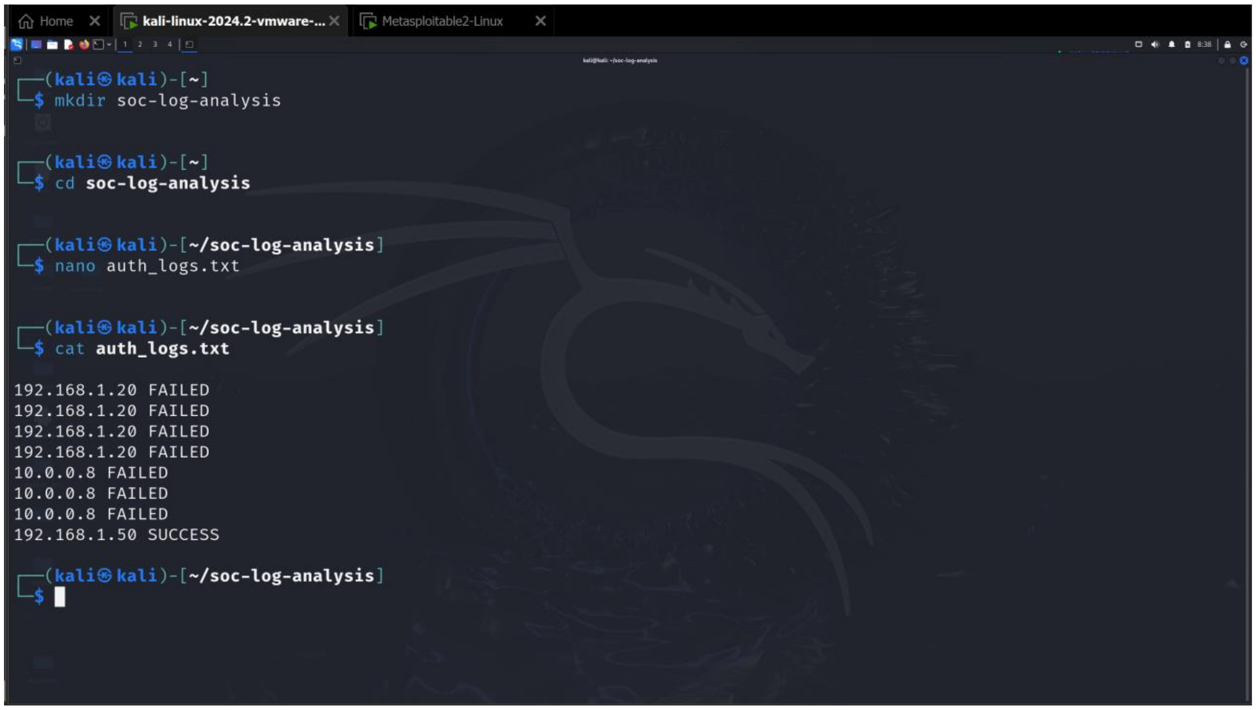
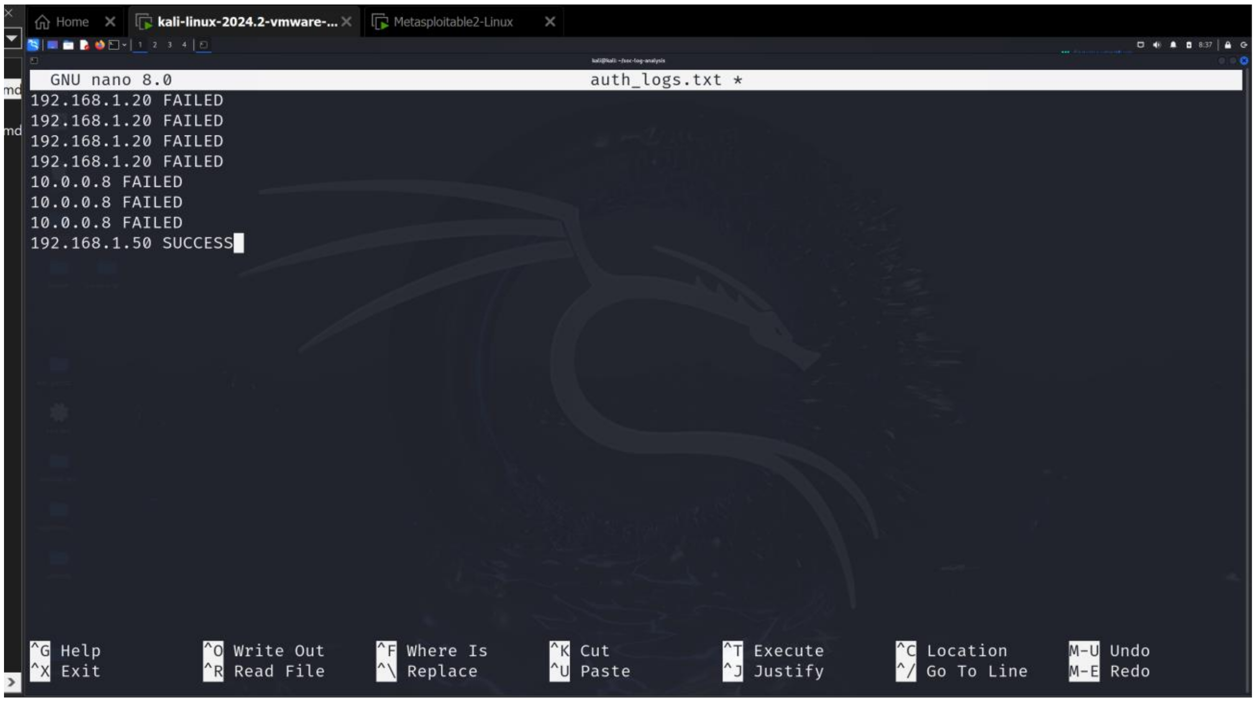
Brute Force Attack Detection & Response System

Step 1: Create Authentication Logs



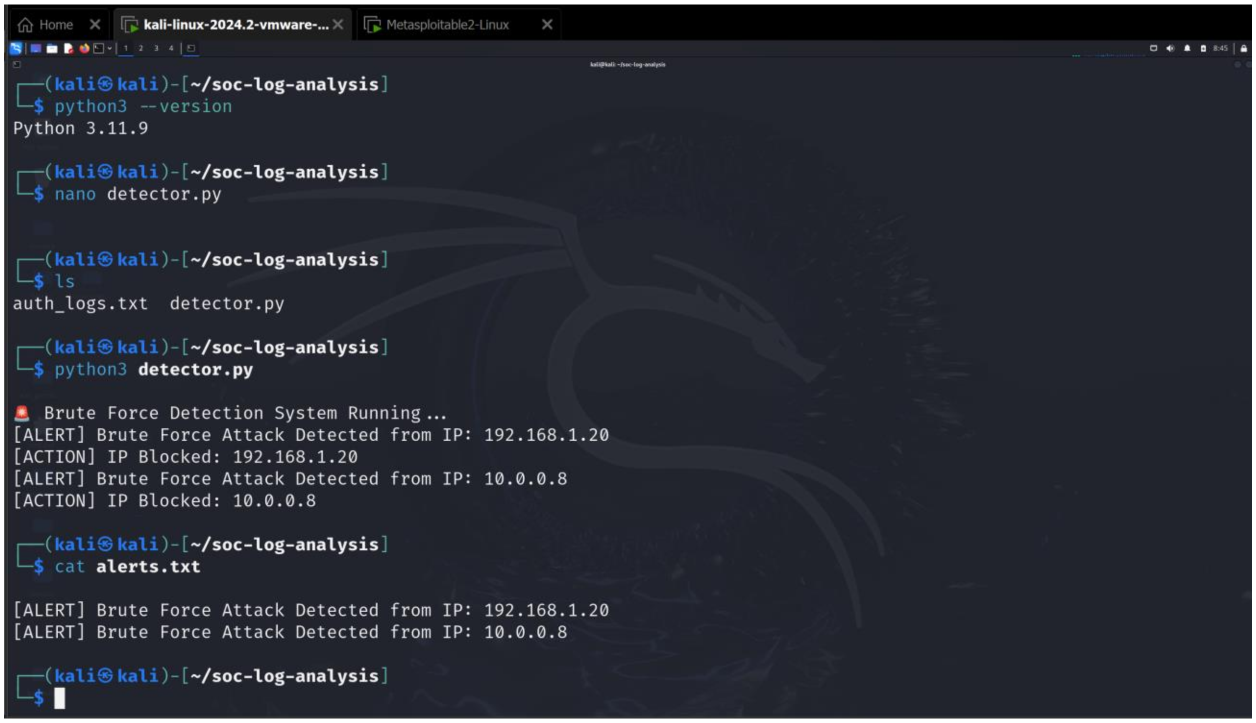
```
┌──(kali㉿kali)-[~]
└─$ mkdir soc-log-analysis

┌──(kali㉿kali)-[~]
└─$ cd soc-log-analysis

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ nano auth_logs.txt

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ cat auth_logs.txt

192.168.1.20 FAILED
192.168.1.20 FAILED
192.168.1.20 FAILED
192.168.1.20 FAILED
10.0.0.8 FAILED
10.0.0.8 FAILED
10.0.0.8 FAILED
192.168.1.50 SUCCESS

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ 
```

Nano auth_logs.txt



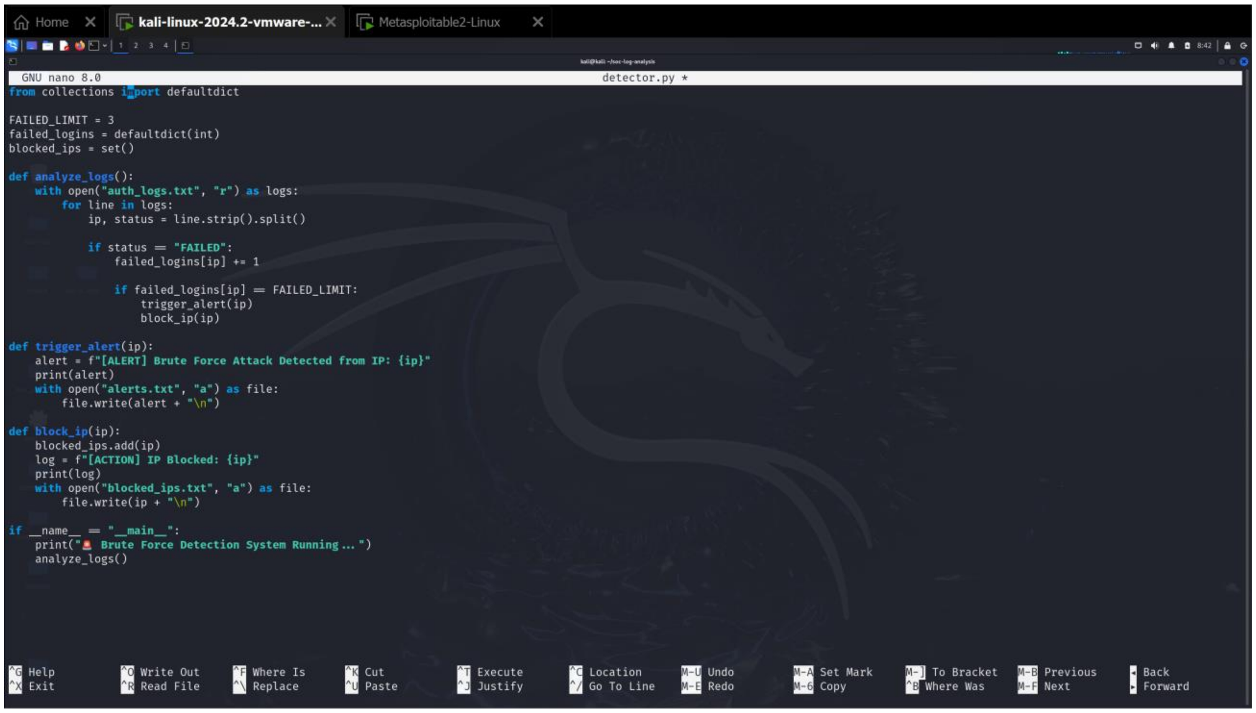
```
  GNU nano 8.0                              auth_logs.txt *
192.168.1.20 FAILED
192.168.1.20 FAILED
192.168.1.20 FAILED
192.168.1.20 FAILED
10.0.0.8 FAILED
10.0.0.8 FAILED
10.0.0.8 FAILED
192.168.1.50 SUCCESS
```

Step 2: Write Detection & Response Script



```
┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ python3 --version
Python 3.11.9

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ nano detector.py


┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ ls
auth_logs.txt  detector.py

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ python3 detector.py

🚨 Brute Force Detection System Running...
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ACTION] IP Blocked: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ACTION] IP Blocked: 10.0.0.8

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ cat alerts.txt

[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ 
```

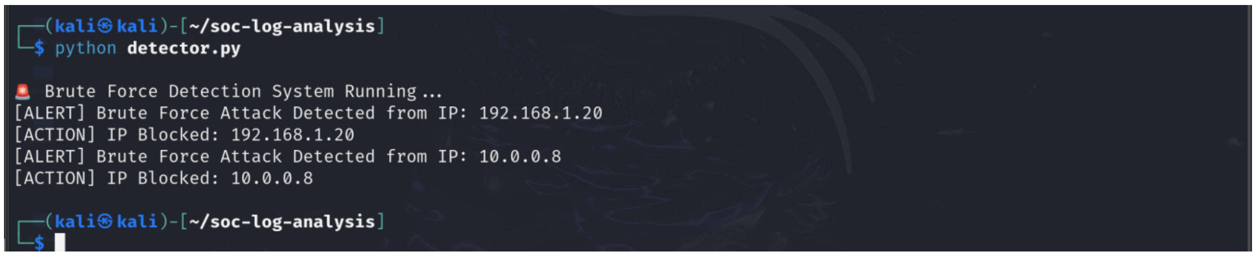
Nano detector.py



```
  GNU nano 8.0                                detector.py *
from collections import defaultdict

FAILED_LIMIT = 3
failed_logins = defaultdict(int)
blocked_ips = set()

def analyze_logs():
    with open("auth_logs.txt", "r") as logs:
        for line in logs:
            ip, status = line.strip().split()

            if status == "FAILED":
                failed_logins[ip] += 1

                if failed_logins[ip] == FAILED_LIMIT:
                    trigger_alert(ip)
                    block_ip(ip)

def trigger_alert(ip):
    alert = f"[ALERT] Brute Force Attack Detected from IP: {ip}"
    print(alert)
    with open("alerts.txt", "a") as file:
        file.write(alert + "\n")

def block_ip(ip):
    blocked_ips.add(ip)
    log = f"[ACTION] IP Blocked: {ip}"
    print(log)
    with open("blocked_ips.txt", "a") as file:
        file.write(ip + "\n")

if __name__ == "__main__":
    print("🚨 Brute Force Detection System Running...")
    analyze_logs()
```

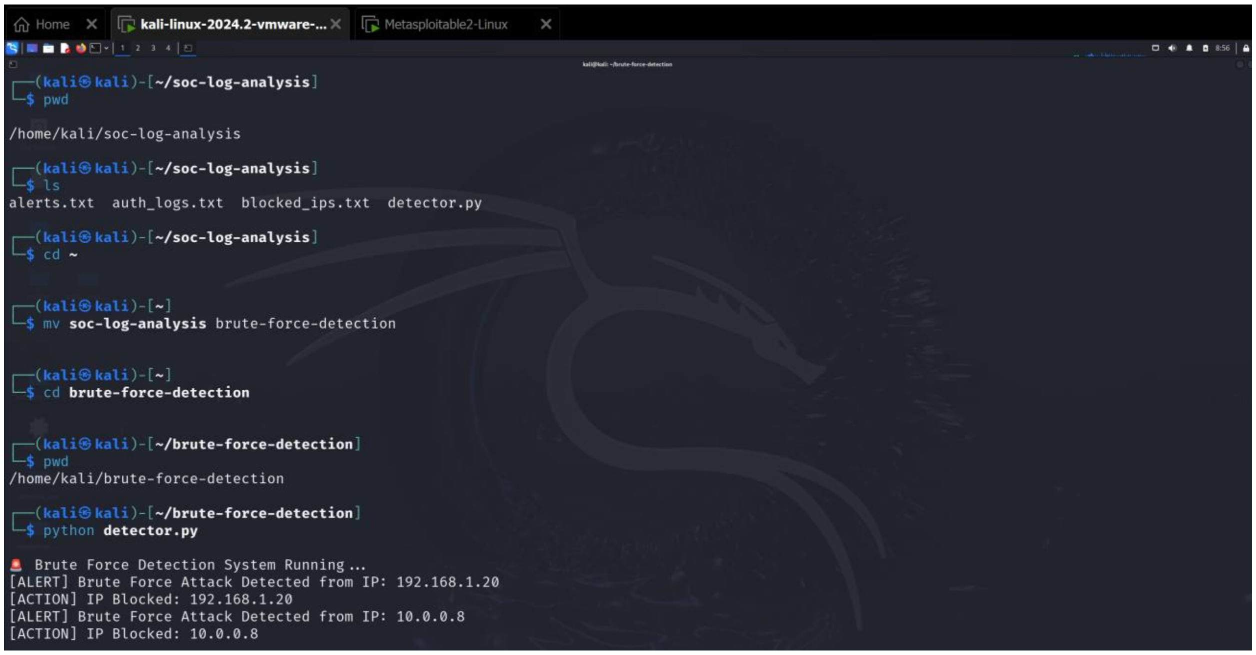
Step 3: Run the Project

Python detector.py



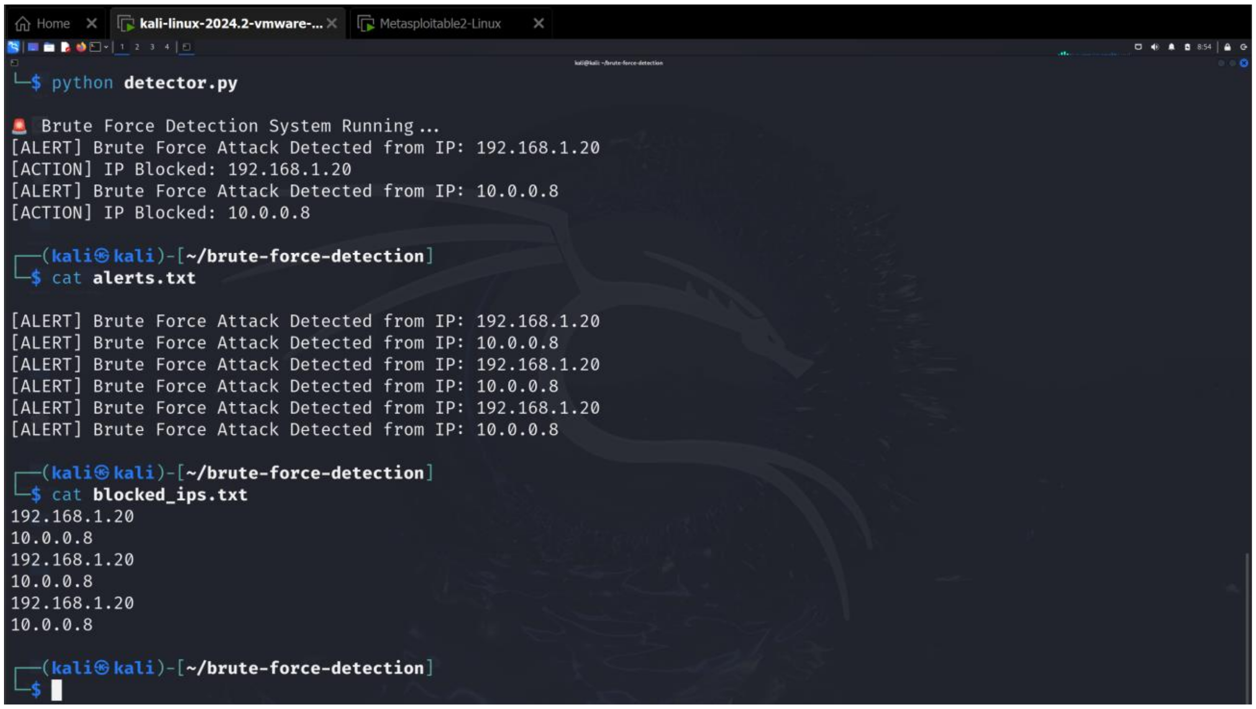
```
┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ python detector.py

🚨 Brute Force Detection System Running...
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ACTION] IP Blocked: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ACTION] IP Blocked: 10.0.0.8

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$
```

Step 4: Output Files



```
┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ pwd

/home/kali/soc-log-analysis

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ ls
alerts.txt  auth_logs.txt  blocked_ips.txt  detector.py

┌──(kali㉿kali)-[~/soc-log-analysis]
└─$ cd ~


┌──(kali㉿kali)-[~]
└─$ mv soc-log-analysis brute-force-detection


┌──(kali㉿kali)-[~]
└─$ cd brute-force-detection


┌──(kali㉿kali)-[~/brute-force-detection]
└─$ pwd
/home/kali/brute-force-detection

┌──(kali㉿kali)-[~/brute-force-detection]
└─$ python detector.py

🚨 Brute Force Detection System Running...
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ACTION] IP Blocked: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ACTION] IP Blocked: 10.0.0.8
```

alerts.txt blocked_ips.txt



```
└─$ python detector.py

🚨 Brute Force Detection System Running ...
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ACTION] IP Blocked: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ACTION] IP Blocked: 10.0.0.8

┌──(kali㉿kali)-[~/brute-force-detection]
└─$ cat alerts.txt

[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
[ALERT] Brute Force Attack Detected from IP: 10.0.0.8

┌──(kali㉿kali)-[~/brute-force-detection]
└─$ cat blocked_ips.txt
192.168.1.20
10.0.0.8
192.168.1.20
10.0.0.8
192.168.1.20
10.0.0.8

┌──(kali㉿kali)-[~/brute-force-detection]
└─$ 
```

Step 5: Cyber Security Explanation

STEP 5.1: Understand the Security Problem

Problem: Brute Force Attack

A brute force attack happens when:

- An attacker tries many passwords
- From the same IP address
- Until access is gained

This is one of the most common real-world attacks.

STEP 5.2: Explain the Log File (auth_logs.txt)

Your log file simulates authentication attempts.

Example:

192.168.1.20 FAILED

192.168.1.20 FAILED

192.168.1.20 FAILED

What this means:

- Same IP
- Multiple failed logins
- Suspicious behavior

In real systems, these logs come from:

- SSH
- Web login pages
- VPN
- Firewalls

STEP 5.3: Explain the Detection Logic (VERY IMPORTANT)

My script uses this rule:

Your script uses this rule:

Detection Logic:

1. Read each log entry
2. Count FAILED attempts per IP
3. If FAILED ≥ 3
4. Trigger an alert
5. Block the IP

This is called **Threshold-Based Detection**

Used in:

- SIEM systems
- IDS/IPS
- SOC monitoring tools

STEP 5.4: Explain the Alert System

When an attack is detected, this happens:

```
[ALERT] Brute Force Attack Detected from IP: 192.168.1.20
```

Why alerts matter:

- SOC analysts monitor alerts

- Alerts allow quick response
- Prevent account compromise

Stored in:

alerts.txt

This simulates **SIEM alert logs**

STEP 5.5: Explain the Response Action (Blocking)

After alerting, the system blocks the IP:

192.168.1.20

Stored in:

blocked_ips.txt

Real-world equivalent:

- Firewall rule
- Fail2Ban
- IP blacklisting
- WAF rule

This shows **incident response skills**

STEP 5.6: Map This Project to Real SOC Work

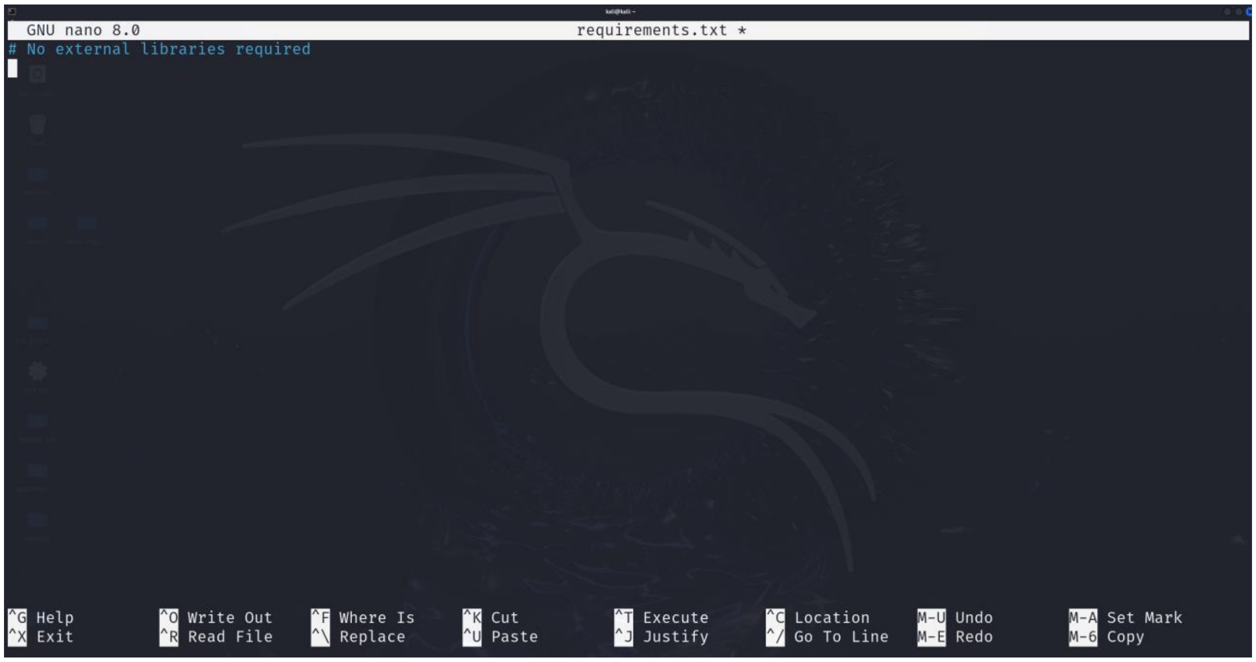
Project Feature	Real SOC Equivalent
Log analysis	SIEM (Splunk, ELK)
Threshold detection	IDS rules
Alerts	SOC alerts
IP blocking	Firewall / IPS
Python automation	SOAR

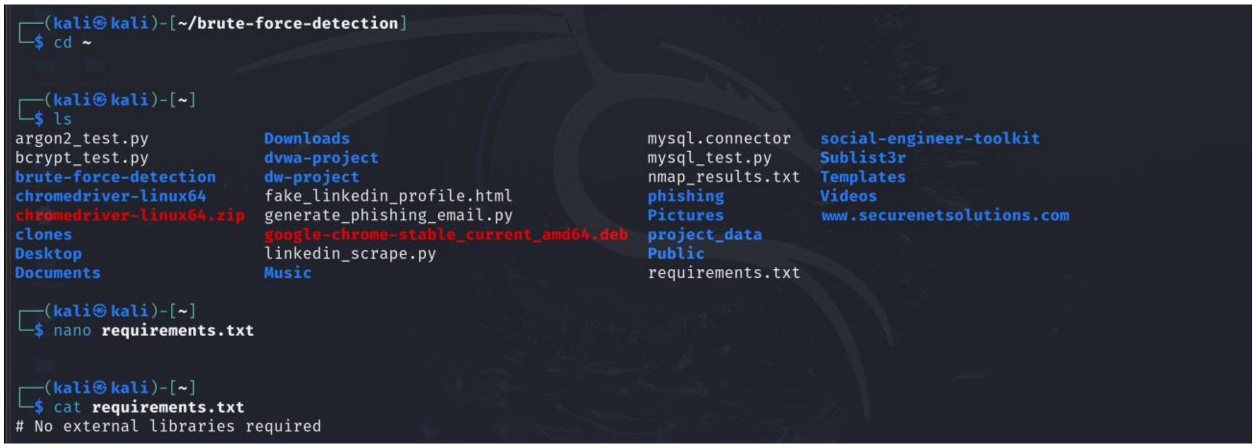
"I built a Python-based brute-force detection system that analyzes authentication logs, triggers alerts when suspicious behavior is detected, and simulates automated IP blocking. This project reflects real SOC detection and response workflows."

Step 6: requirements.txt

Nano requirements.txt

# No external libraries required


```
  GNU nano 8.0                           requirements.txt *
# No external libraries required


^G Help       ^O Write Out  ^F Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo      M-6 Copy
```


```
┌──(kali㉿kali)-[~/brute-force-detection]
└─$ cd ~

┌──(kali㉿kali)-[~]
└─$ ls
argon2_test.py            Downloads                                mysql.connector   social-engineer-toolkit
bcrypt_test.py            dvwa-project                             mysql_test.py     Sublist3r
brute-force-detection     dw-project                               nmap_results.txt  Templates
chromedriver-linux64      fake_linkedin_profile.html               phishing          Videos
chromedriver-linux64.zip  generate_phishing_email.py               Pictures          www.securenetsolutions.com
clones                    google-chrome-stable_current_amd64.deb   project_data
Desktop                   linkedin_scrape.py                       Public
Documents                 Music                                    requirements.txt

┌──(kali㉿kali)-[~]
└─$ nano requirements.txt

┌──(kali㉿kali)-[~]
└─$ cat requirements.txt
# No external libraries required
```