ISCC2019-WEB4-Writeup

原创

菜鸟之小菜 于 2019-06-03 20:08:29 发布 229 收藏
分类专栏： 安全 Writeup 文章标签： ISCC2019 writeup web

0x00

这道题整体思路是变量覆盖：

PHP中会产生变量覆盖的函数有以下几种可能：

$$使用不当（PHP中这种写法表示可变变量）

extract函数使用不当

parse_str函数使用不当（本题就是这个函数导致的变量覆盖）

import_request_variables使用不当

开启全局变量

以上这些变量覆盖的例子可查看如下博客：https://www.jianshu.com/p/a4d782e91852

0x01

分析源代码

打开web页面看到源代码如下：

```php
<?php
error_reporting(0);
include("flag.php");
$hashed_key = 'ddbafb4eb89e218701472d3f6c087fdf7119dfdd560f9d1fcbe7482b0feea05a';
$parsed = parse_url($_SERVER['REQUEST_URI']);
if(isset($parsed["query"])){
        $query = $parsed["query"];
        $parsed_query = parse_str($query);
        if($parsed_query!=NULL){
                $action = $parsed_query['action'];
        }

        if($action=="auth"){
                $key = $_GET["key"];
                $hashed_input = hash('sha256', $key);
                if($hashed_input!==$hashed_key){
                        die("<img src='cxk.jpg'>");
                }

                echo $flag;
        }
}else{
        show_source(__FILE__);
}?>
```


拿到源代码，一开始并不是能够直接看懂，PHP中有很多不常用的函数，所以就一点点查各个函数的功能，下面贴出对源代码的分析结果：

```php
<?php
error_reporting(0);
include("flag.php"); //包含flag.php文件，里面放着flag
$hashed_key = 'ddbafb4eb89e218701472d3f6c087fdf7119dfdd560f9d1fcbe7482b0feea05a'; //hash单向散列得出的字符串，下面分析代码得知是sha256哈希
$parsed = parse_url($_SERVER['REQUEST_URI']); //通过parse_url函数来分解超全局变量$_SERVER接收的URL传递来的参数，分解后是一个数组，放到$parsed变量中
if(isset($parsed["query"])){
    $query = $parsed["query"];              //将传递来的参数赋值给$query变量（注意，$parsed["query"]也是一个数组，->
    $parsed_query = parse_str($query);      //也就是说$parsed接收到的$_SERVER传递来的数组中还嵌套这数组，这种数据类型在PHP中是允许的
    if($parsed_query!=NULL){
        $action = $parsed_query['action'];  //将URL传递来的参数中的  action参数  赋值给  $action变量
    }

    if($action==="auth"){                   //很显然，传递的action要等于auth  而且是真等于（这个是PHP的特性，PHP是弱数据类型的语言）
        $key = $_GET["key"];                //由通过$_GET方式获取key
        $hashed_input = hash('sha256', $key);           //将url传递来的key通过sha256哈希后与源代码中保存的hash值进行比较，如果真相同就
        if($hashed_input!==$hashed_key){                //打印出flag
            die("<img src='cxk.jpg'>");
        }

        echo $flag;
    }
}else{
    show_source(__FILE__);
}?>
```


0x02

思路的转折

经过分析代码，会不会想到要破解hashed_key，但是那是不可逆的，想通过碰撞得到什么时候才能拿到答案？？？

PHP弱类型？又分析分析代码，不是弱类型

有没有疑惑，为什么接收的都是URL传递的参数，action用的是$_SERVER接收的而key是$_GET接收的（这个是后来在做其他题时想到的，第一想法就是变量覆盖，只有这样才能够顺利的解决哈希值这一步）

知道了是变量覆盖就好办了，查了查PHP变量覆盖就看到了parse_url函数

OK，思路有了，下面就开始构造payload了，通过URL传参把hashed_key覆盖就好了

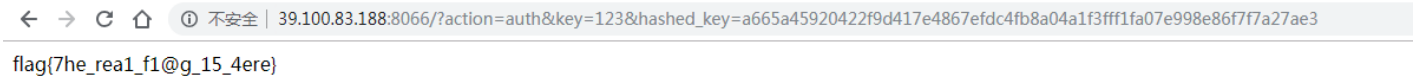
0x03

构造payload如下：

http://39.100.83.188:8066/?action=auth&key=123&hashed_key=a665a45920422f9d417e4867efdc4fb8a04a1f3fff1fa07e998e86f7f7a27ae3

后面那个hashed_key是key这个参数的值经过sha256得到的结果

然后提交就看到了下面的页面



flag{7he_rea1_f1@g_15_4ere}