Privilege Escalation Techniques and Prevention

Privilege escalation is **a process where an attacker gains elevated access or permissions on a system, typically moving from a lower privilege level (e.g., user) to a higher one (e.g., administrator or root)**. Privilege escalation exploits are a critical step in many attack chains and can lead to full system compromise.

1. Types of Privilege Escalation

a. Vertical Privilege Escalation

- Definition: An attacker elevates their privileges **from a lower level (e.g., a normal user) to a higher level (e.g., root or administrator)**.
- Example: Exploiting a vulnerability in a system service to gain administrative privileges.

b. Horizontal Privilege Escalation

- Definition: An attacker **gains access to another account at the same privilege level**, such as another user's account, to access restricted resources.
- Example: Stealing another user's session cookie to impersonate them.

2. Techniques for Privilege Escalation

a. Exploiting Misconfigurations

- **World-Writable Files**
  - An attacker modifies files or binaries owned by privileged users.
  - Example: Modifying a world-writable /etc/passwd file to add a new root user.
  - Fix: Audit file permissions regularly; restrict write access.
- **Setuid/Setgid Binaries**
  - Exploiting binaries with the setuid or setgid bit set to execute as the owner or group.
  - Example: Running a misconfigured setuid binary to execute commands as root.
  - Fix: Minimize the use of setuid binaries and monitor for unusual changes.

b. Exploiting Vulnerabilities

- **Kernel Vulnerabilities**
  - Exploiting bugs in the operating system kernel to execute arbitrary code with kernel privileges.
  - Example: Using Dirty COW (CVE-2016-5195) to modify files the attacker doesn't own.
  - Fix: Keep the system updated with security patches.
- **Application Vulnerabilities**
  - Exploiting poorly coded or vulnerable applications running with elevated privileges.
  - Example: Buffer overflows in services like sudo or cron.
  - Fix: Regularly patch and update software.

c. Credential Dumping

- Definition: **Extracting credentials from memory or configuration files** to impersonate higher-privileged accounts.

- Example: Using tools like **Mimikatz** to dump credentials from memory on Windows systems.
- Fix: Use memory protections, enable Credential Guard (Windows), and enforce MFA.

d. Exploiting Weak Credentials

- Definition: Using **brute force or password spraying** to guess passwords of privileged accounts.
- Example: Using default passwords for admin accounts.
- Fix: Enforce strong password policies and disable default credentials.

e. DLL/Library Hijacking

- Definition: **Placing malicious libraries** in locations where privileged processes load them.
- Example: A process loads an attacker-controlled DLL with elevated privileges.
- Fix: Validate library paths and use secure loading mechanisms.

f. Abusing Scheduled Tasks

- Definition: **Modifying or creating scheduled tasks or cron jobs** to execute malicious commands as privileged users.
- Example: Editing root-owned cron jobs on Linux to run malicious scripts.
- Fix: Restrict access to scheduling tools and audit scheduled tasks.

g. Exploiting Insecure APIs

- Definition: **Calling privileged APIs** directly or bypassing access controls.
- Example: Exploiting cloud APIs to elevate privileges in misconfigured IAM roles.
- Fix: Harden APIs, enforce access controls, and monitor API activity.

3. Prevention Techniques

a. Principle of Least Privilege (PoLP)

- Ensure users and processes only have the **minimum privileges** necessary.
- Example: Preventing a regular user from executing administrative commands.

b. Regular Updates and Patches

- Keep operating systems, software, and firmware **up to date** to mitigate known vulnerabilities.

c. Secure Configuration Management

- Harden systems by **disabling unnecessary services, removing default accounts, and auditing permissions**.

d. Multi-Factor Authentication (MFA)

- **Enforce MFA for privileged accounts** to reduce the impact of credential theft.

e. Logging and Monitoring

- **Enable logging of privilege escalation attempts and monitor for unusual activity**.

- Example: Alert on unauthorized changes to sudoers files or cron jobs.

f. Use Security Tools

- **SELinux/AppArmor**
  - Constrain processes to predefined policies, limiting their ability to escalate privileges.
- **Endpoint Protection**
  - Use tools like EDR (Endpoint Detection and Response) to detect exploitation attempts.

g. Restrict Access to Sensitive Files

- **Limit access to files** like /etc/shadow, Windows SAM, or sensitive configuration files.

h. Audit and Harden Elevated Accounts

- **Regularly audit** the use of elevated accounts and remove unnecessary privileges.

i. Enforce Secure Coding Practices

- **Train developers to write secure code** and avoid common vulnerabilities like buffer overflows.

4. Tools for Privilege Escalation Detection and Prevention

Tool	Purpose
Lynis	Auditing and hardening Linux systems.
Mimikatz	Credential dumping (used by attackers).
OSQuery	Monitoring file and process changes.
Auditd	Linux auditing for sensitive operations.
Sysmon	Windows process and event monitoring.

5. Summary

Aspect	Details
Types of Escalation	Vertical (low-to-high privilege), Horizontal (user-to-user).
Common Techniques	Exploiting misconfigurations, vulnerabilities, weak credentials, or APIs.
Prevention Strategies	PoLP, patching, secure configurations, MFA, monitoring, and SELinux/AppArmor.
Detection Tools	Sysmon, Lynis, OSQuery, Auditd.

Privilege escalation is a critical threat in modern systems. By understanding common techniques attackers use, such as exploiting misconfigurations or vulnerabilities, and implementing robust preventive measures, organizations can significantly reduce the risk of privilege escalation and ensure better overall system security.