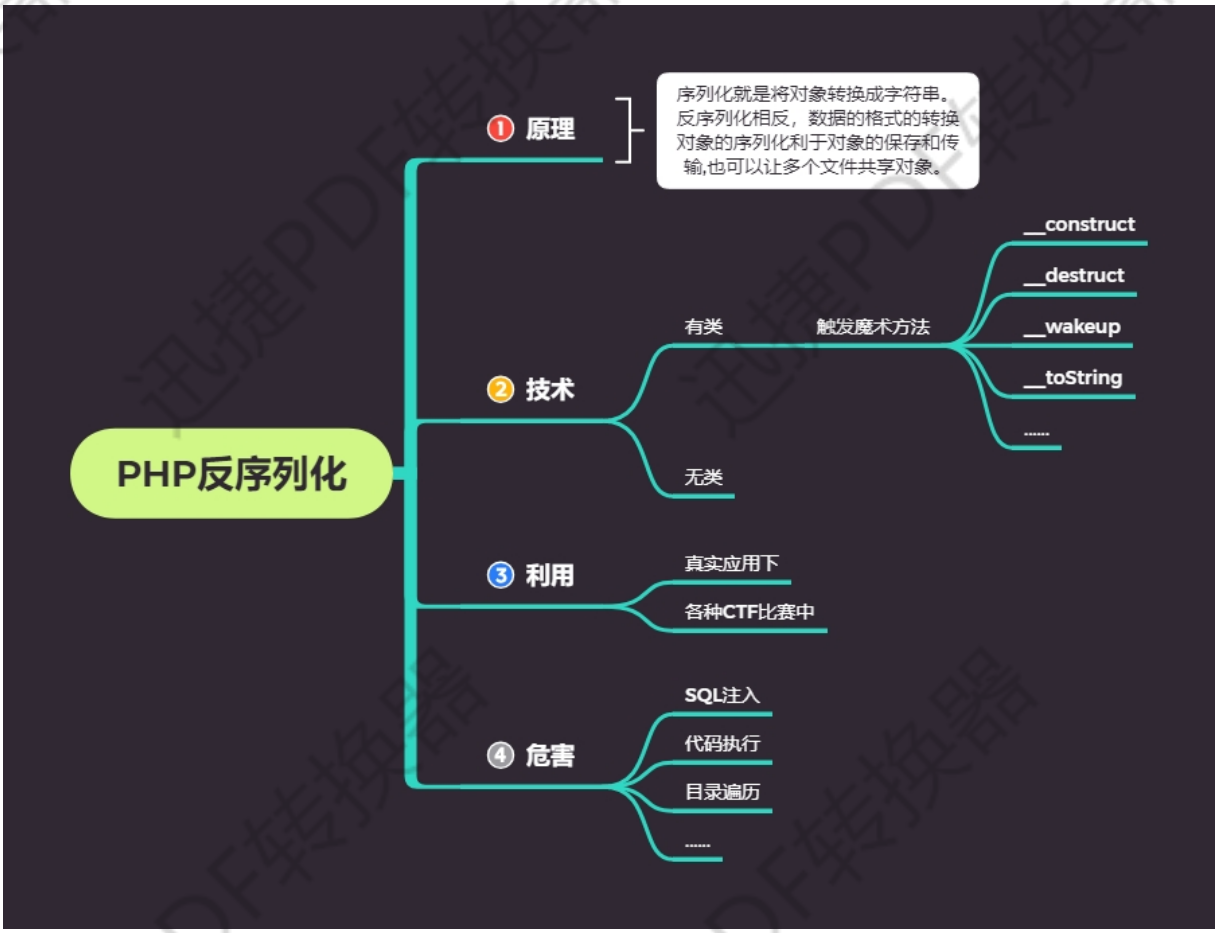
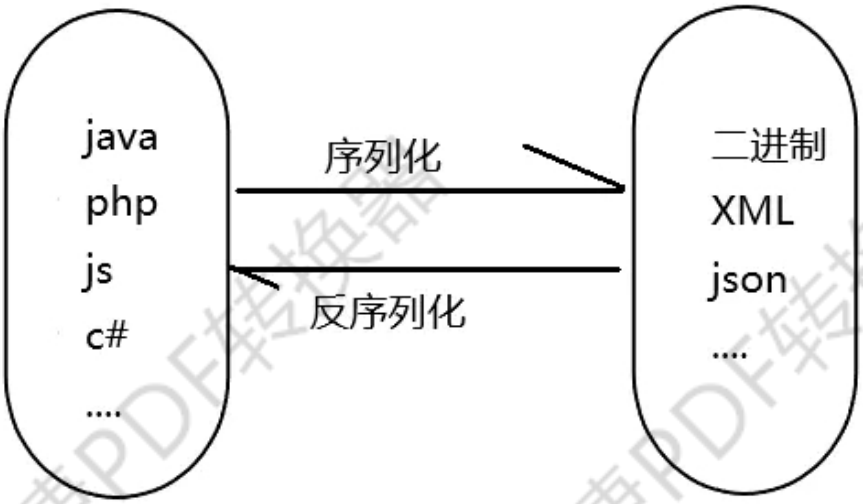
WEB 漏洞-反序列化之 PHP&JAVA 全解(上)





#PHP 反序列化

原理：未对用户输入的序列化字符串进行检测，导致攻击者可以控制反序列化过程，从而导致代码执行，SQL 注入，目录遍历等不可控后果。在反序列化的过程中自动触发了某些魔术方法。当进行反序列化的时候就有可能会触发对象中的一些魔术方法。

serialize()　　//将一个对象转换成一个字符串

unserialize()　//将字符串还原成一个对象

触发：unserialize 函数的变量可控，文件中存在可利用的类，类中有魔术方法：

参考：https://www.cnblogs.com/20175211lyz/p/11403397.html

__construct()//创建对象时触发

__destruct() //对象被销毁时触发

__call() //在对象上下文中调用不可访问的方法时触发

__callStatic() //在静态上下文中调用不可访问的方法时触发

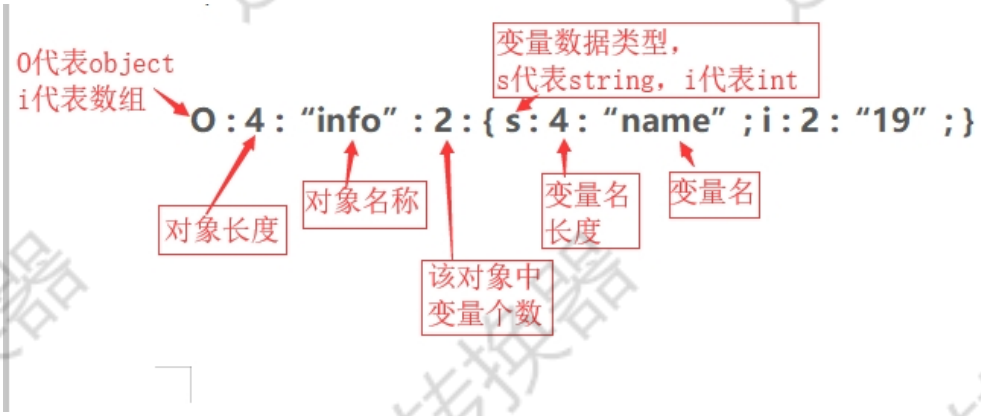
__get() //用于从不可访问的属性读取数据

__set() //用于将数据写入不可访问的属性

__isset() //在不可访问的属性上调用 isset()或 empty()触发

__unset() //在不可访问的属性上使用 unset()时触发

__invoke() //当脚本尝试将对象调用为函数时触发



演示案例：

- 先搞一把 PHP 反序列化热身题稳住-无类问题-本地
- 在撸一把 CTF 反序列化小真题压压惊-无类执行-实例
- 然后抗一把 CTF 反序列化练习题围观下-有类魔术方法触发-本地
- 最后顶一把网鼎杯 2020 青龙大真题舒服下-有类魔术方法触发-实例

首先 ctf 命名及代码函数 unserialize 判断反序列化知识点
第一：获取 flag 存储 flag.php
第二：两个魔术方法__destruct __construct
第三：传输 str 参数数据后触发 destruct，存在 is_valid 过滤
第四：__destruct 中会调用 process,其中 op=1 写入及 op=2 读取
第五：涉及对象 FileHandler，变量 op 及 filename,content，进行构造输出

```
<?php
class FileHandler{
public $op=' 2';//源码告诉我们 op 为 1 时候是执行写入为 2 时执行读
public $filename="flag.php";//文件开头调用的是 flag.php
public $content="xd";
}
$flag = new FileHandler();
$flag_1 = serialize($flag);
echo $flag_1;
?>
```

涉及：反序列化魔术方法调用，弱类型绕过，ascii 绕过
使用该类对 flag 进行读取，这里面能利用的只有__destruct 函数（析构函数）。__destruct 函数对 $this->op 进行了===判断并内容在 2 字符串时会赋值为 1，process 函数中使用==对$this->op 进行判断（为 2 的情况下才能读取内容），因此这里存在弱类型比较，可以使用数字 2 或字符串' 2'绕过判断。

is_valid 函数还对序列化字符串进行了校验，因为成员被 protected 修饰，因此序列化字符串中会出

现 ascii 为 0 的字符。经过测试，在 PHP7.2+的环境中，使用 public 修饰成员并序列化，反序列化后成员也会被 public 覆盖修饰。

涉及资源：

http://www.dooccn.com/php/

https://www.ctfhub.com/#/challenge

https://ctf.bugku.com/challenges#flag.php

https://cgctf.nuptsast.com/challenges#Web

https://www.cnblogs.com/20175211lyz/p/11403397.html

```
<?php
error_reporting(0);
include "flag.php";
$KEY = "xiaodi";
$str = $_GET['str'];
if (unserialize($str) === "$KEY")
{
echo "$flag";
}
show_source(__FILE__);

class ABC{
public $test;
function __construct(){
$test =1;
echo '调用了构造函数<br>';
}
function __destruct(){
echo '调用了析构函数<br>';
}
function __wakeup(){
echo '调用了苏醒函数<br>';
}
}
echo '创建对象 a<br>';
$a = new ABC;
echo '序列化<br>';
$a_ser=serialize($a);
```

```
echo '反序列化<br>';
$a_unser = unserialize($a_ser);
echo '对象快要死了！ ';
?>
```
