Privilege escalation, Pivoting and Persistence

Martin Stanek

2024

Table of Contents

Privilege escalation in general

- obtaining access to privileged account
  - root (Linux), Administrator, SYSTEM (Windows)
  - SYSDBA (Oracle DB), sa (MS SQL)
  - user in privileged groups (wheel, Administrators, Backup Operators)
  - user with additional privileges (SeTakeOwnershipPrivilege)
- vulnerabilities in kernel, system utilities and programs
- vulnerabilities in 3rd party app
- configuration problems
- lack of timely patching and lax administration
- today: few examples for Linux and Windows

File permissions

```
-rw-r----- 1 root shadow 1489 Jan 28 05:42 shadow
```

- file permissions (owner/group/others model)
  - sensitive information can be read
  - configuration of important services/utilities can be changed
- examples:
  - readable `/etc/shadow` – dictionary or brute-force attacks
  - writable `/etc/shadow` or `/etc/passwd` – replace password or create new root user
  - writable `/etc/sudoers` or something from `@includedir`, e.g. `sudoers.d/*`
- directory permissions
  - add or replace a configuration file
  - add a malicious library/program in the path

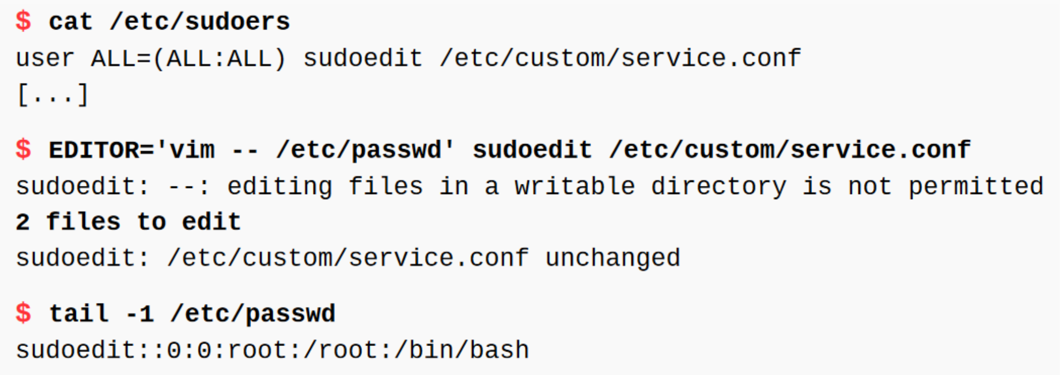
Linux Kernel – *Dirty Pipe*

- CVE-2022-0847
  - since version 5.8, fixed in 5.16.11, 5.15.25 and 5.10.102
- attacker can overwrite *arbitrary* (must have read permission) file on the system
- page caching problem, basic idea:
  - files are read to page cache
  - set `PIPE_BUF_FLAG_CAN_MERGE` flag for a pipe – writing data to the page cache
  - `splice()` system call, moves data between two file descriptors
  - splice data from read only file to pipe with the flag set
  - modify data in the pipe – cached file data are overwritten
- easy exploitation, e.g. overwrite `/etc/passwd`, overwrite SUID binary
- Linux kernel privilege escalation auditing tool: LES (Linux Exploit Suggester)

System utility – sudo

- CVE-2021-3156 (*Sudo Baron Samedit*)
  - affected versions: 1.8.2-1.8.31p2 and 1.9.0-1.9.5p1
  - heap-based buffer overflow, almost 10 years in the source code
  - any user can escalate to root
- another problem: CVE-2023-22809
  - sudoedit allows a user with sudoedit privileges to edit arbitrary files
  - user-specified editor may contain a "--" argument that defeats a protection mechanism (where "--" is used as a separator)
  - affected versions: 1.8.0-1.9.12.p1 (*see next slide*)
- patching is important

sudoedit problem CVE-2023-22809



```
$ cat /etc/sudoers
user ALL=(ALL:ALL) sudoedit /etc/custom/service.conf
[...]

$ EDITOR='vim -- /etc/passwd' sudoedit /etc/custom/service.conf
sudoedit: --: editing files in a writable directory is not permitted
2 files to edit
sudoedit: /etc/custom/service.conf unchanged

$ tail -1 /etc/passwd
sudoedit::0:0:root:/root:/bin/bash
```

Source: Synacktiv, Sudoedit bypass in Sudo <= 1.9.12p1

Escalating privilege – sudo

- `sudo` – delegating authority to run commands as a privileged user (usually root)
  - some utilities allow privilege escalation
  - examples: `vim`, `dd`, `zip`, `find` etc.
- GTFOBins (`gtfobins.github.io`)
  - collection of Unix binaries
  - how to bypass local security restrictions in misconfigured systems
  - SUID, sudo, read/write files, spawning an interactive shell
- `NOPASSWD` – executing some commands without knowing the password

Libraries

- environment variables and configuration
- `LD_PRELOAD` – preloading (malicious) library
  - security feature: preload ignored if real UID is different from effective UID
  - potentially vulnerable sudo option: `env_keep+=LD_PRELOAD`
- `LD_LIBRARY_PATH` – where to search for a library
  - similar to the previous case
- `/etc/ld.so.conf` and configuration files in specified paths
  - paths where libraries are searched for
  - writable configuration or paths allow to inject malicious library

Sensitive information stored in readable way

- passwords, API keys and other data
- places:
  - configuration files
  - scripts (profile, scheduled, etc.)
  - environment variables
  - shell history
  - logs
  - backups

Vulnerable cron jobs

- cron jobs – scheduling tasks
    - system-wide (`/etc/crontab`), and user specific crontabs
    - run as root at 2am every Monday and Wednesday (crontab fragment):

      ```
      SHELL=/bin/sh
      PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
      0 2 * * 1,3 root /root/backup.sh > /root/backup-report.txt
      ```
- vulnerable, if the attacker can
    - modify scheduled program/script directly
    - abuse vulnerability of scheduled task
    - inject script in the path before the original script
- similar functionality (cron alternative): timers in systemd
    - similar problems/opportunities for privilege escalation

SUID, SGID

```
-rwsr-xr-x 1 root root 68248 Nov 11 03:28 passwd
```

- SUID – execute with the same permissions as the owner
- SGID – execute with the same permissions as the group
- GTFOBins – again, exploiting common tools with SUID
- simple `find` can enumerate SUID binaries (2000 for SGID):

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

Capabilities

```
$ getcap /usr/bin/ping
/usr/bin/ping cap_net_raw=ep    (permitted, effective)
```

- more granular approach to allow privileged operations
- examples of what can be achieved with a capability:
  - `cap_setuid` – arbitrary manipulations of process UIDs
  - `cap_sys_ptrace` – transfer data to or from the memory of arbitrary processes
  - `cap_dac_override` – bypass file read, write, and execute permission checks
- misconfiguration can open privilege escalation possibilities

Services

```
SERVICE_NAME: ekrn
  TYPE               : 20  WIN32_SHARE_PROCESS
  START_TYPE         : 2   AUTO_START
  BINARY_PATH_NAME   : "C:\Program Files\ESET\ESET NOD32 Antivirus\ekrn.exe"
  DISPLAY_NAME       : ESET Service
  DEPENDENCIES       :
  SERVICE_START_NAME : LocalSystem
```

- background processes (some are part of the OS, some are part of installed apps)
- manage: `sc` command, GUI, PowerShell

Services – paths, permissions

- Unquoted Service Paths
  - binary path name without quotes, e.g.
    `BINARY_PATH_NAME : C:\Program Files (x86)\Some App\progam.exe`
  - evaluated as `C:\Program.exe`, `C:\Program Files (x86)\Some.exe`, `C:\Program Files (x86)\Some App\progam.exe`
  - ability to create/overwrite any of those files leads to an exploit
  - (re)start the service or wait for a reboot
- permission to change service configuration, i.e. `BINARY_PATH_NAME`
  - replace with malicious executable
- permission to replace service binary with own executable

Startup and Autoruns

- programs that run when OS is starting or after user logs in to Windows
  - system-wide or user-specific
  - placed in defined folders or in the registry
  - GUIs (e.g. `Startup Apps`, Task Manager) consolidate various sources of startup applications
- vulnerable, if user has permission to insert additional application, e.g. to the system-wide Startup folder
  - wait for administrator to log in (running with administrator's privileges)
- Autoruns from Sysinternals (deep dive into auto-starting components)

Passwords at rest

- stored as LM hash (weak) and NT hash
    - LM hashes disabled by default since Windows Vista and Windows Server 2008
    - unsalted, hash values are encrypted
- Active Directory: stored in NTDS.DIT file
- domain members, workstations:
    - local users in the Security Account Manager (SAM database); file/Registry
- supplementalCredentials – additional forms of the cleartext password, e.g.
    - Primary:Kerberos – hashes of the cleartext password for the Kerberos protocol
- access to SAM database is restricted, otherwise:
    - CVE-2021-36934 (HiveNightmare)
    - overly permissive Access Control Lists (ACLs), read any Registry hives
    - SAM, SYSTEM, SECURITY – access to password hashes
- brute-force or dictionary attacks, pass-the-hash

Other sources of passwords (hashes)

- memory dumps
- cached domain credentials
- elevated privileges required
- tools: Mimikatz, Impacket

- Net-NTLMv1, Net-NTLMv2 authentication protocols
- relaying authentication requests (SMB Signing disabled)
- tools: Responder, Inveigh

DLL Hijacking

- tricking an application to load a malicious DLL (and execute a code in the DLL)
- methods – examples:
  - missing DLL for a process (that can be substituted)
  - modifying PATH variable
  - replace a legitimate DLL with a modified version
  - abusing DLL search order
- assumption: process that runs with elevated privileges and DLL hijacking possible
- default search order for Windows (unpackaged apps, SafeDllSearchMode enabled):
  - 12 steps

    > 1. DLL redirection (`<your app name>.local` file)

    > 7. the folder from which the application loaded

    > 11. the current folder

    > 12. PATH environment

Scheduled tasks

- scripts and programs running when triggered
- defined in the registry
  - `HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\Taskcache\Tasks`
  - GUI: Task Scheduler, CLI: `schtasks`, PowerShell
- modern versions of Windows require local admin to create a scheduled task
- weak file permissions for scheduled task
  - replace or overwrite

Automation

- enumerate various configuration problems in the system
- large number of potential problems
  - tedious and error-prone to check manually
- faster result, but you should know what a how is tested
  - false sense of security (if nothing is detected)
  - unwanted impact of some tests
- tools
  - Linux: LinEnum, LinPEAS (PEAS-ng), etc.
  - Windows: PrivescCheck, WinPEAS (PEAS-ng), etc.

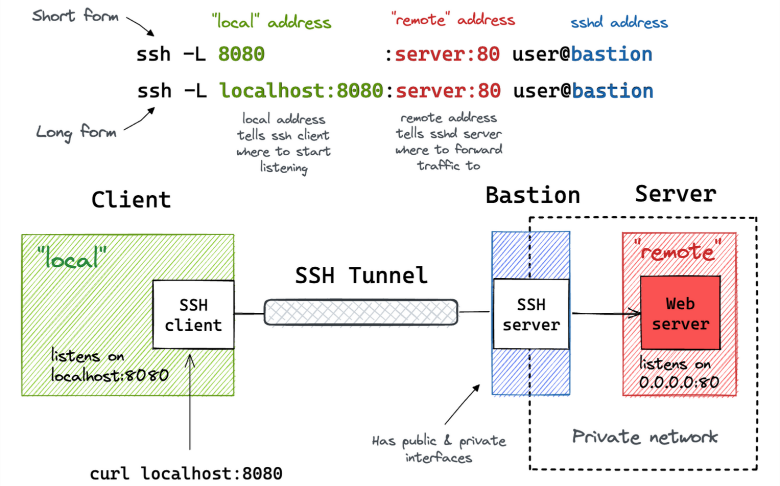
Pivoting

- using access to one system to perform reconnaissance/enumeration and exploitation of other systems
  - separate networks, firewall rules
  - might bypass network security controls
  - might avoid triggering network security monitoring controls
- criteria for tools selection
  - privileged or unprivileged account on pivot machine
  - pivot communication for a single port or multiple ports
  - native tools or additional software required (on pivot machine)
  - configuration complexity

SSH – port forwarding

- local port forwarding (“jump” server)
  - `ssh -L 8080:internal_server:80 user@ssh_server`
  - local port (8080) forwarded to internal server (port 80) through an SSH tunnel
  - e.g. accessing internal web from the outside
- remote port forwarding
  - `ssh -R 2222:internal_server:22 user@my_server`
  - `my_server` port (2222) forwarded to internal server (22) through SSH tunnel
  - e.g. creating a backdoor into the internal network
- server must enable `AllowTcpForwarding`
  - *Note that disabling TCP forwarding does not improve security unless users are also denied shell access, as they can always install their own forwarders.*
- dynamic port forwarding (client as a SOCKS proxy server)

SSH – local port forwarding

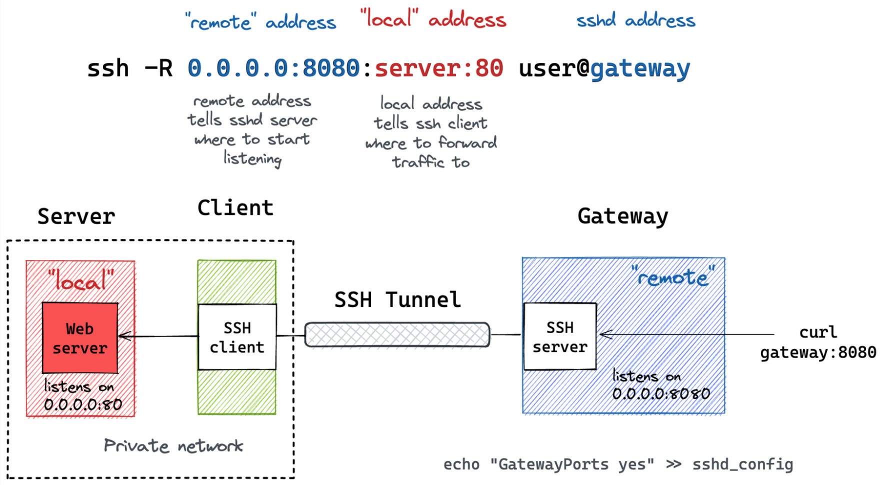


```
ssh -L 8080          :server:80 user@bastion
ssh -L localhost:8080:server:80 user@bastion
```

```
curl localhost:8080
```

Source: Ivan Velichko, A Visual Guide to SSH Tunnels: Local and Remote Port Forwarding

SSH – remote port forwarding



```
ssh -R 0.0.0.0:8080:server:80 user@gateway
```

```
echo "GatewayPorts yes" >> sshd_config
```

Source: Ivan Velichko, A Visual Guide to SSH Tunnels: Local and Remote Port Forwarding

SOCKS, Ncat, socat

- SOCKS
  - proxy for any TCP (and UDP since SOCKS5) traffic
  - client must support SOCKS protocol to use the proxy (e.g. web browsers)
  - proxychains for tools that do not support SOCKS natively
- Ncat (part of nmap project)
  - advanced alternative to netcat (`nc`)
  - connections through SOCKS and HTTP proxies
  - redirect or proxy TCP/UDP traffic to other ports or hosts, etc.
- socat (data relay)
  - creates two bidirectional data streams and connects them
  - streams: files, pipes, sockets (TCP, UDP), etc.
  - port forwarding, relaying, etc.
- many other tools exist (chisel, . . .)

Persistence – general

- retaining access after compromise
  - exploit hard to reproduce (e.g. depends on successful phishing)
  - easier access than the original exploit
  - avoiding detection
- security testing perspective
  - testing detection and reaction capabilities of the target

Persistence – Linux

- create a new user
  - possibly in sudo/wheel group
- ssh authorized keys
  - adding a new public key to `authorized_keys` (or create file if not present)
- cron jobs or systemd timers with a backdoor
- modify files that are executed at login/logoff or starting a shell
  - systemwide or user-specific
  - `/etc/profile`, `.bashrc`, `.profile`, etc.
- modify files that are executed when system starts (boots)
- set SUID for an installed program or for a prepared script
- create or modify a systemd service

Persistence – Windows

- add user to a special group
  - Administrators, Backup Operators, etc.
- assign a special privilege (for example `SeBackupPrivilege`)
- modifying executable files, shortcuts, file associations
- creating or modifying a service
- plant a backdoor in task scheduler
- StartUp folder, Run/RunOnce registry keys, Winlogon registry keys, etc.
- login screen – replace helper tools with other programs (e.g. `cmd.exe`)
  - sticky keys (`sethc.exe`), Ease of Access options (`utilman.exe`)

Exercises

1. TryHackMe: Linux PrivEsc, Windows PrivEsc
    - Don’t just copy&paste the instructions, think about the root cause.
    - What went wrong and how you would test for each particular privilege escalation vector?
    - What privilege escalation “opportunity” you think is the most prevalent (one for each OS)? Justify your answer.

Resources

1. HackTricks
2. I. Velichko, *A Visual Guide to SSH Tunnels: Local and Remote Port Forwarding*, 2023