前面几篇文章讲了字节码、javaagent、ASM，有了这些储备知识，终于可以开始讲无痛破解这个主题了。整个过程的核心思想还是通过 javaagent 和 asm 来进行字节码改写。这篇文章将以 censum、gceasy 两款软件为例进行破解过程的讲解。

0x01 换个花样破解 censum

我们要做的事情就是

```
private static int getYear() {
    return 2016;
}
改为
private static int getYear() {
    return 2026;
}
```

对应的字节码是

```
SIPUSH 2016
IRETURN
```

我们把方法getYear注入一下就可以了，相当于强行插入 return 2026;，插入这行在 Java 语言无法编译通过，但是在字节码层面完全没有任何问题。

```
private static int getYear() {
    // 强行插入
    return 2026;
    return 2016;
}
```

我们来看下具体的代码

```
// javaagent的入口
public class AgentMain {
    public static void premain(String
agentArgument, Instrumentation instrumentation)
throws Exception {
        instrumentation.addTransformer(new
MyClassFileTransformer(), true);
    }
}

public class MyClassFileTransformer implements
ClassFileTransformer {
    @Override
    public byte[] transform(ClassLoader loader,
String className, Class<?> classBeingRedefined,
ProtectionDomain protectionDomain, byte[]
classBytes) throws IllegalClassFormatException {
        // 只注入 CensumStartupChecks 类
        if
(className.equals("com/jclarity/censum/CensumStar
tupChecks")) {
            ClassReader cr = new
ClassReader(classBytes);
            ClassWriter cw = new ClassWriter(cr,
ClassWriter.COMPUTE_FRAMES);
```

```
            ClassVisitor cv = new
MyClassVisitor(cw);
            cr.accept(cv, ClassReader.SKIP_FRAMES
| ClassReader.SKIP_DEBUG);
            byte[] bytes = cw.toByteArray();
            // 把转换以后的文件写入到文件中，方便进行检
查是否改写正确
            writeByteArrayToFile(bytes, new
File("./CensumStartupChecks-modify.class"));
            return bytes;
        }
        return classBytes;
    }

    public static class MyClassVisitor extends
ClassVisitor {
        @Override
        public MethodVisitor visitMethod(int
access, String name, String desc, String
signature, String[] exceptions) {
            MethodVisitor mv =
super.visitMethod(access, name, desc, signature,
exceptions);
            // 只改写 getYear 函数
            if (name.equals("getYear")) {
                return new MyMethodVisitor(mv,
access, name, desc);
            }
            return mv;
        }
    }

    public static class MyMethodVisitor extends
```

```
AdviceAdapter {
        @Override
        protected void onMethodEnter() {
            // 插入 return 2016;
            mv.visitIntInsn(SIPUSH, 2026);
            mv.visitInsn(IRETURN);
        }
    }
}
```

执行mvn clean package 编译出 my-javaagent.jar

执行java -javaagent:/path/to/my-javaagent.jar -jar censum-full.jar，发现已经成功地绕过了过期检查

我们来对比一下改写前后的类文件javap -v -p CensumStartupChecks.class vs
javap -v -p CensumStartupChecks-modify.class
改写前

```
private static int getYear();
descriptor: ()I
flags: ACC_PRIVATE, ACC_STATIC
Code:
  stack=1, locals=0, args_size=0
     0: sipush        2016
     3: ireturn
  LineNumberTable:
    line 37: 0
```

改写后

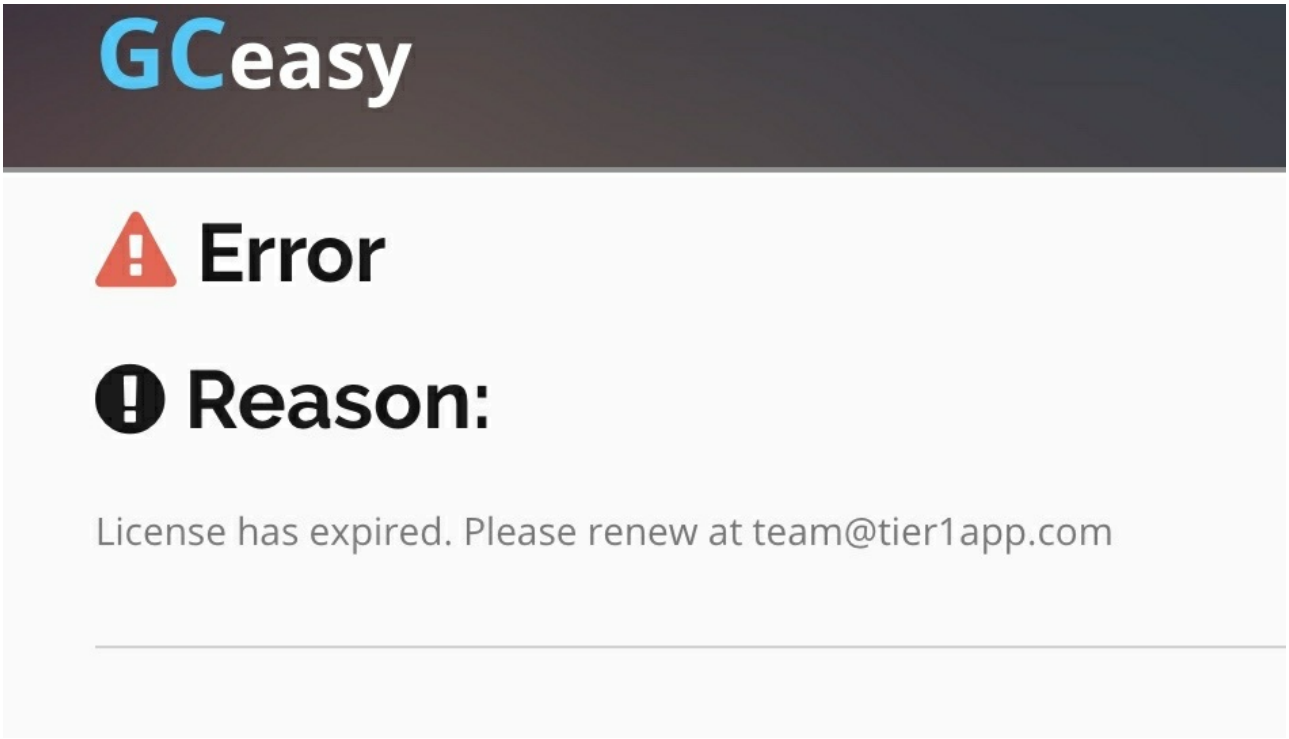
```
private static int getYear();
descriptor: ()I
flags: ACC_PRIVATE, ACC_STATIC
Code:
  stack=1, locals=0, args_size=0
     0: sipush        2026
     3: ireturn
     4: nop
     5: nop
     6: nop
     7: athrow
```

可以看到改写以后的字节码，加载 2026 然后直接 return 了

0x02 破解 gceasy

前面破解 censum 分别使用了直接修改字节码打包和 javaagent 动态注入的方法，今天再来介绍一个软件的破解，大名鼎鼎的分析 gc 日志和 jstack 线程日志的工具 [gceasy (http://gceasy.io/)](http://gceasy.io/) 和 [fastthread (http://fastthread.io/)](http://fastthread.io/)。

这里只供大家学习测试，掌握一些方法，请勿在正式环境使用。



通过 jdgui 工具来看 war 包的代码，发现 license 处理是放在 `com.websina.license.LicenseManagerImpl`中的

```
public boolean isValid() throws
GeneralSecurityException
{
    // 我们要做的事情非常简单，在这里强行字节码插入
return true; 就可以了
    String signature = this.lic.getSignature();
    if ((signature == null) ||
(signature.trim().length() == 0)) {
      return false;
    }
    boolean valid =
SignatureUtil.verify(this.lic.format(),
ByteHex.convert(signature), this.key);
    if (!valid) {
      return false;
    }
    if (daysLeft() < 0) {
      return false;
    }
    return true;
}
```

首先需要弄清楚return true;对应的字节码是什么

```
private boolean foo() {
    return true;
}
对应于
ICONST_1
IRETURN
```

改写为 ASM 的语句就是

```
mv.visitInsn(ICONST_1);
mv.visitInsn(IRETURN);
```

有了上一篇文章的铺垫，我们来直接看实现的代码

```
public class MyClassFileTransformer implements
ClassFileTransformer {
    @Override
    public byte[] transform(ClassLoader loader,
String className, Class<?> classBeingRedefined,
ProtectionDomain protectionDomain, byte[]
classBytes) throws IllegalClassFormatException {
        // 只注入 LicenseManagerImpl 类
        if
(className.equals("com/websina/license/LicenseMan
agerImpl")) {
            ClassReader cr = new
ClassReader(classBytes);
            ClassWriter cw = new ClassWriter(cr,
ClassWriter.COMPUTE_FRAMES);
            ClassVisitor cv = new
MyClassVisitor(cw);
            cr.accept(cv, ClassReader.SKIP_FRAMES
| ClassReader.SKIP_DEBUG);
            byte[] bytes = cw.toByteArray();
            return bytes;
        }
        return classBytes;
    }

    public static class MyClassVisitor extends
ClassVisitor {
        @Override
```

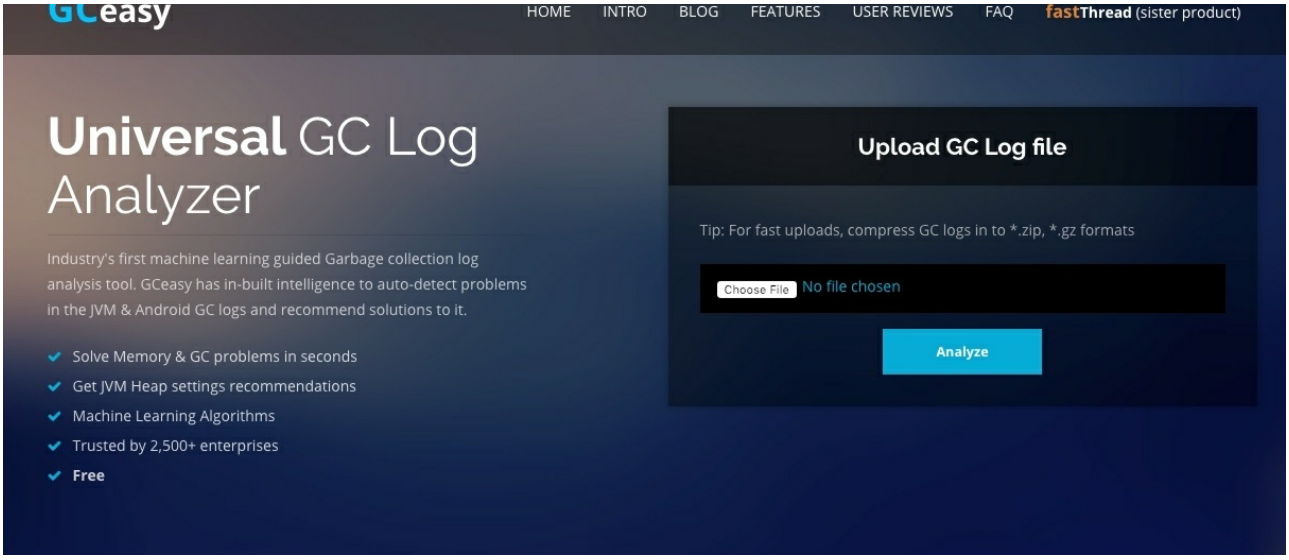
```
        public MethodVisitor visitMethod(int
access, String name, String desc, String
signature, String[] exceptions) {
            MethodVisitor mv =
super.visitMethod(access, name, desc, signature,
exceptions);
            // 只注入 isValid 函数
            if (name.equals("isValid")) {
                return new MyMethodVisitor(mv,
access, name, desc);
            }
            return mv;
        }
    }

    public static class MyMethodVisitor extends
AdviceAdapter {
        @Override
        protected void onMethodEnter() {
            // 强行插入 return true;
            mv.visitInsn(ICONST_1);
            mv.visitInsn(IRETURN);
        }
    }
}
```

mvn clean package生成my-javaagent.jar

然后修改一下启动命令加上 javaagent 参数

```
java -javaagent:/path/to/my-javaagent.jar -
DuploadDir=. -jar webapp-runner-8.0.33.4.jar --
port 8080 gceasy.war
```

大功告成，就是这么简单



0x03 小结

这篇文章，我们讲解了如何通过 javaagent 和 ASM 的方式来破解 censum 和 gceasy 两款软件，回顾一下重点：要通过反编译工具找到相关的 license 检查函数在哪里，然后通过 javaagent 的 premain 函数在类加载之前动态修改字节码，绕过 license 检查机制。

0x04 思考

留一道作业：读者反馈发现新版本的 censum 的 license 检查方式做了修改，这种方式已经不适用，你能通过我们介绍的工具和方法搞定新版本的 censum 破解吗?

另外留一个思考题，如果你去做一个商业版本的软件，有哪些手段可以防止别人用类似的手段破解呢?

欢迎你在留言区留言，和我一起讨论。