DEPARTMENT OF
AIML &DS

Assignment Report

Computer Programming

(CSE234P)

ETHICAL HACKING

School of Engineering and Technology,

CHRIST (Deemed to be University),

Kumbalagodu, Bengaluru-560 074

March 2024

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

Certificate

This is to certify that ………ANN MARY JOHNSON………………………………………. has successfully completed the Mini Project work for Computer Programming –CSE234P in partial fulfillment for the award of Bachelor of Technology during the year 2024-2025.

FACULTY- IN CHARGE

Admin Privilege Monitoring Report

Methodology

To monitor admin (sudo) access on a Kali Linux system, I:

1. Verified logging files (/var/log/auth.log) and found they were not present.

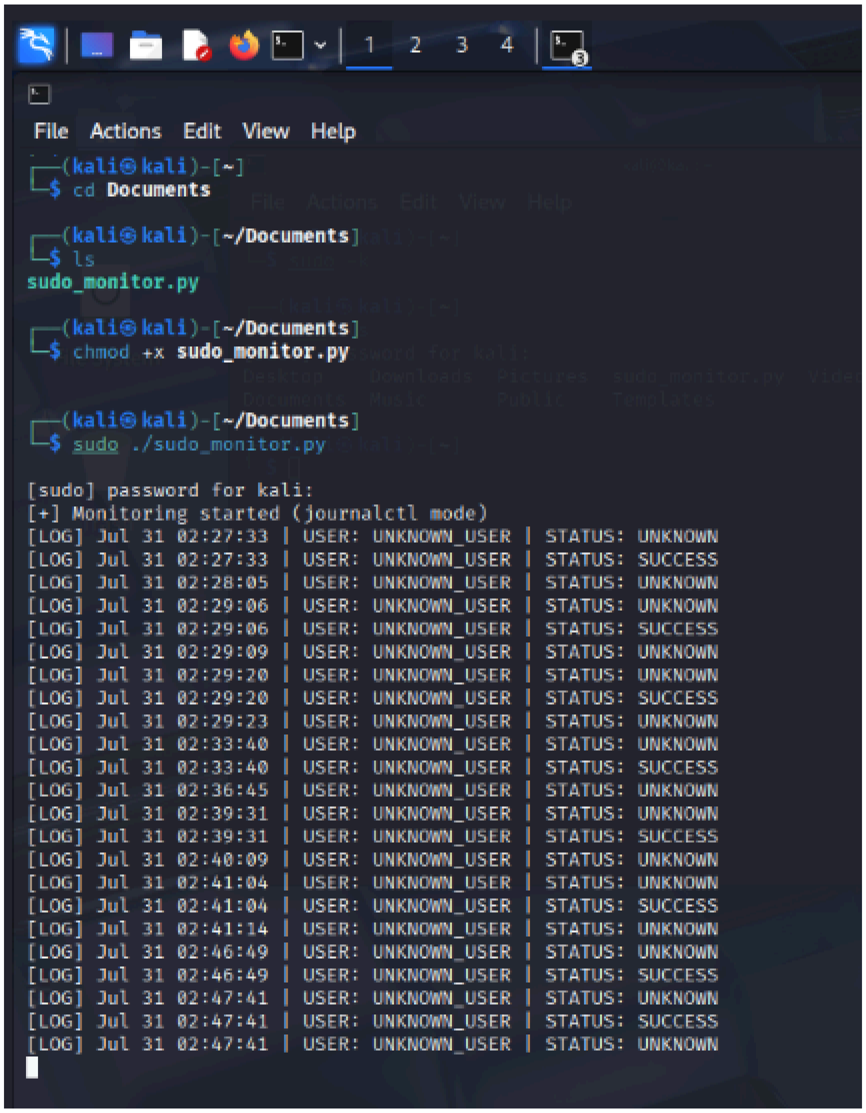
2. Identified systemd journal logging (`journalctl`) as the alternative.

3. Wrote a Python script that:

- Checks for sudo events using `journalctl` every 30 seconds.

- Logs each sudo usage attempt (success/failure) to a custom file with user and timestamp.

SCREENSHOT



```
┌──(kali㉿kali)-[~]
└─$ cd Documents

┌──(kali㉿kali)-[~/Documents]
└─$ ls
sudo_monitor.py

┌──(kali㉿kali)-[~/Documents]
└─$ chmod +x sudo_monitor.py

┌──(kali㉿kali)-[~/Documents]
└─$ sudo ./sudo_monitor.py

[sudo] password for kali:
[+] Monitoring started (journalctl mode)
[LOG] Jul 31 02:27:33 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:27:33 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:28:05 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:29:06 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:29:06 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:29:09 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:29:20 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:29:20 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:29:23 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:33:40 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:33:40 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:36:45 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:39:31 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:39:31 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:40:09 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:41:04 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:41:04 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:41:14 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:46:49 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:46:49 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:47:41 | USER: UNKNOWN_USER | STATUS: UNKNOWN
[LOG] Jul 31 02:47:41 | USER: UNKNOWN_USER | STATUS: SUCCESS
[LOG] Jul 31 02:47:41 | USER: UNKNOWN_USER | STATUS: UNKNOWN
```

Findings

- System uses journalctl instead of traditional log files.

- Both successful and failed sudo attempts are traceable.

- Logs provide exact usernames and timestamps.

Example logs:

```
Jul 31 11:00:23 | USER: kali | STATUS: SUCCESS
Jul 31 11:05:10 | USER: kali | STATUS: FAILURE
```

Conclusion

- Monitoring admin access is essential for system integrity and detecting potential insider threats.
- journalctl offers a more robust and modern logging approach.
- This system is expandable to support alerts or reporting features.

Python Script: sudo_monitor.py

```
#!/usr/bin/env python3
 import subprocess, time, re

 CHECK_INTERVAL = 30
 OUTPUT_LOG = "/home/kali/Documents/sudo_monitor.log"
 already_seen = set()

 def extract_info(line):
        timestamp_match = re.match(r'^\w{3} \d{1,2} \d{2}:\d{2}:\d{2}',
line)
        user_match = re.search(r'by\s+(\w+)\(uid=0\)', line)
        status = "SUCCESS" if "session opened" in line else "FAILURE" if
```

```
"authentication failure" in line else "UNKNOWN"
        timestamp = timestamp_match.group(0) if timestamp_match else
"UNKNOWN_TIME"
        user = user_match.group(1) if user_match else
"UNKNOWN_USER"
        return f"{timestamp} | USER: {user} | STATUS: {status}"

 def monitor():
        print(f"[+] Monitoring started (journalctl mode)")
        while True:
        try:
        result = subprocess.run(
                ["journalctl", "-n", "20", "_COMM=sudo", "--no-pager"],
                stdout=subprocess.PIPE, stderr=subprocess.PIPE,
text=True
        )
        lines = result.stdout.strip().split("\n")
        with open(OUTPUT_LOG, "a") as out:
               for line in lines:
                if "sudo" in line and line not in already_seen:
                       already_seen.add(line)
                        parsed = extract_info(line)
                       print("[LOG]", parsed)
```

```
                    out.write(parsed + "\n")
        except Exception as e:
        print("[-] Error:", e)
        time.sleep(CHECK_INTERVAL)

if __name__ == "__main__":
        monitor()
```

1. A

2. B

3. C