Valkyrie Documentation

Release 1.0.0

Marcelo Aguiar Rodrigues

Dec 11, 2021

Contents:

Trying to create a simple table of contents

CHAPTER 1

Valkyrie Sandbox Library

Valkyrie is a sandbox library for applications that run plugins. It provides a more comprehensive way to setup a set of permissions for each classloader that runs a plugin inside the main application while maintaining full permissions for the application itself.

With Valkyrie is possible to create a *SecurityProfile* that will be used for your application plugins. Each *SecurityProfile* is associated with a *ClassLoader*.

SecurityProfile is a interface that must be implemented by the application to set the *PermissionCollection* for the plugins. There are two implementations in Valkyrie, one with no permissions at all and one with all permissions, where the latter is used by default as the application set of permissions.

The *ClassLoader* can be any custom classloader that will run the plugins.

1.1 Usage

1.1.1 Installation

You may use any of the following package managers to use Valkyrie.

Maven

```
<repositories>
    <repository>
        <id>jitpack.io</id>
        <url>https://jitpack.io</url>
    </repository>
</repositories>
```

```
<dependency>
    <groupId>com.github.marceloaguiarr</groupId>
    <artifactId>valkyrie</artifactId>
    <version>1.0.0</version>
</dependency>
```

Gradle

```
allprojects {
        repositories {
            ...
            maven { url "https://jitpack.io" }
        }
    }
```

```
compile group: 'com.github.marceloaguiarr', name: 'valkyrie', version: '1.0.0'
```

Sbt

```
resolvers += "jitpack" at "https://jitpack.io"
```

```
libraryDependencies += "com.github.marceloaguiarr" % "valkyrie" % "1.0.0"
```

1.1.2 Adding to your application

Valkyrie acts as a wrapper for the SecurityManager rabbit hole to provide a simpler way to secure your application and the plugins it might run. To do that you are going to define a set o permissions that the plugins have.

The way this works is as a whitelist of permissions, allowing them to do what is explicit described and denying anything else.

Plugins must run in a separate classloader.

To create a set of permissions create a class that implements the *com.github.marceloaguiarr.valkyrie.profiles.SecurityProfile* interface. This interface has only one method called *getPermissions* that returns a *java.security.PermissionCollection* object. An example is shown below.

```
public class PluginSecurityProfile implements SecurityProfile {

    @Override
    public PermissionCollection getPermissions() {
        Permissions permissions = new Permissions();
        permissions.add(new PropertyPermission("*", "read"));
        permissions.add(new FilePermission("<<ALL FILES>>", "read"));
        permissions.add(new FilePermission("/home/user/tmp/*", "write"));
        permissions.add(new SocketPermission("*", "connect, resolve"));

        return permissions;
    }

}
```

A complete list of permissions can be found here.

With that done now you can start Valkyrie

```
SecurityProfile pluginSecurityProfile = new PluginSecurityProfile(); (1)

Valkyrie.addProfile(URLClassLoader.class, pluginSecurityProfile); (2)
Valkyrie.setSecurityManager(SecurityManagers.DEFAULT); (3)

Valkyrie.start(); (4)
```

(1) • Create an instance of the *SecurityProfile* you created

(2) • Set the profile you created to the ClassLoader you will use to run your plugins

(3) • Set the *SecurityManager* for the application

(4) • Start Valkyrie

That is it. Any code executed under the ClassLoader defined in (2) will be submitted to the set of permissions given to the *PluginSecurityProfile* class. You can add multiple *SecurityProfile* distinct ClassLoaders.

1.1.3 Disabling Valkyrie

Even though the application has a set of AllPermissions that are still some actions that the SecurityManager will not allow. This might prompt the developer to want to stop Valkyrie to execute their code. This is not advised and Valkyrie does not provide a functionality to stop itself.

If your business logic requires that the application execute a snippet of code that is being blocked by Valkyrie there is a *doPrivileged* method.

Usage:

```
Valkyrie.doPrivileged(() -> {
    // your privileged code here
    return null;
});
```

This will run the code with elevated privileged without making your application vulnerable.