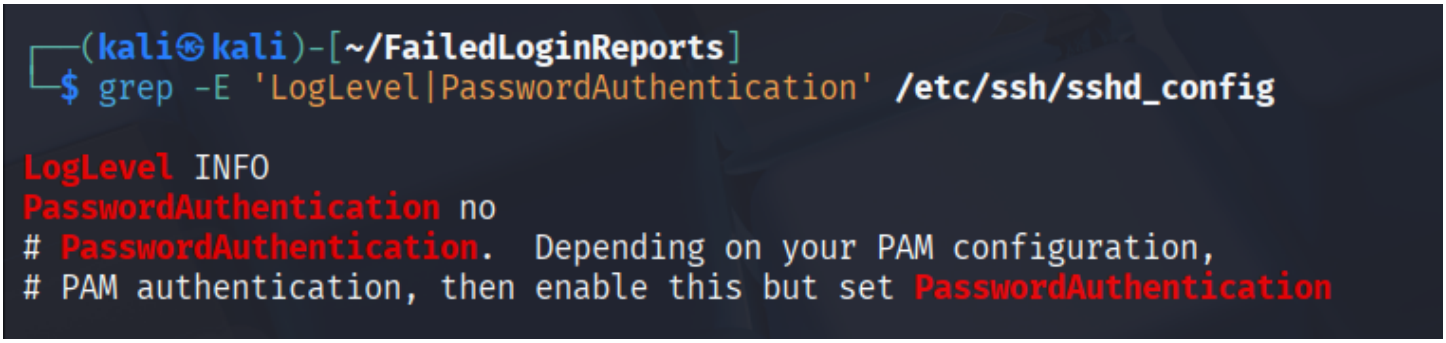
Title: Failed Login Incidents — Daily SSH Report
Date: 2025-08-22
Host: kali (Lab)
Log Source: journalctl (tag:sshd, fallback unit:ssh)
Author: ***Muhaned Alhashimy***

1) Executive Summary

During a controlled lab exercise, we recorded and analyzed failed SSH login attempts to validate a lightweight reporting pipeline. No compromise occurred. Password authentication was enabled temporarily for testing and then reverted to keys-only.

Key KPIs (today)

Metric	Value
Total failed password attempts	33
Invalid-user attempts	15
Top IP	127.0.0.1 (33)
Most targeted usernames	kali (18), usernotexists (15)



[SSHD config before/after]

Alt text: “sshd_config shows LogLevel INFO and PasswordAuthentication no”

2) Environment

- OS: Kali (systemd-journald)
- Service: OpenSSH (ssh)
- Test traffic was generated via sshpass from localhost with wrong passwords.

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ for i in {1..6}; do sshpass -p WrongPass ssh -o PreferredAuthentications=password

-o PubkeyAuthentication=no -o StrictHostKeyChecking=no usernotexists@127.0.0.1 true || true; done

usernotexists@127.0.0.1: Permission denied (publickey).
usernotexists@127.0.0.1: Permission denied (publickey).
usernotexists@127.0.0.1: Permission denied (publickey).
usernotexists@127.0.0.1: Permission denied (publickey).
usernotexists@127.0.0.1: Permission denied (publickey).
usernotexists@127.0.0.1: Permission denied (publickey).
```

[Generating failed attempts]

Alt text: “sshpass loop sending wrong passwords to localhost”

3) Methodology

1. **Collect raw events** from journald
   journalctl -t sshd -n 2000 *(fallback: journalctl -u ssh -n 2000)*
2. **Filter by regex**
   Failed password|Invalid user|authentication failure|PAM.*authentication failure
3. **Parse & report**
   - Markdown: reports/failed_login_report_YYYY-MM-DD.md
   - CSV: reports/failed_login_YYYY-MM-DD.csv
4. **Alert** if failed passwords ≥ threshold *(default: 20)*

Raw sample in terminal

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ sudo journalctl -t sshd -n 2000 | grep -E 'Failed password|Invalid user|authentication failure|PAM.*authentication failure' | sed -n '1,15p'
```

Raw count & head

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ wc -l logs/auth_failed_$(date +%F).log
81 logs/auth_failed_2025-08-22.log

┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ head -n 5 logs/auth_failed_$(date +%F).log
Aug 22 10:24:55 kali sshd-session[37139]: Invalid user usernotexists from 127.0.0.1 port 38194
Aug 22 10:24:55 kali sshd-session[37139]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=127.0.0.1
Aug 22 10:24:57 kali sshd-session[37139]: Failed password for invalid user usernotexists from 127.0.0.1 port 38194 ssh2
Aug 22 10:24:58 kali sshd-session[37167]: Invalid user usernotexists from 127.0.0.1 port 40484
Aug 22 10:24:58 kali sshd-session[37167]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=127.0.0.1
```

4) Findings (Today)

- **Total failed password attempts: 33**
- **Invalid-user attempts: 15**
- **Top IPs:** 127.0.0.1 — 33
- **Most targeted usernames:** kali — 18, usernotexists — 15
- **Peak hours (hour → count):** 10:00 → 8, 12:00 → 18, 13:00 → 7

Run report script

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ python3 scripts/report_auth_fail.py
Done: reports/failed_login_report_2025-08-22.md
```

Markdown report preview

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ head -n 5 logs/auth_failed_$(date +%F).log
Aug 22 10:24:55 kali sshd-session[37139]: Invalid user usernotexists from 127.0.0.1 port 38194
Aug 22 10:24:55 kali sshd-session[37139]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=127.0.0.1
Aug 22 10:24:57 kali sshd-session[37139]: Failed password for invalid user usernotexists from 127.0.0.1 port 38194 ssh2
Aug 22 10:24:58 kali sshd-session[37167]: Invalid user usernotexists from 127.0.0.1 port 40484
Aug 22 10:24:58 kali sshd-session[37167]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=127.0.0.1

┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ python3 scripts/report_auth_fail.py
Done: reports/failed_login_report_2025-08-22.md

┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ sed -n '1,80p' reports/failed_login_report_$(date +%F).md
# Failed Login Incidents Report — 2025-08-22

- Total failed password attempts: **33**
- Invalid-user attempts: **15**

## Top IPs
- 127.0.0.1: 33

## Most targeted usernames
- kali: 18
- usernotexists: 15

## Peak hours (hour → count)
- 10:00 → 8
- 12:00 → 18
- 13:00 → 7

## Log samples
- `Aug 22 10:24:57 kali sshd-session[37139]: Failed password for invalid user usernotexists from 127.0.0.1 port 38194 ssh2`
- `Aug 22 10:24:59 kali sshd-session[37167]: Failed password for invalid user usernotexists from 127.0.0.1 port 40484 ssh2`
- `Aug 22 10:25:02 kali sshd-session[37187]: Failed password for invalid user usernotexists from 127.0.0.1 port 40488 ssh2`
- `Aug 22 10:25:05 kali sshd-session[37219]: Failed password for invalid user usernotexists from 127.0.0.1 port 40500 ssh2`
- `Aug 22 10:25:09 kali sshd-session[37247]: Failed password for invalid user usernotexists from 127.0.0.1 port 40502 ssh2`

## Quick recommendations
- Enable Fail2ban; prefer SSH keys over passwords; restrict access via UFW; disable unused accounts.
```

CSV export

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ python3 scripts/export_csv.py && ls -l reports/*$(date +%F).csv
CSV: reports/failed_login_2025-08-22.csv
-rw-rw-r-- 1 kali kali 1646 Aug 22 16:44 reports/failed_login_2025-08-22.csv
```

Threshold alert

```
┌──(kali㉿kali)-[~/FailedLoginReports]
└─$ th=20; cnt=$(grep -c 'Failed password' logs/auth_failed_$(date +%F).log); echo "[ALERT] $cnt failed logins today"
[ALERT] 33 failed logins today
```

5) Evidence

- Raw log file: logs/auth_failed_2025-08-22.log *(81 lines)*
- Sample entry:
  Aug 22 10:24:57 kali sshd-session[37139]: Failed password for invalid user usernotexists from 127.0.0.1 port 38194 ssh2

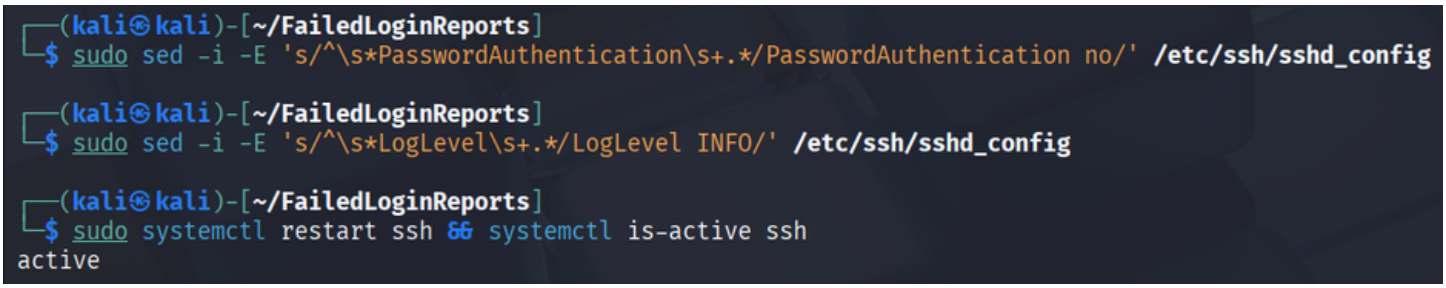
6) Impact

Lab-only scope; attempts were intentionally generated from localhost. No unauthorized access observed.

7) Actions Taken

- Reverted SSH to keys-only: PasswordAuthentication no
- Restored normal logging level: LogLevel INFO

Hardened back



8) Recommendations (Next Steps)

1. Enable **Fail2ban** (sshd jail; tune maxretry, findtime, bantime).
2. Keep SSH keys only; disable/lock unused accounts.
3. If exposed to the internet, restrict via **UFW** to trusted sources.
4. Optional: schedule a daily run (cron) and centralize to a SIEM later.

9) Project Structure

FailedLoginReports/

├── logs/

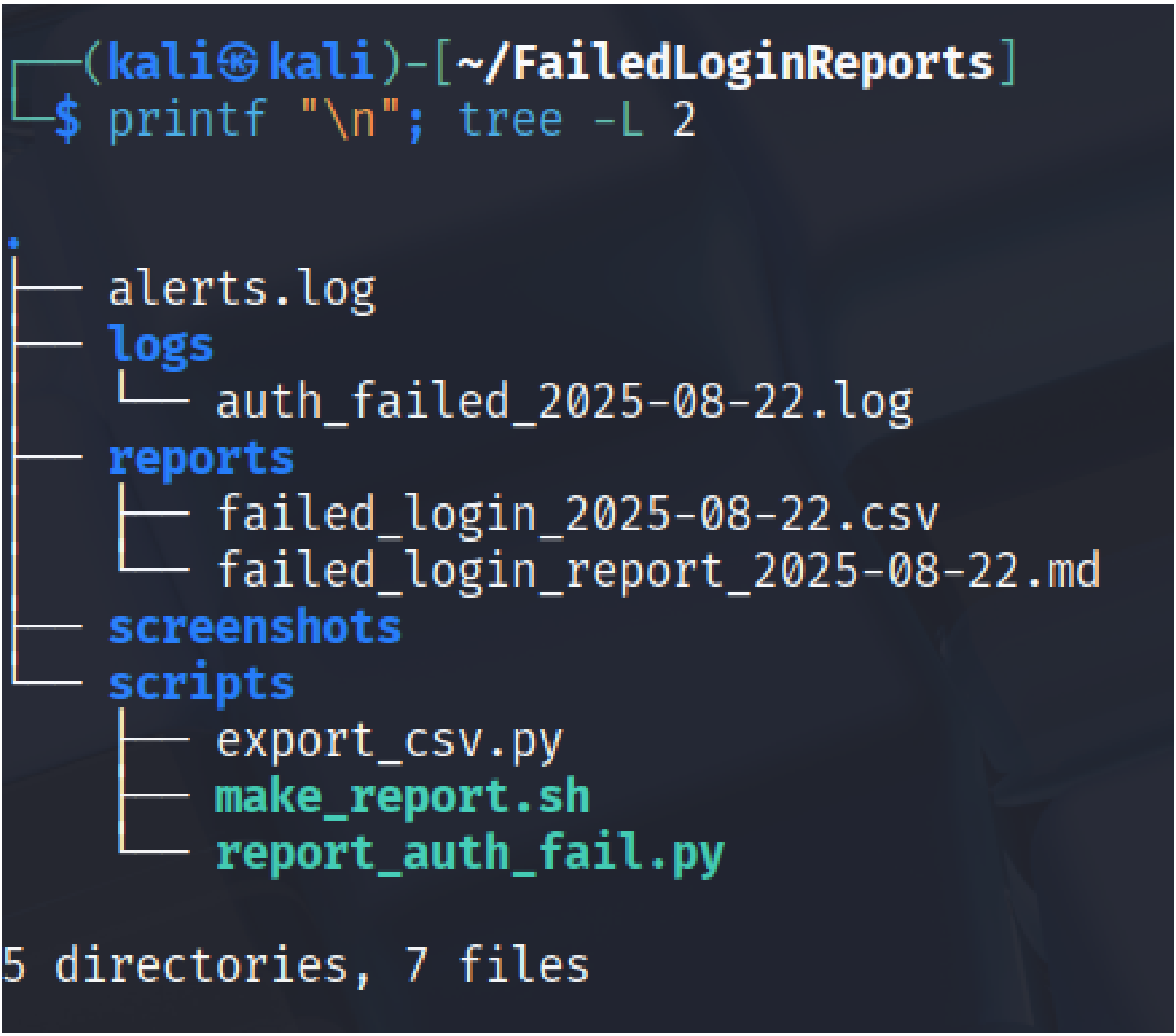
├── reports/

└── scripts/

  ├── report_auth_fail.py

  ├── export_csv.py

  └── make_report.sh



10) Reproduction (Commands)

Generate attempts (lab)

```
for i in {1..6}; do sshpass -p WrongPass ssh \
  -o PreferredAuthentications=password -o PubkeyAuthentication=no \
  -o StrictHostKeyChecking=no usernotexists@127.0.0.1 true || true; done
```

Collect raw

```
sudo journalctl -t sshd -n 2000 | \
  grep -E 'Failed password|Invalid user|authentication failure|PAM.*authentication failure' \
  > logs/auth_failed_$(date +%F).log
```

Build reports

python3 scripts/report_auth_fail.py

python3 scripts/export_csv.py

Threshold alert

th=20; cnt=$(grep -c 'Failed password' logs/auth_failed_$(date +%F).log)

["$cnt" -ge "$th"] && echo "[ALERT] $cnt failed logins today"

Conclusion

We built a lightweight, repeatable SSH failed-login reporting pipeline that parses journalctl, generates Markdown and CSV outputs, and triggers a simple threshold alert. The activity was performed in a controlled lab and no compromise occurred. After testing, SSH was reverted to keys-only and normal logging.

Recommendations (Priority)

1. **Enable Fail2ban** for sshd (e.g., maxretry=5, findtime=10m, bantime=1h).
2. **Restrict SSH via UFW** to trusted IPs/networks only.
3. **Schedule a daily run** (cron) and retain RAW + reports for 30–90 days.
4. **Harden accounts**: keys-only auth, lock unused users, lower MaxAuthTries.
5. **Iterate later**: add Windows Event ID 4625 coverage, auto HTML/PDF export, and GeoIP/IP reputation tagging.

TL;DR: A clean, auditable SSH failed-login reporting workflow (MD + CSV + alerts) with clear next steps for prevention and monitoring.