【Writeup】Boston Key Party CTF 2015(部分题目)

原创

lymh 于 2015-03-05 11:10:08 发布
分类专栏： 其它记录 文章标签： php 漏洞 CTF

假期试着做了一下这场美国的CTF比赛，无奈题目看了一遍都没什么想法，只好等比赛结束再学习了。在这里总结一下学到的姿势。

（以下是六道php代码审计题目）

1.Prudential

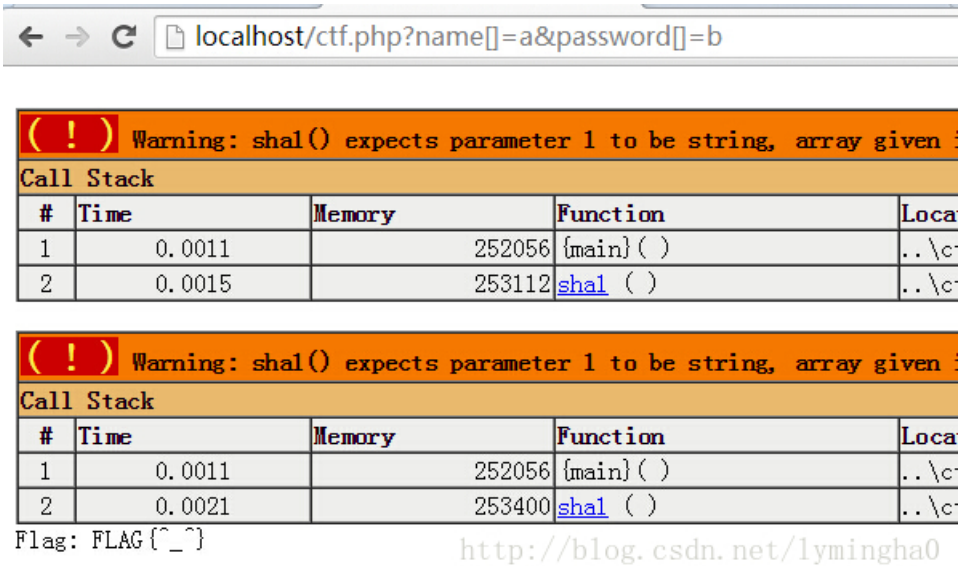
I don't think that sha1 is broken. Prove me wrong.

代码如下：

```
<html>
<head>
 <title>level1</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
require 'flag.php';

if (isset($_GET['name']) and isset($_GET['password'])) {
    if ($_GET['name'] == $_GET['password'])
        print 'Your password can not be your name.';
    else if (sha1($_GET['name']) === sha1($_GET['password']))
      die('Flag: '.$flag);
    else
        print '<p class="alert">Invalid password.</p>';
}
?>

<section class="login">
 <div class="title">
  <a href="./index.txt">Level 1</a>
 </div>

 <form method="get">
  <input type="text" required name="name" placeholder="Name"/><br/>
  <input type="text" required name="password" placeholder="Password" /><br/>
  <input type="submit"/>
 </form>
</section>
</body>
</html>
```

分析代码逻辑，发现GET了两个字段name和password，获得flag要求的条件是：name != password & sha1(name) == sha1(password)，乍看起来这是不可能的，其实可以利用sha1()函数的漏洞来绕过。如果把这两个字段构造为数组，如：**?name[]=a&password[]=b**，这样在第一处判断时两数组确实是不同的，但在第二处判断时由于sha1()函数无法处理数组类型，将报错并返回false，if 条件成立，获得flag。
经验证md5()函数同样存在此漏洞。

测试截图：



2.Symphony

A less than four characters number, bigger than 999? Maybe the bug is elsewhere.

代码如下：

```
<html>
<head>
 <title>level2</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
require 'flag.php';

if (isset($_GET['password'])) {
 if (is_numeric($_GET['password'])){
  if (strlen($_GET['password']) < 4){
   if ($_GET['password'] > 999)
    die('Flag: '.$flag);
   else
    print '<p class="alert">Too little</p>';
  } else
    print '<p class="alert">Too long</p>';
 } else
  print '<p class="alert">Password is not numeric</p>';
}
?>

<section class="login">
        <div class="title">
                <a href="./index.txt">Level 2</a>
        </div>

        <form method="get">
                <input type="text" required name="password" placeholder="Password" /><br/>
                <input type="submit"/>
        </form>
</section>
</body>
</html>
```

这个相对简单，只要脑洞够大，能想到数字还能用 **1e9** 这样的形式来表示。。。

3.Northeastern Univ.

> Of course, a timing attack might be the answer, but I'm quite sure that you can do better than that.

代码如下：

```
<html>
<head>
 <title>level3</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
require 'flag.php';

if (isset($_GET['password'])) {
    if (strcmp($_GET['password'], $flag) == 0)
  die('Flag: '.$flag);
    else
  print '<p class="alert">Invalid password.</p>';
}
?>

<section class="login">
        <div class="title">
                <a href="./index.txt">Level 3</a>
        </div>

        <form method="get">
                <input type="text" required name="password" placeholder="Password" /><br/>
                <input type="submit"/>
        </form>
</section>
</body>
</html>
```

乍看也算是个比较严密的验证逻辑，但正如第一题一样，strcmp()函数也只能处理字符串参数，传个数组进去就能返回false，又由于它与0的比较用的是==而非===（允许类型转换后比较），就满足了这个 if 的条件。Payload：**?password[]=a**

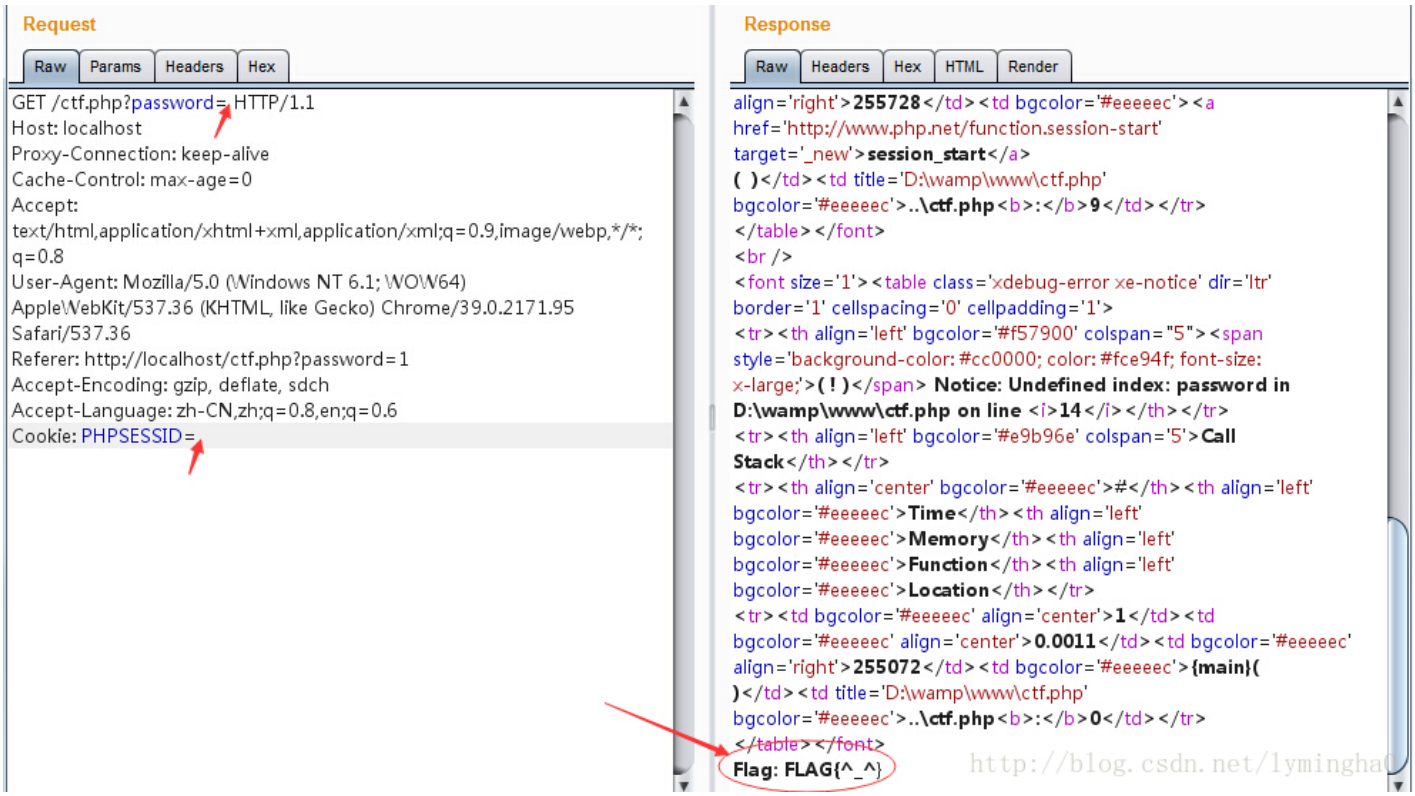
4.Museum of Fine Arts

Because cryptography is hard, we only implemented a hand-made PRNG. What could possibly go wrong?

代码如下：

```
<html>
<head>
 <title>level4</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
session_start();

require 'flag.php';

if (isset ($_GET['password'])) {
    if ($_GET['password'] == $_SESSION['password'])
        die ('Flag: '.$flag);
    else
        print '<p class="alert">Wrong guess.</p>';
}

// Unpredictable seed
mt_srand((microtime() ^ rand(1, 10000)) % rand(1, 10000) + rand(1, 10000));
?>

<section class="login">
        <div class="title">
                <a href="./index.txt">Level 4</a>
        </div>

  <ul class="list">
  <?php
  for ($i=0; $i<3; $i++)
   print '<li>' . mt_rand (0, 0xffffff) . '</li>';
  $_SESSION['password'] = mt_rand (0, 0xffffff);
  ?>
  </ul>

        <form method="get">
                <input type="text" required name="password" placeholder="Next number" /><br/>
                <input type="submit"/>
        </form>
</section>
</body>
</html>
```

最初看完以后感觉不可做，莫非真要分析出rand()函数的规律？！但是那就不是CTF的难度了吧。。。后来发现如果将session手动清除掉，然后password字段也提交为空，不就可以绕过了嘛，也是蛮笨的。。。



5.Longwood Medical

Because we dont trust mysqli_real_escape_string, we wrote our own military-grade sanitization method.

代码如下：

```
<html>
<head>
 <title>level5</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php

require 'flag.php';

if (isset ($_GET['name']) and isset ($_GET['password'])) {
    $name = $_GET['name'];
    $password = $_GET['password'];

    if (ctype_alnum ($name) and ctype_alnum ($password)) {
        $request = 'SELECT login FROM user where login = ' . $name . ' AND password = ' . $password . ';';
        $db = new SQLite3 (sha1($flag).'.db', SQLITE3_OPEN_READONLY); // Ghetto anti-database-download
        $result = $db->querySingle ($request);
        $db->close ();

        if ($result === FALSE)
            echo '<p class="alert">"Invalid login or password</p>';
        else
            die('Flag: ' . $flag);
    } else
        echo '<p class="alert">Invalid chars detected</p>';
}
?>

<section class="login">
        <div class="title">
                <a href="./index.txt">Level 5</a>
        </div>

        <form method="get">
                <input type="text" required name="name" placeholder="Name"/><br/>
                <input type="text" required name="password" placeholder="Password" /><br/>
                <input type="submit"/>
        </form>
</section>
</body>
</html>
```

注入题，代码逻辑还不是很明白。其他人的Payload： **?name=0&password=0**，为什么可注还没有看懂。。。

6.Brigham Circle

Sanitization is hard, lets use regexp!

代码如下：

```
<html>
<head>
 <title>level6</title>
    <link rel='stylesheet' href='style.css' type='text/css'>
</head>
<body>

<?php
require 'flag.php';

if (isset ($_GET['password'])) {
 if (ereg ("^[a-zA-Z0-9]+$", $_GET['password']) === FALSE)
  echo '<p class="alert">You password must be alphanumeric</p>';
 else if (strpos ($_GET['password'], '--') !== FALSE)
  die('Flag: ' . $flag);
 else
  echo '<p class="alert">Invalid password</p>';
}
?>

<section class="login">
        <div class="title">
                <a href="./index.txt">Level 6</a>
        </div>

        <form method="get">
                <input type="text" required name="password" placeholder="Password" /><br/>
                <input type="submit"/>
        </form>
</section>
</body>
</html>
```

关于ereg()函数的一个漏洞——%00截断，可以构造一个这样的提交：**?password=a%00–**，ereg()匹配到%00就截止了，所以会认为提交串合法，但strpos()不受此影响，成功绕过。

此外还有一个非常微妙的方法。把password构造为数组，如：**?password[]=a**，由于ereg()也是只能处理字符串的，遇到数组做参数返回NULL，注意第一处判断用的是 === ，要求类型也相同，而NULL跟FALSE类型是不同的；第二处判断strpos()的参数同样不能为数组，否则返回NULL，而判断用的是 !== ，所以这里的条件成立，也能得到flag。

注：这里的第二处判断 !== 并非多此一举，因为当strpos()返回位置为0时用 == 将不知道到底是找到了还是没找到，于是不转换类型的 === 或 !== 是实际中普遍使用的。