Team4

ICS344 Project

Layal albuhussain 202160450
Narjes Alsaad 202162050
Maha albasri 202180230

Phase 3

Introduction

In this phase, we implemented and tested two independent security defense mechanisms on our vulnerable SSH server:

1- Fail2Ban:

A tool that monitors log files and blocks IP addresses after multiple failed login attempts, protecting against brute-force attacks.

2- SSH Configuration Hardening:

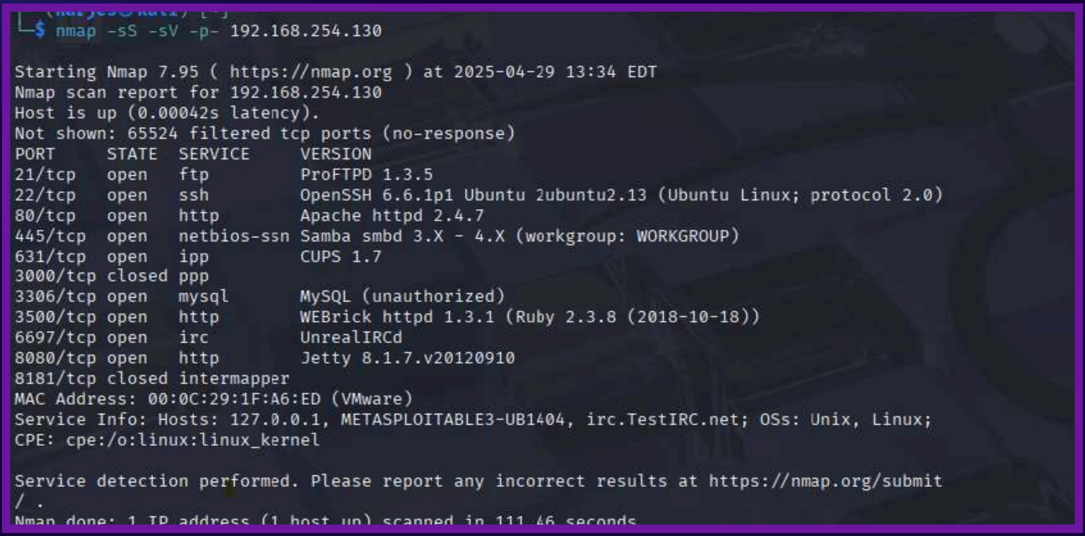
Manual adjustments to the SSH server's configuration file to disable password-based logins and enforce key-based authentication only, reducing the attack surface.

Before and after applying each mechanism, we assessed the system's security status using Nmap scans and attack tools to confirm the effectiveness of our changes.

Defense Mechanism 1 Fail2Ban

Step 1 : Using nmap to verify security status before installing Fail2Ban

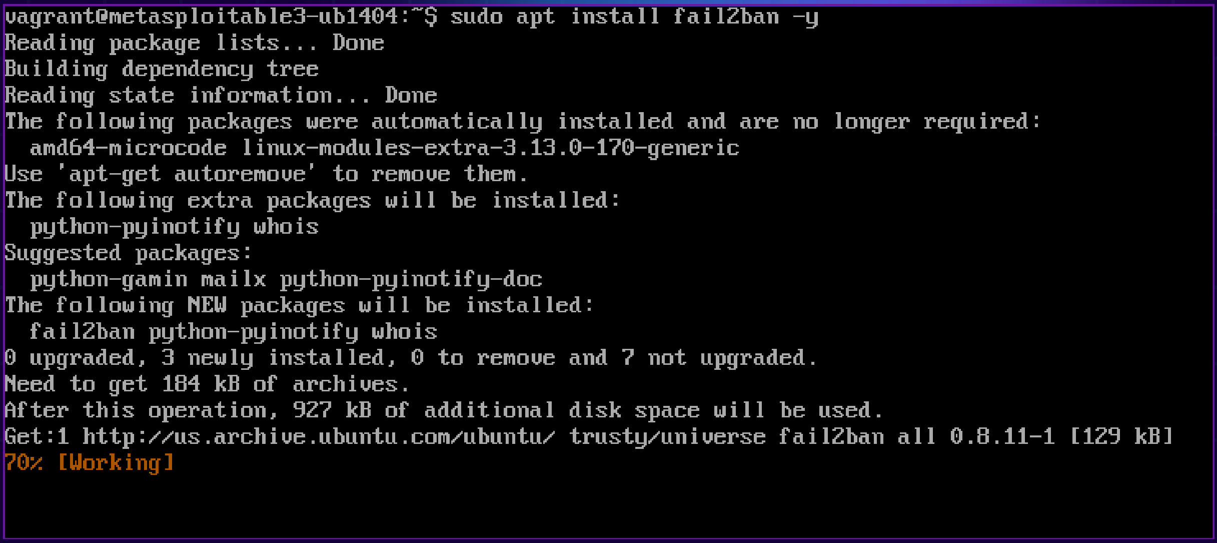
We performed a full TCP SYN and service version detection scan on 192.168.254.130 using Nmap with the command nmap -sS -sV -p- 192.168.254.130. The scan revealed multiple open ports and services, including FTP, SSH, HTTP, MySQL, IRC, and Jetty. The target was identified as Metasploitable3-UB1404, confirming it's a deliberately vulnerable system often used for penetration testing.



```
└─$ nmap -sS -sV -p- 192.168.254.130

Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-29 13:34 EDT
Nmap scan report for 192.168.254.130
Host is up (0.00042s latency).
Not shown: 65524 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)
3500/tcp open   http        WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
6697/tcp open   irc         UnrealIRCd
8080/tcp open   http        Jetty 8.1.7.v20120910
8181/tcp closed intermapper
MAC Address: 00:0C:29:1F:A6:ED (VMware)
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net; 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 111.46 seconds
```

Step 2: Installing fail2ban

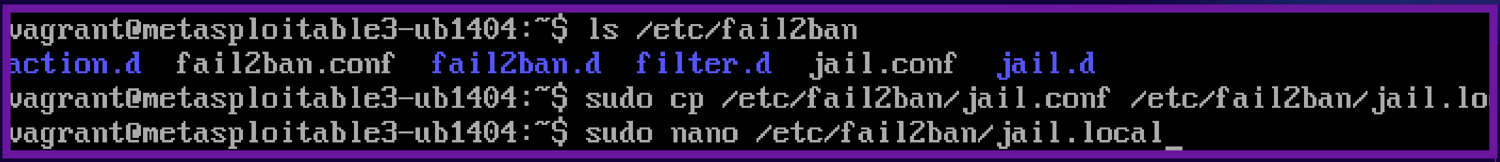
We installed Fail2Ban on the target system to protect it against brute-force attacks using the command sudo apt install fail2ban -y. This added the necessary packages and configured the service to enhance SSH login security.



```
vagrant@metasploitable3-ub1404:~$ sudo apt install fail2ban -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  amd64-microcode linux-modules-extra-3.13.0-170-generic
Use 'apt-get autoremove' to remove them.
The following extra packages will be installed:
  python-pyinotify whois
Suggested packages:
  python-gamin mailx python-pyinotify-doc
The following NEW packages will be installed:
  fail2ban python-pyinotify whois
0 upgraded, 3 newly installed, 0 to remove and 7 not upgraded.
Need to get 184 kB of archives.
After this operation, 927 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu/ trusty/universe fail2ban all 0.8.11-1 [129 kB]
70% [Working]
```

Step 3: Copying the Configuration File

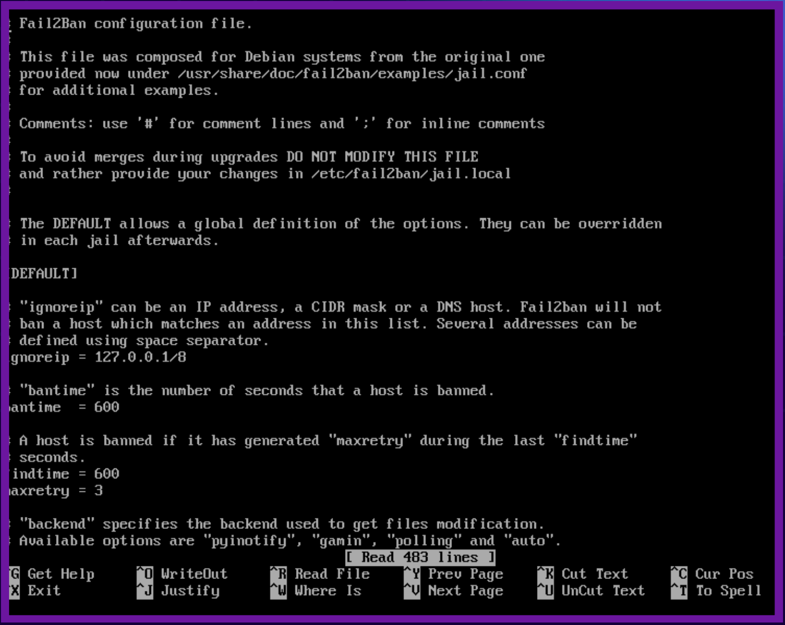
After installing the Fail2Ban package, we created a local configuration file (/etc/fail2ban/jail.local) by copying the default jail.conf file. This approach ensures that our custom settings remain intact during system updates. We then edited the jail.local file using nano /etc/fail2ban/jail.local



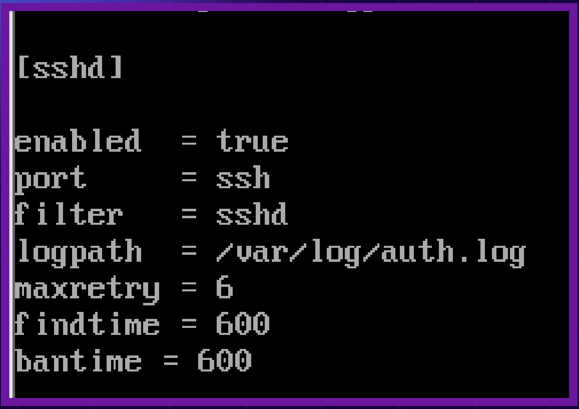
```
vagrant@metasploitable3-ub1404:~$ ls /etc/fail2ban
action.d  fail2ban.conf  fail2ban.d  filter.d  jail.conf  jail.d
vagrant@metasploitable3-ub1404:~$ sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.lo
vagrant@metasploitable3-ub1404:~$ sudo nano /etc/fail2ban/jail.local_
```

Step 4: Editing SSHD Settings

We opened the jail.local file using nano /etc/fail2ban/jail.local and enabled the [sshd] section. Then we added key parameters such as maxretry (maximum failed attempts), findtime (time window for monitoring), and bantime (duration of the ban) so that Fail2Ban can detect and block repeated failed SSH login attempts.



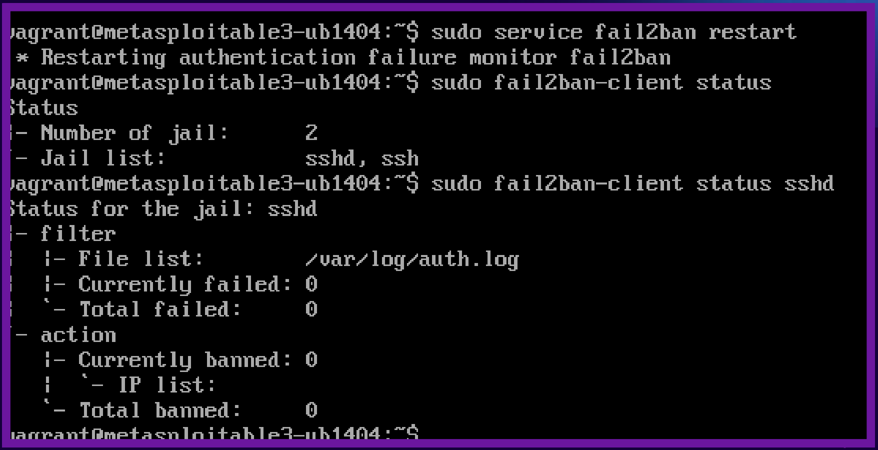
```
: Fail2Ban configuration file.
:
: This file was composed for Debian systems from the original one
: provided now under /usr/share/doc/fail2ban/examples/jail.conf
: for additional examples.
:
: Comments: use '#' for comment lines and ';' for inline comments
:
: To avoid merges during upgrades DO NOT MODIFY THIS FILE
: and rather provide your changes in /etc/fail2ban/jail.local
:

: The DEFAULT allows a global definition of the options. They can be overridden
: in each jail afterwards.

DEFAULT]

: "ignoreip" can be an IP address, a CIDR mask or a DNS host. Fail2ban will not
: ban a host which matches an address in this list. Several addresses can be
: defined using space separator.
gnoreip = 127.0.0.1/8

: "bantime" is the number of seconds that a host is banned.
antime  = 600

: A host is banned if it has generated "maxretry" during the last "findtime"
: seconds.
indtime = 600
axretry = 3

: "backend" specifies the backend used to get files modification.
: Available options are "pyinotify", "gamin", "polling" and "auto".
                                 [ Read 483 lines ]
G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```



```
[sshd]

enabled   = true
port      = ssh
filter    = sshd
logpath   = /var/log/auth.log
maxretry = 6
findtime = 600
bantime = 600
```

Step 5: Restarting Fail2Ban and Checking Status

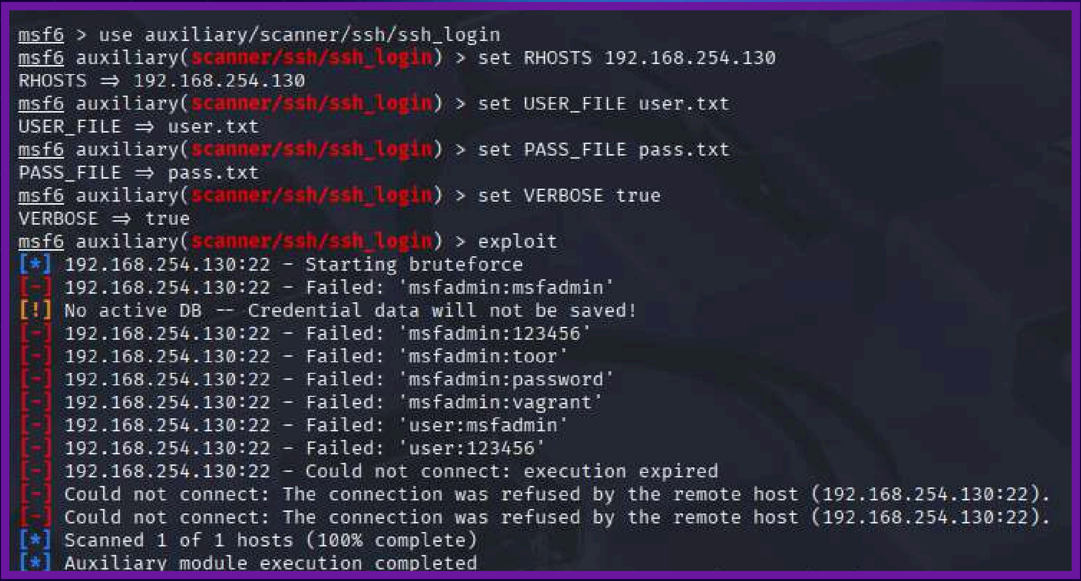
After completing the configuration, we restarted the Fail2Ban service to apply the changes. We then verified its status using fail2ban-client status and fail2ban-client status sshd, confirming that the system was active and monitoring. Note that all counters showed zero because no failed login attempts had occurred yet.



```
vagrant@metasploitable3-ub1404:~$ sudo service fail2ban restart
 * Restarting authentication failure monitor fail2ban
vagrant@metasploitable3-ub1404:~$ sudo fail2ban-client status
Status
|- Number of jail:      2
`- Jail list:           sshd, ssh
vagrant@metasploitable3-ub1404:~$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- filter
|  |- File list:        /var/log/auth.log
|  |- Currently failed: 0
|  `- Total failed:     0
`- action
   |- Currently banned: 0
   |  `- IP list:
   `- Total banned:     0
vagrant@metasploitable3-ub1404:~$
```

Step 6: Attacking the System with SSH Brute-Force

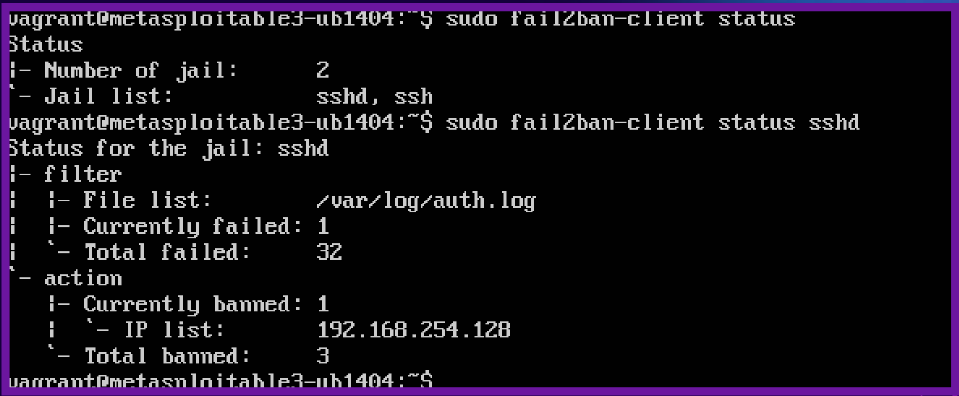
We used a brute-force SSH attack using Metasploit's module, like what we did in Phase 1. As shown in the results, after several failed login attempts, the connection was eventually refused by the remote host (192.168.254.130:22). This confirms that Fail2Ban successfully detected the repeated failed attempts and blocked further access, verifying that our defense mechanism worked as intended.



```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.254.130
RHOSTS ⇒ 192.168.254.130
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE user.txt
USER_FILE ⇒ user.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE pass.txt
PASS_FILE ⇒ pass.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE ⇒ true
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.254.130:22 - Starting bruteforce
[-] 192.168.254.130:22 - Failed: 'msfadmin:msfadmin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.254.130:22 - Failed: 'msfadmin:123456'
[-] 192.168.254.130:22 - Failed: 'msfadmin:toor'
[-] 192.168.254.130:22 - Failed: 'msfadmin:password'
[-] 192.168.254.130:22 - Failed: 'msfadmin:vagrant'
[-] 192.168.254.130:22 - Failed: 'user:msfadmin'
[-] 192.168.254.130:22 - Failed: 'user:123456'
[-] 192.168.254.130:22 - Could not connect: execution expired
[-] Could not connect: The connection was refused by the remote host (192.168.254.130:22).
[-] Could not connect: The connection was refused by the remote host (192.168.254.130:22).
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Step 7: Rechecking the Fail2Ban Status

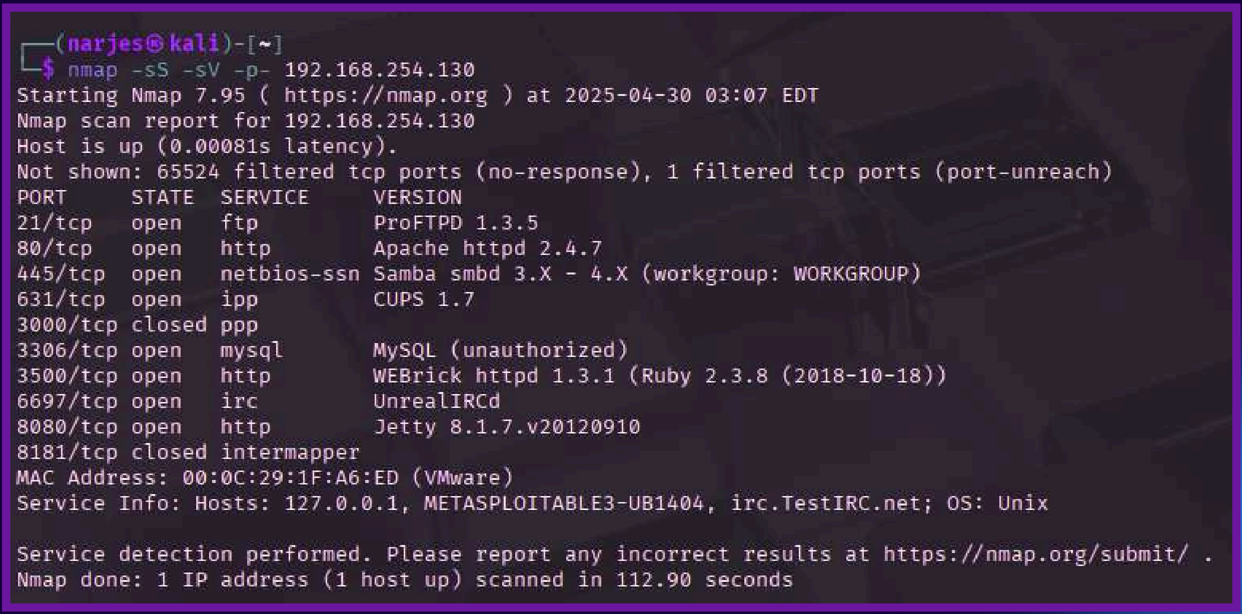
After the brute-force SSH attack, we checked Fail2Ban's status to confirm it had blocked our Kali machine's IP (192.168.254.128). We clearly saw that the Kali IP appeared in the list of banned IPs, with the currently banned counter showing 1, confirming that Fail2Ban successfully detected and blocked the attack.



```
vagrant@metasploitable3-ub1404:~$ sudo fail2ban-client status
Status
|- Number of jail:      2
`- Jail list:           sshd, ssh
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:     32
`- action
   |- Currently banned: 1
   |  `- IP list:       192.168.254.128
   `- Total banned:     3
vagrant@metasploitable3-ub1404:~$
```

Step 8: Checking the Security Status After Blocking

We then used nmap -sS -sV -p- 192.168.254.130, just as we did before installing Fail2Ban, to scan the system. This time, we clearly observed that the SSH port (22) was no longer accessible, confirming that Fail2Ban had successfully enforced the ban and secured the system against further SSH login attempts from the attacking machine.

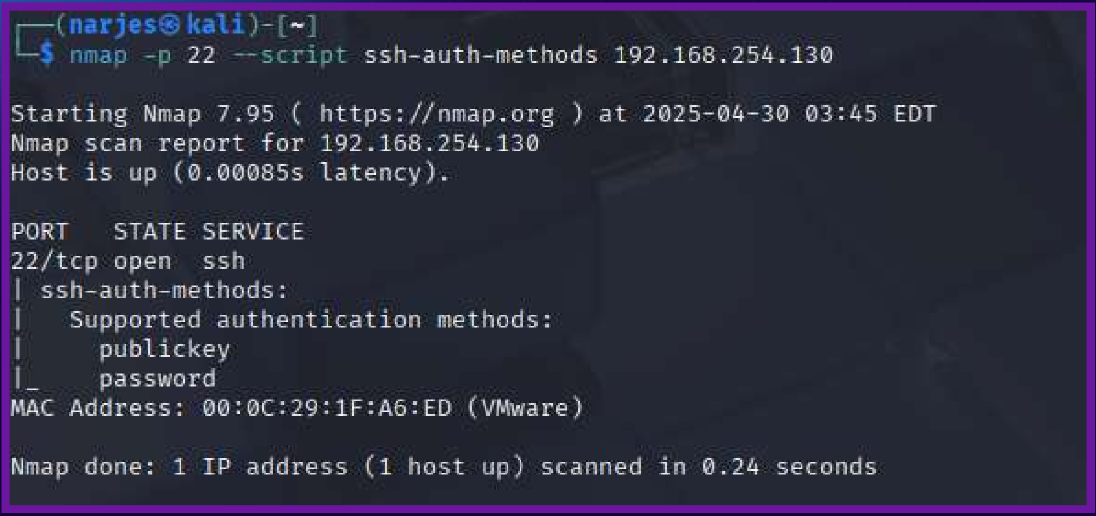


```
┌──(narjes㉿kali)-[~]
└─$ nmap -sS -sV -p- 192.168.254.130
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 03:07 EDT
Nmap scan report for 192.168.254.130
Host is up (0.00081s latency).
Not shown: 65524 filtered tcp ports (no-response), 1 filtered tcp ports (port-unreach)
PORT      STATE  SERVICE     VERSION
21/tcp    open   ftp         ProFTPD 1.3.5
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)
3500/tcp  open   http        WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
6697/tcp  open   irc         UnrealIRCd
8080/tcp  open   http        Jetty 8.1.7.v20120910
8181/tcp  closed intermapper
MAC Address: 00:0C:29:1F:A6:ED (VMware)
Service Info: Hosts: 127.0.0.1, METASPLOITABLE3-UB1404, irc.TestIRC.net; OS: Unix

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 112.90 seconds
```

Defense Mechanism 2
SSH Configuration Hardening

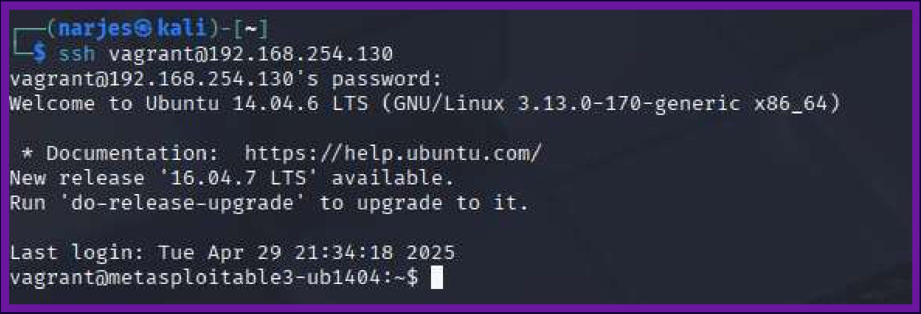
Step 1 : Using nmap to verify security status before implementing the defense mechanism

Before making any configuration changes, we used the nmap -p 22 --script ssh-auth-methods 192.168.254.130 command to scan the SSH service. This allowed us to identify the supported authentication methods, and we observed that it currently supports both publickey and password.



```
┌──(narjes㉿kali)-[~]
└─$ nmap -p 22 --script ssh-auth-methods 192.168.254.130

Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-30 03:45 EDT
Nmap scan report for 192.168.254.130
Host is up (0.00085s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|_    password
MAC Address: 00:0C:29:1F:A6:ED (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds
```

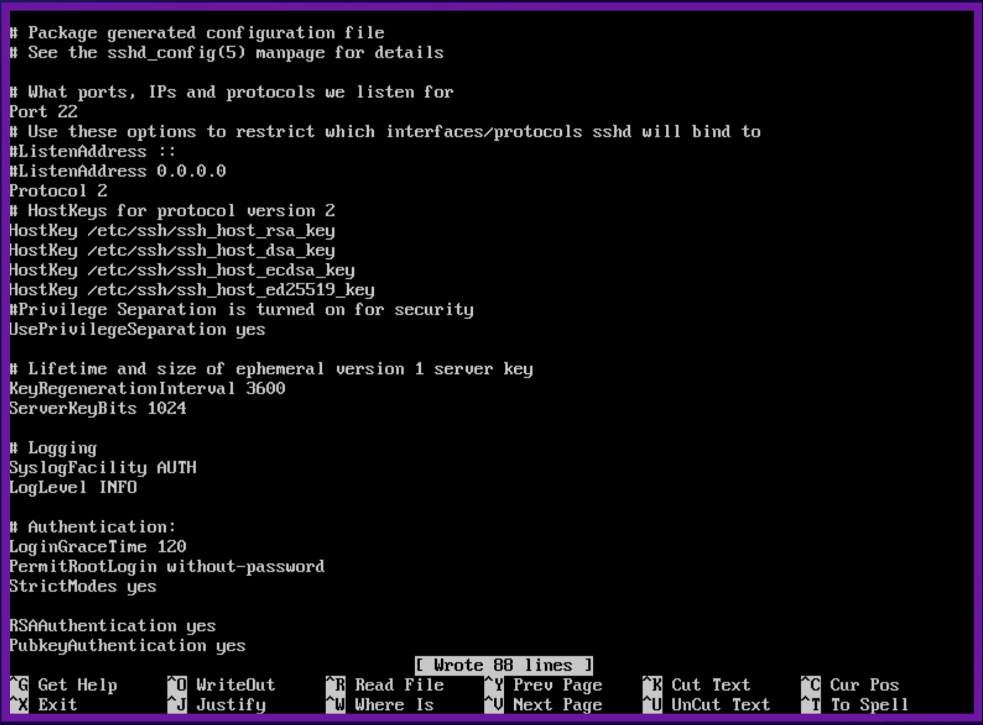
We also tested the SSH connection by connecting from the Kali machine to the target (ssh vagrant@192.168.254.130) to ensure the service was running normally and was accessible before applying any configuration changes.



```
┌──(narjes㉿kali)-[~]
└─$ ssh vagrant@192.168.254.130
vagrant@192.168.254.130's password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Apr 29 21:34:18 2025
vagrant@metasploitable3-ub1404:~$ 
```

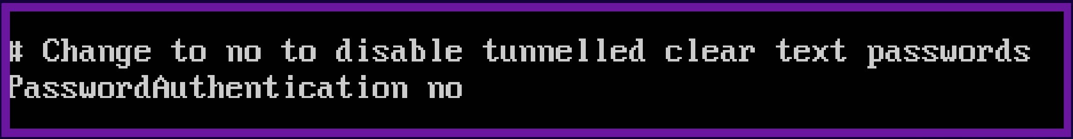
Step 2 : SSH Configuration Hardening

We accessed the SSH server configuration file (/etc/ssh/sshd_config) using a text editor (sudo nano /etc/ssh/sshd_config).



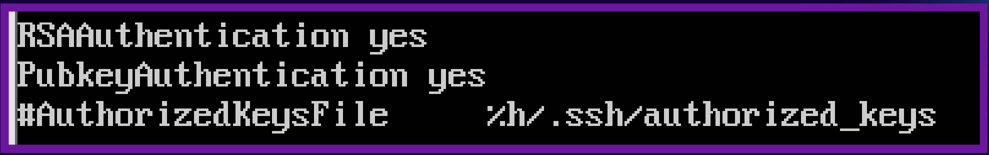
```
# Package generated configuration file
# See the sshd_config(5) manpage for details

# What ports, IPs and protocols we listen for
Port 22
# Use these options to restrict which interfaces/protocols sshd will bind to
#ListenAddress ::
#ListenAddress 0.0.0.0
Protocol 2
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
#Privilege Separation is turned on for security
UsePrivilegeSeparation yes

# Lifetime and size of ephemeral version 1 server key
KeyRegenerationInterval 3600
ServerKeyBits 1024

# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin without-password
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
                                         [ Wrote 88 lines ]
^G Get Help     ^O WriteOut     ^R Read File    ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where Is     ^V Next Page    ^U UnCut Text   ^T To Spell
```

While reviewing the /etc/ssh/sshd_config file, we found that PasswordAuthentication was set to yes, allowing users to log in using passwords, which presents a potential security risk. To enhance security, we changed this setting to no, thereby enforcing key-based authentication only.



```
# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no
```

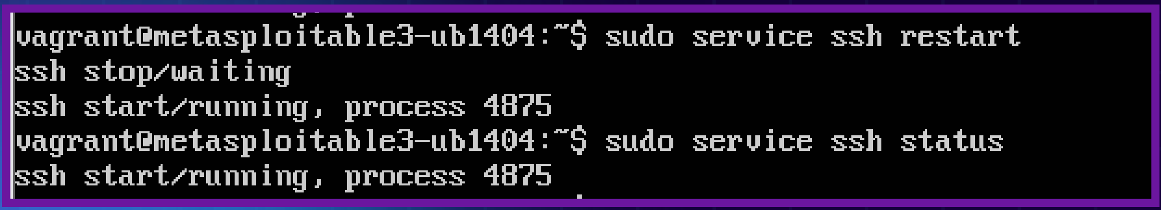
Also, we ensured that PubkeyAuthentication was set to yes to allow public key authentication, which is a more secure method of logging in compared to password-based authentication.



```
RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile     %h/.ssh/authorized_keys
```

Step 3 : Restarting the SSH Service

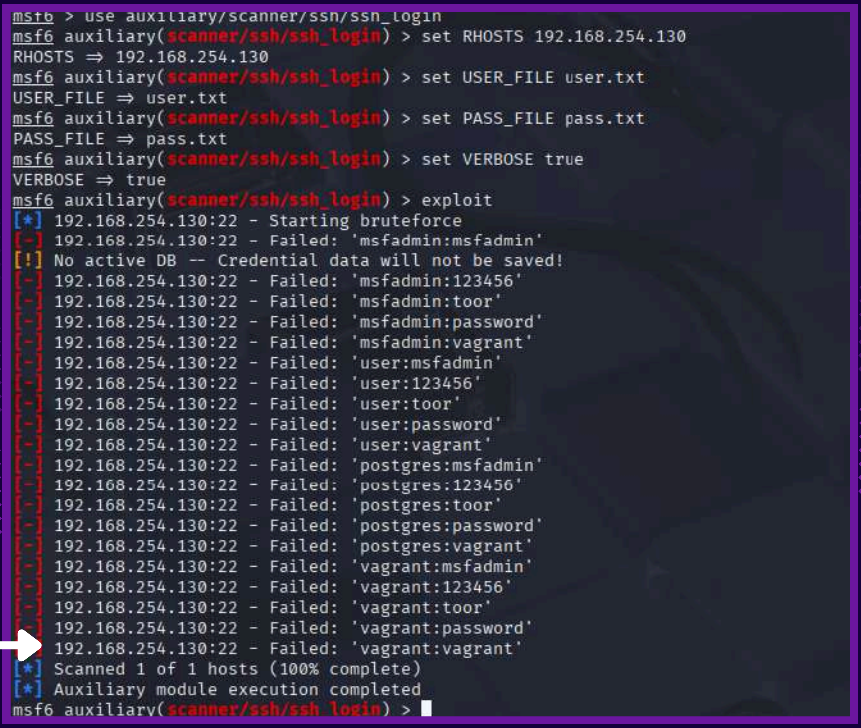
After saving the configuration changes, we restarted the SSH service using sudo service ssh restart to apply the new settings. We then used sudo service ssh status to verify that the service was running correctly and that the new settings had been applied.



```
vagrant@metasploitable3-ub1404:~$ sudo service ssh restart
ssh stop/waiting
ssh start/running, process 4875
vagrant@metasploitable3-ub1404:~$ sudo service ssh status
ssh start/running, process 4875
```

Step 4 : Attacking the System with SSH Brute-Force

We used the same brute-force SSH attack used in Phase 1; however, even though the correct username and password (vagrant:vagrant) were provided, the attack failed. This is because we had disabled password-based authentication in the SSH configuration (PasswordAuthentication no), effectively blocking any login attempts that rely on passwords.

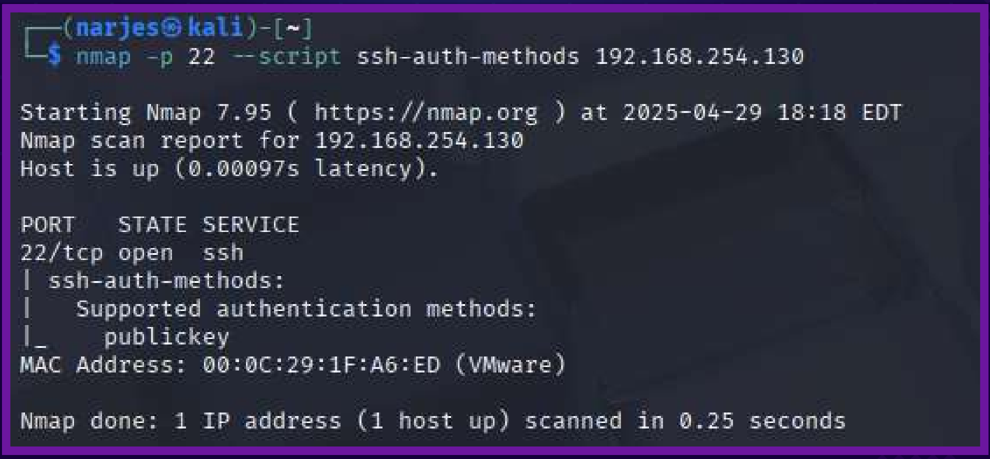


```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.254.130
RHOSTS => 192.168.254.130
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE user.txt
USER_FILE => user.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE pass.txt
PASS_FILE => pass.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set VERBOSE true
VERBOSE => true
msf6 auxiliary(scanner/ssh/ssh_login) > exploit
[*] 192.168.254.130:22 - Starting bruteforce
[-] 192.168.254.130:22 - Failed: 'msfadmin:msfadmin'
[!] No active DB -- Credential data will not be saved!
[-] 192.168.254.130:22 - Failed: 'msfadmin:123456'
[-] 192.168.254.130:22 - Failed: 'msfadmin:toor'
[-] 192.168.254.130:22 - Failed: 'msfadmin:password'
[-] 192.168.254.130:22 - Failed: 'msfadmin:vagrant'
[-] 192.168.254.130:22 - Failed: 'user:msfadmin'
[-] 192.168.254.130:22 - Failed: 'user:123456'
[-] 192.168.254.130:22 - Failed: 'user:toor'
[-] 192.168.254.130:22 - Failed: 'user:password'
[-] 192.168.254.130:22 - Failed: 'user:vagrant'
[-] 192.168.254.130:22 - Failed: 'postgres:msfadmin'
[-] 192.168.254.130:22 - Failed: 'postgres:123456'
[-] 192.168.254.130:22 - Failed: 'postgres:toor'
[-] 192.168.254.130:22 - Failed: 'postgres:password'
[-] 192.168.254.130:22 - Failed: 'postgres:vagrant'
[-] 192.168.254.130:22 - Failed: 'vagrant:msfadmin'
[-] 192.168.254.130:22 - Failed: 'vagrant:123456'
[-] 192.168.254.130:22 - Failed: 'vagrant:toor'
[-] 192.168.254.130:22 - Failed: 'vagrant:password'
[-] 192.168.254.130:22 - Failed: 'vagrant:vagrant'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Here we we provided the correct credentials (vagrant:vagrant), but it failed

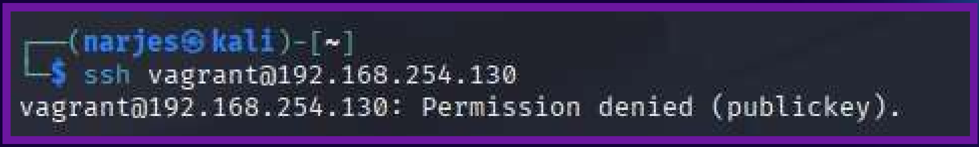
Step 5: Verifying Security Status After Implementing the Defense Mechanism

We ran the command nmap -p 22 --script ssh-auth-methods 192.168.254.130 to check which SSH authentication methods were still available. The result confirmed that only publickey authentication was supported, explaining why our password-based attempts failed.



```
┌──(narjes㉿kali)-[~]
└─$ nmap -p 22 --script ssh-auth-methods 192.168.254.130

Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-29 18:18 EDT
Nmap scan report for 192.168.254.130
Host is up (0.00097s latency).

PORT   STATE SERVICE
22/tcp open  ssh
| ssh-auth-methods:
|   Supported authentication methods:
|_    publickey
MAC Address: 00:0C:29:1F:A6:ED (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

Also, when we tried to connect manually using ssh vagrant@192.168.254.130, the system rejected password prompts, showing that only key-based logins were permitted.



```
┌──(narjes㉿kali)-[~]
└─$ ssh vagrant@192.168.254.130
vagrant@192.168.254.130: Permission denied (publickey).
```