链滴

Solr 安全: 提示、技巧及需要知道的事情

作者：llh

原文链接：https://ld246.com/article/1497573086984

来源网站：链滴

随着solr5.2的发行，Solr有了认证和授权的API。让你能使用RuleBasedAuthorizationPlugin和BasicuthPlugin这两个插件来定义用户、角色和权限。但美中不足的是，这些插件虽然很强大，但是配置来却不那么直观。因此，我花了好一些时间来研究Solr的安全架构，搞清楚了两件事情，它的架构是样工作的以及它有哪些特性。最后，我总结了一个在配置和管理Solr安全时需要注意的清单。

使用Solr认证和授权的第一步是要开通solr security功能。当你开通了solr security功能，你就能用ap来更新你的安全配置了。你需要用zookeeper来上传security.json文件。首先你要打开命令行，cd到solr安装目录。然后启动solr，执行

```
bin/solr start -cloud
```

在那里，先要确保security.son文件在安装根目录，然后执行命令：

```
server/scripts/cloud-scripts/zkcli.sh -zkhost localhost:9983 -cmd putfile /security.json security.son
```

这样会安装security.json到zookeeper和所有可用节点，并且会使solr安全框架启动起来。

security.json 例子

```
{
"authentication":{
   "class":"solr.BasicAuthPlugin",
   "credentials":{"solr":"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdiVC3otuq0= Ndd7LKvVBAaZF0QAVi1ekCfAJXr1GGfLtRUXhgrF8c="}
},
"authorization":{
   "class":"solr.RuleBasedAuthorizationPlugin",
   "permissions":[{"name":"security-edit",
      "role":"admin"}]
   "user-role":{"solr":"admin"}
}}
```

一个security.json文件的解析

在上面的例子中，我们看到两个顶层的属性声明：authentication和authorization。它们分别代表要给两个api的组合数据。“class”属性定义用于处理这些api的处理类。BasicAuthPlugin和RuleBaseAuthorizationPlugin都打包到solr中了，你也可以定义自己的处理类。

在这个例子中，自定义鉴权，你需要这样做

```
extend AuthenticationPlugin implements ConfigEditorPlugin, SpecProvider
```

在这个例子中，自定义授权，你需要这样做

```
implements AuthorizationPlugin,ConfigEditablePlugin,SpecProvider
```

继续，在authentication对象中，有一个“credential”属性定义。它包含了一个列表，列表中是授

了的用户和SHA-256加密的密码。用户管理会在这篇文章的后面讨论。

在authorization对象中，有一个“permissions"属性定义和一个用户角色列表，用username:role来义。在这里，你能关联你的环境中用到的角色和在authentication定义的用户。你想创建多少用户角就能创建多少，但是，需要记住

- 所有用户必须是系统中注册的用户。
- 用户角色定义得越多，权限矩阵就越复杂。

权限数组是安全定义的最小粒度。它使用先前定义的用户角色来绑定单独的权限。这是很重要的，也为什么人们觉得solr安全设置不大方便的原因。权限和角色是一对一的关系。唯一例外的是通配符“，能用在所有的角色定义中，意味着所有的角色都拥有该权限。有重叠定义的情况下，可以使用“before”关键字，来使一个权限优先于其他权限。其他情况下，一个权限只限制在一个角色中。这意味着你要仔细考虑怎样构造你权限矩阵。举个例子，如果你要给系统中的所有用户“读”权限，你需要不仅是定义{“name”：“read”，“role”：“”}。乍一看，或直观上，你会觉得，这样就覆盖了所有要读的东西。但事实上不是这样的。给所有角色赋予读权限，你需要这样做

{"name":"read","role":""},{"name":"schema-read","role":""},{"name":"config-read","role":""},{"nme":"collection-admin-read","role":""},{"name":"metrics-read","role":""},{"name":"core-admin-ead","role":""}

这样，你的所有用户就能没有限制的访问solr管理控制台所能看到的所有东西了。

一个真实运行的例子

最近，我们有一个客户咨询了关于实现solr文档级别的安全的问题。Solr的权限框架允许你在控制台置collection级别的安全，但是，如果你要实现文档级别的安全，则需要编写自己的服务来对每个文进行ACP（access control policy）控制。如上所述，你能从控制台配置到collection级别的安全。要创建怎样的collection级别的权限矩阵呢?


```
{"authentication":{
"class":"solr.BasicAuthPlugin",
"credentials":{"solr":"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdiVC3otuq0= Ndd7LKvVBAaZI
0QAVi1ekCfAJXr1GGfLtRUXhgrF8c=","devuser":"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdi
C3otuq0= Ndd7LKvVBAaZIF0QAVi1ekCfAJXr1GGfLtRUXhgrF8c="}
},
"authorization":{
"class":"solr.RuleBasedAuthorizationPlugin",
"user-role":{"solr":"admin","devuser":"dev"},
"permissions":[{"name":"read","role":"*"},{"name":"schema-read","role":"*"},{"name":"config-re
d","role":"*"},{"name":"collection-admin-read","role":"*"},{"name":"metrics-read","role":"*"},{"n
me":"core-admin-read","role":"*"},{ "name":"secure-collection1-permission",
                    "collection":"securecollection",
                    "path":"/select",
                    "before": "collection-admin-read",
                    "role": "admin"
  }]
}}
```


从上面的例子所看到的，我定义了两个用户（solr/devuser）和两个角色（admin/dev），然后我分

了完全可读的权限给所有用户，除了一个collection，它只能通过admin角色访问。按照配置，这个限定义优先于collection-admin-read角色。这意味着，只有admin角色能看到名为“securecollection”的collection。我定义了授权路径“/select",这意味着用户可以查询collection。权限的名字是任的，只限制不能和内置的权限重名（比如”read"、"update"等）。在这个例子中，我将它命名为“seure-collection1-permission”。

预先定义的权限名，从PermissionNameProvider这个类里可以看到：

```
COLL_EDIT_PERM("collection-admin-edit", null),
COLL_READ_PERM("collection-admin-read", null),
CORE_READ_PERM("core-admin-read", null),
CORE_EDIT_PERM("core-admin-edit", null),
READ_PERM("read", ""), UPDATE_PERM("update", ""),
CONFIG_EDIT_PERM("config-edit", ""),
CONFIG_READ_PERM("config-read", ""),
SCHEMA_READ_PERM("schema-read", ""),
SCHEMA_EDIT_PERM("schema-edit", ""),
SECURITY_EDIT_PERM("security-edit", null),
SECURITY_READ_PERM("security-read", null),
METRICS_READ_PERM("metrics-read", null),
ALL("all", unmodifiableSet(new HashSet<>(asList("*", null))))
```

- collection-admin-edit: 允许该角色的用户更新collection.
- collection-admin-read: 允许该角色的用户读取collection.
- core-admin-edit: 允许该角色的用户编辑管理接口的特定方面（不安全）。
- core-admin-read: 允许该角色的用户在管理控制台读取特定的条目。
- read: 提供的基本的‘read'权限给用户。注意：该权限不是给角色完全的读权限。
- config-read: 允许已鉴权用户读取配置信息。
- config-edit: 允许已鉴权用户编辑Solr配置信息。
- schema-read: 允许该角色的用户读取 collection schema 信息.
- schema-edit: 允许该角色的用户编辑collection schema 信息.
- metics-read: 允许该角色的用户Solr metric 数据.
- all: 提供相关角色所有权限。

权限属性包括

```
"collection", "role", "params", "path", "method", "name","index","before"
```

- ‘collection‘是你环境中的collection的名字
- ‘role‘ 是你预先定义的角色
- ‘params‘ 说明允许的查询参数（比如&wt=json)
- ‘path‘ 说明要允许的url(比如 ‘/select). Solr中有效的path是：

/admin/mbeans,/browse,/update/json/docs,/admin/luke,/export,/get,/admin/properties,/elevate,/update/json,/admin/threads,/query,/analysis/field,/analysis/+ document,/spell,/update/csv/sql,/graph,/tvrh,/select,/admin/segments,/admin/system,/replication,/config,/stream,/schema

/admin/plugins,/admin/logging,/+ admin/ping,/update,/admin/file,/terms,/debug/dump,/update/extract

- ‘method ‘ 指的是HTTP协议的方法, 支持的方法: GET,PUT,POST,DELETE,HEAD. 注意 ‘OPTIONS’ 不是一个支持的方法, 还有 ‘TRACE’ , ‘CONNECT’ 和 ‘PATCH’ 也不是。
- ‘name ‘ 权限的名称，可以是内置的，也可以是自定义的。
- ‘index ‘ 指的是权限的索引，权限一旦在系统中定义了，就会赋予一个索引。 注意: + 一些老的明文档中使用权限名称来限定 ‘delete-permission’ 行为，然而，你需要使用权限索引（一个整数而不是一个名称（字符串）。你可以调用授权api获取权限的索引。
- ‘before’ 表明这个权限会比指定的权限优先。

管理用户

Solr的用户管理api是很简练的，且由于缺少必要的操作而备受争议。你能增加、修改、删除用户密码然而，一旦你上传了security.json文件，开始使用api，你初始的security.json文件就过期了，而如果重新上传security.json文件，它会覆盖你之前的任何修改操作。这就是为什么保持security.json为最版本是个好主意。这样，如果您需要迁移节点集群，就不必担心需要从头建立用户权限。你只需要上你的文件security.json，然后其他都不需要做了。

最后，让我们看一下一个用户管理流程的最佳实践。

1.上传 security.json

2.增加一个用户:

```
curl –user solr:SolrRocks http://localhost:8983/solr/admin/authentication -H 'Content-type:application/json' -d '{


"set-user": {"tom" : "TomIsCool"}'
```

3.现在，然我们获取tom的密码SHA-256哈希码:

```
curl –user solr:SolrRocks http://localhost:8983/solr/admin/authentication
```

应答大概是这样:

```
{
  "responseHeader":{
    "status":0,
    "QTime":3},
  "authentication.enabled":true,
  "authentication":{
    "class":"solr.BasicAuthPlugin",
    "credentials":{
      "solr":"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdiVC3otuq0= Ndd7LKvVBAaZIF0QAVi1eCfAJXr1GGfLtRUXhgrF8c=",
      "tom":"IV0EHq1OnNrj6gvRCwvFwTrZ1+z1oBbnQdiVC3otuq0= Ndd7LKvVBAaZIF0QAVi1eCfAJXr1GGfLtRUXhgrF8c="}}}
```

4.现在，拷贝 ‘tom’ 的条目，加到你的security.json文件中。 这样，你维护了一个授权用户的最新表，而如果你需要改变一个权限，改变你的security.json文件，或迁移你的环境，你也不需要重新创

它。 不可否认，这样做是有点绕，但这会节省你的时间，且很多时候，是非常有效的。

权限的故障排除

权限的故障排除往往是非常困难的。往往权限系统不会提供足够的错误信息，甚至，一点错误信息都有。但Solr不是这样的。Solr在未授权的情况下会返回403错误。然而，如果你上传了一个security.json文件，但它的表现并非如你所愿，试下找到solr.log文件，查看一下里面的错误信息。这是你调试Solr的权限矩阵的最好的朋友。这是一些常见的错误

- 没有授权用户:

2017-03-08 15:47:55.876 INFO (qtp1348949648-16) [ ] o.a.s.s.HttpSolrCall USER_REQUIRED auth header null context : [FAILED toString()]

- 无效角色:

“role”:” admin”}, The principal [principal: devuser] does not have the right role

这些是用户权限最常见的错误。

简而言之：使用Solr安全时的重要的提示和技巧

- 你的Solr安装在防火墙里面会比较安全。
- 启动Solr的用户只有Solr home目录的写权限。（Solr home 目录(默认是 $SOLR_INSTALL/server/solr)
- 权限是先进先出的。就是说，他们按照声明的顺序进行计算。
- 角色和权限的关系是一对一的。一个权限（比如‘read'）只会被计算一次，就是在它第一次在列表出现的时候。如果你有第二个’ read ‘在下面的行中定义，它会被忽略掉。
- 你可以用星号()来分配所有权限给所有用户。{ “name”：“read”，“role”：“”}
- ‘role’ 属性可以传递(*)或一个角色名称字符串作为参数。你不能传递一个数组或逗号分隔的字符。
- 你可以使用 ‘all’ 权限来分配所有权限给给定角色{ “name”：“all”，“role”：“admin”}
- 你应该尽量分配少点权限以免犯错。只分配你所需要的。 “少即是多.” 就是说，所需之间保持平，避免走向极端。
- 当删除权限时，你必须使用权限下标而不是权限名称。
- 通配符要好好利用。
- 如果你分配控制台的普通读权限，你还需要同时分配collection-admin-read and core-admin-read这两个权限。类似的，分配 ‘update'权限，也需要同时分配core-admin-edit and + collection-admin-edit给相同的角色。
- 定义一个权限，只有’ role'参数是必须的。
- 使用库比如Solrj来和ZooKeeper通讯。
- 当使用一个预定义的权限，可选的参数只有name, role, collection, index.

英文链接

https://lucidworks.com/2017/04/14/securing-solr-tips-tricks-and-other-things-you-really-need-to-know/