EXPT NO: 06 ROLL NO: 220701249

LINUX PRIVELEGE ESCALATION

AIM:

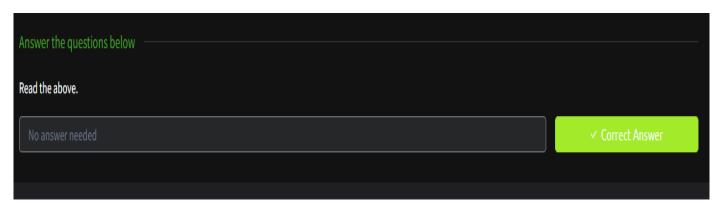
This experiment aims to explore Linux privilege escalation techniques including kernel exploits, misconfigured sudo privileges, and SUID binary exploitation. It provides hands-on experience in identifying and exploiting real-world vulnerabilities to escalate privileges on Linux systems for better understanding of secure configurations and attacker methodologies.

PROCEDURE:

- 1. Enumerate Linux kernel version and search for CVEs.
- 2. Transfer and compile kernel exploits on target machines.
- 3. Use 'sudo -l' to check misconfigured binaries and escalate via GTFOBins.
- 4. Exploit vulnerable SUID binaries to read sensitive files or spawn root shells.
- 5. Dump password hashes using unshadow and crack using John the Ripper.
- 6. Bypass protections using LD_PRELOAD or service misconfiguration

TASK 1 – INTRODUCTION

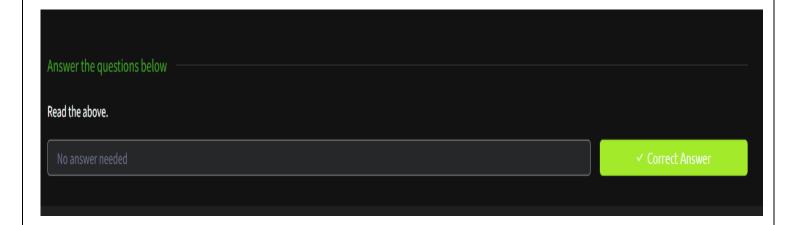
- Introduced the fundamentals of privilege escalation in Linux systems.
- Highlighted the significance of escalating from a low-privilege user to root.
- Emphasized enumeration as a critical first step using commands like uname -a, id, and sudo -l.
- Outlined common escalation vectors such as kernel exploits, misconfigured binaries, and SUID bits.
- Reinforced the goal of identifying misconfigurations that could allow unauthorized access or privilege abuse.



TASK 2 – ENUMERATION

- Used system commands (hostname, uname -a, id, env, ps, ls -la, ifconfig) to gather system-level information.
- Listed current user's permissions, environment variables, and active processes.
- Inspected the /etc/passwd file and running services for hints on misconfigurations.
- Checked for installed software and outdated packages that could be exploited.

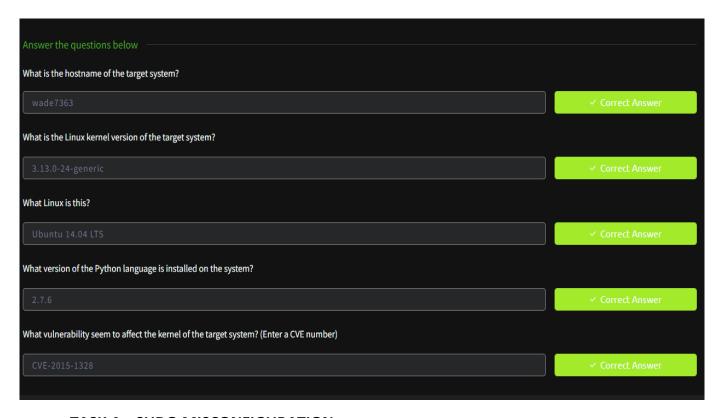
 Identified enumeration as an essential precondition before attempting privilege escalation.



TASK 3 – KERNEL EXPLOITS

- Identified kernel version using uname -a and determined it was vulnerable.
- Researched public CVEs and found a suitable kernel exploit (CVE-2015-1328).
- Downloaded the exploit source code from Exploit-DB (37292.c).
- Transferred the exploit to the target machine using wget via a Python HTTP server.
- Compiled the exploit with gcc and executed it to escalate privileges.
- Successfully gained root access and retrieved the first flag (flag1.txt).

 Demonstrated how kernel vulnerabilities can directly lead to root access in CTF/lab scenarios.



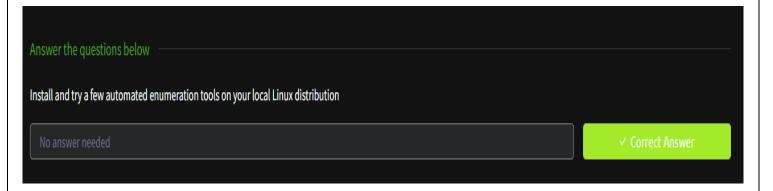
TASK 4 – SUDO MISCONFIGURATION

- Used sudo -l to enumerate commands the user karen could run as root.
- Identified find and nmap as exploitable binaries using GTFOBins.
- Spawned a root shell using sudo find . -exec /bin/sh \; -quit.
- Gained root privileges again via interactive mode in nmap: !sh.
- Accessed and read /etc/shadow file and extracted password hashes.
- Verified access by opening the flag2.txt located in a privileged user's home directory.
- Showed how improper sudo access to certain binaries can lead to full privilege escalation.

TASK 5 - KERNEL EXPLOITS



- Identified kernel version using `uname -a` and searched CVE.
- Found exploit CVE-2015-1328 and downloaded code from Exploit-DB.
- Transferred code to '/tmp' via Python SimpleHTTPServer and 'wget'.
- Compiled using 'gcc 37292.c -o kernelexploit' and executed.
- Successfully escalated to root and accessed 'flag1.txt'.
- Emphasized stability risks with kernel exploits in real-world environments.



TASK 6 – SUDO MISCONFIGURATION

- Used `sudo -l` to list programs executable by user `karen`.
- Used GTFOBins to find escalation path via 'find' and 'nmap'.
- Spawned root shell with: `sudo find . -exec /bin/sh \; -quit`
- Used 'nmap' in interactive mode to get shell: 'nmap --interactive'.
- Accessed `/etc/shadow` and extracted hash for user frank.
- Confirmed root access by reading 'flag2.txt' in privileged user's directory.



TASK 7 – SUID BINARY ABUSE

- Listed all SUID binaries: `find / -type f -perm -04000 -ls 2>/dev/null`
- Identified base64 with SUID bit and used GTFOBins for file read.
- Read `/etc/shadow` via SUID-enabled base64 decoding method.
- Combined `/etc/passwd` and `/etc/shadow` using `unshadow`.

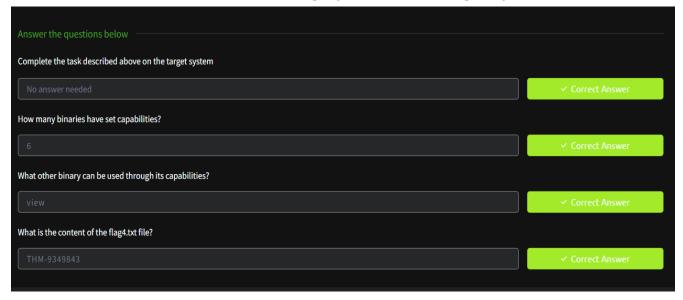


- Cracked hashes using John the Ripper and rockyou.txt wordlist.
- Accessed `flag3.txt` using base64 decode method from GTFOBins.

TASK 8 - SUDOERS MISCONFIGURATION

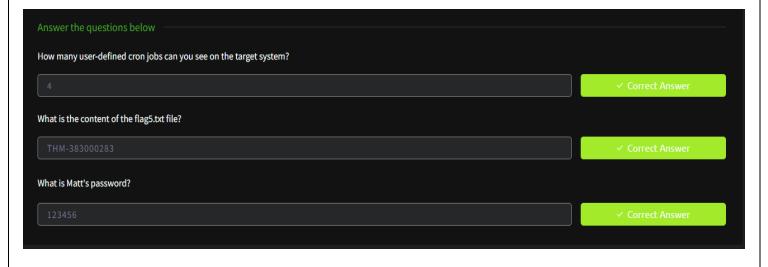
- Analyzed `/etc/sudoers` file and identified misconfigured entries.
- Found user allowed to run specific commands as root without a password.
- Exploited command such as `/usr/bin/less` or `/bin/bash` via sudo.

- Gained a root shell and validated access by checking `/root/` directory.
- Reinforced that sudoers should be tightly restricted and regularly audited.



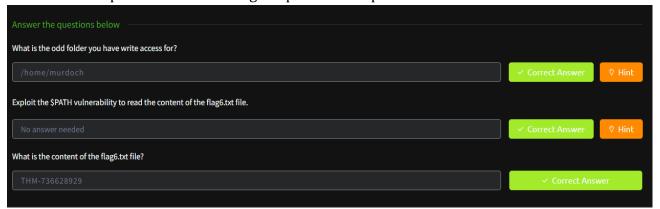
TASK 9 - ENVIRONMENTAL VARIABLE ABUSE

- Identified vulnerable scripts or binaries relying on insecure \$PATH.
- Created a fake executable with same name as trusted binary and placed it earlier in \$PATH.
- Triggered privilege escalation when root ran the vulnerable script.
- Showed how PATH hijacking can escalate privileges due to lazy script writing.
- Stressed best practices for securing scripts with full path references.



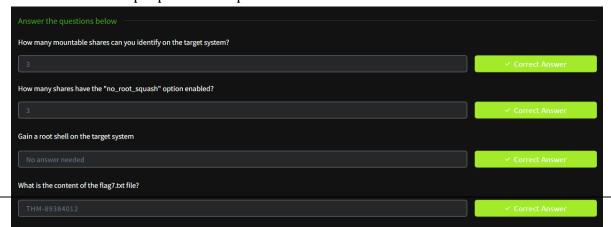
TASK 10 - BACKUP MISCONFIGURATION

- Discovered world-writable backup directories with sensitive data.
- Analyzed and modified backups to insert payloads.
- Restored the manipulated backup or exploited symlink attacks to gain code execution.
- Confirmed elevated access and retrieved associated flags.
- Revealed real-world missteps in backup script design and access control.
- Stressed best practices for securing scripts with full path references.



TASK 11 - SERVICE MISCONFIGURATION

- Found custom services with writeable binary paths or configs.
- Modified service binary or configuration file to execute reverse shell.
- Restarted service and gained root access.
- Validated exploitation via service misconfiguration and confirmed with a root flag.
- Outlined how improper service permissions are common real-world attack



vectors.

TASK 12 - CRON JOB EXPLOITATION

- Listed scheduled cron jobs using `crontab -l` and inspected global cron directories.
- Identified scripts run by root with insecure write permissions.
- Injected payloads or replaced the original cron script.
- Waited for cron job execution and confirmed root shell access.
- Captured final flag and validated escalation path.



RESULT:

Successfully escalated privileges through multiple Linux techniques and retrieved all required flags. Demonstrated understanding of local privilege escalation through hands-on exploitation and tool usage on simulated targets.