirtualBox:~$

```

Fig 5: Ubuntu 22.04 VM

Configuration

The rsh server program uses two files for authentication, *.rhosts* and */etc/hosts.equiv*. We set up the *.rhosts* file on X-Terminal using commands shown in the *screenshot* below.



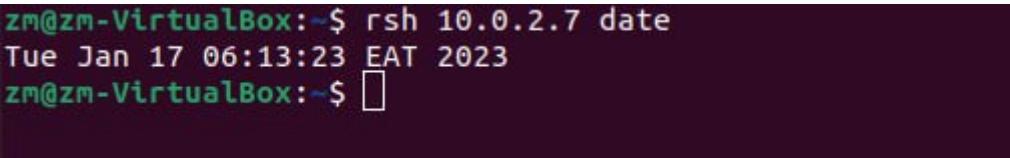
```

zm@zm-VirtualBox:~$ touch .rhosts
zm@zm-VirtualBox:~$ echo 10.0.2.8 > .rhosts
zm@zm-VirtualBox:~$ chmod 644 .rhosts
zm@zm-VirtualBox:~$ ifconfig

```

Fig 5: Configuration in X-terminal

To verify your configuration, *rsh 10.0.2.7 date* command on the trusted server.



```

zm@zm-VirtualBox:~$ rsh 10.0.2.7 date
Tue Jan 17 06:13:23 EAT 2023
zm@zm-VirtualBox:~$

```

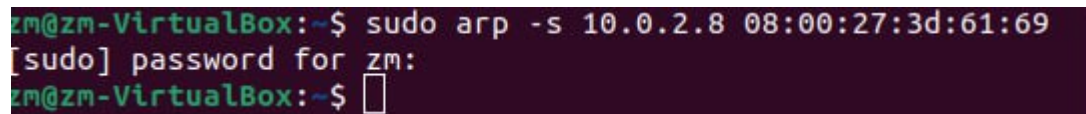
Fig 6: Verifying configuration in trusted server

The command prints the current date and time, so our configuration is working. Now we proceed to do the tasks since we have finished our configuration.

Task 1: Simulated SYN flooding

The operating systems at the time of the Mitnick Attack were vulnerable to SYN flooding attacks, which could mute the target machine or even shut it down. However, SYN flooding can no longer cause such a damage for modern operating systems. We simulate this effect by manually stopping the trusted server VM and then run the following command on X-Terminal to permanently add an entry to the ARP cache:

```
# arp -s [Server's IP] [Server's MAC ]
```



```
zm@zm-VirtualBox:~$ sudo arp -s 10.0.2.8 08:00:27:3d:61:69
[sudo] password for zm:
zm@zm-VirtualBox:~$
```

Now we have “brought down” the trusted server.

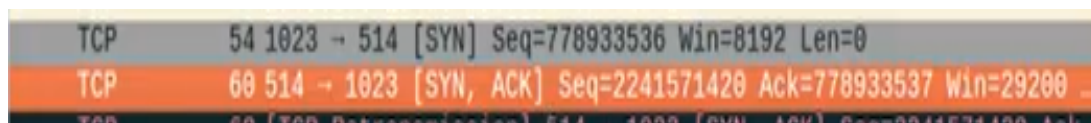
Task 2: Spoof TCP Connections and rsh Sessions

2.1. Spoof the first connection

After X-Terminal receives the SYN packet, it will in turn send a SYN+ACK packet to the trusted server. Since the server has been brought down, it will not reset the connection. The attacker, which is on the same network, can sniff the packet and get the sequence number.

Step 1: Spoof a SYN packet

To spoof a syn packet between x-terminal and the trusted server , we use a python code(mitnick_spoof.py) and after running it in the attacker machine, we observe a SYN + ACK response from the x-terminal. We have used wireshark to capture this packet.



TCP	54	1023	→	514	[SYN]	Seq=778933536	Win=8192	Len=0
TCP	60	514	→	1023	[SYN, ACK]	Seq=2241571420	Ack=778933537	Win=29200
TCP	60				[TCP Retransmission]	514	→	1023 [SYN, ACK] Seq=2241571420 Ack

Step 2: Respond to the SYN+ACK packet.

After X-Terminal sends out a SYN+ACK, the trusted server needs to send out an ACK packet to complete the three-way handshake protocol. To accomplish this we have written a python code, we named it as mitnick_attack

Step 3: Spoof the rsh data packet.

Once the connection is established, the attacker needs to send rsh data to X-Terminal.

Here also we have used a python script, we have modified a mitnick_spoof.py file that we used in the previous step. Then when we observe it in the wireshark the rsh session is established.



Task 2.2: Spoof the Second TCP Connection

This connection is used by rshd to send out error messages. The connections have been successfully established, and rshd executes the command contained in the rsh data packet.

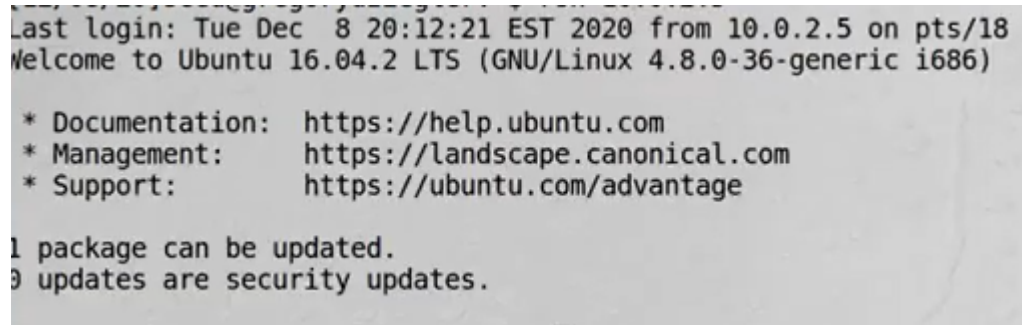
/tmp/xyz is created in /tmp folder.



Task 3: Set Up a Backdoor

Instead of launching the attack again and again, we created a backdoor in X-Terminal after the initial attack. This backdoor allowed us to log into X-Terminal normally anytime we wanted, without typing any password.

First we have modified the data in the python code from /xyz to “echo ++ > .rhosts”, then we are able to login to the x-terminal from the attacker machine.



```
Last login: Tue Dec  8 20:12:21 EST 2020 from 10.0.2.5 on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

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

1 package can be updated.
0 updates are security updates.
```