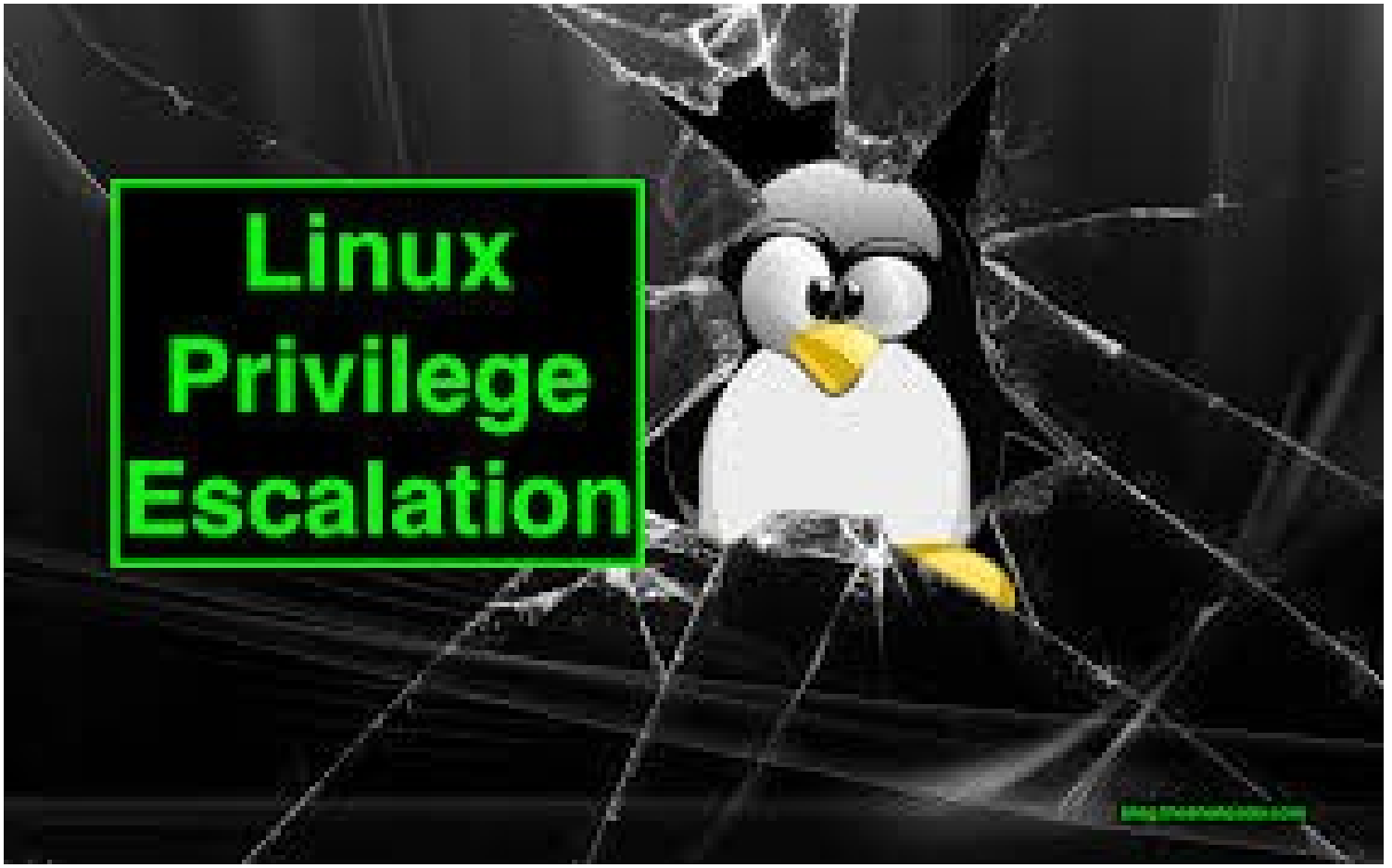
Escalada de Privilegios – Linux



Índice de contenido

Introducción

- En Linux, los permisos o derechos que los usuarios pueden tener sobre determinados archivos se establecen en tres niveles. Estos niveles son los siguientes:
  - Permisos del propietario.
  - Permisos del grupo.
  - Permisos del resto de usuarios.
- Dentro de los sistemas Linux existe la figura del administrador, superusuario o **root**.
- Estos privilegios se establecen tanto para el directorio HOME de cada usuario como para los directorios y archivos a los que el administrador decida que el usuario pueda acceder.
- Cada archivo y directorio define sus permisos en términos de usuario, grupo y “otros”.

Introducción

- Las cuentas de usuario en Linux se configuran en el archivo "*/etc/passwd*" y los hash de las contraseñas de usuario se almacenan en el archivo "*/etc/shadow*".
- Cada uno de los usuarios en el sistema se identifica bajo un identificador llamado **UID**.
- La cuenta de usuario **root** es una cuenta especial en Linux. Esta posee un **UID de 0** y el sistema otorga a este usuario acceso a todos los archivos.

Introducción

- Los grupos de usuario en Linux se configuran en el archivo "*/etc/group*". Dentro del sistema, cada usuario tiene un grupo primario y puede tener múltiples grupos secundarios.
- De forma predeterminada, el grupo principal de un usuario tiene el mismo nombre que su cuenta de usuario.
- Todos los archivos y directorios dentro de Linux poseen un único propietario y un único grupo, así, los permisos se definen en términos de operaciones de lectura (r), escritura (w) y ejecución (x).
- Es importante saber que solo el propietario del directorio o archivo puede cambiar los permisos.

Introducción

- Aún hay otro tipo de permisos que hay que considerar. Se trata del bit de permisos **SUID (*Set User ID*)** y el bit de permisos **SGID (*Set Group ID*)**.

- **SUID:** este bit es asignable a ficheros ejecutables, y permite que cuando un usuario ejecute dicho fichero, el proceso adquiera los permisos del propietario del fichero ejecutado.

- **SGID:** este bit permite adquirir los privilegios del grupo asignado al fichero, también es asignable a directorios. Esto será muy útil cuando varios usuarios de un mismo grupo necesiten trabajar con recursos dentro de un mismo directorio.

Introducción

- Cuando ejecutamos la sentencia "*ls –l*" obtenemos
  - **-rwxr-xr-x** 1 root root fecha nombre_fichero

- Los caracteres en negrita indican los permisos establecidos en el archivo o directorio.
- El primer carácter indica el tipo ('-' para archivo y 'd' para directorio).
- Los siguientes caracteres representan los 3 conjuntos de permisos (propietario,grupo,otros).
- **Los permisos SUID/SGID están representados por una 's' en la posición de ejecución**.

Herramientas de escalada de privilegios

- ✓ Existen herramientas automáticas que permiten enumerar todos los posibles vectores de escalada de privilegios del sistema. Antes de usar estas herramientas, es importante conocer y dominar las técnicas de enumeración manuales.
- ✓ El uso de herramientas nos sirve para automatizar el proceso de reconocimiento y poder identificar las brechas potenciales que nos ayudarán a escalar privilegios.
- ✓ Es importante conocer qué herramientas utilizar y qué estamos buscando para seleccionar la adecuada.

Herramienta: Linux Smart Enumeration

- **Linux Smart Enumeration (*lse.sh*)**: este script mostrará información sobre la seguridad del sistema Linux local. Es una herramienta que posee diferentes niveles de búsquedas de brechas de seguridad.
- También puede monitorear procesos para descubrir ejecuciones recurrentes de programas.
- Está basada en LinEnum.
- La herramienta se encuentra en el siguiente link: https://github.com/diego-treitos/Linux-Smart-enumeration

Herramienta: LinEnum

- La herramienta **LinEnum** es un script que extrae una gran cantidad de información útil del sistema de destino la cual se nos mostrará dividida en segmentos por diferentes colores y nos ayudará a focalizar nuestra búsqueda de brechas de seguridad.
- Esta herramienta busca archivos que contengan una palabra clave, por ejemplo “password”.
- Lleva varios años sin actualización.
- La herramienta puede descargarse del siguiente link: https://github.com/rebootuser/LinEnum

Herramienta: Linux Exploit Suggester 2

- Este script es útil para encontrar rápidamente vulnerabilidades de escalada de privilegios.
- La herramienta se basa en la búsqueda de exploits específicamente de kernel y nos será muy útil durante la escalada de privilegios para detectar las vulnerabilidades existentes.
- La herramienta puede descargarse del siguiente link: https://github.com/jondonas/Linux-exploit-suggester-2

Herramienta: LinPEAS

- ✓ Posiblemente el script más completo de todos. Desarrollado por Carlos Polop.
- ✓ La explicación de las vías de escalada de privilegios que utiliza el script aparecen en la sección correspondiente de book.hacktricks.xyz (blog escrito también por Carlos Polop).
- ✓ La herramienta puede descargarse del siguiente link: https://github.com/carlospolop/PEASS-ng/tree/master/linPEAS

Otras herramientas

- Algunas de las siguientes herramientas también nos sirven para escalada de privilegios en entornos Linux.

- https://github.com/DominicBreuker/pspy
- https://github.com/linted/linuxprivchecker
- http://pentestmonkey.net/tools/Audit/unix-privesc-check
- https://book.hacktricks.xyz/linux-unix/privilege-escalation

Escalada de privilegios con el Kernel

Explotación del Kernel de Linux

- El kernel es el encargado de que el software y el hardware de cualquier ordenador puedan trabajar juntos en un mismo sistema, para lo cual, administra la memoria de los programas y procesos ejecutados, el tiempo de procesador que utilizan los programas, o se encarga de permitir el acceso y el correcto funcionamiento de periféricos y otros elementos físicos del equipo.

- El kernel tiene control completo sobre el sistema operativo por lo que explotar una vulnerabilidad del kernel puede resultar en una escalada de privilegios a un usuario root.

Explotación del Kernel de Linux

- La búsqueda de exploits de kernel suele seguir los siguientes pasos:
  1. Enumerar la versión del sistema y kernel:

     $ uname -a
  2. Buscar un exploit relacionado (ExploitDB, GitHub).
  3. Compilar el exploit y posteriormente ejecutarlo.

- Se debe tener en cuenta que los exploits de kernel son inestables y solo nos funcionarán una vez antes de que el sistema reciba un crash y se reinicie.

Explotación del Kernel de Linux

- ✓ Algunos exploits de Kernel famosos:
  1. DirtyCow (CVE-2016-5195).
  2. DirtyPipe (CVE-2022-0847).
  3. Sudo Baron Samedi (CVE-2021-3156).
  4. OverlayFS (CVE-2021-3493).
  5. PolKit (CVE-2021-3560).
  6. PwnKit (CVE-2021-4034).

Escalada de privilegios en servicios

Explotación de servicios

- Un servicio es un programa simple que funciona en segundo plano, normalmente se inician cuando se carga el sistema operativo y aceptan entradas o realizan tareas regulares.
- Si los servicios vulnerables se ejecutan como root, explotarlos puede conducir a la ejecución de comandos como root.
- Los exploits de servicio se pueden encontrar usando Searchsploit, Google y GitHub, al igual que con los exploits de kernel.

Explotación de servicios root

- El siguiente comando nos brinda como resultado todos los procesos que actualmente se encuentran corriendo como root.
  - *$ ps aux | grep "^root"*
- Dentro de los resultados obtenidos intentaremos identificar el número de versión de los programas que se están ejecutando.
- En distribuciones Debian, el comando "*dpkg -l*" muestra los programas instalados y sus versiones.
- En las distribuciones que usan "*rpm*", se puede verificar de la siguiente forma: *rpm –qa | grep <program>*

Escalada de privilegios con archivos

Permisos en archivos

- Se pueden aprovechar ciertos archivos del sistema para realizar escalada de privilegios si los permisos en ellos son demasiado débiles.
- Si un archivo del sistema tiene información confidencial que podemos leer, este puede ser utilizado para obtener acceso a la cuenta root.
- Si se puede escribir en un archivo del sistema, podemos modificar la forma en que funciona el sistema operativo y obtener acceso a root de esa manera.

Permisos en archivos

- Para la verificación de permisos débiles en archivos comenzaremos buscando programas para los que se establece el bit ***setuid*** y podamos utilizar.
  - *$ find / -type f –perm -04000 –ls 2>/dev/null*
- Buscaremos todos los archivos que podemos leer y escribir en el directorio “etc”.
  - *$ find /etc –maxdepth 1 –readable –type f*

Archivo shadow

- El archivo "***/etc/shadow***" contiene valores hash de contraseña de usuario, de forma predeterminada, ningún usuario puede leerlo, excepto el root.
- Si podemos leer el contenido del archivo "*/etc/shadow*", es posible que podamos descifrar el hash de la contraseña del usuario root.
- Si podemos modificar el archivo "/etc/shadow", podemos reemplazar el hash de contraseña del usuario root con uno que conozcamos.

Archivo passwd

- La cuenta del usuario root en el archivo "***/etc/passwd***" usualmente está configurada de la siguiente forma:
  - *root:x:0:0:root: /root: /bin/bash*
- La "***x***" en el segundo campo indica a Linux que busque el hash de contraseña en el archivo "***/etc/shadow***".
- En algunas versiones de Linux, es posible eliminar simplemente la "x", que Linux interpreta como el usuario que no tiene contraseña:
  - root::0:0:root: /root/: /bin/bash

Archivo backup

- ✓ Una máquina puede tener los permisos correctos en archivos importantes o confidenciales, pero un usuario pudo haber creado copias de seguridad inseguras de estos archivos.

- ✓ Siempre vale la pena explorar el sistema de archivos en busca de archivos de respaldo legibles. Algunos lugares comunes incluyen directorios de inicio de usuarios, el directorio ***/root***, ***/tmp*** y ***/var/backups***.

Escalada de privilegios con sudo/su

Comando sudo

- El comando "***sudo***" permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario de manera segura.

- Por defecto, el usuario debe autenticare con su contraseña al ejecutar *sudo*. Una vez se ha autenticado, si el archivo de configuración ***/etc/sudoers*** permite dar al usuario acceso al comando requerido, el sistema lo ejecuta.

- Existe la opción de habilitar el parámetro **NOPASSWD** con el fin de evitar introducir la contraseña de usuario a la hora de ejecutar el comando.

Comando su

- El comando "***su***" permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar sesión. Comúnmente se usa para obtener permisos de root para operaciones administrativas sin tener que salir y reentrar al sistema.

- La función "***su***" normalmente se ejecuta desde la línea de comandos de un terminal. Cuando se ejecuta, "su" pide la contraseña de la cuenta a la que se quiere acceder, y si es aceptada, da acceso a dicha cuenta.

- A diferencia de su, sudo pide a los usuarios su propia contraseña en lugar de la del usuario requerido.

GTFOBins

- ***GTFOBins*** (https://gtfobins.github.io/) es una lista de binarios de Unix que puede ser explotada por un atacante para evitar las restricciones de seguridad locales.

- Estaremos utilizando estas secuencias de escape en los casos en que tengamos permisos de ejecución "sudo" sobre algunas aplicaciones. Esto nos ayudará a migrar de nuestra Shell restringida a una Shell con permisos root. Dado que el programa inicial se ejecuta con privilegios de root, al migrarnos a una secuencia del mismo, obtendremos una Shell con privilegios altos.

Abusando de funciones

- ✓ En el caso en que un programa no tenga una secuencia de escape, aún puede ser posible usarlo para escalar privilegios. Podremos lograrlo si podemos leer los archivos propiedad de root. Al hacer esto podremos extraer información útil (por ejemplo, contraseñas, hashes,..).

- ✓ En el caso en que podamos escribir en archivos propiedad de root, podremos insertar o modificar información.

Variables de entorno

- Los programas que ejecutan a través de sudo pueden heredar las variables de entorno del entorno del usuario.

- En el archivo de configuración ***/etc/sudoers***, si se establece la opción "***env_reset***", sudo ejecutará programas en un entorno nuevo y mínimo. La opción "***env_keep***" se puede usar para mantener ciertas variables de entorno del entorno del usuario.

- ***LD_PRELOAD*** es una variable de entorno que se puede establecer en la ruta de un archivo de objeto compartido (***.so***). Cuando se establece, el objeto compartido se cargará antes que cualquier otro.

Variables de entorno

- Al crear un objeto compartido personalizado y crear una función ***init()***, podemos ejecutar código tan pronto como se carga el objeto.

- LD_PRELOAD no funcionará si el ID de usuario real es diferente desde la identificación de usuario efectiva. A su vez, la función “sudo” debe estar configurada para preservar la variable LD_PRELOAD de entorno utilizando la opción env_keep.

- Otra variable de entorno común es PATH. Si no está bien configurada, es posible escalar privilegios muy fácilmente (Priv Esc using PATH variable).

Escalada de privilegios con tareas programadas

Tareas programadas

- Los atacantes suelen aprovechar las tareas programadas en los ataques de escalada de privilegios.
- Los sistemas que actúan como servidores a menudo ejecutan periódicamente varias tareas automatizadas y programadas que pueden ser vulnerables. En sistemas Unix las tareas programadas las ejecuta el servicio ***cron***.
- Los trabajos de ***cron*** se ejecutan con el nivel de seguridad del usuario que los posee. Por defecto, los trabajos cron se ejecutan utilizando el Shell ***/bin/sh*** con variables de entorno limitadas.

https://www.hackingarticles.in/linux-privilege-escalation-by-exploiting-cron-jobs/

Tareas programadas

- Los archivos de tabla Cron (***crontabs***) almacenan la configuración para trabajos cron. Estas tareas de usuario generalmente se encuentran en los directorios ***/var/spool***, ***/cron/*** o ***/var/spool/cron/crontabs/***. El crontab de todo el sistema se encuentra en ***/etc/crontab***.

- La configuración incorrecta de los permisos de archivo asociados con los trabajos cron puede conducir a una escalada de privilegios fácil.

- Si podemos escribir en un programa o script que se ejecuta como parte de un trabajo cron, podemos reemplazarlo con nuestro propio código.

Variable de entorno

- La variable de entorno crontab PATH está configurada por defecto en /usr/bin:/bin. Esta variable PATH se puede sobrescribir en el archivo crontab.

- Si un programa/script de trabajo cron no utiliza una ruta absoluta, y nuestro usuario puede escribir en uno de los directorios PATH, es posible que podamos crear un programa/script con el mismo nombre que el trabajo cron y con esto tomar control del ejecutable crontab.

Wildcards/Comodines

- Cuando se proporciona un carácter comodín (*) a un comando como parte de un argumento, el Shell primero realizará la expansión del nombre del archivo (también conocido como ***globbing***) en el comodín.

- Este proceso reemplaza el comodín con una lista separada por espacios de los nombres de archivo y directorio en el directorio actual.

- Una manera fácil de ver esto en acción es ejecutar el siguiente comando desde su directorio de inicio: echo *

Wildcards/Comodines

- Dado que los sistemas de archivos en Linux son generalmente muy permisivos con los nombres de archivo, y la expansión del nombre de archivo ocurre antes de que se ejecute el comando, es posible pasar opciones de línea de comando (por ejemplo, -h, --help) a los comandos creando archivos con estos nombres.

- Ejemplo

  user@debian$ ls *

  % touch ./-l

  user@debian$ ls *

Wildcards/Comodines

- ✓ Los nombres de archivo no se limitan simplemente a opciones simples como –h o –help.
- ✓ De hecho, podemos crear nombres de archivo que coincidan con opciones complejas: --option = key = value
- ✓ GTFOBins puede ayudar a determinar si un comando tiene opciones de línea de comando que serán útiles para nuestros propósitos.
  - https://gtfobins.github.io

Escalada de privilegios con SUID/SGUID

SUID/SGUID

- En Linux, algunos de los comandos y binarios existentes pueden ser utilizados por usuarios no root para escalar los privilegios de acceso root si el bit SUID está habilitado.
- Hay algunos comandos ejecutables famosos de Linux que pueden permitir la escalada de privilegios: bash, cat, cp, echo, fin, les, more, nano, nmap, vim,...
- Debemos tener en cuenta que en SUID los archivos se ejecutan con los privilegios del archivo, mientras que en SGUID los archivos se ejecutan con los privilegios del grupo de archivos.

SUID/SGUID

- Esto nos lleva a lo siguiente: si el archivo es propiedad de root, se ejecuta con privilegios de “root” y podemos usarlo para escalar privilegios.

- Para poder identificar brechas vulnerables de los binarios SUID/SGID podremos utilizar la siguiente sentencia que nos ayudará a identificar binarios en los cuales tengamos permisos.

- *$ find / -type f –a \ (-perm –u+s –o –perm –g+s \) –exec ls –l {} \; 2> /dev/null*

SUID/SGUID

- ✓ Así como pudimos usar secuencias de escape de Shell con programas que se ejecutan a través de sudo, podemos hacer lo mismo con los archivos SUID/SGID.

- ✓ Podremos encontrar una lista de programas con sus secuencias de escape de Shell aquí: https://gtfobins.github.io/

Inyección de objetos compartidos

- Cuando se ejecuta un programa, este intentará cargar los objetos compartidos que requiere para su ejecución.
- Al usar un programa llamado "***strace***", podemos rastrear estas llamadas al sistema y determinar si no se encontraron objetos compartidos.
- Si podemos escribir en la ubicación que el programa intenta abrir, podemos crear un objeto compartido y generar un Shell root cuando se carga.

https://www.hackingarticles.in/linux-privilege-escalation-using-ld_preload/

https://www.boiteaklou.fr/Abusing-Shared-Libraries.html

https://book.hacktricks.xyz/linux-unix/privilege-escalation/ld.so.conf-example

Mal uso o descubrimiento de contraseñas

Contraseñas

- Puede parecer una posibilidad remota, pero el almacenamiento de contraseñas dentro del sistema, una contraseña débil y la reutilización de contraseñas pueden ser formas fáciles de escalar privilegios.
- Si bien la contraseña de la cuenta del usuario root está cifrada y almacenada de forma segura en /etc/shadow, otras contraseñas, como las de los servicios, pueden almacenarse en texto plano sin formato en archivos de configuración.
- En el caso en que el usuario root reutilizó su contraseña para un servicio, esa contraseña se puede encontrar y reutilizar para cambiar al usuario root.

Contraseñas

- Es importante mantener un registro de los comandos que se han ingresado en el Shell. Afortunadamente, bash mantiene un historial de comandos. Esto nos permite ver los comandos emitidos por los usuarios mientras utilizan ciertos programas.

- Si un usuario escribe una contraseña como parte de un comando, esta contraseña puede almacenarse en un archivo de historial.

- Siempre es una buena idea intentar cambiar al usuario root con una contraseña descubierta.

Contraseñas

- ✓ Muchos servicios y programas usan archivos de configuración (config) para almacenar configuraciones. Si un servicio necesita autenticarse en algo, podría almacenar las credenciales en un archivo de configuración.

- ✓ Si se puede acceder a estos archivos de configuración y los usuarios privilegiados reutilizan las contraseñas que almacenan, es posible que podamos usarlo para iniciar sesión como ese usuario.

Escalada de privilegios con NFS

NFS (Network File System)

- ✓ NFS es un protocolo de sistema de archivos distribuido desarrollado por Sun Microsystems. Permite a un usuario en un ordenador cliente acceder a los archivos a través de una red informática como si estuvieran en un almacenamiento montado localmente.

- ✓ NFS a menudo se usa con sistemas Linux y normalmente es inseguro en su implementación. Puede ser algo difícil de configurar de forma segura, por lo que no es raro encontrar recursos compartidos de NFS abiertos al mundo.

NFS (Network File System)

- ✓ Esto es bastante conveniente para nosotros como pentesters, ya que podríamos aprovecharlos para recopilar información confidencial y escalar privilegios.

- ✓ De manera predeterminada, los archivos creados heredan la identificación del usuario remoto y la identificación del grupo, incluso si no existen en el servidor NFS.

https://www.hackingarticles.in/linux-privilege-escalation-using-misconfigured-nfs/

https://www.errno.fr/nfs_privesc.html

NFS (Network File System)

- Existen los siguientes comandos que nos llevarán a identificar la presencia del protocolo NFS:

```
# nmap –sV –p 111 –script=rpcinfo 192.168.1.10
# nmap –p 111 –script nfs* 192.168.1.10
# mount –o rx,vers=2 <target>:<share> <local_directory>
```

- El archivo */etc/exports* contiene un registro para cada directorio que espera compartir dentro de una máquina de red. Cada registro describe cómo se comparte un directorio o archivo.

NFS (Network File System)

- Hay varias opciones que definirán qué tipo de privilegio tendrá esa máquina sobre el directorio compartido.

  - **rw:** permitir a los clientes leer y escribir en el directorio compartido.
  - **ro:** permitir a los clientes acceso de solo lectura.
  - **Root_Squash:** esta opción evita la solicitud de archivos realizada por el usuario root en la máquina cliente porque los recursos compartidos NFS cambian el usuario root al usuario nfsnobody, que es una cuenta de usuario sin privilegios.
  - **No_root_squash:** otorga autoridad al usuario root del cliente para acceder a los archivos del servidor NFS como root.

NFS (Network File System)

- Una configuración del archivo export débil podría llevar a la escalada de privilegios, es bueno tener claro cómo configurar el archivo /etc/export usando una opción en particular. Un sistema NFS se considera débil o mal configurado cuando se edita la entrada/registro siguiente para compartir cualquier directorio.
  - /home *(rw,no_root_squash)

- La entrada anterior muestra que hemos compartido el directorio /home y hemos permitido que el usuario root del cliente acceda a los archivos para la operación de lectura/escritura y el signo * indica la conexión desde cualquier máquina host.

Escalada de Privilegios – Linux

Fin