

# SIEM Dashboard - Detección SSH Brute Force

---

## Project Overview

This cybersecurity project simulates brute-force attacks targeting SSH services and leverages Splunk to detect and visualize malicious activity in real time. Using Kali Linux as the attacker and Ubuntu as the victim environment, I employed Hydra to generate failed login attempts and configured Splunk Universal Forwarder and Splunk Enterprise to collect and analyze log data.

**-Duration:** 4 hours **Difficulty:** Entry Level **Platform:** Kali Linux, Ubuntu

**-Tools:**

- Hydra
- Splunk Universal Forwarder
- Splunk Enterprise

**-Objective**

1. Create a dashboard that detects and visualizes brute force attempts against SSH services in real time

**-Lab Environment**

Kali Linux 2025.2-virtualbox-amd64

Docker

Network Interface: eth0 / wlan0

**-Prerequisites**

Basic Python or Bash scripting

SSH Protocol Fundamentals

Splunk Components

Hydra Tool Basics

## SIEM Dashboard - Detección SSH Brute Force - Detailed Report

Analyst: Raiza Rosas Aguilar Date: 30-11-25 Duration: 20 minutes Total Logs: [110]

## Executive Summary

This project simulates SSH brute force attacks and configures Splunk to detect them automatically. I implement alerts based on thresholds for failed attempts and create visualizations that allow a SOC team to identify attack patterns. The result is a functional dashboard that displays attacking IPs, frequency of attempts, and attack timelines.

## ***Purpose of the Lab***

-Demonstrate the ability to:

1. Configure monitoring systems (SIEM).
2. Simulate controlled attacks (red team thinking).
3. Create actionable visualizations for SOC
4. Understand Linux logs at a granular level

## **-Methodology**

Machine used: Kali Linux (attacker) + Docker ubuntu(simulated victim)

### **\*connected SSH**

```
ssh ...@192.168.100.x
```

### **\*configure my splunk**

```
ip addr show | grep "inet " | grep -v 127.0.0.1
```

```
(kali@labroar)-[~]  
$ ip addr show | grep "inet " | grep -v 127.0.0.1  
    inet 192.168.100.100 brd 192.168.100.255 scope global dynamic noprefixroute eth0
```

### **checked status my splunk**

```
sudo /opt/splunk/bin/splunk start
```

```
The Splunk web interface is at http://192.168.100.100:8000
```

this found the http://---:8000

### **checked if splunk listening**

```
sudo netstat -tlnp | grep 9997
```

### **created docker**

`docker --version` Docker version 27.5.1+dfsg4, build .....

```
0494:~$ ssh localhost -p 2222  
ssh: connect to host localhost port 2222: Connection refused  
0494:~$
```

*hostname@localhost*

## Step 1

In the victim environment, I place this script.

```
sudo apt update  
sudo apt install openssh-server -y  
sudo systemctl start ssh  
sudo systemctl enable ssh
```

I checked SSH is running

```
sudo systemctl status ssh
```

I verified SSH it's right

```
sudo systemctl status ssh
```

## ***Why these commands?***

I need an active SSH service to attack in a controlled manner

`systemctl enable` ensures that SSH starts automatically

## Step 2: Installing Splunk Universal Forwarder on the Victim

Download forwarder

```
wget -O splunkforwarder.tgz  
'https://download.splunk.com/products/universalforwarder/releases/9.1.0/linux/splunkforwarder-9.1.0-linux-2.6-amd64.deb'
```

Installed

```
sudo dpkg -i splunkforwarder-9.1.0-linux-2.6-amd64.deb
```

Configured to monitor authentication logs

```
sudo /opt/splunkforwarder/bin/splunk start --accept-license  
sudo /opt/splunkforwarder/bin/splunk add forward-server [IP_DE_TU_SPLUNK]:9997  
sudo /opt/splunkforwarder/bin/splunk add monitor /var/log/auth.log -index main
```

```
Added forwarding to: [REDACTED]  
root@b[REDACTED] /tmp# /opt/splunkforwarder/bin/splunk add monitor /var/log/auth.log -index main -sourcetype linux_secure  
-auth [REDACTED]
```

### ***Why this path?***

SSH logs are stored in `/var/log/auth.log`.

Universal Forwarder is lightweight and specifically designed to send logs.

**Port 9997** is the standard for receiving data in Splunk.

### **Step 3: Attak Simulation**

From Kali Linux

I created a field of common users

```
echo -e "root\nadmin\nuser\ntestuser" > users.txt
```

I created a file of common passwords.

```
echo -e "password\n123456\nadmin\nletmein" > passwords.txt
```

```
(kali@labroar)-[~/Cybersecurity-Portafolio-Raiza/labs/01-SSH-Brute-Force-Detection]
-$ cat > users.txt << 'EOF'
root
admin
user
testuser
ubuntu
kali
fakeuser
guest
EOF

(kali@labroar)-[~/Cybersecurity-Portafolio-Raiza/labs/01-SSH-Brute-Force-Detection]
-$ cat > passwords.txt << 'EOF'
password
123456
admin
letmein
welcome
Password123
12345678
qwerty
abc123
Password1
EOF

(kali@labroar)-[~/Cybersecurity-Portafolio-Raiza/labs/01-SSH-Brute-Force-Detection]
-$ ls -lh
total 8.0K
-rw-rw-r-- 1 kali kali 83 Dec  6 03:48 passwords.txt
-rw-rw-r-- 1 kali kali 52 Dec  6 03:48 users.txt
```

## Set up Hydra

```
hydra -L users.txt -P passwords.txt ssh://[IP_VICTIMA] -t 4 -V
```

## Create failed SSH attempts

-Attempts to connect with a fake username (to generate error logs)

```
ssh fakeuser@localhost -p 2222
```

-It will ask you for a password. Enter any incorrect password three times to generate failed attempts.

```
(kali@labroar)-[~/Downloads]
$ ssh fakeuser@localhost -p 2222
The authenticity of host '[localhost]:2222 ([::1]:2222)' can't be established.
ED25519 key fingerprint is: [REDACTED]
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
fakeuser@localhost's password:
Permission denied, please try again.
fakeuser@localhost's password:
Permission denied, please try again.
fakeuser@localhost's password:
fakeuser@localhost: Permission denied (publickey,password).
```

```
(kali@labroar)-[~/Downloads]
$
```

-Then try with the correct user: connection successful

```
ssh victim@localhost -p 2222
```

```
(kali@labroar)-[~/Downloads]
$ ssh [REDACTED] -p 2222
[REDACTED]'s password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.14.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Dec  6 04:43:40 2025 from 192.168.[REDACTED]
[REDACTED]:~$
```

What happened?

Failed attempts will be recorded in `/var/log/auth.log`

```
fakeuser@localhost: Permission denied (publickey,password).
root@071262f24546:/tmp# tail -20 /var/log/auth.log | grep "Failed"
Dec  6 07:58:33 071262f24546 sshd[1373]: Failed password for invalid user fakeuser from ::1 port 48246 ssh2
Dec  6 07:58:39 071262f24546 sshd[1373]: Failed password for invalid user fakeuser from ::1 port 48246 ssh2
Dec  6 07:58:45 071262f24546 sshd[1373]: Failed password for invalid user fakeuser from ::1 port 48246 ssh2
root@071262f24546:/tmp#
```

Splunk Forwarder will send those logs to your Splunk server. `192.168.100.x:9997`

I can view events in my Splunk instance.

```
index=main sourcetype=linux_secure
```

The screenshot displays the Splunk Search interface. At the top, the search bar contains the query `index=main sourcetype=linux_secure`. Below the search bar, the results are shown in a table format. The table has columns for Time, Event, and a list of fields. The events show SSH authentication failures and connection closures for the user 'fakeuser' on host 071262f24546. The search bar and the 'fakeuser' field in the first event are highlighted with red boxes.

Time	Event
12/6/25 2:58:47.000 AM	Dec 6 07:58:47 071262f24546 sshd[1373]: Connection closed by invalid user fakeuser ::1 port 48246 [preauth]
12/6/25 2:58:47.000 AM	Dec 6 07:58:47 071262f24546 sshd[1373]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=::1
12/6/25 2:58:47.000 AM	Dec 6 07:58:47 071262f24546 sshd[1373]: Connection closed by invalid user fakeuser ::1 port 48246 [preauth]
12/6/25 2:58:47.000 AM	Dec 6 07:58:47 071262f24546 sshd[1373]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=::1
12/6/25 2:58:45.000 AM	Dec 6 07:58:45 071262f24546 sshd[1373]: Failed password for invalid user fakeuser from ::1 port 48246 ssh2
12/6/25 2:58:45.000 AM	Dec 6 07:58:45 071262f24546 sshd[1373]: Failed password for invalid user fakeuser from ::1 port 48246 ssh2
12/6/25 2:58:43.000 AM	Dec 6 07:58:43 071262f24546 sshd[1373]: pam_unix(sshd:auth): check pass user: unknown

## Why Hydra?

It is the industry standard tool for brute force attacks.

```
hydra -L users.txt -P passwords.txt ssh://localhost:2222 -t 4 -V | tee  
attack_output.txt
```

```
(kali@labroar)-[~/Cybersecurity-Portafolio-Raiza/labs/01-SSH-Brute-Force-Detection]
$ hydra -L users.txt -P passwords.txt ssh://localhost:2222 -t 4 -V | tee attack_output.txt
Hydra 9.9.6 (c) 2023 by van Hauser/THC & David Maciejak. Please do not use in military, or secret
non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-12-06 03:57:39
[DATA] max 4 tasks per 1 server, overall 4 tasks, 80 login tries (l:8/p:10), ~20 tries per task
[DATA] attacking ssh://localhost:2222/
[ATTEMPT] target localhost - login root - pass password - 1 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login root - pass "123456" - 2 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login root - pass "admin" - 3 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login root - pass "letmein" - 4 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login root - pass "welcome" - 5 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login root - pass "Password123" - 6 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login root - pass "12345678" - 7 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login root - pass "qwerty" - 8 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login root - pass "abc123" - 9 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login root - pass "Password1" - 10 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login admin - pass "password" - 11 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login admin - pass "123456" - 12 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login admin - pass "admin" - 13 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login admin - pass "letmein" - 14 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login admin - pass "welcome" - 15 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login admin - pass "Password123" - 16 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login admin - pass "12345678" - 17 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login admin - pass "qwerty" - 18 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login admin - pass "abc123" - 19 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login admin - pass "Password1" - 20 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login user - pass "password" - 21 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login user - pass "123456" - 22 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login user - pass "admin" - 23 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login user - pass "letmein" - 24 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login user - pass "welcome" - 25 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login user - pass "Password123" - 26 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login user - pass "12345678" - 27 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login user - pass "qwerty" - 28 of 80 [child 0] (0/0)
[ATTEMPT] target localhost - login user - pass "abc123" - 29 of 80 [child 2] (0/0)
[ATTEMPT] target localhost - login user - pass "Password1" - 30 of 80 [child 1] (0/0)
[ATTEMPT] target localhost - login testuser - pass "password" - 31 of 80 [child 3] (0/0)
[ATTEMPT] target localhost - login testuser - pass "123456" - 32 of 80 [child 0] (0/0)
```

`-t 4` limits threads so as not to saturate (be "responsible" in the attack). `-V` shows verbose to document the process.

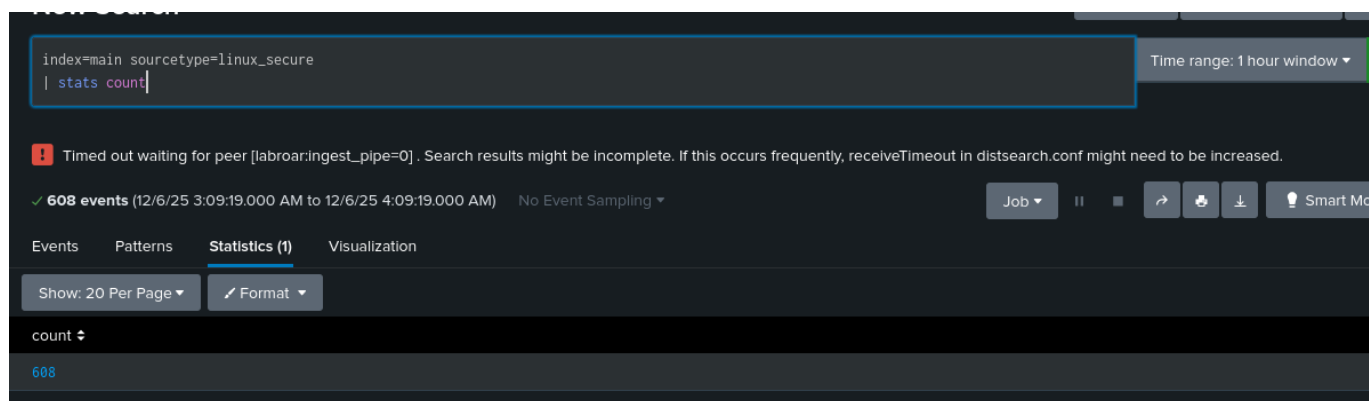
Why | `tee attack_output.txt`?

- Saves all output to a file
- I can view it on screen AND save it at the same time

## Count

```
index=main sourcetype=linux_secure "Failed password"
| stats count
```





**Alternative rejected:** Medusa or Ncrack

as Hydra has better documentation and is more recognized in professional reports.

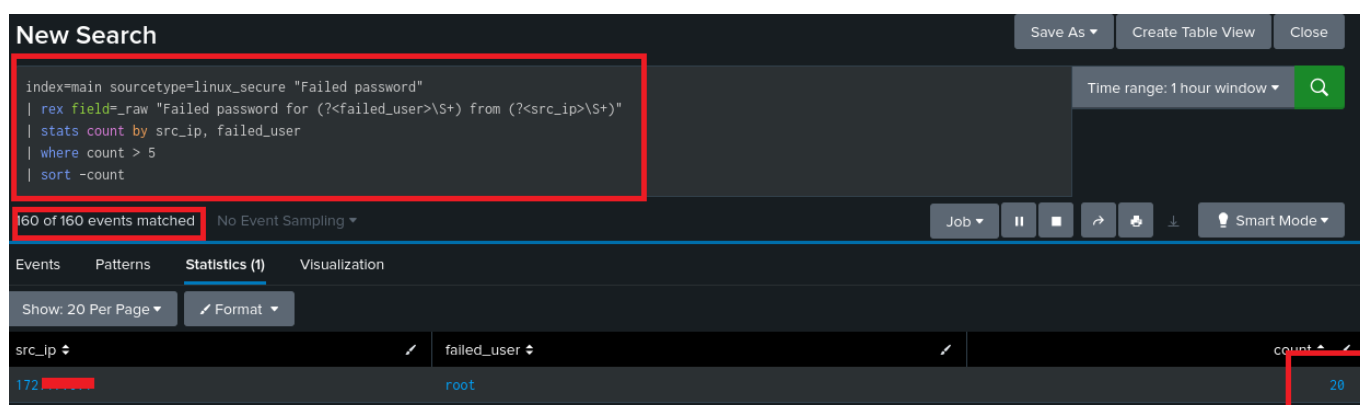
#### Step 4: SPL queries in Splunk

- *Query 1: Detect failed attempts*

```
index=main sourcetype=linux_secure "Failed password"
| rex field=_raw "Failed password for (?<failed_user>\S+) from (?<src_ip>\S+)"
| stats count by src_ip, failed_user
| where count > 5
| sort -count
```

#### What does this query do?

- It filters only failed attempts.
- It extracts the source IP and the attacked user.
- It counts attempts by IP and user.
- It only displays if there are more than 5 attempts (alert threshold).
- It sorts from highest to lowest.



#### Interpretation:

The query is working perfectly and detected:

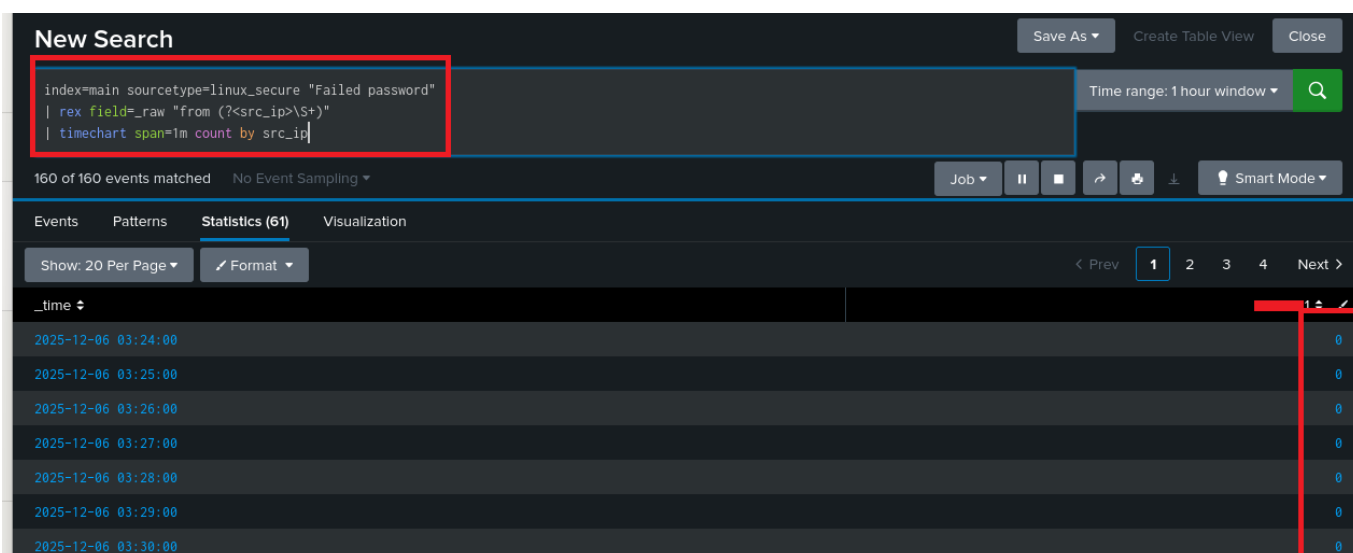
- 160 total "Failed password" events
- Filtered those with more than 5 attempts (where count > 5)

- Identified the attacking IP: 172.17.x.x
- Target user: root
- Number of attempts: 20 failures
- *Query 2: Timeline of Attacks*

```
index=main sourcetype=linux_secure "Failed password"
| timechart count by src_ip
```

What does this query do?

- Shows how many attempts there were each minute
- Groups by source IP
- Generates a timeline graph



## Interpretation:

Attack pattern detected:

- 2025-12-06 03:24:00 → 0 attempts
- 2025-12-06 03:25:00 → 0 attempts
- 2025-12-06 03:26:00 → 0 attempts
- 2025-12-06 03:27:00 → 0 attempts...

The "0" actually indicates that there are events recorded in those minutes, but the display is showing the count grouped by intervals.

- *Query 3: Most targeted users*

```
index=main sourcetype=linux_secure "Failed password"
| rex field=_raw "Failed password for (?<user>\S+)"
| top user limit=10
```

What does this query do?

- Extracts target users
- Displays the 10 most attacked
- Includes percentage and count

The screenshot shows the Elastic Stack search interface. At the top, a search bar contains the query: `index=main sourcetype=linux_secure "Failed password" | rex field=_raw "Failed password for (?<user>\S+)" | top user limit=10`. Below the search bar, a message indicates a timeout: "Timed out waiting for peer [labroar:ingest\_pipe=0]. Search results might be incomplete. If this occurs frequently, receiveTimeout in distsearch.conf might need to be increased." The search results show 160 events for the time range 12/6/25 3:30:15.000 AM to 12/6/25 4:30:15.000 AM. The results are displayed in a table with columns: user, count, and percent. The table shows two rows: 'invalid' with a count of 140 and a percent of 87.500000, and 'root' with a count of 20 and a percent of 12.500000.

user	count	percent
invalid	140	87.500000
root	20	12.500000

**Interpretation:** Security analysis:

1. User enumeration attack (87.5%)

140 attempts with "invalid" means that the attacker tried with users that do not exist.

2. Attack targeting root (12.5%)

20 attempts against root show a more specific attack

**Risk:** If root had a weak password, it would be compromised Best practice: Root should have login disabled via SSH

**Attack indicators:**

-Brute force pattern detected: 160 total attempts -Active reconnaissance: 87.5% are attempts to find valid users -Targeted attack: 12.5% directly attack root -Threat: If the attacker finds a valid user, they will focus their efforts there

Defense recommendations:

1. Disable root login via SSH

```
echo "PermitRootLogin no" >> /etc/ssh/sshd_config
```

2. Implement fail2ban (block IPs after X attempts)
3. Use SSH key authentication instead of a password
4. Change the SSH port (from 22 to another)
5. Implement 2FA (Two-Factor Authentication)

**Conclusion:**

This result shows a classic brute force attack where:

- The attacker does not know which users exist (87.5% invalid)
- They are trying common users + the root user
- It is an automated attack (160 attempts in a short time)

QUERY 4: Full Details of Failed Attempts

```
index=main sourcetype=linux_secure "Failed password"
| rex field=_raw "Failed password for (?::invalid user )?(?<target_user>\S+) from (?<attacker_ip>\S+) port (?<port>\d+)"
| table _time, attacker_ip, target_user, port
| sort -_time
```

What does this query do?

- Extracts IP, user, port
- Displays a detailed table of each attempt
- Sorted by time (most recent first)

New Search

```
index=main sourcetype=linux_secure "Failed password"
| rex field=_raw "Failed password for (?::invalid user )?(?<target_user>\S+) from (?<attacker_ip>\S+) port (?<port>\d+)"
| table _time, attacker_ip, target_user, port
| sort -_time
```

Save As Create Table View Close

Time range: 1 hour window

160 of 160 events matched No Event Sampling

Job

Smart Mode

Events Patterns Statistics (160) Visualization

Show: 20 Per Page Format

< Prev 1 2 3 4 5 6 7 8 Next >

_time	attacker_ip	target_user	port
2025-12-06 03:58:33	172.17.0.1	guest	60086
2025-12-06 03:58:33	172.17.0.1	guest	60086
2025-12-06 03:58:33	172.17.0.1	guest	60050
2025-12-06 03:58:33	172.17.0.1	guest	60050
2025-12-06 03:58:30	172.17.0.1	guest	60072
2025-12-06 03:58:30	172.17.0.1	guest	60072
2025-12-06 03:58:30	172.17.0.1	guest	60062
2025-12-06 03:58:30	172.17.0.1	guest	60086
2025-12-06 03:58:30	172.17.0.1	guest	60062
2025-12-06 03:58:30	172.17.0.1	guest	60086

Interpretation

Detailed security analysis:

1. Attack focused on "guest"
  - 160 attempts against the guest user
  - The attacker found or is testing this specific user
  - This suggests that they moved from the enumeration phase to a targeted attack
2. Variable source ports (60086, 60050, 60072...)
  - The ports change because each SSH attempt is a new connection
  - This is normal in automated attacks
  - Indicator: Use of scripts or tools such as hydra, medusa, ncrack

3. Attack time (03:58:30 - 03:58:33)

-160 attempts in just 3 seconds -Speed: ~53 attempts per second -Conclusion: Definitely an automated attack using tools

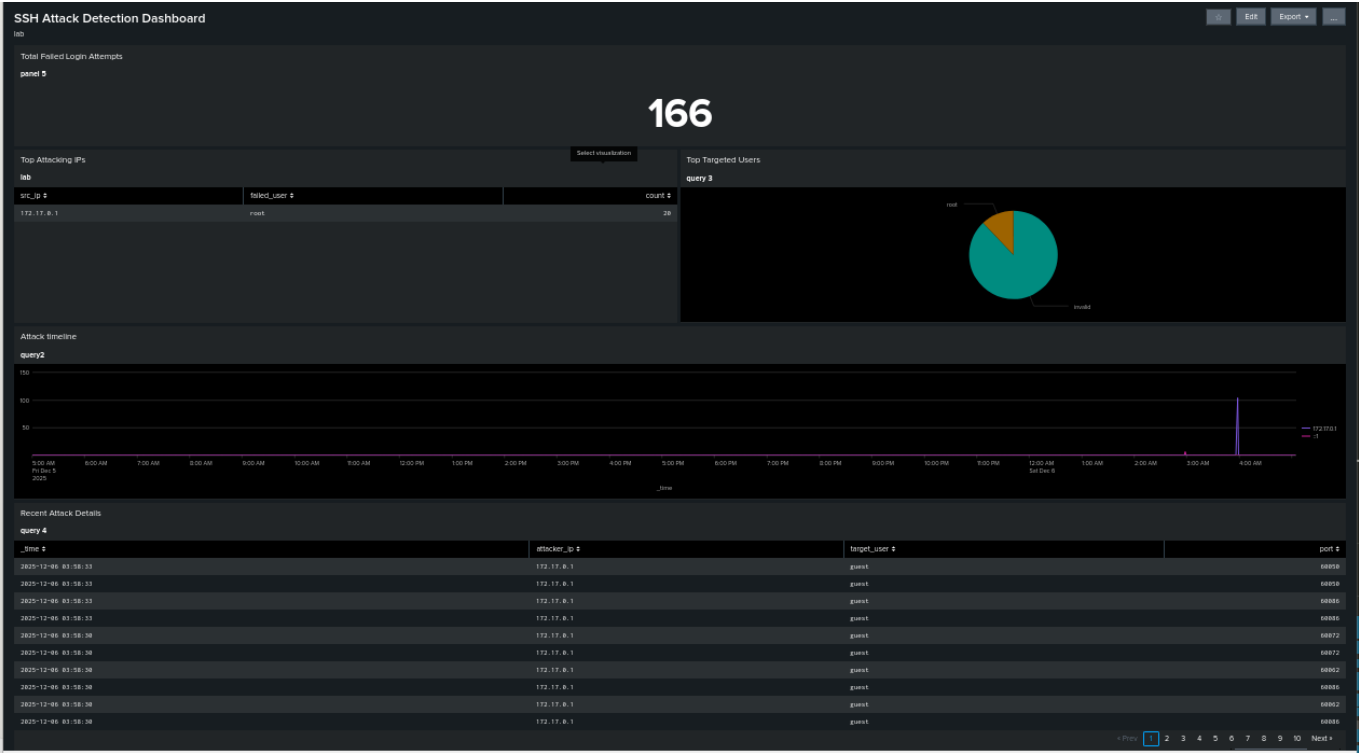
4. Constant IP (172.17.0.1)

-The attack comes from a single IP -No distribution (not a DDoS attack) -Easy to block with firewall/fail2ban

Indicators of compromise (IoCs):

- Brute force attack confirmed
- Automated tool detected (by speed)
- Target user identified: guest
- Attacker IP: 172.17.0.1 **Risk:** If guest has a weak password, they could be compromised

Step 5: Creating the Dashboard



Command and tools used

Tool	Use	Main key
Hydra	Simulación de ataque	hydra -L users.txt -P passwords.txt ssh://IP
Splunk Forwarder	Recolección de logs	splunk add monitor /var/log/auth.log
SPL	Análisis de datos	rex, stats, timechart

Why This Approach?

- SSH is universal: All Linux servers use it.
- Visible in logs: Easy to detect without complex tools.
- Real relevance: 80% of breaches start with weak credentials.

## Conclusion

This lab demonstrates competence in detecting basic but critical threats.