easy_serialize_php

发表于 2021-01-16 分类于 Challenge ， 2019 ， 安洵杯 ， Web
Challenge | 2019 | 安洵杯 | Web | easy_serialize_php

点击此处获得更好的阅读体验

WriteUp来源

https://xz.aliyun.com/t/6911

题目考点

- *变量覆盖*
- *预包含*
- *反序列化中的对象逃逸*

解题思路

首先打开网页界面，看到source_code，点击就可以直接看到源码。

从上往下阅读代码，很明显的可以发现一个变量覆盖。至于这个覆盖怎么用，暂时还不知道，这是第一个考点。

往下看，可以看到我们可以令function为phpinfo来查看phpinfo，此时就可以看到我的第二个考点：

我在php.ini中设置了auto_prepend_file隐式包含了d0g3_f1ag.php，直接访问可以发现没有任何内容，说明我们需要读取这个文件的内容。

接着往下看代码，可以看到最终执行了一个`file_get_contents`，从这个函数逆推回去`$userinfo["img"]`的值，可以发现这个值虽然是我们可控的，但是会经过sha1加密，而我没有解密，导致无法读取任何文件。

此时需要把注意力转移到另外一个函数`serialize`上，这里有一个很明显的漏洞点，数据经过序列化了之后又经过了一层过滤函数，而这层过滤函数会干扰序列化后的数据。

> *PS：任何具有一定结构的数据，**如果经过了某些处理而把结构体本身的结构给打乱了**，则有可能会产生漏洞。*

先放payload：

```
1 _SESSION[user]=flagflagflagflagflagflag&_SESSION[function]=a";s:3:"img";s:20:"ZDBnM19mMWFnLnBocA==";s:2:"dd";s:1:"a";}&function=show_image
```

这里我令_SESSION[user]为flagflagflagflagflagflag，正常情况下序列化后的数据是这样的：

```
1 a:3:{s:4:"user";s:24:"flagflagflagflagflagflag";s:8:"function";s:59:"a";s:3:"img";s:20:"ZDBnM19mMWFnLnBocA==";s:2:"dd";s:1:"a";}";s:3:"img";s:28:"L3VwbG9hZC9ndWVzdF9pbWcuanBn";}
```

而经过了过滤函数之后，序列化的数据就会变成这样：

```
1 a:3:{s:4:"user";s:24:"";s:8:"function";s:59:"a";s:3:"img";s:20:"ZDBnM19mMWFnLnBocA==";s:2:"dd";s:1:"a";}";s:3:"img";s:28:"L3VwbG9hZC9ndWVzdF9pbWcuanBn";}
```

可以看到，user的内容长度依旧为24，但是已经没有内容了，所以反序列化时会自动往后读取24位：

会读取到上图的位置，然后结束，由于user的序列化内容读取数据时需要往后填充24位，导致后面function的内容也发生了改变，吞掉了其双引号，导致我们可以控制后面的序列化内容。

而php反序列化时，当一整段内容反序列化结束后，后面的非法字符将会被忽略，而如何判断是否结束呢，可以看到，前面有一个a:3，表示序列化的内容是一个数组，有三个键，而以{作为序列化内容的起点，}作为序列化内容的终点。

所以此时后面的`";s:3:"img";s:28:"L3VwbG9hZC9ndWVzdF9pbWcuanBn";}`在反序列化时就会被当作非法字符忽略掉，导致我们可以控制$userinfo["img"]的值，达到任意文件读取的效果。

在读取完`d0g3_flag.php`后，得到下一个hint，获取到flag文件名，此时修改payload读根目录下的flag即可。

- ***本文作者：** CTFHub*
- ***本文链接：** https://writeup.ctfhub.com/Challenge/2019/安洵杯/Web/4C3GdGAG19S68x7k6hiCVP.html*
- ***版权声明：** 本博客所有文章除特别声明外，均采用 BY-NC-SA 许可协议。转载请注明出处！*

 # Challenge # Web # 2019 # 安洵杯
easy_web
不是文件上传