

# java应用破解之破解 jeb mips 2.3.3

## 前言

由于要去学习路由器相关的漏洞利用技术的学习，而许多的路由器都是 mips 架构的，IDA 又不能反编译 mips，发现 jeb 的新版支持 mips 的反编译，于是去官网申请一个试用版，试用版的限制还是比较多的，比如 使用时间验证，没法复制粘贴 等，于是想尝试看看能否破解，同时填了 java 破解 这个坑。

本文主要记录的是破解过程中的思路和使用的一些工具，技巧。文末有处理后的数据。

## 正文

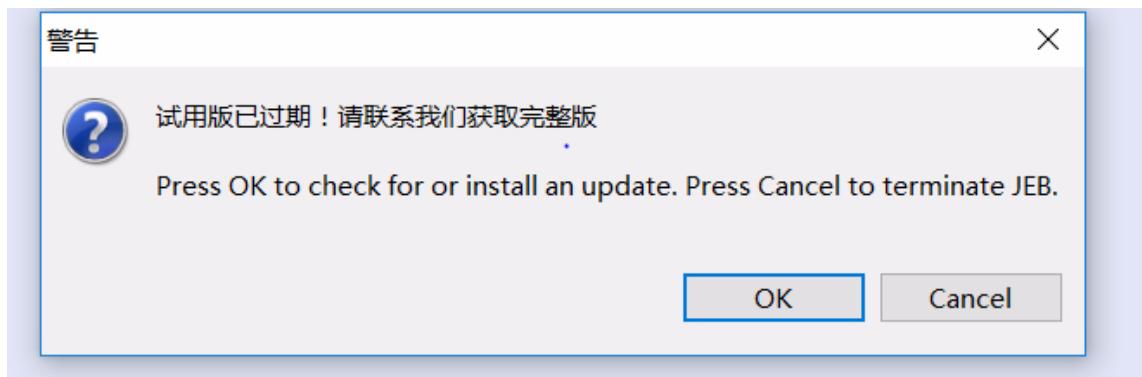
jeb 的主要逻辑在 jeb.jar 中，该文件会在程序运行起来后释放到程序目录中的其中一个子目录下，使用 Everything 搜 jeb.jar 就可以找到文件的位置。找到文件后就可以逆向分析了。本文重点不在逆向这方面，而是要介绍我破解这个软件的一个大概的流程。

下面介绍几个在整个流程中起到重要作用的工具。

- Btrace ----> hook java 系统函数，打印堆栈，找关键方法
- javassist ----> 修改字节码
- IDEA -----> 动态调试 jar 包

试用版的一个最无语的限制就是必须要联网才能使用，不联网就会直接退出了，就是如此暴力。但是这对我们来说则是绝佳的条件。我们可以使用 Btrace hook java.lang.System.exit 函数，然后打印堆栈信息，就可以定位到在退出前所调用的方法，一般来说，在方法之间肯定会有离关键方法很近的方法，或者直接就是我们要找的目标方法。

这个是之前破解的，现在我重新测试时，提示 超过试用期，然后就退出了。



不管怎样有异常就好，然后 hook java.lang.System.exit 打印堆栈信息就可以看到一些 jeb 自己写一些方法的信息了。

Btrace 脚本如下

```
import com.sun.btrace.annotations.*;
import static com.sun.btrace.BTraceUtils.*;

@BTrace
public class TraceHelloWorld {
    @OnMethod
    (clazz = "java.lang.System", method = "exit")
    public static void Trace_exit()
    {
        println("jstack() is :");
        println("[");
        jstack();
        println("]");
    }
}
```

```
}
```

```
F:\hackworkplace\java\jeb_reverse
\ bttasklist.exe | grep j
java.exe          5152 Console      1    102,928 K
java.exe          5008 Console      1    105,620 K
javaw.exe        4412 Console      1    271,036 K

F:\hackworkplace\java\jeb_reverse
\ tbtrace 4412 TraceHelloWorld.java
jstack() is :
[
java.lang.System.exit(Unknown Source)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
java.lang.reflect.Method.invoke(Unknown Source)
com.pnfsoftware.jeb.client.AbstractContext.terminate(SourceFile:281)
com.pnfsoftware.jeb.rcpclient.RcpClientContext$AppStartupCompleteEventHandler.handleEvent(RcpClientContext
org.eclipse.e4.ui.services.internal.events.UIEventHandler$1.run(UIEventHandler.java:40)
org.eclipse.swt.widgets.RunnableLock.run(RunnableLock.java:35)
org.eclipse.swt.widgets.Synchronizer.runAsyncMessages(Synchronizer.java:182)
org.eclipse.swt.widgets.Display.runAsyncMessages(Display.java:4211)
org.eclipse.swt.widgets.Display.readAndDispatch(Display.java:3827)
org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine$4.run(PartRenderingEngine.java:1121)
org.eclipse.core.databinding.observable.Realm.runWithDefault(Realm.java:336)
org.eclipse.e4.ui.internal.workbench.swt.PartRenderingEngine.run(PartRenderingEngine.java:1022)
org.eclipse.e4.ui.internal.workbench.E4Workbench.createAndRunUI(E4Workbench.java:150)
org.eclipse.e4.ui.internal.workbench.swt.E4Application.start(E4Application.java:161)
org.eclipse.equinox.internal.app.EclipseAppHandle.run(EclipseAppHandle.java:196)
org.eclipse.core.runtime.internal.adaptor.EclipseAppLauncher.runApplication(EclipseAppLauncher.java:134)
```

这里有个小问题,如果你是通过运行jeb\_wincon.bat 或者 jeb.exe来启动jeb的话你是看不到他开启的 java进程的, 所以可以使用 everything 搜索org.eclipse.equinox.launcher\*.jar

org.eclipse.equinox.launcher*.jar	
Name	Path
org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar	F:\\$RECYCLE.BIN\\$-1-5-21-4084953826-1610909797-2112784352-500\RN9060M.201708221725
org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar	F:\\$RECYCLE.BIN\\$-1-5-21-4084953826-1610909797-2112784352-500\RZYXQDE.201708221725
org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar	F:\jeb\jeb-2.3.3\jeb-demo-mips-2.3.3\bin\plugins
org.eclipse.equinox.launcher_1.3.201.v20161025-1711.jar	F:\jeb\jeb-2.3.3\原版\jeb-demo-mips-2.3.3.201708221725-JEBDecompilerDemo-12182046498738

然后运行那个 jar 包就可以正常的找到 jeb启动的 java 进程了, 这样我们才可以使用 Btrace 脚本进行 hook. 至于为什么是这样的, 我也不记得当初是怎样找到的。可以去逆向 jeb. exe 或者 看使用 org.eclipse.equinox 开发的教程可以弄清楚。其实通过 Btrace 然后配合着静态分析就可以解决这个软件了我认为。

Jeb里面会使用一个函数对字符串进行加密, 所以在逆向的时候会很不方便, 当初我是用 IDEA 调试它, 然后 在 IDEA 的调试环境里面, 调用解密函数(使用IDEA的自带的功能), 把加密后的字符串解密后, 然后再分析的。

The screenshot shows the JBoss Seam IDE environment. The top part displays a Java code editor with several methods and variables:

```
public static String dl(String var0) {
    return var0;
}

public static String dl(byte[] var0, int var1, int var2) { var0: [41, 14, 6, 24, 92, 5, 6, 6, 19, 14, + 5 more] var1: 2 var2: 169
}
}

GN(String var1) {
    this.dl = var1;
}

public GN(byte[] var1, int var2, int var3) {
    this.sC = var1;
```

A red circle highlights the call to `(new GN(var0, var1, var2)).dl()` in the code editor.

The bottom part shows the debugger interface with a "Variables" tab open, displaying the values of `var0`, `var1`, and `var2`.

A large red circle highlights the "Evaluate Expression" dialog box, which is open over the debugger interface. The dialog box contains the expression `(new GN(var0, var1, var2)).dl()` and shows the result as `result = "java.lang.Class"`. A red arrow points from the highlighted code in the editor up to this dialog box.

使用IDEA调试其实非常简单，我们只需要先新建一个 project，然后把相关的jar包添加到 Project 的 lib，然后调用 jar 包中的函数即可。比如

```
> org.eclipse.equinox.ds_1.4.400.v20160226-2036.jar lib import javassist.CtNewMethod;
> org.eclipse.equinox.event_1.3.200.v20160324-1850.jar lib import com.pnfsoftware.jeb.installer.*;
> org.eclipse.equinox.frameworkadmin.equinox_1.0.700.v20160102-2223.jar library root libncher.Main;
> org.eclipse.equinox.frameworkadmin_2.0.300.v20160509 lib import com.pnfsoftware.jeb.client.Licensing;
> org.eclipse.equinox.launcher_1.3.201.v20161025-1712 lib
> org.eclipse.equinox.p2.artifact.repository_1.1.500.v20160404 lib
> org.eclipse.equinox.p2.core_2.4.100.v20160419-0834 lib
> org.eclipse.equinox.p2.director.app_1.0.500.v20160404 lib
> org.eclipse.equinox.p2.director_2.3.300.v20160504-1 lib
> org.eclipse.equinox.p2.engine_2.4.100.v20160419-0834 lib
> org.eclipse.equinox.p2.garbagecollector_1.0.300.v20160404 lib
> org.eclipse.equinox.p2.jarprocessor_1.0.500.v20160504 lib
> org.eclipse.equinox.p2.metadata.repository_1.2.300.v20160404 lib
> org.eclipse.equinox.p2.metadata_2.3.100.v20160427 lib
> org.eclipse.equinox.p2.publisher.eclipse_1.2.100.v20160404 lib
> org.eclipse.equinox.p2.publisher_1.4.100.v20160504-1 lib
> org.eclipse.equinox.p2.repository.tools_2.1.300.v20160404 lib
> org.eclipse.equinox.p2.repository_2.3.200.v20160421 lib
> org.eclipse.equinox.p2.touchpoint.eclipse_2.1.400.v20160404 lib
> org.eclipse.equinox.p2.touchpoint.osgi_2.1.600.v20160504 lib
> org.eclipse.equinox.p2.touchpoint.osgi_2.1.600.v20160504 lib

public class JebCracker {

    public static void main(String[] args) throws Exception {
        // com.pnfsoftware.jeb.installer.Launcher.main(new String[] {"-di"});
        // DES.main(args);
        // Launcher.main(new String[] {"-generate-key"});
        CtClass.debugDump = "./debugDump/";

        System.out.println(Licensing.allowAnyClient());
    }
}

Main.main(args)
```

信息比较杂，看我画圈的那段代码即可。然后进入想要下断点的位置，正常的下个断点就可以了。

比如我们已经知道，程序权限校验的关键逻辑在 jeb.jar 中，我们直接调用 jeb.jar 中的 main 方法，然后进去调试里面的代码即可

```

Decompiled .class file, bytecode version: 51.0 (Java 7)

int var7 = var0.length; var7 (slot_7): 1

for(int var8 = 0; var8 < var7; ++var8) { var8 (slot_8): 0 var7 (slot_7): 1
    String var9 = var6[var8]; var9 (slot_9): "--generate-key" var6 (slot_6): {"--generate-key".
    if (!var9.equalsIgnoreCase( anotherString: "-c")) {
        if (var9.equalsIgnoreCase( anotherString: "--license")) {
            System.out.println("JEB Core v" + AbstractClientContext.app_ver);
            System.out.println(AbstractClientContext.generateLicenseInformation());
            System.exit( status: 0);
        } else if (var9.equalsIgnoreCase( anotherString: "--generate-key")) { var9 (slot_9): "
            LF var10 = new LF(); var10 (slot_10): LF@1115
            var10.initialize(var0); var0: {"--generate-key...
            var10.start(); var10 (slot_10): LF@1115
            var10.stop();
        } else {
            System.out.format("JEB Core v%s - PNF Software, Inc. (c) 2015-2016\n", AbstractClientContext.app_ver);
            System.out.println("");
            System.out.println("Standard options:");
        }
    }
}

```

Main

Variables

- static members of Launcher
- Variables debug info not available
- var0 = [String[1]@1113]
- var1 (slot\_1) = null
- var2 (slot\_2) = 0
- var3 (slot\_3) = null

赏心悦目的调试，美滋滋。分析或者调试 jeb.jar，就可以找到字符串加密的那个方法。

```

Decompiled .class file, bytecode version: 51.0 (Java 7)

package com.pnfsoftware.jebglobal;

import java.nio.charset.Charset;

public class GN {
    private String dL;
    private byte[] sC;
    private int gM;
    private int At;

    public static String dL(String var0) {
        return var0;
    }

    public static String dL(byte[] var0, int var1, int var2) {
        return (new GN(var0, var1, var2)).dL();
    }
}

```

如果没有目标，我们可以使用 Btrace hook 这个函数，打印他的返回值，就可以看到程序中各种被解密后的字符串了。脚本如下

```

import static com.sun.btrace.BTraceUtils.println;
import static com.sun.btrace.BTraceUtils.str;
import static com.sun.btrace.BTraceUtils.strcat;

```

```

import static com.sun.btrace.BTraceUtils.currentTimeMillis;
import static com.sun.btrace.BTraceUtils.jstack;

import com.sun.btrace.annotations.BTrace;
import com.sun.btrace.annotations.Kind;
import com.sun.btrace.annotations.Location;
import com.sun.btrace.annotations.OnMethod;

@BTrace
public class TraceHelloWorld {
    @OnMethod
    (clazz = "com.pnfsoftware.jebglobal.GN", method = "dL")
    public static void Trace_exit()
    {
        println("ret is :   ");
        println("[");
        jstack();
        println("]");
        println("-----");
        println("-----");
        println("-----");
        println("-----");
    }
}

```

经过各种翻看代码，调试， Hook，终于找到一些可能是比较关键的函数，我们该怎么办呢？这时可以使用 javassist 来修改目标方法。

比较懒，把破解 JEB 期间的所有代码都放到一个 函数里面了，做了一定的注释。

```

package me.hacklh;

import com.pnfsoftware.jeb.Launcher;
import javassist.ClassPool;
import javassist.CtClass;
import javassist.CtMethod;
import javassist.CtNewMethod;
import com.pnfsoftware.jeb.installer.*;
import org.eclipse.core.launcher.Main;
import com.pnfsoftware.jeb.client.Licensing;

public class JebCracker {

    public static void main(String[] args) throws Exception {
        // com.pnfsoftware.jeb.installer.Launcher.main(new String[]{"--di"});
        // DES.main(args);
        Launcher.main(new String[] {"--generate-key"});
        CtClass.debugDump = "./debugDump/";
        System.out.println(Licensing.allowAnyClient());
    }
}

```

```
// Main.main(args);

/**
 * 修改安装时的校验，避免去下载网络安装文件，直接使用我们事先下好的文件就行
 */
ClassPool pool = ClassPool.getDefault();
pool.importPackage("com.pnfsoftware.jeb.installer");
CtClass old_class = pool.get("com.pnfsoftware.jeb.installer.Package");
old_class.detach();
CtMethod old_method = old_class.getDeclaredMethod
(
    "verifyData",
    new CtClass[]
    {
        pool.get(byte[].class.getName()),
    }
);
old_method.setBody("return true;");
old_class.writeFile();
/***
 * 修改getStatus， AbstractContext会起几个线程修改status
 */
pool = ClassPool.getDefault();
pool.importPackage("com.pnfsoftware.jeb.client.AbstractContext");
old_class = pool.get("com.pnfsoftware.jeb.client.AbstractContext");
old_class.detach();
old_method = old_class.getDeclaredMethod
(
    "getStatus",
    new CtClass[]
    {
    }
);
old_method.setBody("return 0;");

old_method = old_class.getDeclaredMethod
(
    "terminate",
    new CtClass[]
    {
    }
);
old_method.setBody(";");
old_class.writeFile();

/***
 * internet 检测
 */
pool = ClassPool.getDefault();
pool.importPackage("com.pnfsoftware.jebglobal.tB");
old_class = pool.get("com.pnfsoftware.jebglobal.tB");
```

```
old_class.detach();
old_method = old_class.getDeclaredMethod

(
    "dL",
    new CtClass[]
    {
        pool.get(boolean.class.getName()),
    }
);

old_method.setBody("return true;");

old_method = old_class.getDeclaredMethod
(
    "run",
    new CtClass[]
    {
    }
);

old_method.setBody(";");
old_class.writeFile();

/***
 * 增加许可证的过期时间
 */
pool = ClassPool.getDefault();
pool.importPackage("com.pnfsoftware.jeb.client.Licensing");
old_class = pool.get("com.pnfsoftware.jeb.client.Licensing");
old_class.detach();

old_method = old_class.getDeclaredMethod
(
    "getExpirationTimestamp",
    new CtClass[]
    {
    }
);

old_method.setBody("return real_license_ts + 345600000;");

old_method = old_class.getDeclaredMethod
(
    "isInternetRequired",
    new CtClass[]
    {
    }
);

old_method.setBody("return false;");

old_method = old_class.getDeclaredMethod
(
    "isFullBuild",

```

```
        new CtClass[]
        {
        }

);

old_method.setBody("return true;");

old_method = old_class.getDeclaredMethod
(
    "canUseCoreAPI",
    new CtClass[]
    {
    }

);

old_method.setBody("return true;");

old_method = old_class.getDeclaredMethod
(
    "canUseCoreAPI",
    new CtClass[]
    {
    }

);

old_method.setBody("return true;");

old_class.writeFile();

/***
 * patch 掉与网络下载有关的函数，禁止升级
 */
pool = ClassPool.getDefault();
pool.importPackage("com.pnfsoftware.jeb.util.net.Net");
old_class = pool.get("com.pnfsoftware.jeb.util.net.Net");
old_class.detach();
old_method = old_class.getDeclaredMethod

(
    "downloadBinary",
    new CtClass[]
    {

        pool.get(String.class.getName())
    }

);

old_method.setBody("return null;");

old_method = old_class.getDeclaredMethod
(
    "httpPost",
    new CtClass[]
    {

        pool.get(String.class.getName()),
        pool.get(String.class.getName()),
        pool.get(long[].class.getName())
    }

);

old_method.setBody("return null");
```

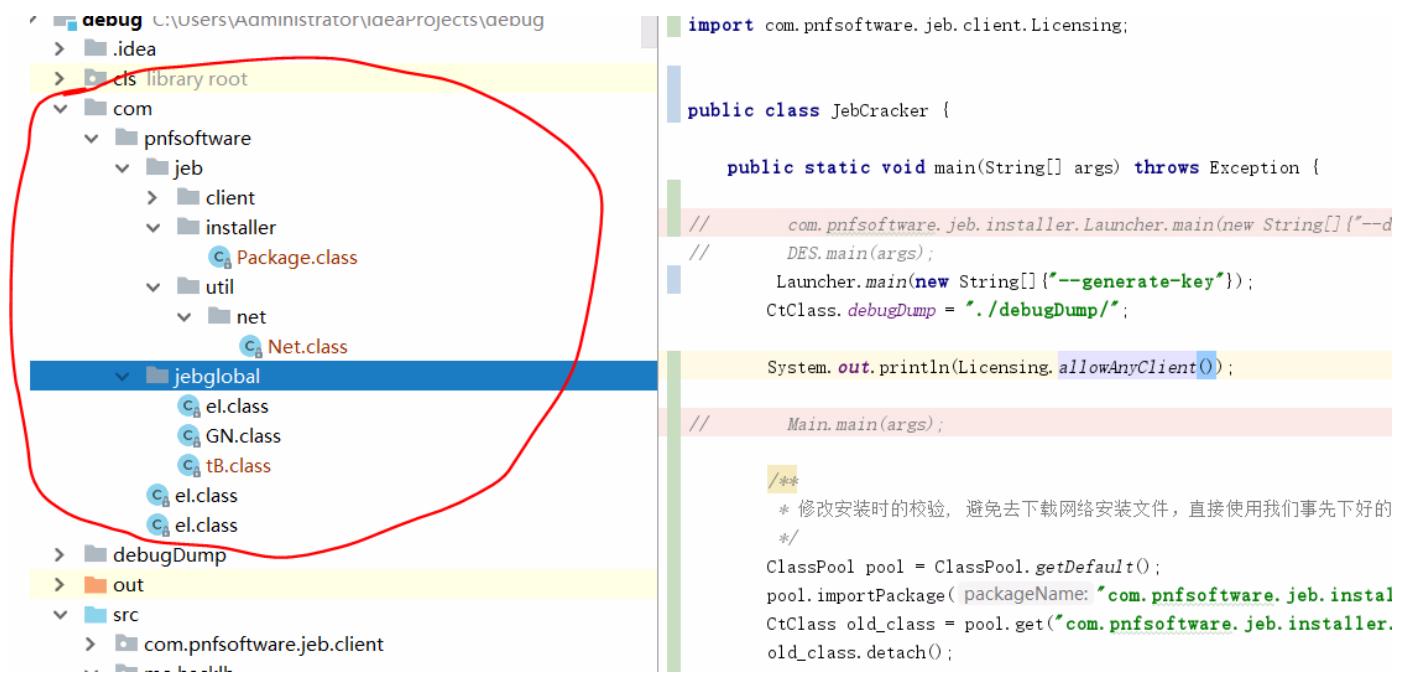
```

        old_class.writeFile();
    }

}

```

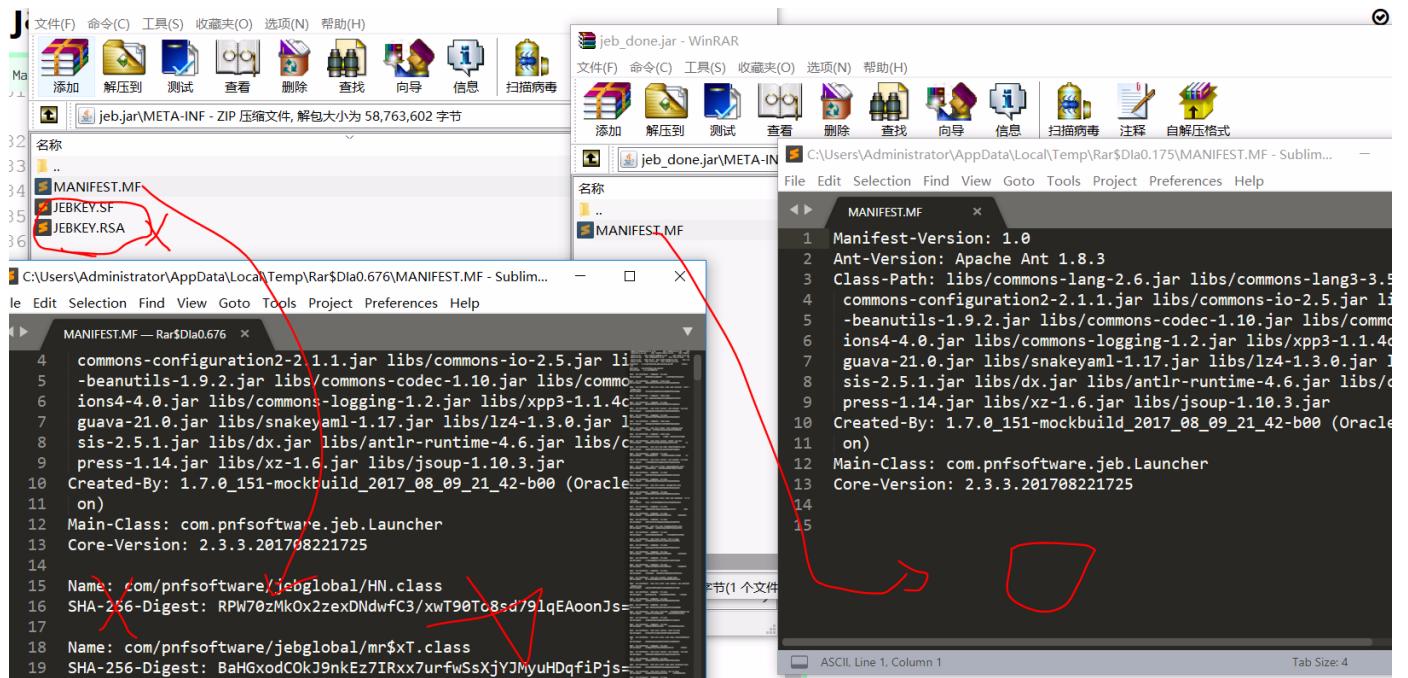
运行后会在工程目录生成一个文件夹，以你修改的类名为目录结构。



把这些 class文件替换到来的 jar 包里面就可了。

可以使用 winrar 把 jar 包打开，找到对应目录，拖进去替换就行了。

替换之后要去 META-INF 删掉一些东西。具体看下图



这样就完成了jar包的修改。

最后说下静态分析jar包的工具，使用 JEB 就可以。首先把jar 转换为 dex。

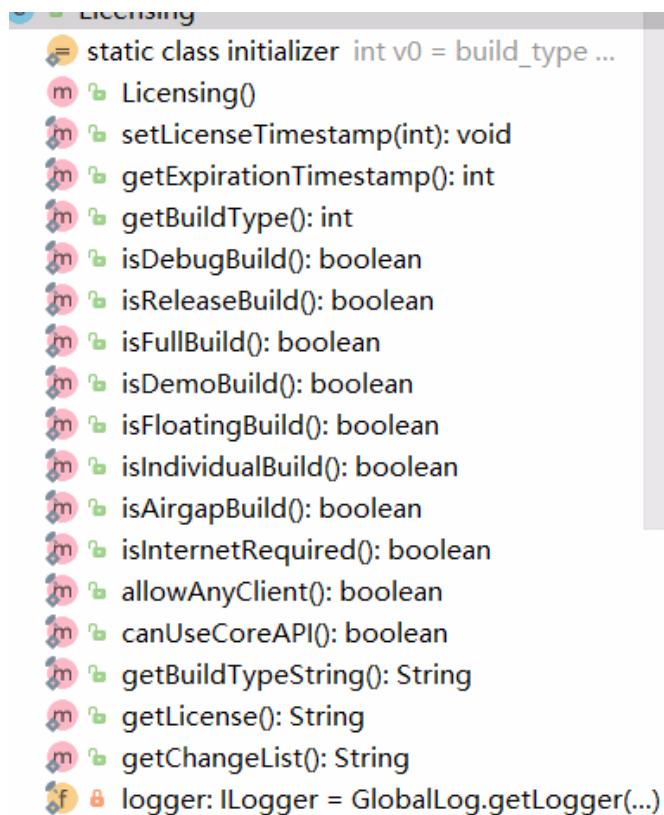
```
dx.bat --dex --output=d:\dst.dex src.jar
```

然后拿起jeb分析就行了。

## 最后

如果你看到了这里，并且按我前面所说的方式一步一步破解了jeb，那么恭喜你和我一样被坑了。弄得差不多的时候，我发现有一个神奇的类。

com.pnfsoftware.jeb.client.Licensing



```
public static final int license_ts = 0;
public static final int license_validit
public static int real_license_ts = 0;
public static int build_type = 0;
public static final int FLAG_AIRGAP = 8
public static final int FLAG_ANYCLIENT =
public static final int FLAG_COREAPI =
public static final int FLAG_DEBUG = 1;
public static final int FLAG_FLOATING =
public static final int FLAG_FULL = 2;
public static final int FLAG_JEB2 = 128

public Licensing() {
}

public static final void setLicenseTime
public static final int getExpirationTi
public static final int getBuildType()
```

瞬间爆炸，我们只要修改这里的函数的返回值，或者直接重写这个类，就可以基本搞定这款软件了。52破解上的jeb 2.2.7 中延长使用时间就是修改的这个类的方法（后面才看的，悲伤~）

```
import javassist.ClassPool;
import javassist.CtClass;
import javassist.CtMethod;

public class Loader
{
    public static void premain(String agentOps, Instrumentation inst)
    {
        inst.addTransformer(new ClassFileTransformer()
        {
            public byte[] transform(ClassLoader loader, String className, Class<?> classBeingRedefined, ProtectionDomain protectionDomain, byte[] bytecode) throws IllegalClassFormatException
            {
                className = className.replace("/", ".");
                if (className.equals("com.pnfsoftware.jeb.client.Licensing"))
                {
                    try
                    {
                        ClassPool pool = ClassPool.getDefault();
                        CtClass ctClass = pool.makeClass(new ByteArrayInputStream(classfileBuffer));
                        CtMethod a = ctClass.getDeclaredMethod("getExpirationTimestamp", null);
                        System.out.println("loader加载成功100%");
                        a.setBody("return 2000000000;");
                        return ctClass.toBytecode();
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                }
                else if (className.equals("com.pnfsoftware.jeb.client.AbstractClientContext"))
                {
                    ClassPool pool = ClassPool.getDefault();
                    CtClass ctClass = pool.makeClass(new ByteArrayInputStream(classfileBuffer));
                    CtMethod a = ctClass.getDeclaredMethod("startFloatingClient", null);
                    System.out.println("loader加载成功50%");
                    a.setBody("return;");
                    return ctClass.toBytecode();
                }
            }
        });
    }
}
```

编译后 Class文件，[点我](#)，用它去替换jeb.jar中的相应文件即可。具体替换方法，文中有介绍。

分析过程的一些笔记

### 2.3.3

com.pnfsoftware.jebglobal.cF 用于获取serial, uuid 生成字符串

.At--> get uuid  
.GQ--> get serial number  
.dL-----> get\_md5、

com.pnfsoftware.jebglobal.eI sc方法会检测运行时间，定时退出 校验运行时间 patch

com.pnfsoftware.jeb.client.Licensing licensing 校验 ， 修改该类的方法的返回值可以拿到大量的结果

com.pnfsoftware.jebglobal.Wr 重要函数，程序初始化，保存功能

### 分析过程中的另外的 Btrace脚本

```
/* BTrace Script Template */  
import com.sun.btrace.annotations.*;  
import static com.sun.btrace.BTraceUtils.*;  
  
/*  
  
@BTrace  
public class TracingScript {  
  
    @OnMethod(  
        clazz = "com.pnfsoftware.jebglobal.Wr",  
        method = "saveProject")  
    public static void traceExecute() {  
        jstack();  
        println(strcat("-----:--\n", "*****\n"));  
    }  
  
}  
  
@BTrace  
public class TracingScript {  
  
    @OnMethod(  
        clazz = "com.pnfsoftware.jebglobal.qI",  
        method = "getKey",  
        location=@Location(Kind.RETURN)  
    )  
    public static void traceExecute(@Self com.pnfsoftware.jebglobal.qI object, @Return String result){  
        println(strcat("ret: ", str(result)));  
        jstack();  
        println(strcat("-----:--\n", "*****\n"));  
    }  
  
}
```

```
@BTrace
public class TracingScript {

    @OnMethod(
        clazz = "com.pnfsoftware.jebglobal.GN",
        method = "dL",
        location=@Location(Kind.RETURN)
    )
    public static void traceExecute(@Self com.pnfsoftware.jebglobal.GN object, byte[] var0, int var1, int var2, @Return String result) {
        println(strcat("ret: ", str(result)));
        jstack();
        println(strcat("-----:--\n", "*****\n"));
    }
}
```

来源: <https://www.cnblogs.com/hac425/p/9416950.html>