Linux Privilege Escalation Automation Toolkit
(Project Documentation)

1. Project Overview / Description

This project focuses on developing an automated toolkit capable of scanning a Linux system to detect privilege escalation opportunities.

Privilege escalation happens when a normal user gains unauthorized root/admin access due to weak configurations, mismanaged permissions,

or vulnerable services. The toolkit will practically scan, analyze, and report potential risks without executing harmful exploitation steps.

The project reflects real-world red-team enumeration and blue-team auditing practices used in penetration testing and SOC operations.

2. Practical Motivation

Linux servers often contain overlooked misconfigurations. Manual auditing is slow and error-prone.

This project allows students to practically implement automated scanning techniques, helping them understand how attackers escalate privileges

and how defenders can detect and mitigate these misconfigurations.

Practical benefits include:

- Hands-on understanding of Linux permissions and SUID/SGID behavior

- Real-world security auditing experience

- Creating automated scripts used by cybersecurity professionals

3. Project Objectives

1. Develop a fully automated scanner for privilege escalation weaknesses.

2. Teach students to identify real misconfigurations practically through testing.

3. Equip students with knowledge to perform red-team style enumeration.

4. Produce a practical security report usable by blue-team defenders.

5. Ensure toolkit is safe (detection only).

4. Practical Scope of the Toolkit

The toolkit will practically perform the following checks:

A. SUID/SGID Binary Discovery:

- Scan entire filesystem for binaries with SUID/SGID bits.
- Identify binaries that can be exploited using GTFOBins techniques (practical attack paths).
- Generate high-risk binary list (e.g., awk, find, perl, vim).

B. Weak File & Directory Permissions:

- Find world-writable files that can be modified by any user.
- Detect writable cron scripts, service files, logs, or binaries used by privileged processes.
- Scan for incorrect permissions on /etc/passwd, /etc/shadow, and home directories.

C. Misconfigured Services:

- Identify systemd services that run as root but point to user-controlled files.
- Check insecure PATH variables in service files.
- Detect sudo misconfigurations like NOPASSWD privilege escalation.

D. Cron Job Vulnerabilities:

- Extract system-level cron entries.
- Identify writable scripts executed by root.
- Highlight timing-based attack vectors such as cron replacement attacks.

E. Kernel Exploit Detection:

- Capture kernel version and match against known CVEs.

- Detect outdated, end-of-life kernels.

- Suggest mitigation without running live exploits.

5. Tools & Technologies Used (Practical Tools)

Programming Languages:

- Python (recommended)

- Bash scripting

Linux Utility Commands Students Will Use Practically:

- find (search for binaries and permissions)

- ls -laR (recursive listing for permission analysis)

- systemctl (service configuration discovery)

- getcap (capabilities scan)

- sudo -l (sudo privilege analysis)

- crontab -l and /etc/cron.* (cron enumeration)

- uname -a (kernel information)

- grep, awk, sed (data parsing)

Documentation Tools:

- MS Word / Google Docs

- Draw.io (flowchart creation)

6. Practical Techniques Students Will Implement

Red Team Practical Techniques:

- Enumerating SUID binaries for exploit paths.

- Investigating writable directories that lead to privilege escalation.

- Identifying root cron jobs and modifying attack chains.

- Analyzing misconfigured systemd services.

Blue Team Practical Techniques:

- Permission hardening.

- Reviewing service configurations and fixing misconfigurations.

- Kernel patching and secure update procedures.

- Creating alerts/logs for suspicious privilege escalation behaviors.

7. Workflow / Architecture (Practical Explanation)

The toolkit workflow must follow these practical steps:

STEP 1: System Information Collection

- Detect logged-in user, groups, kernel version, OS release.

- Identify whether the user is root or limited.

STEP 2: Privilege Escalation Vector Scanning

- Run modules for SUID, cron, services, kernel vulnerabilities, etc.

STEP 3: Analysis Engine

- Match findings against predefined vulnerability patterns.

- Identify high-risk vectors.

STEP 4: Report Generation

- Export a structured report with:

  * Findings

  * Severity

  * Exploitation possibility

  * Suggested mitigation steps

8. Detailed Flowchart (Text Version for Word File)

START

↓

System Information Gathering

↓

Scan for SUID/SGID Binaries

↓

Scan for Weak Permissions

↓

Scan for Misconfigured Services

↓

Scan for Vulnerable Cron Jobs

↓

Scan for Kernel Exploit Possibilities

↓

Analyze All Findings

↓

Generate Final Security Report

↓

END

9. Expected Practical Output

Students must provide:

- Working automated toolkit (Python/bash).

- Practical screenshots of running commands and toolkit output.

- A findings report categorizing each vulnerability.

- Mitigation steps for each identified issue.

- A final presentation.

Expected Toolkit Output Example:

- List of exploitable SUID binaries

- Writable cron scripts

- Kernel vulnerabilities with CVE references

- Misconfigured sudo rules

10. Learning Outcomes (Practical Skills)

Students will learn:

- Practical Linux privilege escalation methods.

- How attackers escalate privileges in real environments.

- How defenders audit systems for weaknesses.

- How to automate security testing.

- Real-world scripting and cybersecurity reporting skills.

11. Student Deliverables

1. Complete project documentation (Word/PDF)

2. Automated scanning script/toolkit

3. Flowcharts and architecture diagrams

4. Screenshots of practical testing

5. Final PPT for presentation