1.前言：

必读：

【1 Java安全基础\2 类加载】文件夹下的 **JAVA安全基础（一）--类加载器（ClassLoader）**

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

Java 反序列化 Commons-Collections 篇 04-CC3 链

---

defineClass的作用：

将byte 字节流解析成JVM能够识别的class对象，**但不实例化。**想实例化需要使用 `.newInstance` 方法

为什么需要CC3：

为了绕过一些规则对InvokerTransformer的限制，简单来说就是改变命令执行的方式,以达到绕过黑名单。

CC3的本质：

利用 com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter去加载字节码

为什么calc.java需要继承AbstractTranslet

TemplatesImpl.java 的第418行会判断是否继承了**ABSTRACT_TRANSLET**这个父类

注意 ConstantTransformer 的执行顺序：

这个语句：

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

的执行顺序：

```
new ConstantTransformer(TrAXFilter.class)
    new InstantiateTransformer(
                        new Class[] { Templates.class },
                        new Object[] { obj })
        InstantiateTransformer.tranform(TrAXFilter);
            con = TrAXFilter.getConstructor(Templates.class);
                con.newInstance(obj);
```

简易流程：

- InstantiateTransformer实现了Transformer接口的类，他的**作用就是调用构造方法**
- TemplatesImpl.newTransformer()方法用于执行传入字节码的类构造器。

利用 **org.apache.commons.collections.functors.InstantiateTransformer** 来调用到 **com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilter** 的构造方法，再利用**TrAXFilterd**的构造方法里的 templates.newTransformer()

（直接免去了我们使用InvokerTransformer手工调用 newTransformer() 方法这一步【参考：CC1链】）

调用到 TemplatesImpl 里的字节码。

CC1,CC3链子图：

直接引入 Drun1baby 师傅的图：