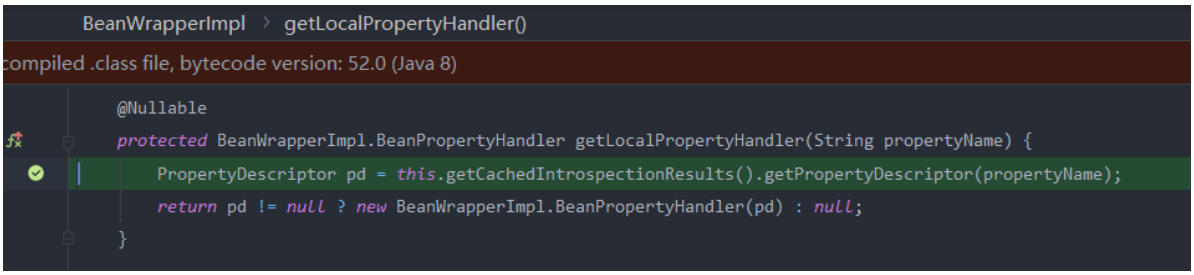
简介

该洞算是对`CVE-2010-1622`的绕过,关于`CVE-2010-1622`可以看:http://rui0.cn/archives/1158

比较关键的一点:

> 成员是public或者提供set方法，否则是不能赋值的。但是数组类型的成员在非public且没有提供set方法的情况下，可以通过这种方式被赋值。

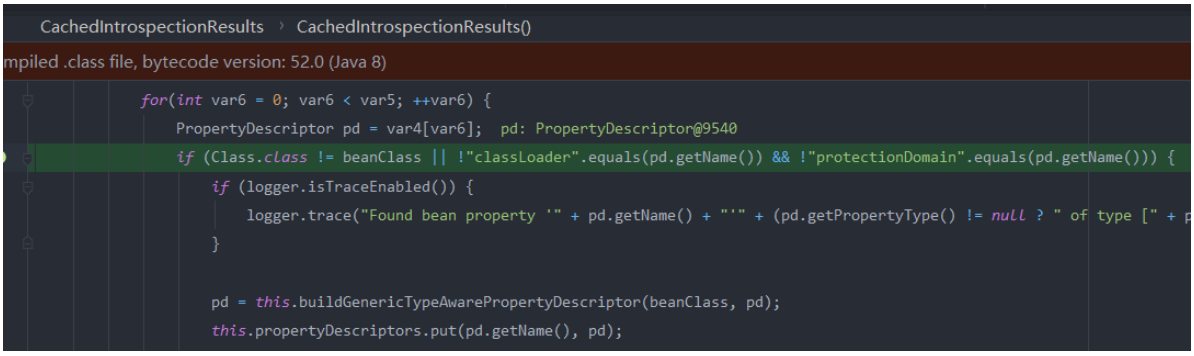
调用getter:



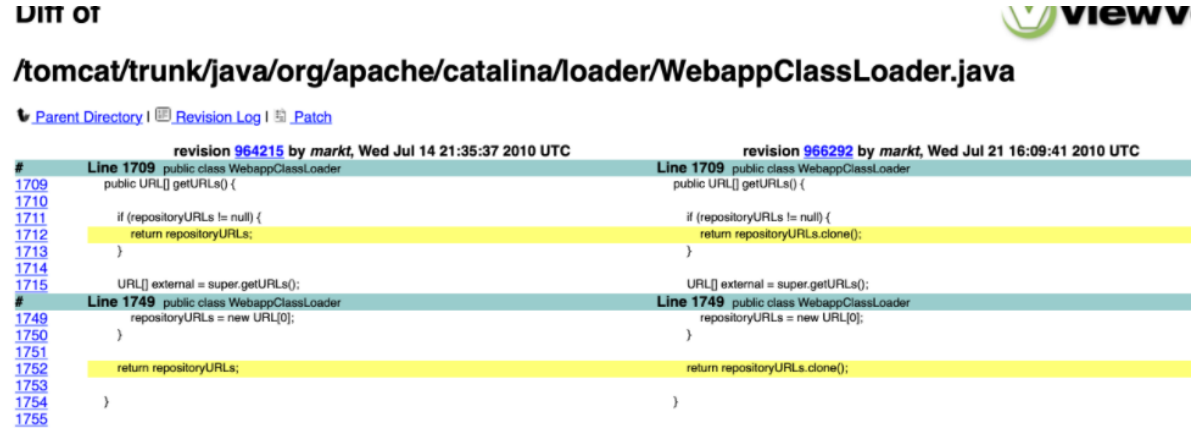
调用setter:

CVE-2010-1622修复

主要修复两点,spring将class.classLoader加入了黑名单

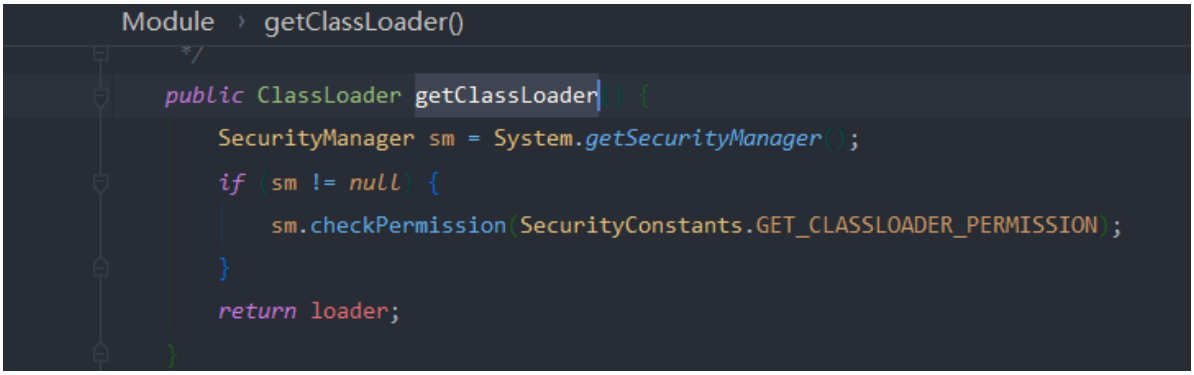


tomcat让我们无法修改URLs



绕过

第一个`class.classloader`使用了`class.mould.classloader`进行绕过,class中有对应的getModule方法,而且Module中有`getClassLoader`方法



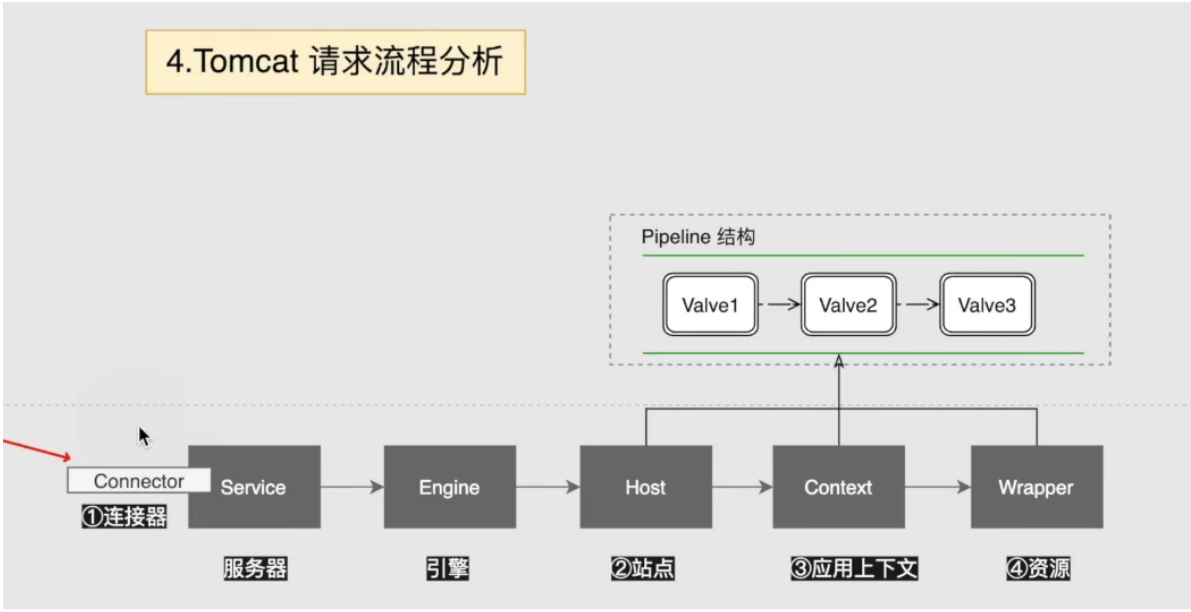
mould是JDK9新增的,这也就是为什么利用条件中需要JDK9及以上版本的原因

第二个修复点则采用了tomcat的`Pipeline`,修改了默认的日志文件位置并写入了webshell

https://www.bilibili.com/video/BV11w411f7XC?spm_id_from=333.337.search-card.all.click

tomcat中container有四种，分别是engine,host,context,wrapper,这4个。container的实现类分别是StandardEngine,StandardHost,StandardContext,StandardWrapper。四个容器是包含关系，engine包含host,host包含context,context，包含wrapper，wappper代表最基础的一个servlet。

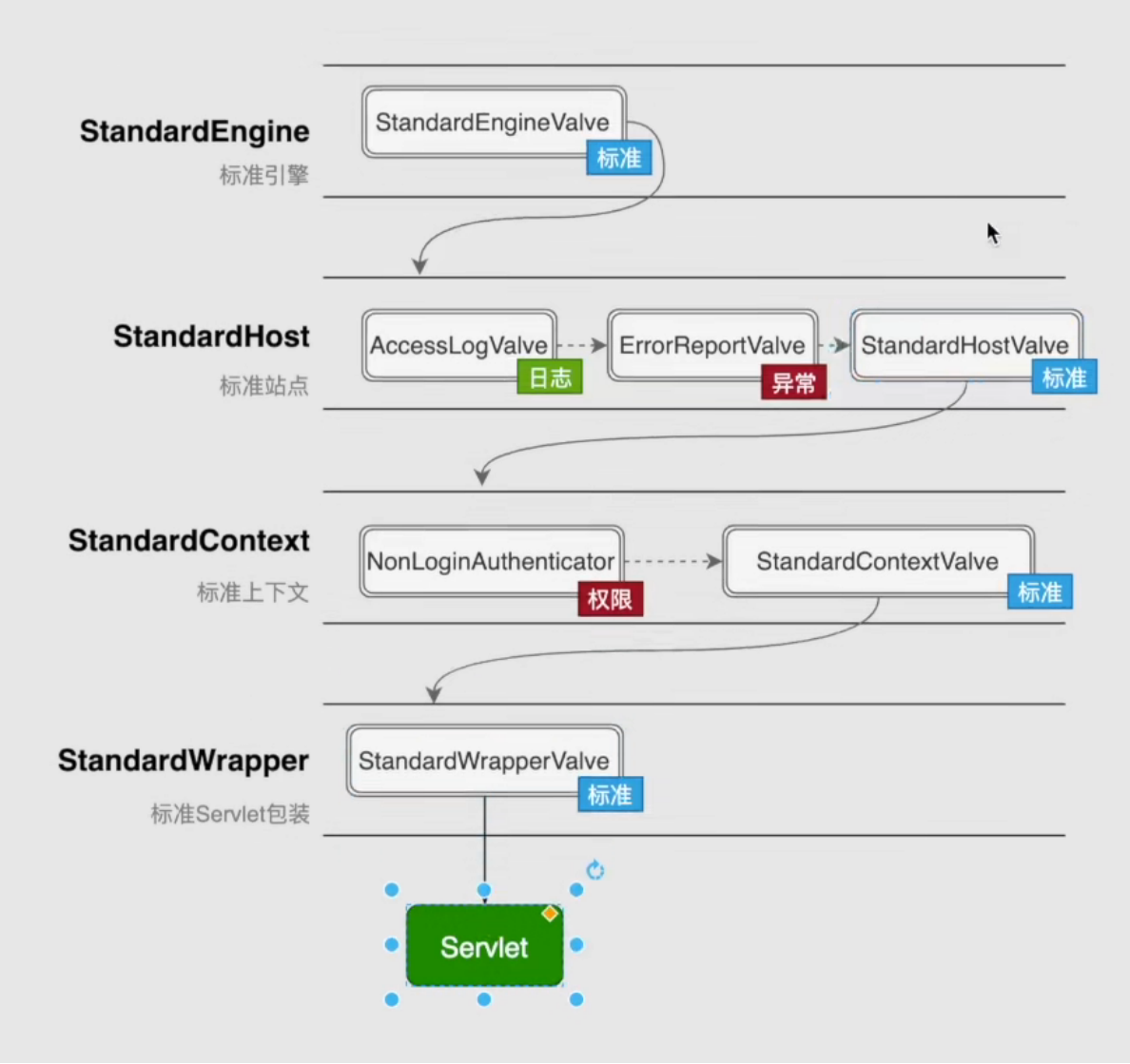
```
<!--医院  -->
<Server>
    <!--      妇幼保健大楼-->
    <Service name="HelloBaby" >
            <Connector port="8080" protocol ="HTTP/1.1"/>
            <Connector port="7070" protocol ="HTTP/1.1"/>
            <!--       挂号  -->
        <Engine name="hello" default="www.baby.com">
                <!--       科室       -->
            <Host name="www.baby.com" appBase="/Users/tommy/git/coderead-tomcat/Baby" >
                <Context path="/zhangshan" docPath="zhangshan"/>
            </Host>
            <Host name="www.women.com" >

            </Host>

        </Engine>

    </Service>

</Server>
```



当请求到达Engine容器的时候，Engine并非是直接调用对应的Host去处理相关的请求，而是调用了自己的一个组件去处理，这个组件就叫做pipeline组件,跟pipeline相关的还有个也是容器内部的组件，叫做valve组件。

在Catalina中，我们有4种容器，每个容器都有自己的Pipeline组件，每个Pipeline组件上至少会设定一个Valve(阀门)，这个Valve我们称之为BaseValve（基础阀）。基础阀的作用是连接当前容器的下一个容器(通常是自己的自容器),可以说基础阀是两个容器之间的桥梁。

Pipeline定义对应的接口Pipeline,标准实现了StandardPipeline。Valve定义对应的接口Valve,抽象实现类ValveBase,4个容器对应基础阀门分别是StandardEngineValve,StandardHostValve,StandardContextValve,StandardWrapperValve。



参数作用

在exp中改了几个参数,第一个是匹配写入日志的格式,这里是为了写入我们的payload,然后就是改后缀,路径,文件名,然后最后一个是将格式化,关于这个属性单独说一下

```
  try:
# class.module.classLoader.resources.context.parent.pipeline.first.pattern=%{c2}i if("j".equals(request.getParameter("pwd"))){ java.io.InputStre
# class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp&
# class.module.classLoader.resources.context.parent.pipeline.first.directory=webapps/ROOT/aaa&
# class.module.classLoader.resources.context.parent.pipeline.first.prefix=tomcatwaraaaa&
# class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=
```

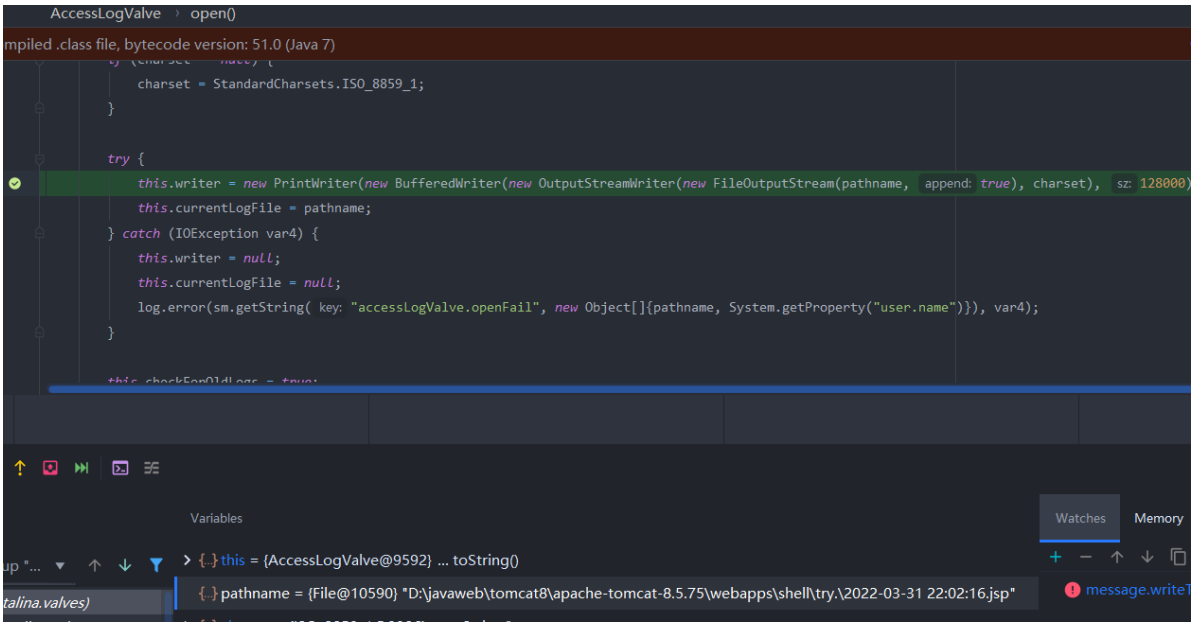
pattern

可以看看这篇文章,设定了写入日志的格式 https://blog.csdn.net/lixiangchibang/article/details/84024253

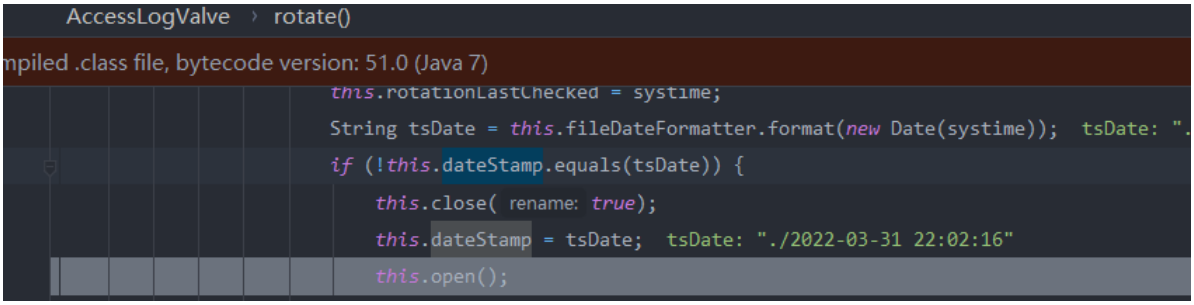
fileDataFormat

日志是在这里记录的:

这里的writer是



open方法在这里调用:



这里会判断上一次记录的时间和这次的时间,如果不相同的话才会更新`writer`,默认`fileDateFormat=.yyyy-MM-dd`,每天都会创建一个新的日志文件。在创建`writer`的时候,又会将我们转换后的时间作为文件名,所以这里exp中将`fileDateFormat`设置为了空,让我们的shell文件名彻底可控。而且这里之前在测试的时候有个问题,就是这个exp只能打一遍,现在的话算是懂了,因为我们把`fileDataFormat`设置为了空,导致我们每次计算出来的时间都是一样的,就不会更新我们的`writer`,解决方法就是先`fileDateFormat=.%2Fyyyy-MM-dd%20HH%3Amm%3Ass`打一遍,然后再设置为空,再打一遍

修复

Tomcat和Spring都做出了修复

- Tomcat：`WebappClassLoaderBase#getResources`方法直接返回null，在10.1.x版本后也会直接删除该方法，阻断了我们获取StandardRoot

- Spring：更新了CVE-2010-1622对于属性的限制。对于Class，只允许获取name相关的属性，对于所有类，禁止获取ClassLoader和ProtectionDomain，因此如果还想bypass的话，需要再找其他的属性链了