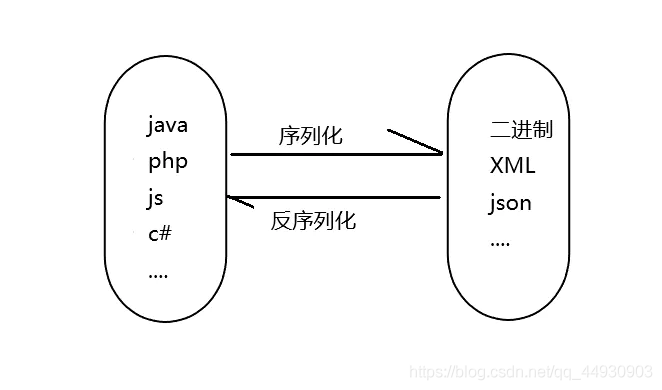
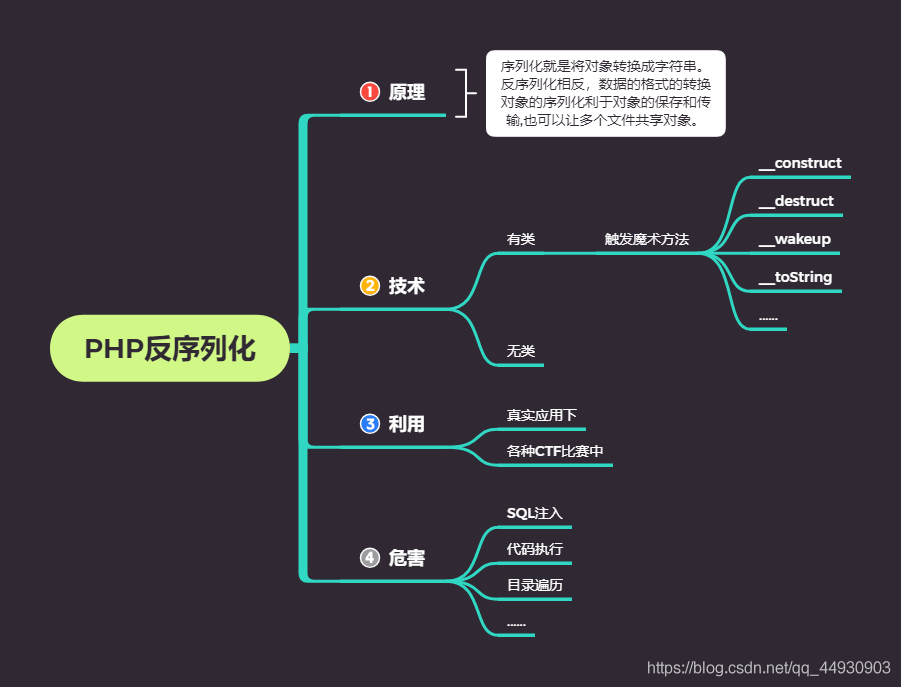
# 37 WEB漏洞-反序列化之PHP&JAVA全解(上)



#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

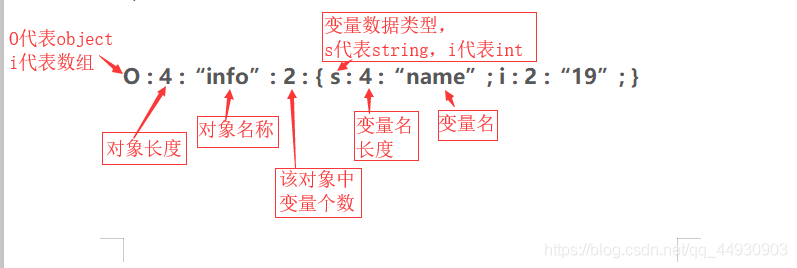
\_\_get() //用于从不可访问的属性读取数据

\_\_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



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。

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

<?php

error\_reporting(0);

include "flag.php";

$KEY = "liuyifei";

$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！';

?>

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