Hadoop Sandbox LCR

——LinuxContainerRuntime

当运行未知Java程序的时候，该程序可能有恶意代码（删除系统文件、重启系统等），为了防止运行恶意对系统产生影响，需要对运行的代码权限进行控制，这需要需要Java安全管理器（SecurityManager），Java安全应该包括两个方面的内容：

* Java平台（即Java运行环境）的安全性，由Java平台开发者来维护，平台安全主要是对class文件进行检查：变量、方法定义、类型转化等
* Java语言开发的应用程序安全性，由程序开发人员维护，主要是Java类对文件读取权限、java方法的使用权限

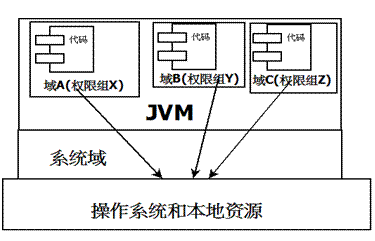
Java为应用程序安全性提供安全管理机制-SecurityManager，每个Java应用都可以拥有自己的安全管理器，在运行阶段检查需要保护资源的访问权限和其他规定的操作权限，保护系统免受恶意操作攻击，以达到系统的安全策略。

# Sandbox与SecurityManager

## 1.1 基本概念

Java安全模型的核心是Java Sandbox，沙箱是一个限制程序运行的环境，将Java代码限定在虚拟机特定的运行范围内，并且严格限制代码对本地系统资源的访问，从而隔离代码来保护系统资源（CPU、内存、文件系统等）和程序本身。

Java的安全机制中有域（Domain）的概念，虚拟机会把所有代码加载到不同的系统域和应用域，系统域负责与关键资源进行交互，而应用域则通过系统域的部分代理对各种需要的资源进行访问。不同的受保护域（Protected Domain）对应不同的权限（Permission），存在不同域的类文件具有当前域的全部权限，如下图所示：



组成沙箱的基本组件包括以下部分：

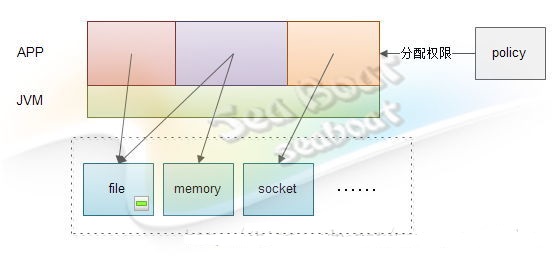
* 字节码校验器，确保Java类文件遵循的Java语言规范，可以帮助Java程序实现内存保护，但是并不是所有类文件都经过字节码校验，比如核心类
* 类装载器，在3个方面对Java沙箱起作用

1. 防止恶意代码去干涉善意的代码
2. 守护了被信任的类库边界
3. 将代码归入保护域，确定代码可以进行哪些操作

虚拟机为不同的类加载器加入的类提供不同的命名空间，命名空间有唯一的标识，被装载的类都属于这个命名空间，命名空间互相之间不可见

* 存取控制器(Access Controller)，存取控制器可以控制核心API对操作系统的存取权限，而这个控制策略的设定，可以由用户指定
* 安全管理器(Security Manager)，核心API和操作系统之间的主要接口，实现权限控制，比存取控制器优先级高
* 安全软件包，java.security下的类和扩展包下的类，允许用户为自己的应用增加新的安全特性，包括安全提供者、消息摘要、数字签名、加密及认证等

其中安全管理器（SecurityManager）是核心API与操作系统之间的主要接口，其实现权限控制。安全管理器会根据Policy文件所描述的策略给程序不同模块分配权限，如下图：



应用程序分成不同权限的三块：文件读取权限、文件和内存读取权限、监听Socket权限，通过这个机制可以控制程序各个部分的操作权限。SecurityManager的启动方式有两种：

* 启动参数方式

*-Djava.security.manager*

指定配置文件的位置参数示例如下:

*-Djava.security.policy=c://protect.policy*

* 编码方式启动

*System.setSecurityManager(new SecurityManager());*

建议通过参数启动，其本质也是通过编码启动，但是参数方式更加灵活。

## **1.2 使用示例**

1）示例程序，SecurityManager的使用示例：

*public class SecurityManagerTest {*

*public static void main(String[] args) throws Exception {*

*System.out.println("SecurityManager:" + System.getSecurityManager());*

*BufferedReader reader =*

*new BufferedReader(new FileReader("C:\\protect.txt"));*

*String line = null;*

*while( (line = reader.readLine() ) != null){*

*System.out.println(line);*

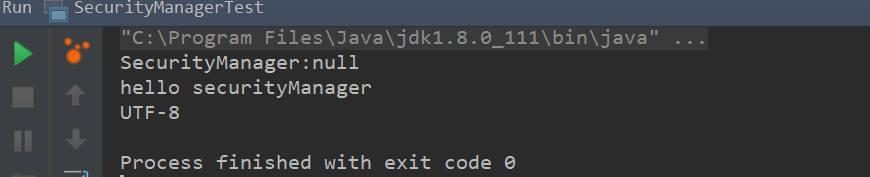
*}*

*System.out.println(System.getProperty("file.encoding"));*

*}*

*}*

不添加启动参数直接运行，则相当于没有启动安全管理器，SecurityManager打印出null，且能正确读取protect.txt文件跟file.encoding属性，输出内容如下：



2）添加SecurityManager启动参数

*-Djava.security.manager -Djava.security.policy=c://protect.policy*

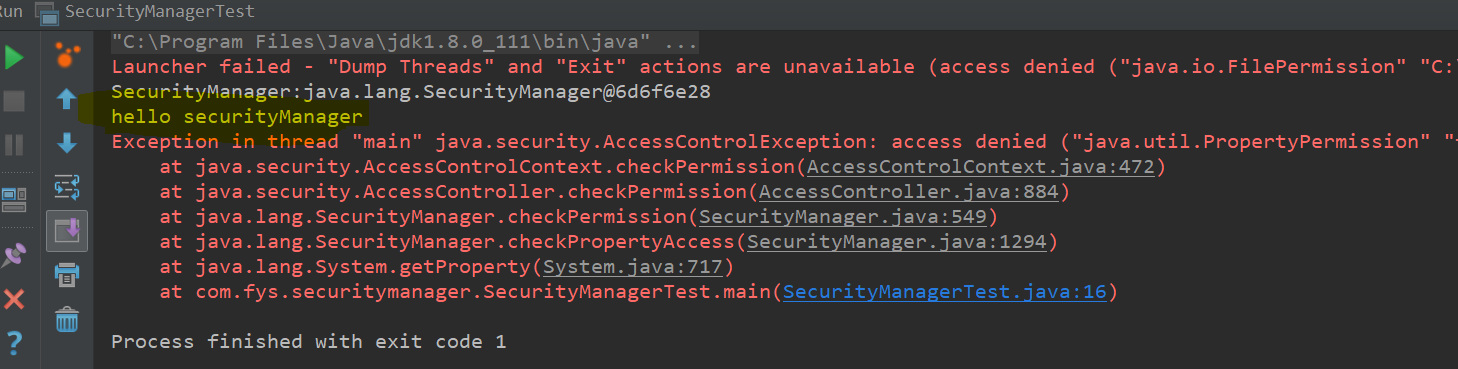
在protect.policy文件中添加以下授权语句：

*grant {*

*permission java.io.FilePermission "c:/protect.txt", "read";*

*};*

执行后输出如下：



此时SecurityManager不为空，并且有权读取protect.txt文件，但最终还是会抛一个安全异常，但是没有权限读取file.encoding系统属性

1. 将protect.policy授权语句改为如下：

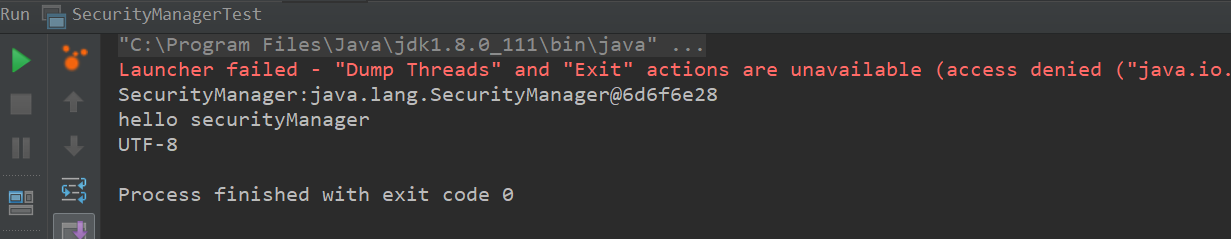
*grant {*

*permission java.io.FilePermission "c:/protect.txt", "read";*

*permission java.util.PropertyPermission "file.encoding", "read";*

*};*

输出结果如下：



程序运行正常，可以读取文件和系统属性，不再抛出安全异常。

## **1.3 Sandbox要素**

对沙箱的控制就是对以下要素的限制

1. 代码源，代码源是类所在的位置，表示为URL地址
2. 权限，允许代码执行的操作，包括三部分：

* 权限类型，实现了权限的Java类名
* 权限名，对资源进行操作的资源定位(文件名或者通配符、网络主机等)，一般基于权限类型来设置，但是java.security.AllPermission不需要权限名
* 允许的操作，和权限类型对应，指定目标可以执行的操作行为，比如读、写等

*permission java.security.AllPermission; //权限类型 permission java.lang.RuntimePermission "stopThread"; //权限类型+权限名 permission java.io.FilePermission "/tmp/foo" "read"; //权限类型+权限名+允许的操作*

标准权限如下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 说明 | 类型 | 权限名 | 操作 |
| 文件权限 | java.io.FilePermission | 文件名 | read/write/delete/execute |
| 套接字权限 | java.net.SocketPermission | 主机名:端口 | accept/listen/connect,resolve |
| 属性权限 | java.util.PropertyPermission | JVM属性名 | read/write |
| 运行时权限 | java.lang.RuntimePermission | 多种权限名 | 无 |
| AWT权限 | java.net.AWTPermission | 6种权限名 | 无 |
| 网络权限 | java.net.NetworkPermission | 3种权限名 | 无 |
| 安全权限 | java.security.SecurityPermission | 多种权限名 | 无 |
| 序列化权限 | java.io.SerializablePermission | 2种权限名 | 无 |
| 反射权限 | java.lang.ReflectPermission | suppressAcces  sChecks | 无 |
| 完全权限 | java.security.AllPermission | 无 | 无 |

1. 保护域，组合代码源和权限，声明比如由代码A可以做权限B这样的事情
2. 策略文件，控制沙箱的管理要素，有一个或多个保护域的项。策略文件完成代码权限的指定任务，包括全局和用户专属两种。JVM可以使用多个策略文件，不过两个最常用：

* 全局，$JRE\_HOME/lib/security/java.policy，作用于JVM所有实例
* 用户自定义，存储到用户主目录下，策略文件可以使用JDK自带的policytool工具编辑

默认java.policy，文件内容如下：

*grant codeBase "file:${{java.ext.dirs}}/\*" {*

*permission java.security.AllPermission;*

*};*

*//默认extenstions获取所有权限*

*grant {*

*//允许所有线程调用Thread.stop*

*permission java.lang.RuntimePermission "stopThread";*

*//允许监听任何端口*

*permission java.net.SocketPermission "localhost:0", "listen";*

*//任何用户可以读取以下参数*

*permission java.util.PropertyPermission "java.version", "read";*

*permission java.util.PropertyPermission "java.vendor", "read";*

*permission java.util.PropertyPermission "java.vendor.url", "read";*

*.......*

*permission java.util.PropertyPermission "java.vm.name", "read";*

*};*

策略文件的内容格式如上所示，grant授权允许操作某个权限，这个默认的策略文件指明JDK扩展包可以有全部权限，允许代码stop线程，允许监听端口等。

java.security，参数文件，这个文件和策略文件在同一个目录下，这个参数文件定义沙箱的一些参数，比如默认的沙箱文件是这样的

*policy.url.1=file:${java.home}/lib/security/java.policy*

*policy.url.2=file:${user.home}/.java.policy*

*policy.allowSystemProperty=true*

*policy.expandProperties=true*

policy.url.\*这个属性指明了使用的策略文件，默认的两个位置就在这里配置，用户可以自行更改顺序和存储位置，policy.allowSystemProperty指明是否允许用户自行通过命令行指定policy文件

5）密钥库，保存密钥证书的地方

https://www.cnblogs.com/MyStringIsNotNull/p/8268351.html

# **JavaSandbox LRE**

在YARN集群中启动的应用会给集群带来不稳定因素，用户程序可以在YARN Container中创建子进程、打开任意网络连接、创建文件等，通过Java SecurityManager可以减轻程序误操作造成的风险。在YARN JVM Sandbox的使用如下：

1. 在yarn-site.xml中增加配置

*<property>*

*<name>yarn.nodemanager.runtime.linux.allowed-runtimes</name>*

*<value>default,docker,javasandbox</value>*

*</property>*

*<property>*

*<name>yarn.nodemanager.runtime.linux.sandbox-mode</name>*

*<value>permissive</value>*

*</property>*

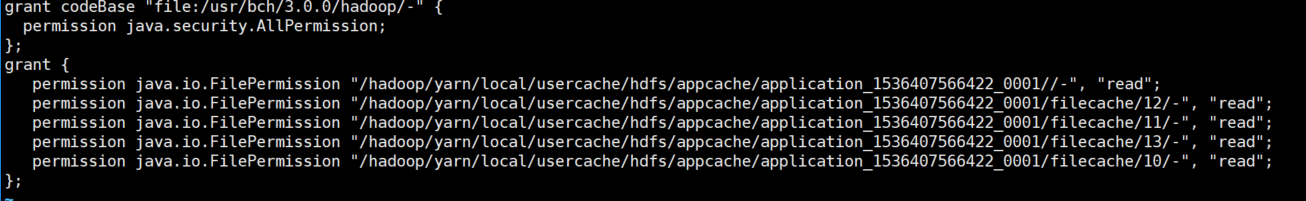
1. 启动应用程序

*hadoop jar hadoop-mapreduce-examples-3.1.0-bc3.0.0.jar pi 1 1000*

1. Sandbox信息

会在本地生成目录nm-sandbox-policies，生成policy文件：

*container\_e32\_1536407566422\_0001\_02\_000001-java.policy*



*/usr/jdk64/jdk1.8.0\_112/bin/java*

*-Djava.security.manager -Djava.security.policy==/tmp/hadoop-yarn/nm-sandbox-policies/container\_e41\_1536423679017\_0001\_02\_000001-java.policy*

*-Djava.io.tmpdir=$PWD/tmp -Dlog4j.configuration=container-log4j.properties -Dyarn.app.container.log.dir=/hadoop/yarn/log/application\_1536423679017\_0001/container\_e41\_1536423679017\_0001\_02\_000001*

*-Dyarn.app.container.log.filesize=0*

*-Dhadoop.root.logger=INFO,CLA*

*-Dhadoop.root.logfile=syslog -Dbch.version= -Xmx3276m -Dbch.version= org.apache.hadoop.mapreduce.v2.app.MRAppMaster*

1. 源码分析

* 在LCE#prepareContainer，生成policy文件，

调用NMContainerPolicyUtils#generatePolicyFile，源码如下：

*static void generatePolicyFile(OutputStream policyOutStream,*

*List<String> localDirs,*

*List<String> groupPolicyPaths,*

*Map<org.apache.hadoop.fs.Path, List<String>> resources,Configuration conf)*

*throws IOException {*

*String policyFilePath =*

*conf.get(YarnConfiguration.YARN\_CONTAINER\_SANDBOX\_POLICY);*

*String filePermissions =*

*conf.get(YarnConfiguration.YARN\_CONTAINER\_SANDBOX\_FILE\_PERMISSIONS,*

*YarnConfiguration.DEFAULT\_YARN\_CONTAINER\_SANDBOX\_FILE\_PERMISSIONS);*

*.....*

*Formatter filePermissionFormat = new Formatter(policyOutStream,*

*StandardCharsets.UTF\_8.name());*

*filePermissionFormat.format(HADOOP\_HOME\_PERMISSION);*

*filePermissionFormat.format("grant {%n");*

*for(String localDir : localDirs) {*

*filePermissionFormat.format(*

*FILE\_PERMISSION\_FORMAT, localDir, filePermissions);*

*}*

*for(String cacheDir : cacheDirs) {*

*filePermissionFormat.format(*

*FILE\_PERMISSION\_FORMAT, cacheDir, filePermissions);*

*}*

*filePermissionFormat.format("};%n");*

*filePermissionFormat.flush();*

*}*

* 命令行中添加SecurityManager参数，在prepareContainer中执行

NMContainerPolicyUtils#generatePolicyFile

*static void appendSecurityFlags(List<String> commands,*

*Map<String, String> env, Path policyPath, SandboxMode sandboxMode)*

*throws ContainerExecutionException {*

*for(int i = 0; i < commands.size(); i++){*

*String command = commands.get(i);*

*if(validateJavaHome(env.get(JAVA\_HOME.name()))*

*&& command.matches(CONTAINS\_JAVA\_CMD)*

*&& !command.matches(MULTI\_COMMAND\_REGEX)){*

*command = command.replaceAll(CLEAN\_CMD\_REGEX, "");*

*String securityString = JVM\_SECURITY\_CMD + policyPath + " ";*

*commands.set(i, command.replaceFirst(JAVA\_CMD, securityString));*

*} .....*

*}*

命令行中添加参数如下：

*-Djava.security.manager -Djava.security.policy==/tmp/hadoop-yarn/nm-sandbox-policies/container\_e41\_1536423679017\_0001\_02\_000001-java.policy*

https://www.aliyun.com/jiaocheng/774888.html

https://blog.csdn.net/wangyangzhizhou/article/details/38497755

https://issues.apache.org/jira/browse/YARN-5280