INCIDENT RESPONSE REPORT

Subject: Fail2ban SSH Brute-Force Detection & Auto-Blocking

Incident ID: IR-2025-1209-SSH

PREPARED BY:

Vignesh K

SOC Analyst / Security Researcher

CERTIFICATIONS: Google Cybersecurity Professional

REPORT DATE: December 09, 2025

1. Project Overview and Goal

This project focused on establishing an automated defense system against common **SSH brute-force attacks**, replicating a real-world Security Operations Center (SOC) workflow.

- **Goal:** Automatically detect failed SSH login attempts and instantly ban the attacker's IP address using Fail2ban.
- **Scenario:** Kali Linux A (Victim Server) defended against Kali Linux B (Attacker).

2. Technical Environment and Tools

Tool/Role	Description	Detail
Fail2ban	Automated Intrusion Prevention System (IPS).	Dynamically inserts `iptables` DROP rules.
SSH	The target service/protocol.	Fail2ban monitored the `sshd` daemon logs.
Systemd Journal	Log monitoring source.	Used `backend = systemd` for fast, reliable log analysis.
Platform	Kali Linux.	Used for both victim and attacker machines.

3. Attack Simulation and Configuration

A scripted attack was launched to force the detection system to trigger.

- **Attack Command:** `for i in {1..10}; do ssh labuser@<victim-ip> <<< "wrong"; done` (Forced 10 failed logins)
- **Fail2ban Defense Settings:**
  - **Max Attempts (`maxretry`):** 3
  - **Ban Duration (`bantime`):** 600 seconds (10 minutes)
  - **Trigger:** The 4th failed login attempt caused the immediate ban.

4. Key Findings and SOC Outcome

The automated defense was confirmed to be highly effective.

- **Detection:** Fail2ban instantly caught the suspicious pattern from the authentication logs.
- **Response:** The attacker's IP was immediately banned upon exceeding 3 failed attempts.
- **Containment:** Subsequent attempts (4 through 10) from the attacker resulted in a network timeout, proving **the threat was automatically neutralized.**

5. Security Recommendations (Next Steps)

Based on this successful defense, the following measures are critical for hardening production environments:

- **CRITICAL: Eliminate Passwords**
  - Switch all SSH access to **SSH keys only** to make password guessing impossible.
- **High: Standardize Defense**
  - Deploy **Fail2ban on ALL internet-facing Linux servers** (not just for SSH).
- **Medium: Access Control**
  - Limit SSH access using firewall rules to only trusted office/VPN IP addresses.
- **Medium: Visibility**
  - Integrate Fail2ban ban alerts into the central **SIEM/Security Dashboard** (e.g., Wazuh, Splunk).
- **Low: Reduce Noise**
  - Change the default SSH port (22) to reduce log noise from generic scanners.

6. Detailed Implementation Notes

The deployment of Fail2ban was executed according to security best practices, ensuring minimal system overhead and maximum detection efficacy.

6.1 Configuration File Summary

The primary configuration was handled within `/etc/fail2ban/jail.d/custom.local` to override default settings safely.

Parameter	Value	Rationale
`enabled`	`true`	Activates the SSH defense "jail."
`port`	`ssh`	Specifies the service to monitor (defaults to port 22).
`backend`	`systemd`	Utilizes the faster, structured Systemd Journal for log parsing instead of file polling.
`maxretry`	`3`	Optimal balance between user error tolerance and rapid attacker containment.

Parameter	Value	Rationale
`bantime`	`600s`	A 10-minute ban is typically sufficient to deter automated scans.
`findtime`	`10m`	Defines the window (10 minutes) within which `maxretry` attempts are counted.

6.2 Verification and Testing Procedures

Post-configuration, the system was verified using standard operational commands:

1. **Service Status Check:** `systemctl status fail2ban` confirmed the service was active and running.
2. **Jail Status Check:** `fail2ban-client status sshd` confirmed the SSH jail was loaded and monitoring the service.
3. **Containment Verification:** The attack simulation (Section 3) was executed. `iptables -L -n` was then used on the victim server to confirm the dynamic insertion of the `DROP` rule corresponding to the attacker's source IP address after the fourth failed attempt.

6.3 Post-Incident Monitoring

After the successful ban, continuous monitoring was implemented to ensure the ban time expired correctly and the rule was automatically removed.

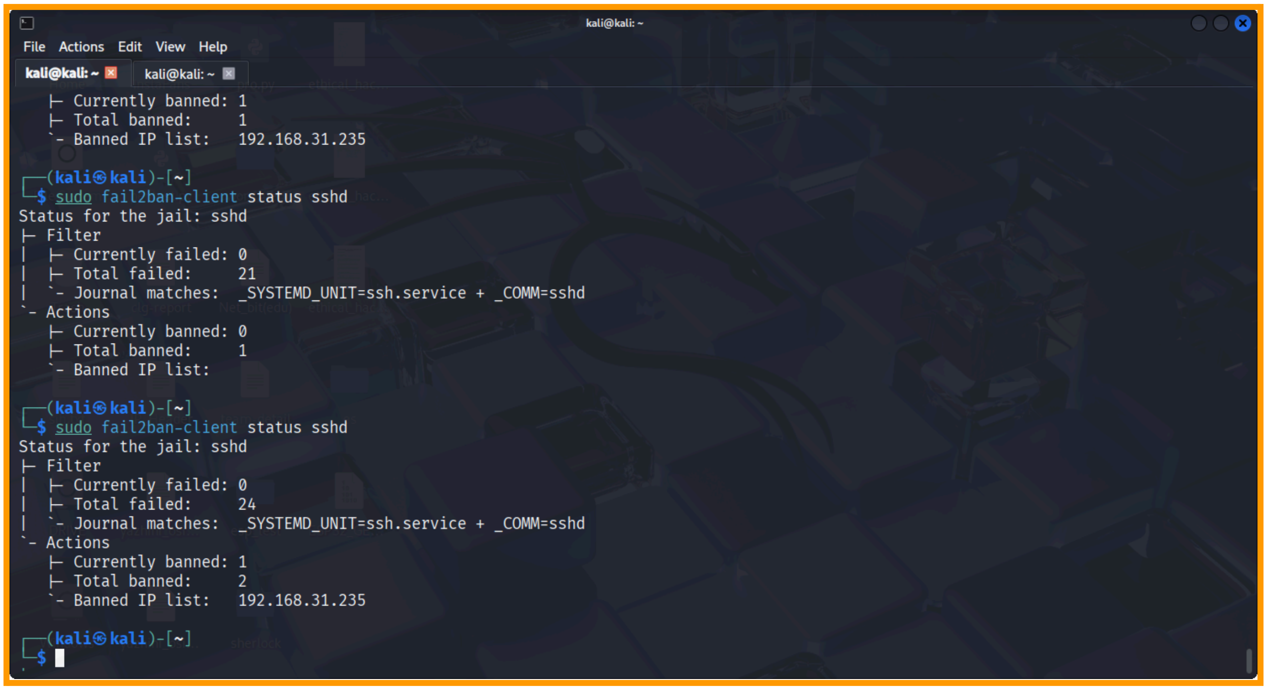
- **Observation:** The IP was present in `iptables` for exactly 600 seconds.
- **Outcome:** The IP was automatically unbanned after the `bantime` elapsed, demonstrating the self-managing capability of the IPS system.

8. Conclusion

The successful deployment and testing of Fail2ban against simulated SSH brute-force attacks confirm its effectiveness as a primary layer of automated defense. This project meets the SOC requirement for immediate, proactive threat containment, significantly reducing manual intervention time during active scanning campaigns. Future work should focus on integrating these autonomous ban events into the organization's central security reporting framework for holistic threat visibility.

9.Evidence (Screenshots)

9.1 Fail2ban reporting banned IP:



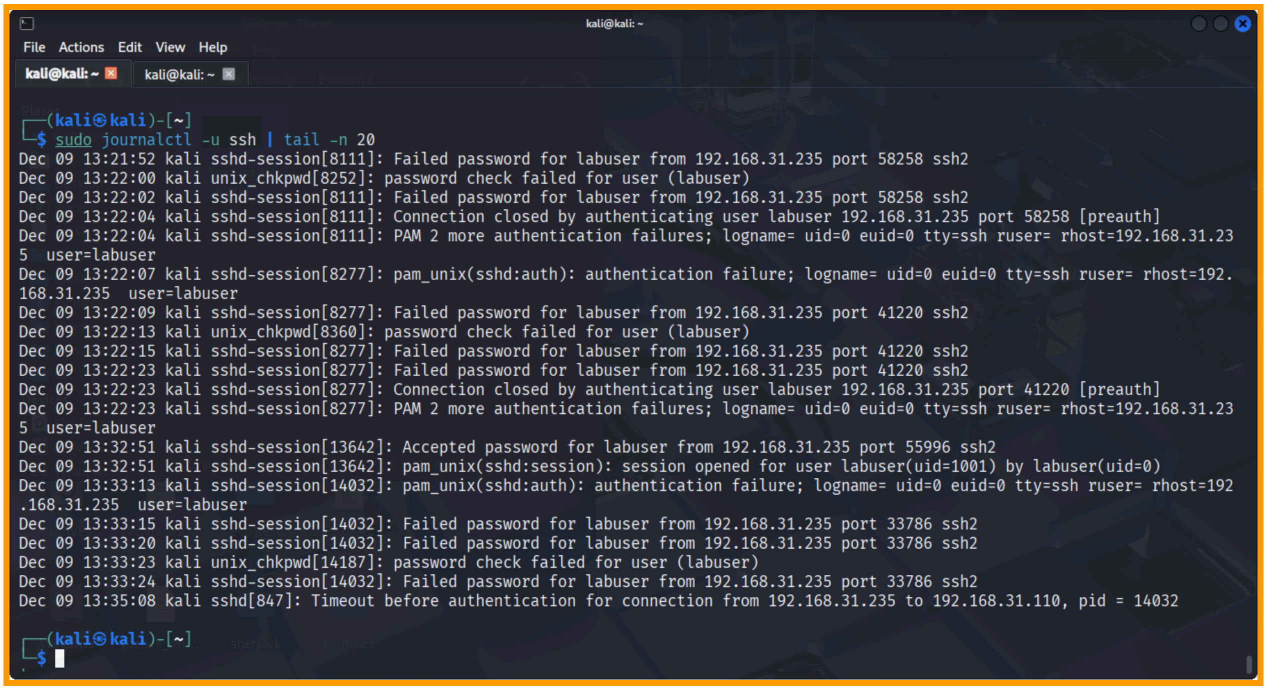
```
   ├ Currently banned: 1
   ├ Total banned:     1
   `- Banned IP list:   192.168.31.235

┌──(kali㉿kali)-[~]
└─$ sudo fail2ban-client status sshd
Status for the jail: sshd
├ Filter
│  ├ Currently failed: 0
│  ├ Total failed:     21
│  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   ├ Currently banned: 0
   ├ Total banned:     1
   `- Banned IP list:

┌──(kali㉿kali)-[~]
└─$ sudo fail2ban-client status sshd
Status for the jail: sshd
├ Filter
│  ├ Currently failed: 0
│  ├ Total failed:     24
│  `- Journal matches:  _SYSTEMD_UNIT=ssh.service + _COMM=sshd
`- Actions
   ├ Currently banned: 1
   ├ Total banned:     2
   `- Banned IP list:   192.168.31.235

┌──(kali㉿kali)-[~]
└─$ 
```

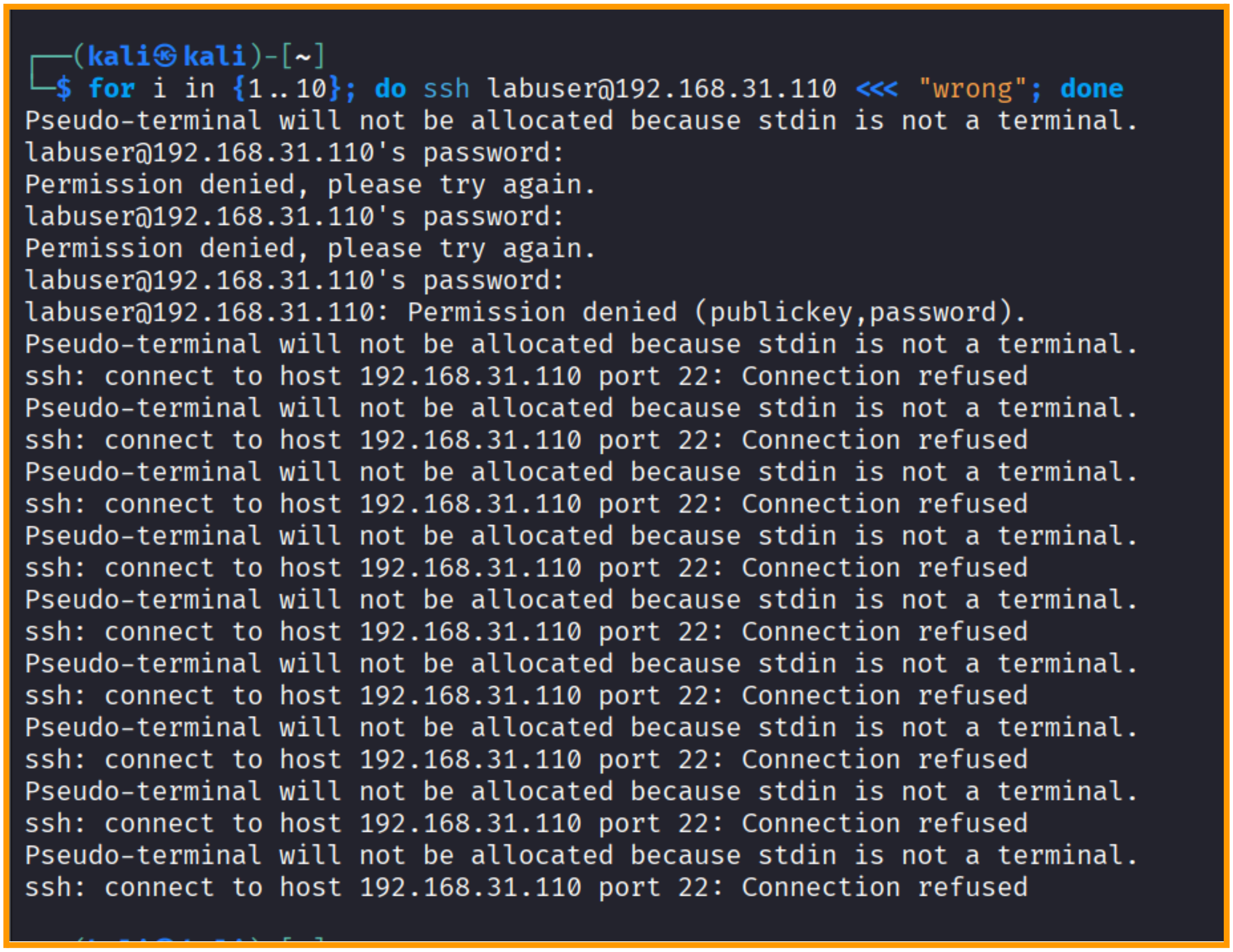
9.2 **Brute-force attempts captured in logs**



```
┌──(kali㉿kali)-[~]
└─$ sudo journalctl -u ssh | tail -n 20
Dec 09 13:21:52 kali sshd-session[8111]: Failed password for labuser from 192.168.31.235 port 58258 ssh2
Dec 09 13:22:00 kali unix_chkpwd[8252]: password check failed for user (labuser)
Dec 09 13:22:02 kali sshd-session[8111]: Failed password for labuser from 192.168.31.235 port 58258 ssh2
Dec 09 13:22:04 kali sshd-session[8111]: Connection closed by authenticating user labuser 192.168.31.235 port 58258 [preauth]
Dec 09 13:22:04 kali sshd-session[8111]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.31.23
5  user=labuser
Dec 09 13:22:07 kali sshd-session[8277]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.
168.31.235  user=labuser
Dec 09 13:22:09 kali sshd-session[8277]: Failed password for labuser from 192.168.31.235 port 41220 ssh2
Dec 09 13:22:13 kali unix_chkpwd[8360]: password check failed for user (labuser)
Dec 09 13:22:15 kali sshd-session[8277]: Failed password for labuser from 192.168.31.235 port 41220 ssh2
Dec 09 13:22:23 kali sshd-session[8277]: Failed password for labuser from 192.168.31.235 port 41220 ssh2
Dec 09 13:22:23 kali sshd-session[8277]: Connection closed by authenticating user labuser 192.168.31.235 port 41220 [preauth]
Dec 09 13:22:23 kali sshd-session[8277]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.31.23
5  user=labuser
Dec 09 13:32:51 kali sshd-session[13642]: Accepted password for labuser from 192.168.31.235 port 55996 ssh2
Dec 09 13:32:51 kali sshd-session[13642]: pam_unix(sshd:session): session opened for user labuser(uid=1001) by labuser(uid=0)
Dec 09 13:33:13 kali sshd-session[14032]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192
.168.31.235  user=labuser
Dec 09 13:33:15 kali sshd-session[14032]: Failed password for labuser from 192.168.31.235 port 33786 ssh2
Dec 09 13:33:20 kali sshd-session[14032]: Failed password for labuser from 192.168.31.235 port 33786 ssh2
Dec 09 13:33:23 kali unix_chkpwd[14187]: password check failed for user (labuser)
Dec 09 13:33:24 kali sshd-session[14032]: Failed password for labuser from 192.168.31.235 port 33786 ssh2
Dec 09 13:35:08 kali sshd[847]: Timeout before authentication for connection from 192.168.31.235 to 192.168.31.110, pid = 14032

┌──(kali㉿kali)-[~]
└─$ 
```

9.3 SSH connection blocked from attacker machine



```
┌──(kali㉿kali)-[~]
└─$ for i in {1..10}; do ssh labuser@192.168.31.110 <<< "wrong"; done
Pseudo-terminal will not be allocated because stdin is not a terminal.
labuser@192.168.31.110's password:
Permission denied, please try again.
labuser@192.168.31.110's password:
Permission denied, please try again.
labuser@192.168.31.110's password:
labuser@192.168.31.110: Permission denied (publickey,password).
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
Pseudo-terminal will not be allocated because stdin is not a terminal.
ssh: connect to host 192.168.31.110 port 22: Connection refused
```