



# php反序列化

主讲人：杨欣然(p5ych)

# PHP基础

- **PHP** (PHP: Hypertext Preprocessor) 即**超文本预处理器**, 是在**服务器端**执行的**脚本语言**, 尤其适用于Web开发并可嵌入HTML中。PHP语法学习了C语言, 吸纳Java和Perl多个语言的特色发展出自己的特色语法, 并根据它们的长项持续改进提升自己, 例如java的面对对象的编程, 该语言当初创建的主要目标是让开发人员快速编写出优质的web网站。PHP同时支持面向对象和面向过程的开发, 使用上**非常灵活**。

- <http://psych.green/psych/web/web%E5%9F%BA%E7%A1%80/php.html>

# 基础知识

- 为什么反序列化？
- 基本函数和方法
- 方法和属性

# PHP反序列化原理

序列化其实就是将数据转化成一种可逆的数据结构，自然，逆向的过程就叫做反序列化。那么为什么要反序列化呢？可以说，序列化就是将对象转换为可以储存，传输的数据。

- **为什么反序列化？**

- **存储需求**

“所有php里面的值都可以使用函数serialize()来返回一个包含字节流的字符串来表示。序列化一个对象将会保存对象的所有变量，但是不会保存对象的方法，只会保存类的名字。”在程序执行结束时，内存数据便会立即销毁，变量所储存的数据便是内存数据，而文件、数据库是“持久数据”，因此PHP序列化就是将内存的变量数据“保存”到文件中的持久数据的过程。

- **传输需求**

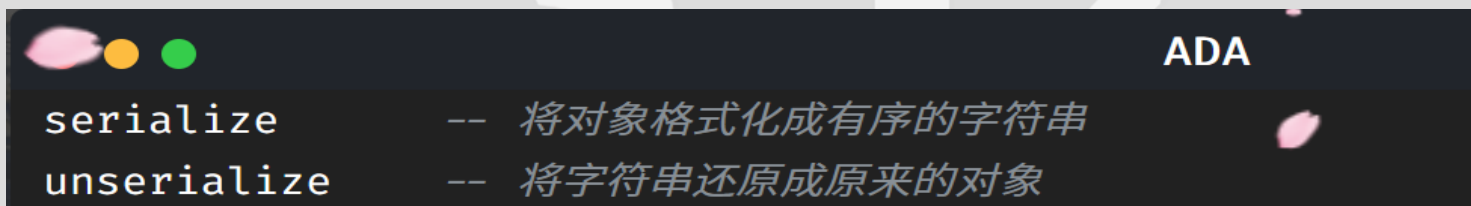
序列化说通俗点就是把一个对象变成可以传输的字符串。比如，json格式，这就是一种序列化，有可能就是通过array序列化而来的。而反序列化就是把那串可以传输的字符串再变回对象。

这样就让对象能够以字节流的形式传输。

# PHP反序列化原理

## 基本函数和方法

- 对于PHP，进行反序列化时主要用到了这两种函数：



```
ADA
serialize    -- 将对象格式化成有序的字符串
unserialize  -- 将字符串还原成原来的对象
```

- 除了这两个函数，还需要了解PHP中常见的魔术方法

# 常见魔术方法



SCSS

`__construct()` 当创建对象时触发，一般用于初始化对象，对变量赋初值  
`__sleep()` 使用`serialize()`时自动触发  
`__wakeup()` 使用`unserialize()`时自动触发  
`__destruct()` 当一个对象被销毁时触发  
`__toString()` 当一个类被当成字符串使用时触发，不仅是echo  
`__invoke()` 当尝试以调用函数的方式调用一个对象时触发  
`__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

# 方法和属性

- 我们可以发现，把对象序列化之后的数据中并不能看到任何一个方法。

1 序列化只序列化他的属性，不序列化方法。

注：也就是说我们在利用序列化攻击的时候，也是依托类属性进行攻击。

实例参考：  
[start.php](#)  
start\_1.php

# 方法和属性

PHP设置了三种不同的类变量，`public`，`private`，`protected`，他们之间存在一些标记上的区别

- `public`无标记，变量名不变，长度不变: `s:2:"op";i:2;`
- `protected`在变量名前添加标记`\00*\00`，长度+3: ``s:5:"\00*\00op";i:2;``
- `private`在变量名前添加标记`\00(classname)\00`，长度+2+类名长度: `s:17:"\00FileHandler_Z\00op";i:2;`

具体应用时建议不要手写序列化字符串，并且配合 url编码解决不可见字符的问题

还有一个细节要注意，`private`变量和`protected`变量和`public`变量不一样，他们是受到保护的，所以我们在创建序列化对对象的时候不能再已经创造出对象了再进行赋值，这是没有权限的。看实例代码帮助理解

[variable.php](#)

`variable_poc.php`



# CTF应用

- 基础技巧
- PHP反序列化字符串逃逸
- Pop链
- Phar反序列化
- PHP-session反序列化
- PHP原生类的利用

# 基础技巧

- 绕过

- 一个很简单的绕过手法就是在数字前面加一个 + , 这样就绕过正则。正常来说正数会省略 + , 但是加上也不算错。

`0:4:"xctf":1:{s:4:"flag";s:3:"111";}` 变成 `0:+4:"xctf":1:{s:4:"flag";s:3:"111";}`

- `serialize(array(a))` a是要反序列化的对象(这样我们序列化的开头就不是O而是a,但同时又不影响作为数组元素的\$a的析构)

- **\_\_wakeup绕过** (CVE-2016-7124)

- 原理

当序列化字符串表示对象属性个数的值大于真实个数的属性时就会跳过\_\_wakeup的执行。

影响版本：PHP>5.6.25；PHP<7.0.10 #注：高版本环境下无法绕过

实例代码：  
[wakeup.php](#)  
[wakeup\\_poc.php](#)

# 基础技巧

- 利用引用绕过原理

推荐博客：[https://inhann.top/2022/05/17/bypass\\_wakeup](https://inhann.top/2022/05/17/bypass_wakeup)

#介绍一个通用的新思路，用以绕过 pop chain 构造过程中遇到的 \_\_wakeup()  
这适用于高版本的PHP中，此时传统的\_\_wakeup绕过不起作用。

原理：

```
a=1;  
b=&a;  
a=a+1;
```

那末最后b得值也会变为2，因为b是引用赋值，b存放的是a的地址，a改变。b的地址没变但是地址对应的值改变，也就是b改变了。同理改变了b也就是通过b找到a的值把它修改了，所以修改b，a也修改了。

参考博客

# PHP反序列化字符串逃逸

- 此类题目的本质就是改变序列化字符串的长度，导致反序列化漏洞
- 这种题目有两个共同点：
  - php序列化后的字符串经过了替换或者修改，导致字符串长度发生变化。
  - 总是先进行序列化，再进行替换修改操作。
- 分类：
  - 替换后字符变多
  - 替换后字符变少

# PHP反序列化字符串逃逸

---

- 过滤后字符变多

实例：[a1.php](#)

---

# pop链の利用

- 实例代码：
  - pop.php
  - pop\_poc.php



# Phar反序列化

- 我们一般利用反序列漏洞，一般都是借助unserialize()函数，不过随着人们安全的意识的提高这种漏洞利用越来越来难了，但是在今年8月份的Blackhat2018大会上，来自Secarma的安全研究员Sam Thomas讲述了一种攻击PHP应用的新方式，利用这种方法可以在不使用unserialize()函数的情况下触发PHP反序列化漏洞。漏洞触发是利用Phar:// 伪协议读取phar文件时，会反序列化meta-data储存的信息。

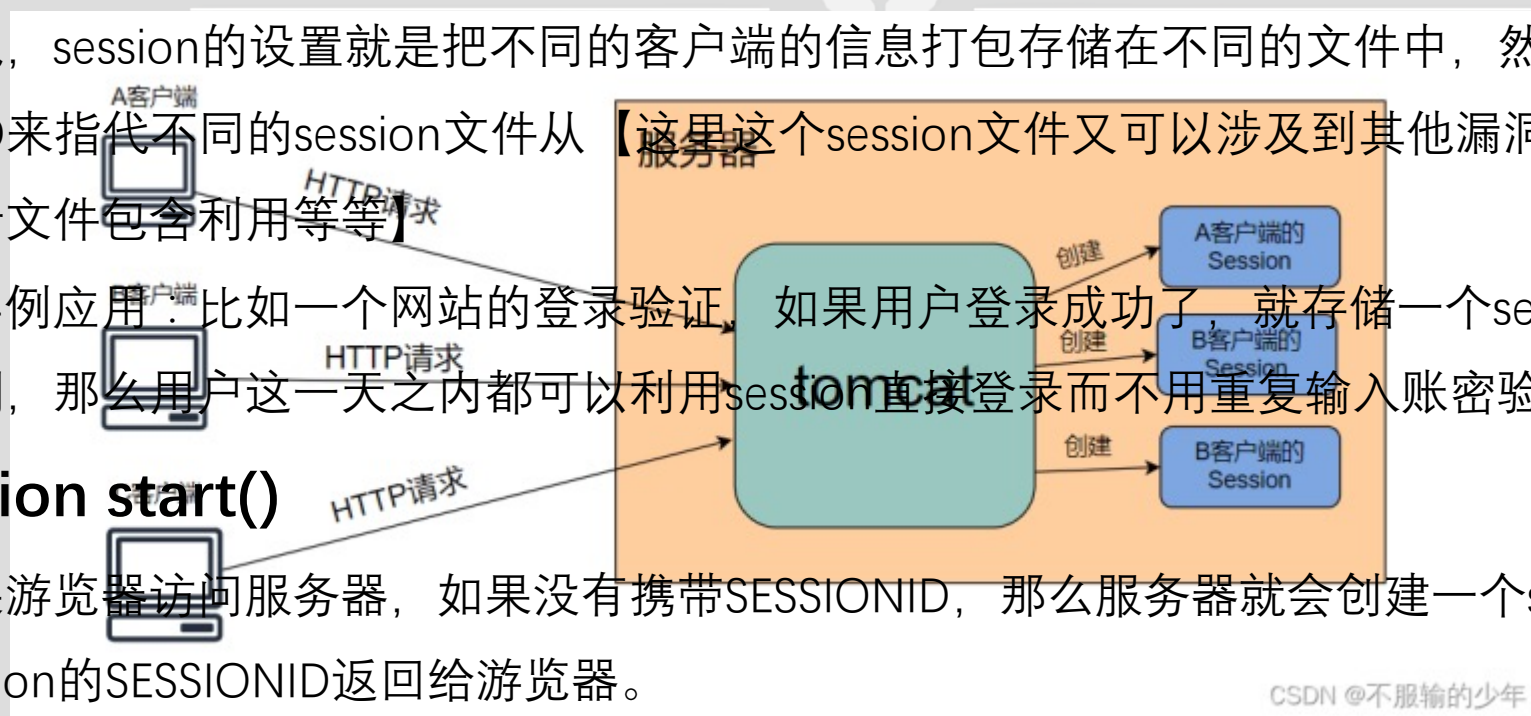
# PHP-session反序列化

简单来说，session的设置就是把不同的客户端的信息打包存储在不同的文件中，然后用cookie里面的sessionID来指代不同的session文件从【这里这个session文件又可以涉及到其他漏洞，比如利用session文件来进行文件包含利用等等】

具体的实例应用：比如一个网站的登录验证，如果用户登录成功了，就存储一个session，并且一天之内不会过期，那么用户这一天之内都可以利用session直接登录而不用重复输入账密验证。

## • Session start()

1. 如果浏览器访问服务器，如果没有携带SESSIONID，那么服务器就会创建一个session，并且把这个session的SESSIONID返回给浏览器。
2. 如果浏览器携带了SESSIONID，那么浏览器在访问时就会携带。而服务器在使用session时，就会使用这个SESSIONID的session。





# PHP-session反序列化

## 1. 流程

1. PHP脚本使用 `session_start()` 时开启 `session` 会话，会自动检测 `PHPSESSID`
  - 如果 `Cookie` 中存在，获取 `PHPSESSID`
  - 如果 `Cookie` 中不存在，创建一个 `PHPSESSID`，并通过响应头以 `Cookie` 形式保存到浏览器
2. 初始化超全局变量 `$_SESSION` 为一个空数组
3. PHP通过 `PHPSESSID` 去指定位置（`PHPSESSID` 文件存储位置）匹配对应的文件
  - 存在该文件：读取文件内容（**通过反序列化方式**），将数据存储在 `$_SESSION` 中
  - 不存在该文件：`session_start()` 创建一个 `PHPSESSID` 命名文件
4. 程序执行结束，将 `$_SESSION` 中保存的所有数据**序列化**存储到 `PHPSESSID` 对应的文件中

# PHP-session反序列化

- PHP session 反序列化机制

在php.ini中存在session.serialize\_handler配置，定义用来序列化/反序列化的**处理器**名字，默认使用php。php中的session中的内容是以文件的方式来存储的存储方式由配置项session.save\_handler确定，默认是以文件的方式存储。

# PHP-session反序列化

- 处理器(引擎)
- session.serialize\_handler是用来设置session的序列化处理器的，除了默认的PHP引擎之外，还存在其他引擎，**不同的引擎所对应的session的存储方式不相同**

PHP中session本身的序列化**机制**是没有问题的

问题出在了如果在序列化和反序列化时选择的**处理器不同**，就会带来安全问题

当使用php引擎的时候，php引擎会以|作为key和value的分隔符，对value多进行一次反序列化，达到我们触发反序列化的目的

	serialize() 函数反序列处理的值
php_serialize (php >= 5.5.4)	经过 serialize() 函数反序列处理的数组

在PHP中默认使用的是PHP引擎，如果要修改为其他的引擎，只需要添加代码  
`ini_set('session.serialize_handler', '需要设置的引擎');`

# PHP原生类的利用

- 原生类就是php内置类，不用定义php自带的类，即不需要在当前脚本写出，但也可以实例化的类。

**PHP原生类就是在标准PHP库中已经封装好的类**，而在其中，有些类具有一些功能，例如文件读取、目录遍历等，这就给了我们可乘之机，我们只需要实例化这些类，就可以实现文件读取这种敏感操作。

在CTF中，有时会遇到一些奇怪的题，比如没有给出反序列化的类，这个时候可能就需要用到PHP原生类了

## 常见的做题会遇见的类有

- Error
- Exception
- SoapClient
- DirectoryIterator
- SimpleXMLElement
- SplFileObject
- ...

## 利用方向

- 读取文件
- 构造xss
- Error绕过
- SSRF
- 获取注释内容

# PHP原生类的利用

——SSRF By SoapClient

SoapClient：PHP 的内置类 SoapClient 是一个专门用来**访问web服务的类**，可以提供一个基于SOAP协议访问Web服务的 PHP 客户端。

```
1 SoapClient {
2     /* 方法 */
3     public __construct ( string|null $wsdl , array $options = [] )
4     public __call ( string $name , array $args ) : mixed
5     public __doRequest ( string $request , string $location , string $action , int $version , bool
    $oneWay = false ) : string|null
6     public __getCookies ( ) : array
7     public __getFunctions ( ) : array|null
8     public __getLastRequest ( ) : string|null
9     public __getLastRequestHeaders ( ) : string|null
10    public __getLastResponse ( ) : string|null
11    public __getLastResponseHeaders ( ) : string|null
12    public __getTypes ( ) : array|null
13    public __setCookie ( string $name , string|null $value = null ) : void
14    public __setLocation ( string $location = "" ) : string|null
15    public __setSoapHeaders ( SoapHeader|array|null $headers = null ) : bool
16    public __soapCall ( string $name , array $args , array|null $options = null ,
    SoapHeader|array|null $inputHeaders = null , array &$outputHeaders = null ) : mixed
17 }
```

# PHP原生类的利用

- 可以看到有一个\_\_call 方法，当\_\_call 方法被触发后，它可以发送 **HTTP 和 HTTPS 请求**。正是这个 \_\_call 方法，使得 SoapClient 类可以被我们运用在 SSRF 中。
- SoapClient采用了HTTP作为底层通讯协议，XML作为数据传送的格式，其采用了SOAP协议(SOAP 是一种简单的基于 XML 的协议,它使应用程序通过 HTTP 来交换信息)。
- 其次我们知道某个实例化的类，如果去调用了一个不存在的函数，会去调用 \_\_call 方法。

# PHP原生类的利用

- 利用上述漏洞构造poc：

```
1 <?php
2 $a = new SoapClient(null,array('location'=>'http://信息.xxx.xxx.xx:xxxxx/aaa',
   'uri'=>'http://xx.xxx.xxx.xx:xxxxx'));
3 $b = serialize($a);
4 echo $b;
5 $c = unserialize($b);
6 $c->a();    // 随便调用对象中不存在的方法，触发__call方法进行ssrf
7 ?>
```

但是，由于它仅限于HTTP/HTTPS协议，所以用处不是很大。而如果这里HTTP头部还存在CRLF漏洞的话，但我们则可以通过SSRF+CRLF，利用组合拳，插入任意的HTTP头。

# PHP原生类的利用

- 为什么存在crlf的问题？

The `cache_wsdl` option is one of `WSDL_CACHE_NONE`, `WSDL_CACHE_DISK`, `WSDL_CACHE_MEMORY` or `WSDL_CACHE_BOTH`.

The `user_agent` option specifies string to use in *User-Agent* header.

The `stream_context` option is a [resource](#) for [context](#).

- 可以看到options参数中还有一个选项为user\_agent，运行我们自己设置User-Agent的值。
- CRLF 指的是回车符(CR, ASCII 13, \r, %0d) 和换行符(LF, ASCII 10, \n, %0a)，操作系统就是根据这个标识来进行换行的。CRLF就是通过注入恶意的换行来构造http请求，于是我们通过SoapClient控制user\_agent，再构造恶意的换行可以控制user\_agent后面的http请求数据。



# 综合应用

- Babyphp(2022安洵杯)
  - 官方wp : <https://bbs.kanxue.com/thread-275369.htm>
  - 考点 :  
session反序列化->soap(ssrf+crlf)->pop链->call\_user\_func激活soap类
- 博客 :
- <http://psych.green/psych/web/ctf/%e7%ac%ac%e4%ba%94%e5%b1%8a%e5%ae%89%e6%b4%b5%e6%9d%af-2022-web-writeup.html#BabyPHP>



谢谢