Phase Three:

Defense Mechanism:

Our vulnerability, which is SSH, was exploit using Brute-Force attack, which means if we need to stop SSH attacks, we need to fix and stop any attempt of trying usernames and passwords. A solution that came to our mind immediately is to limit the amount of tries before blocking the user, which in our case the attacker, from trying different combination, and giving them a time out.

First, we needed to check if there is a limit on the amount of guesses before getting blocked or timed-out, after trying combinations and reading online sources, we discovered that there was no such a thing set for metasploitable 3. This discovery was important, as it confirmed that the victim device will not stop the attacker, and since Brute-Force has a time complexity of $O(k^n)$, k being the number of characters and n the length of the password. The following is an example of time needed to crack a password for n from 1 till 6 for passwords that only use the English alphabet (26 characters):

Value of n	Number of combinations	Time needed to crack
1	$26^1 = 26$	0.026 seconds
2	$26^2 = 676$	0.676 seconds
3	$26^3 = 17{,}576$	17.576 seconds
4	$26^4 = 456{,}976$	7.6 minutes
5	$26^5 = 11{,}881{,}376$	3.3 hours
6	$26^6 = 308{,}915{,}776$	3.6 days

Note: the time her is calculated in worst-case scenario, and if the algorithm is trying 1000 combinations per second.

To solve the issue of Brute-Force, we are going to install a software in our victim device, which will monitor the log-in attempts and prevent any Brute-Force attacks by using time-outs to the user/attacker, and block it if possible. After doing some research, we decided to use Fail2Ban, which is a software used to monitor logs. The decision was based on the way this software works and how it can satisfy our goals the most.

Now, we will implement Fail2Ban into our victim device (metasploitable3), which is via using the following code in the device:

```
$ sudo apt update && sudo apt upgrade
$ sudo apt install fail2ban
$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
$ sudo nano /etc/fail2ban/jail.local

#inside the config file type the following section

[sshd]
enabled = true
maxretry = 3 # Ban after 3 failed attempts
bantime = 3600 # Ban duration (1 hour)
findtime = 600 # Time window for maxretry (10 minuets)
port = ssh # SSH port
filter = sshd
logpath = /var/log/auth.log
banaction = iptables # Use iptables for blocking


# ctrl + x -> y -> Enter to save and exit

#restart after changing the configurations:

$ sudo service fail2ban restart


#run the ssh attack multiple times with incorrect password/username

#check ban list in victim machine

$ sudo fail2ban-client status sshd

#After being banned, the attack will fail even with the correct username/password
```

Testing & Validation:

Figure 14: Successful attack (before defense)

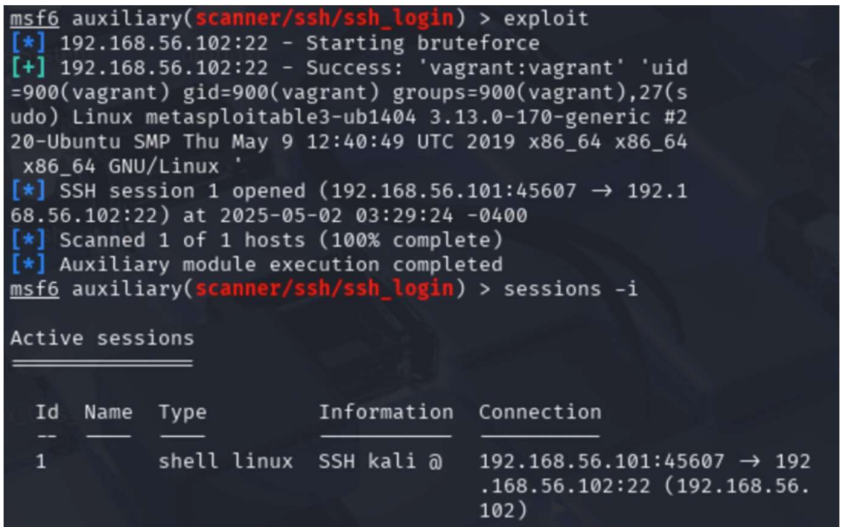

Figure 15: Fail2Ban configuration (Defense mechanism)

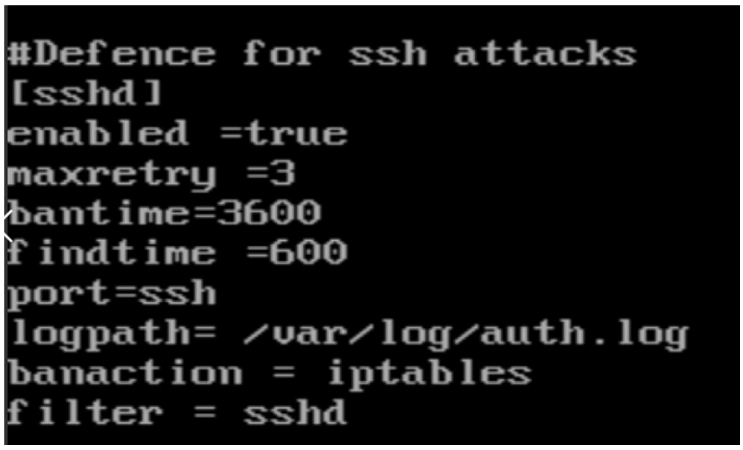

Figure 18: Triggering defense strategy with incorrect password/username

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set rhost 192.168.56.102
rhost ⇒ 192.168.56.102
msf6 auxiliary(scanner/ssh/ssh_login) > set username vagront
username ⇒ vagront
msf6 auxiliary(scanner/ssh/ssh_login) > set password vagront
password ⇒ vagront
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.56.102:22 - Starting bruteforce
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i

Active sessions
===============

No active sessions.
```

Figure 17: Attack Failure (after defense implemntation)

```
msf6 auxiliary(scanner/ssh/ssh_login) > set username vagrant
username ⇒ vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > set password vagrant
password ⇒ vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.56.102:22 - Starting bruteforce
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions i

Active sessions
===============

No active sessions.
```

Figure 16: Banned list with attacker IP

```
vagrant@metasploitable3-ub1404:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- filter
|  |- File list:        /var/log/auth.log
|  |- Currently failed: 1
|  `- Total failed:     4
`- action
   |- Currently banned: 1
   |  `- IP list:       192.168.56.101
   `- Total banned:     1
```

Before-and-After Comparison:

After implementing the defense mechanism, in this part, we will check the log in both victim and attacker machines, to see the difference and to confirm the defense strategy.

Figure 19: Attacker view from splunk

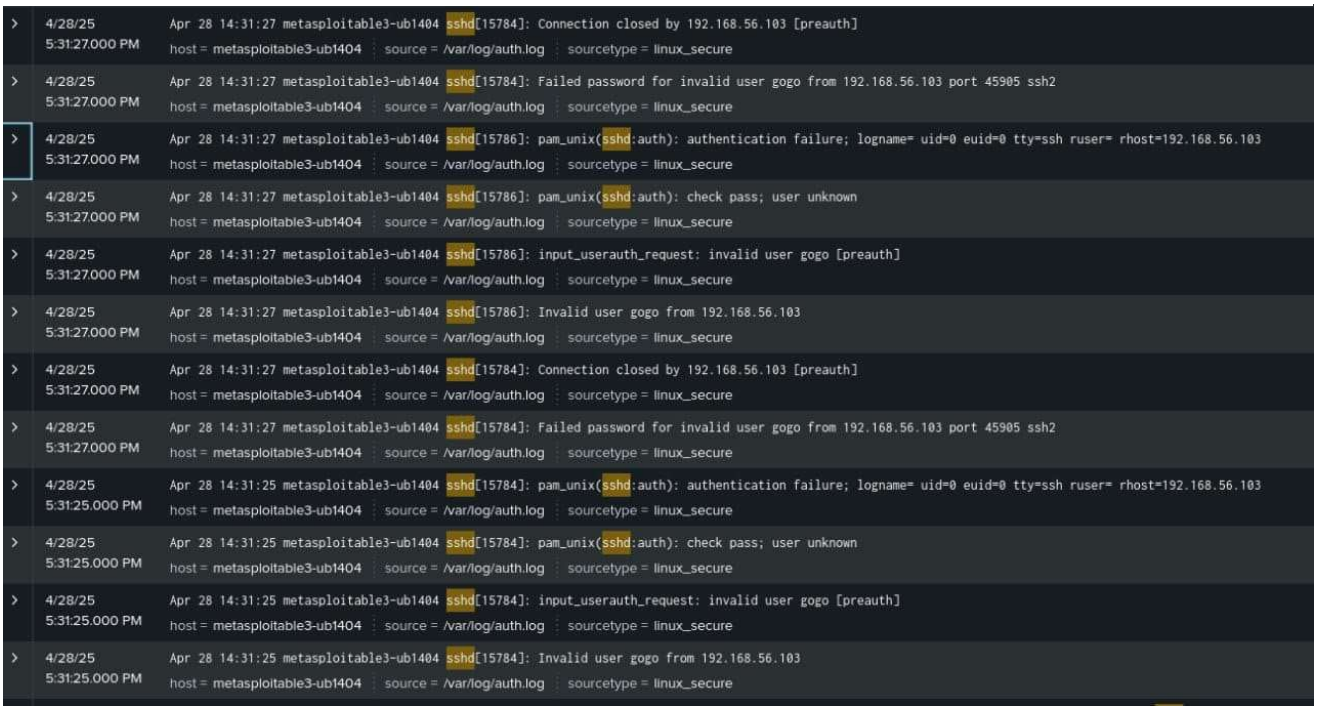

Figure 20: Victim view, shows banned user (who is the attacker)

After trying and failing multiple times, the software, Fail2Ban, blocked the user, which can be shown in Figure 20. This results confirm that our defense mechanism is working fine, and it is blocking hackers from gaining access to SSH via Brute Force attacks.

Conclusion:

To conclude, we were able to develop and enhance our skills in cyber and data security via implementing attacks on vulnerable machine, which was Metasploitable3. Phase 1 was about making both attacker and victim virtual machines ready, and then finding vulnerabilities in victim machine, which we found SSH_Login and FTP to be weak, we chose to continue with SSH. Phase 2 was about visualizing and analyzing the attack with SIEM dashboard, which was done by Splunk as a dashboard. In addition, we were able to take the attack log and visualize it for better understanding. Phase 3 was about implementing a defense strategy that can stop future attacks within our chosen weakness. Since our attack was built fully on brute force, we decided to limit the failed attempts and ban suspicious accounts from trying. Our team worked in parallel, meaning we all started the same phase at the same time, until all of us finish it. The point of this method is to ensure all of us understand each point, making it more beneficial. The project was enjoyable to do, with minimum number of issues faced, such as splunkForwarder not sending the data to the server. Overall, it was important to get hands-on knowledge with this course, which made it easier to connect the course with real life applications.