**Report**

***Telnet is an outdated network protocol that allows users to remotely control a system over a network. However, its lack of encryption makes it highly vulnerable to brute-force attacks. In my project, I set up a Telnet server on an Ubuntu machine using xinetd to manage it."***

***"Once the server was running, I used the Metasploit framework to launch a brute-force attack on the Telnet service, using the well-known password list ‘rockyou.txt’. Metasploit iteratively tried various combinations of usernames and passwords until it found the correct credentials, giving me access to the system."***

**Setting Up a Telnet Vulnerability (Network Vulnerability) and Exploit (For Ubuntu) USER ‘OCTOBER’**

**Introduction**

Telnet is a network protocol used for remote communication, allowing for text-based command input. Despite its use, Telnet lacks encryption, making it vulnerable to brute-force attacks. This report outlines the steps to set up a vulnerable Telnet server on an **Ubuntu** system and perform a brute-force password attack using the **Metasploit Framework** and the **rockyou.txt** wordlist.

**Part 1: Setting Up the Telnet Server on Ubuntu**

**1. Installing the Telnet Server**

Before conducting the attack, the Telnet server must be installed on the target machine.

1. **Update the package list:**

sudo apt-get update

1. **Install the Telnet server:**

sudo apt-get install telnetd

1. **Check if Telnet is Managed by xinetdz**

In Ubuntu, Telnet is often managed by xinetd, which may not be available by default. If needed, install xinetd and configure it to manage the Telnet service.

**Installing xinetd on Ubuntu:**

1. **Install xinetd**:

sudo apt-get install xinetd

**3. Configuring Telnet under xinetd**

1. **Create or modify the Telnet configuration file** located at /etc/xinetd.d/telnet:

sudo nano /etc/xinetd.d/telnet

1. **Add the following configuration** to enable Telnet:

service telnet

{

disable = no

flags = REUSE

socket\_type = stream

wait = no

user = root

server = /usr/sbin/in.telnetd

log\_on\_failure += USERID

}

**4. Restart the xinetd Service**

Once the Telnet configuration is set, restart the xinetd service to apply the changes:

sudo service xinetd restart

**5. Verify Telnet is Running**

To confirm that Telnet is running and accessible, perform the following checks:

1. **Test the Telnet connection locally**:

telnet localhost

1. **Check if Telnet is listening on port 23**:

sudo netstat -tuln | grep 23

If the output shows:

tcp 0 0 0.0.0.0:23 0.0.0.0:\* LISTEN

Telnet is successfully running and listening on port 23.

**Part 2: Exploiting Telnet via a Brute-Force Attack**

**Objective**

**This exercise aimed to exploit a Telnet service running on the target system 10.0.2.15 by using a brute-force attack. The attack was aimed at retrieving valid login credentials by using a list of common passwords.**

**Tools Used**

* **Kali Linux: A popular penetration testing operating system.**
* **Metasploit Framework: Used to conduct the brute-force attack on the Telnet service.**
* **rockyou.txt: A commonly used wordlist for password cracking.**
* **Telnet: The target protocol for this attack.**

After successfully setting up the Telnet server, we can attempt a brute-force attack using the **Metasploit Framework**.

**1. Set Up the Target in Metasploit**

1. **Launch Metasploit** by typing msfconsole in the terminal.
2. **Select the Telnet login module**:

use auxiliary/scanner/telnet/telnet\_login

1. **Configure the target IP address** (RHOSTS):

set RHOSTS 10.0.2.15

1. **Configure the Usernames and Passwords**

**USER\_FILE: A file containing usernames (usernames.txt) was used to test possible accounts on the target.**

1. **Set the username file**:

set USER\_FILE usernames.txt

1. **Set the password file** (in this case, we are using the rockyou.txt wordlist):

**PASS\_FILE**: The widely known password list rockyou.txt was used to attempt passwords for each username.

set PASS\_FILE rockyou.txt

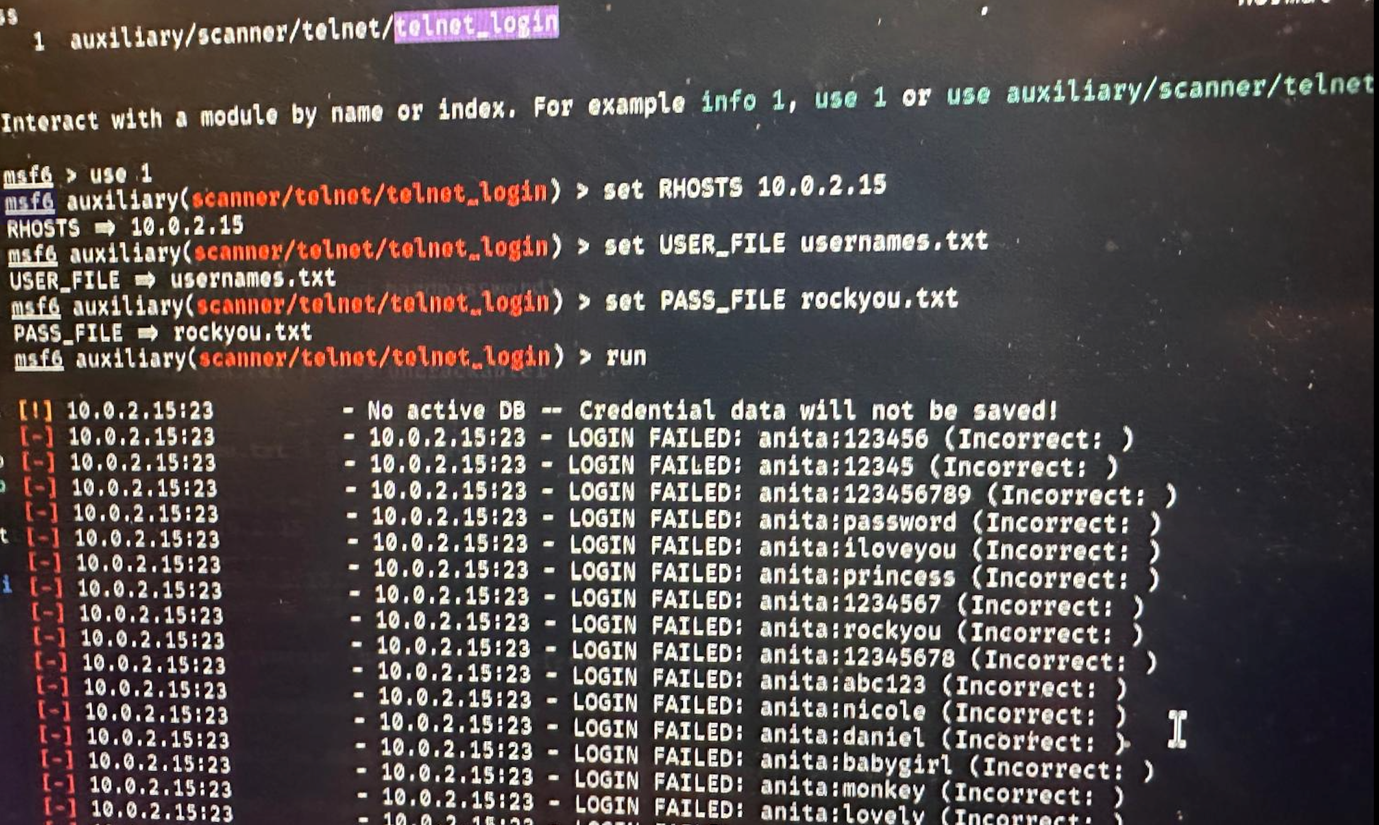
**3. Run the Brute-Force Attack**

**After setting the required parameters, the brute-force attack was initiated using the command run. This caused the Metasploit framework to attempt various username and password combinations. The console output showed a series of login attempts, all of which initially failed.**

Run

Exploit

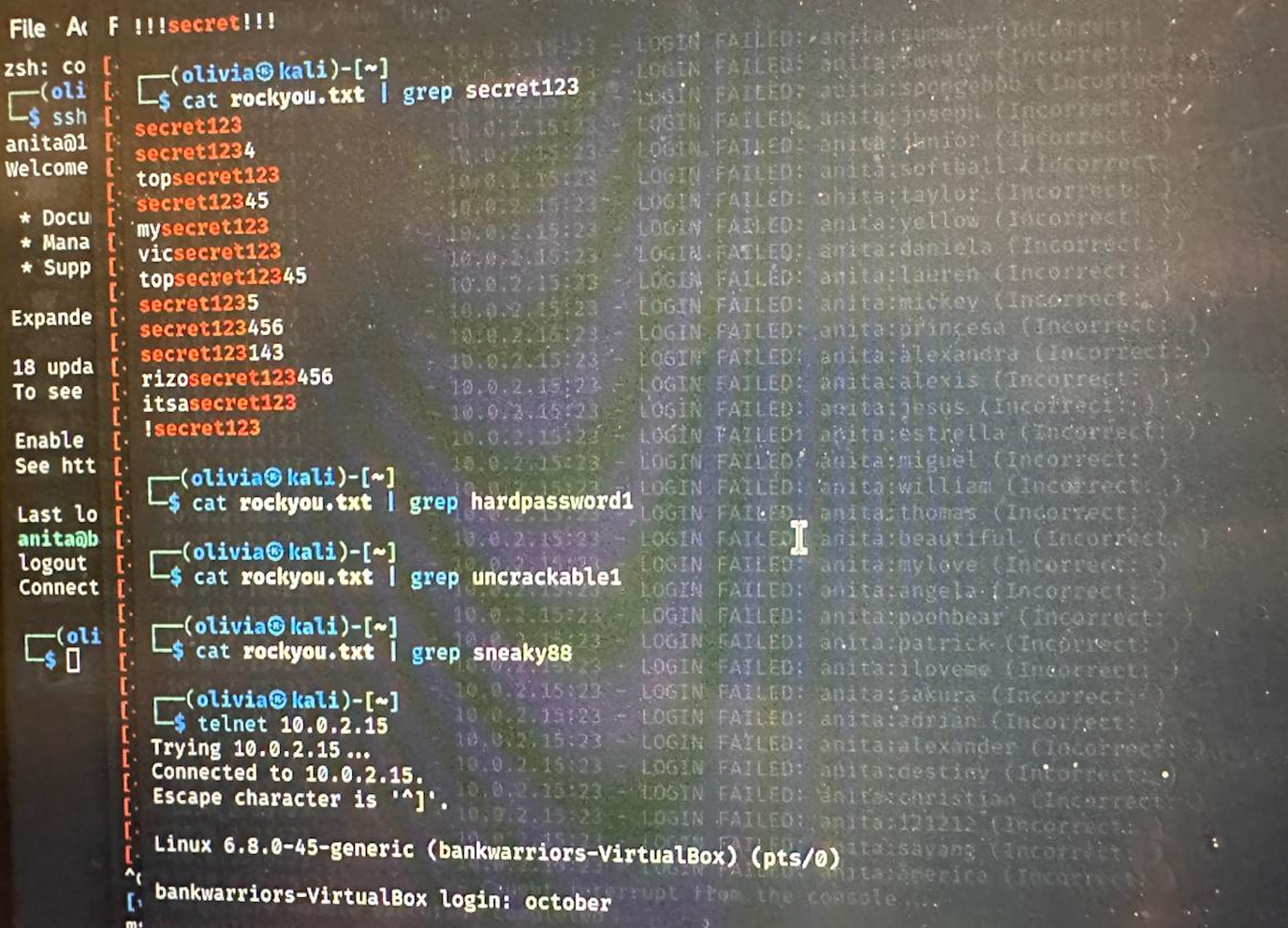
The Metasploit framework will attempt different username and password combinations. The output will indicate the success or failure of each attempt.



**4. Filtering Specific Passwords: Using grep, specific patterns in the password list (such as secret123 and hardpassword1) were searched in rockyou.txt. This allowed us to refine potential guesses if required.**

**grep secret123 rockyou.txt**

**5. Successful Connection Once a valid username-password combination was found, the Telnet session was successfully established, allowing access to the target machine.**

****

**Conclusion**

This report outlines the process for:

1. Installing and configuring the Telnet server on an **Ubuntu** system.
2. Configuring Telnet under xinetd and verifying it is running on port 23.
3. Exploiting Telnet using the **Metasploit Framework** to brute-force credentials.

Due to the lack of encryption in the Telnet protocol, it is highly susceptible to attacks, making it unsuitable for modern systems. It is recommended to use more secure protocols like **SSH** for remote communication.

**Horizontal Privilege Escalation Using Write Access to /etc/shadow for User 'James'**

***I explored a horizontal privilege escalation attack. This form of escalation occurs when a user gains unauthorized access to another user’s account. In this case, I demonstrated how weak file permissions on the /etc/shadow file can allow a non-privileged user to modify the root password."***

***"By having write access to /etc/shadow, I generated a new SHA-512 password hash and replaced the root user’s hash. This allowed me to log in as the root user and gain full administrative rights. Once the attack was complete, I restored the original /etc/shadow file to cover my tracks."***

**Introduction**

In this report, we explore the concept of **horizontal privilege escalation** by manipulating the **/etc/shadow** file, which contains password hashes for users on the system. Specifically, we focus on leveraging weak file permissions where a non-privileged user (james) has written access to **/etc/shadow**, allowing the user to escalate privileges by altering root’s password hash.

**Objective**

The objective is to demonstrate how a user (in this case, james) can escalate their privileges by modifying the **/etc/shadow** file, replacing the root password with a known password. Once the root password is changed, james can gain root access and perform actions with full administrative rights.

**Horizontal Privilege Escalation Process**

The following steps outline leveraging weak permissions on **/etc/shadow** to gain root access through horizontal privilege escalation.

**Step 1: Verifying Write Access to /etc/shadow**

First, ensure that the user james has the required write access to the **/etc/shadow** file. From the screenshot, it is evident that the user james is able to read and possibly write to **/etc/shadow**, which is typically restricted to the root user.

$ ls -l /etc/shadow

-rw-rw-r-- 1 root shadow 3424 Sep 23 18:10 /etc/shadow

Here, **/etc/shadow** is writeable, indicating that user James can potentially modify this

critical file.

**Step 2: Generating a New Root Password Hash**

To proceed with the privilege escalation, the user james will generate a new password hash for root using the **SHA-512** encryption algorithm. This hash will later be inserted into **/etc/shadow** to replace the original root hash, effectively changing the root password.

On a separate attacker machine, use the following command to generate the hash for a new password (e.g., “password”):

mkpasswd -m sha-512 password

This command outputs the SHA-512 hash of the password "password", which will be used to replace the root’s hash in the **/etc/shadow** file

**Step 3: Make a Backup of /etc/shadow**

Before modifying the **/etc/shadow** file, it is important to create a backup. This ensures that the file's original state can be restored if something goes wrong during the modification process. After successfully gaining root access, it also allows the attacker to restore the original password hash for stealth purposes.

cp /etc/shadow /home/james/shadow.bak

This creates a backup of **/etc/shadow** in the home directory of james.

**Step 4: Editing the /etc/shadow File**

Now that we have the hash, edit the **/etc/shadow** file to replace the existing root password hash with the new one.

1. Open the **/etc/shadow** file using a text editor like nano or vim:

nano /etc/shadow

1. Locate the entry for the root user, which typically looks like this:

root:$6$abc123...:19662:0:99999:7:::

1. Replace the existing hash with the new SHA-512 hash generated earlier (e.g., for the password "password"):

root:$6$newhash...:19662:0:99999:7:::



**horizontal privilege escalation** by manipulating the **/etc/shadow** file, which contains password hashes for users on the system. Specifically, we focus on leveraging weak file permissions where a non-privileged user (james) has written access to **/etc/shadow**, allowing the user to escalate privileges by altering root’s password hash.

To proceed with the privilege escalation, the user james will generate a new password hash for root using the **SHA-512** encryption algorithm. This hash will later be inserted into **/etc/shadow** to replace the original root hash, effectively changing the root password.

On a separate attacker machine, use the following command to generate the hash for a new password (e.g., “password”):

mkpasswd -m sha-512 password

This command outputs the SHA-512 hash of the password "password", which will be used to replace the root’s hash in the **/etc/shadow** file

1. Save the file and exit the editor.

**Step 5: Logging in as Root**

With the root password now changed to the known value "password," the next step is to log in as the root user:

su root

When prompted, enter the password password. If successful, james will now have root privileges and can execute commands as the root user.

**Step 6: Restoring the Original /etc/shadow**

To avoid detection, restore the original **/etc/shadow** file from the backup created earlier. This will revert the root password to its original value:

cp /home/james/shadow.bak /etc/shadow

Now, james retains root access but restores the system back to its previous state, making it difficult for system administrators to detect that any changes were made.

**Conclusion**

This report demonstrates how weak file permissions on **/etc/shadow** can be exploited for horizontal privilege escalation. By leveraging write access to this critical file, a non-privileged user like James can modify root’s password, gain full administrative rights, and then cover their tracks by restoring the original **/etc/shadow** file.

It is crucial to ensure that sensitive files like **/etc/shadow** are protected by appropriate file permissions and only accessible by the root user to prevent unauthorized modifications and potential system compromises

**Horizontal Privilege Escalation through Hidden Files on Misconfigured User Account**

***I created user accounts for Lucy and October on an Ubuntu machine. Lucy stored sensitive information, like passwords, in hidden files. By misconfiguring file permissions, October was able to access these hidden files and retrieve Lucy’s password."***

***"October then used Lucy’s password to log into her account and gain unauthorized access. This demonstrates how misconfigured file permissions can lead to security vulnerabilities, even if the sensitive files are hidden from plain sight."*­­­­­­­­**

**A screenshot of a computer

Description automatically generated**

**Introduction**

Horizontal privilege escalation occurs when a user gains access to another user’s account and resources on the same privilege level. In this scenario, the user **October** manages to gain unauthorized access to **Lucy’s** laptop by exploiting misconfigured permissions and accessing hidden files that contain sensitive information, such as a password.

**Scenario Setup on Ubuntu**

**Step 1: Create User Accounts for Lucy and October**

1. **Create User for Lucy:** On the target Ubuntu machine, create a user account for **Lucy**:

sudo adduser lucy

Follow the prompts to set the password and complete the setup.

1. **Create User for October:** Similarly, create a user account for **October**:

sudo adduser october

**Step 2: Set Up Hidden Files on Lucy's Laptop**

1. **Create a Hidden Directory with Sensitive Files:** As **Lucy**, create a hidden directory and add some sensitive files (including password-related information):

su lucy

mkdir /home/lucy/.hidden\_files

touch /home/lucy/.hidden\_files/password.txt

echo "LucySecretPassword123" > /home/lucy/.hidden\_files/password.txt

These hidden files contain sensitive information that should be protected from other users. However, we will misconfigure file permissions to allow October to access these files.

1. **Create Default Files with Sensitive Information:** Inside **Lucy’s** hidden folder, create text files that contain various sensitive information like passwords, system configurations, and personal notes:

echo "LucySecretPassword123" > /home/lucy/.hidden\_files/password.txt

echo "System Configuration Info" > /home/lucy/.hidden\_files/system.conf

echo "Private Notes: Important Project Details" > /home/lucy/.hidden\_files/private\_notes.txt

1. **Misconfigure Permissions for the Hidden Directory and Files:** Set the permissions on **Lucy’s** hidden directory and files to allow other users to read them:

chmod 755 /home/lucy/.hidden\_files

chmod 644 /home/lucy/.hidden\_files/\*

This allows the files to be read by other users, but they are still hidden from casual inspection due to their names starting with a period (.).

**Step 3: Access Lucy's Files as October**

As **October**, you can now access Lucy’s hidden files due to the misconfigured permissions:

1. **Log in as October:**

su October

1. **Navigate to Lucy’s Hidden Directory:** Even though the files are hidden, you can still access them by navigating directly to the folder:

cd /home/lucy/.hidden\_files

1. **View the Hidden Files:** List the hidden files:

ls -la

You should be able to see and read the contents of the password.txt file:

cat password.txt

This will reveal **Lucy’s** password, which October can now use to log in as Lucy.

**Step 4: Logging in as Lucy (Password Theft)**

After retrieving the password from **Lucy’s** hidden file, **October** can now log into **Lucy’s** account using the stolen credentials:

1. **Switch to Lucy’s Account:**

su lucy

When prompted, enter the password you retrieved from password.txt.

**Attack Setup in Kali Linux**

To simulate this attack from a **Kali Linux** machine, follow the steps below:

**Step 1: Set Up SSH Access to the Target Machine**

1. **Enable SSH on the Ubuntu Target:** Install and enable SSH on the Ubuntu target machine:

sudo apt-get install openssh-server

sudo systemctl enable ssh

sudo systemctl start ssh

1. **Retrieve Target’s IP Address:** Find the IP address of the Ubuntu machine using:

ifconfig

**Step 2: Perform SSH Login from Kali Linux**

1. **Initiate SSH Login:** On the **Kali Linux** machine, attempt to log in to **Lucy’s** machine using October’s account via SSH:

ssh october@<target-ip>

Replace <target-ip> with the IP address of the Ubuntu target.

1. **Access Lucy’s Hidden Files:** Once logged in as **October**, repeat the earlier steps to access Lucy’s hidden files:

cd /home/lucy/.hidden\_files

cat password.txt

**Step 3: Exploit the Password and Log in as Lucy from Kali Linux**

Now that **October** has retrieved **Lucy’s** password, she can SSH directly into Lucy’s account from the **Kali** machine:

1. **SSH into Lucy’s Account:** Use the stolen password to log in:

ssh lucy@<target-ip>

1. **Confirm Access:** Once logged in, confirm you have access to **Lucy’s** account and files.

**Conclusion**

This scenario demonstrates a classic horizontal privilege escalation attack in which misconfigured file permissions allowed one user (October) to access another user’s (Lucy’s) hidden files, retrieve sensitive information, and ultimately gain unauthorized access to her account. Proper file permissions and user isolation are essential to prevent such vulnerabilities.