SSH Brute Force Detection using Bash and Cron

Introduction

SSH servers are common targets for brute-force attacks where attackers repeatedly try different passwords to gain access. To detect such attempts, I created a lightweight Bash script that monitors failed SSH login attempts from system logs. This project automates the detection process and provides real-time alerts stored in a log file.

Objective

- Detect repeated failed SSH login attempts.
- Generate alerts with timestamp and source IP.
- Automate monitoring using cron to reduce manual effort.
- Build a simple and extendable solution for Linux systems.

Tools & Environment

- **Operating System**: Kali Linux (VirtualBox VM)
- **Tools Used**:
  - Bash Scripting – for automation
  - journalctl – to read system logs
  - cron – to schedule automatic execution
  - awk, grep, sort, uniq – for log processing

Steps

1. Create the Script

File: log_monitor.sh

```
#!/bin/bash
# Simple Log Monitoring Script for SSH brute-force detection (systemd version)
THRESHOLD=5
ALERT_FILE="$HOME/ssh_alerts1.log"
echo "Monitoring failed SSH logins from journalctl ..."
/usr/bin/journalctl -u ssh -n 2000 | grep "Failed password" | awk '{print $(NF-3)}' | sort | uniq -c | sort -nr | while read count ip; do
  if [ $count -gt $THRESHOLD ]; then
```

```
        ALERT="⚠ ALERT: Suspicious activity from $ip - $count failed attempts"
        echo "$ALERT"
        echo "$(date) - $ALERT" >> "$ALERT_FILE"
    fi
done
```

2. Make Script Executable

Cmd : chmod +x ~/log_monitor.sh

3. Test Manually

Cmd : bash ~/log_monitor.sh

Cmd : cat ~/ssh_alerts1.log

4. Automate with Cron

- Open cron jobs: **Cmd : crontab -e**
- Add job (runs every 5 minutes):

  */5 * * * * /home/kali/log_monitor.sh

5. Verify Cron job

- Check current cron jobs:

  Cmd : crontab -l
- Expected output:

  */5 * * * * /home/kali/log_monitor.sh
- Check if cron service is active:

  Cmd : systemctl status cron
- Output should show: **active (running)**

6. Generate failed login attempts (for testing)

Cmd : ssh kali@localhost

Enter wrong password multiple times to simulate brute-force attempts.

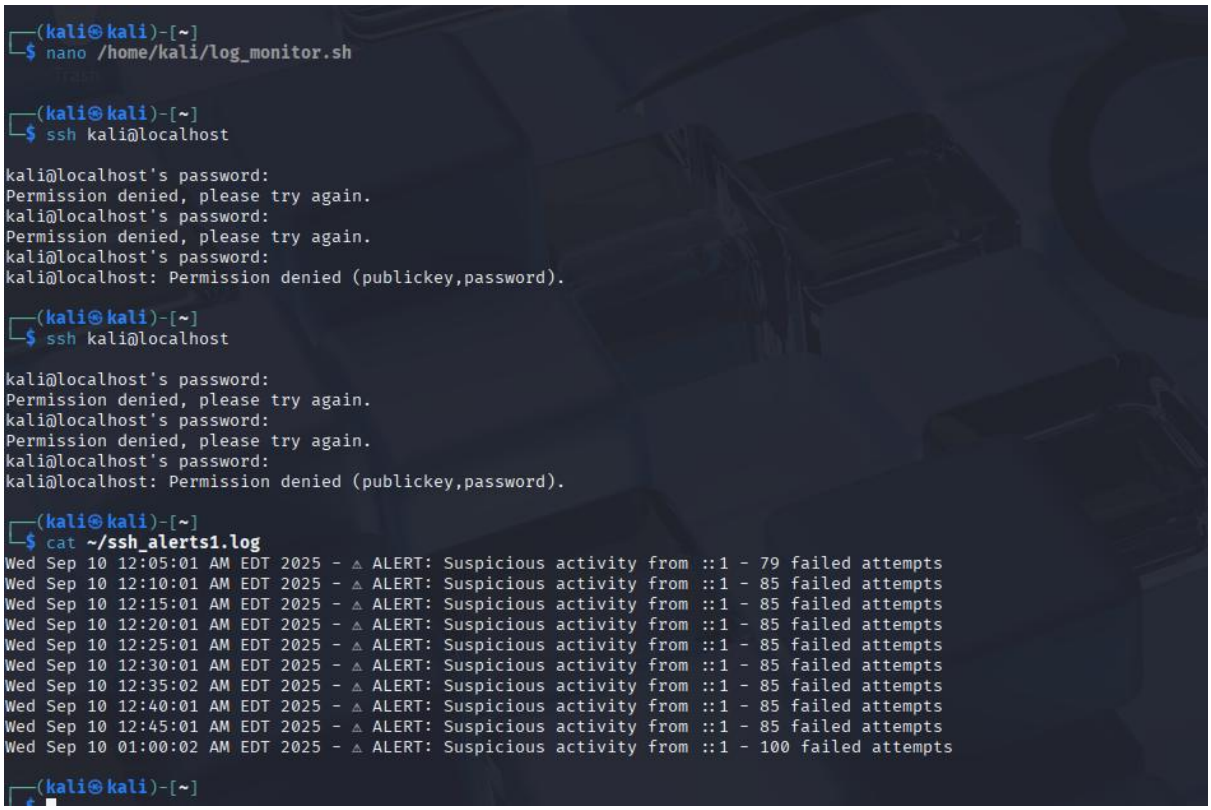
7. Confirm alerts

After 5–10 minutes, check: **Cmd :** cat ~/ssh_alerts1.log

New alerts should be appended with timestamp.

Results

Sample output in log file (ssh_alerts1.log):



- Script successfully detected repeated failed SSH login attempts.
- Alerts were stored in a log file with timestamps.
- Cron ensured automatic monitoring without manual checks.

Conclusion

This project provided a simple but effective way to detect SSH brute-force attempts using Bash and cron. It can be extended further to send alerts via email or integrate with SIEM tools for enterprise-level monitoring.