Simple CTF writeup

This is my walkthrough of the **Simple CTF** room on TryHackMe, but instead of a traditional step-by-step guide, I've written it through the lens of the **Cyber Kill Chain (CKC)**. It helps break down how I approached the box from an attacker's perspective, stage by stage.

🔎 1. Reconnaissance

First things first — I started with recon to gather as much intel on the target as possible.

🔧 Tools Used:

```
nmap -sC -sV -oN simplectf.nmap <TARGET_IP>
```

🔍 Results:

- **FTP** on port 21
- **SSH** on port 22
- **HTTP** on port 80

I also checked the website manually, and did some directory fuzzing with Gobuster:

```
gobuster dir -u http://<IP> -w /usr/share/wordlists/dirb/common.txt
```

That revealed `/simple`, `/robots.txt`, and other interesting directories.

🧨 2. Weaponization

With open FTP and potential web vulnerabilities, I started prepping my attack paths.

🔷 FTP:

- Anonymous login was allowed! username and pass : anonymous
- That gave me access to the FTP directory, and I found the first flag: `user.txt`.

🔷 Web:

- Explored the `/simple` directory and found a PHP-based site.
- Suspected a possible file upload vuln or maybe RFI/LFI.

No malware or exploit development needed — just using what was available

📦 3. Delivery

Time to deliver the payload. Since I already had web access, I uploaded a PHP reverse shell via the web interface (if enabled) or exploited any exposed PHP file that allowed execution.

I also set up my listener:

```
nc -lvnp 4444
```

Then triggered the reverse shell. Used shell from Pentest monkeys.

💣 4. Exploitation

Boom — got a shell.

Upgraded it to something usable:

```
bash
CopyEdit
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

From here, I poked around the file system, looked at permissions, and checked what I could do as the current user.

🏗️ 5. Installation

Once the shell was stable, I treated it as a foothold. I didn't need persistence (since it's a CTF), but I made sure the session was stable enough for privilege escalation.

Used:

```
export TERM=xterm
```

And checked sudo and SUID binaries.

📡 6. Command & Control

My reverse shell was the basic Netcat kind. Nothing fancy like Cobalt Strike or Metasploit C2

```
nc -lvnp 4444
```

The shell was responsive, and I could explore freely.

🎯 7. Actions on Objectives

My end goal: get the **root flag**.

I found a SUID-enabled binary:

```
bash
CopyEdit
find / -perm -4000 -type f 2>/dev/null
```

I noticed Python had the SUID bit set — jackpot.

```
bash
CopyEdit
python -c 'import os; os.setuid(0); os.system("/bin/bash")'
```

That dropped me straight into a root shell. Navigated to `/root`, and boom — `root.txt` captured.

Question and Answers

Q: How many services are running under port 1000?

A: 2

Q: What is running on the higher port?

A: ssh

Q: What's the CVE you're using against the application?

A: CVE-2019-9053

Q: To what kind of vulnerability is the application vulnerable?

A: sqli

Q: What's the password?

A: secret

Q: Where can you login with the details obtained?

A: ssh

Q: What's the user flag?

A: G00d j0b, keep up!

Q: Is there any other user in the home directory? What's its name?

A: sunbath

Q: What can you leverage to spawn a privileged shell?

A: vim

Q: What's the root flag?

A: W3ll d0n3. You made it!