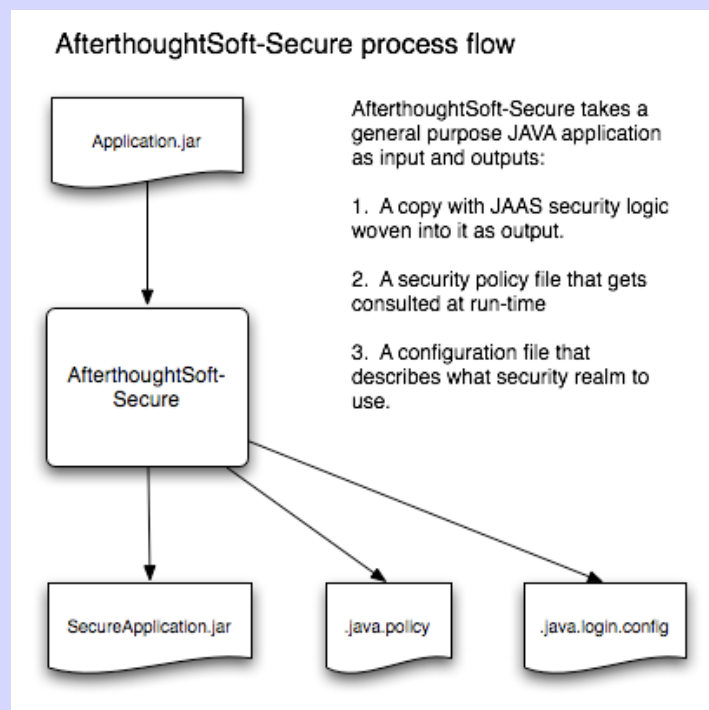


## AfterthoughtSoft-Secure Community Edition

AfterthoughtSoft-Secure is a graphical JAVA based application that simplifies adding [JAAS](#) (Java Authentication and Authorization) security to non-secure JAVA applications. It is designed primarily for client-side applications (ex. Swing GUI applications).

What distinguishes the AfterthoughtSoft-Secure product from other security products is its capability to put security into a JAVA application **after** it has been created. This means:

- \* You can add security to legacy java applications that previously had no security
- \* A separation of concerns -- security can be applied separate from application functionality (non-programmers can do it)

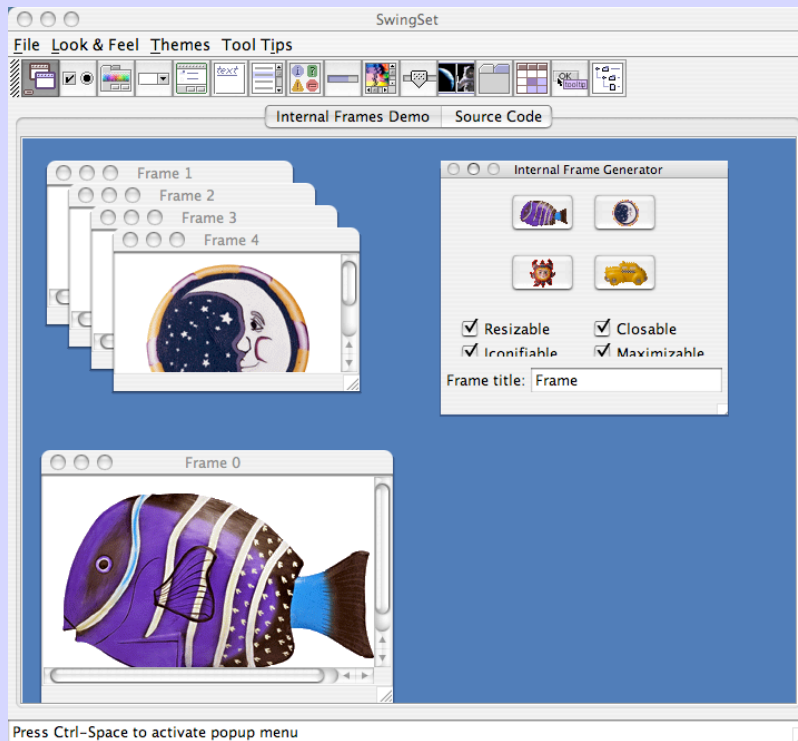


The free community edition provides a glimpse of the capabilities of the AfterthoughtSoft-Secure product line. It allows one to provide RBAC (Role-Based Access Control) against a flat file that contains passwords encoded as MD5 hashed strings.

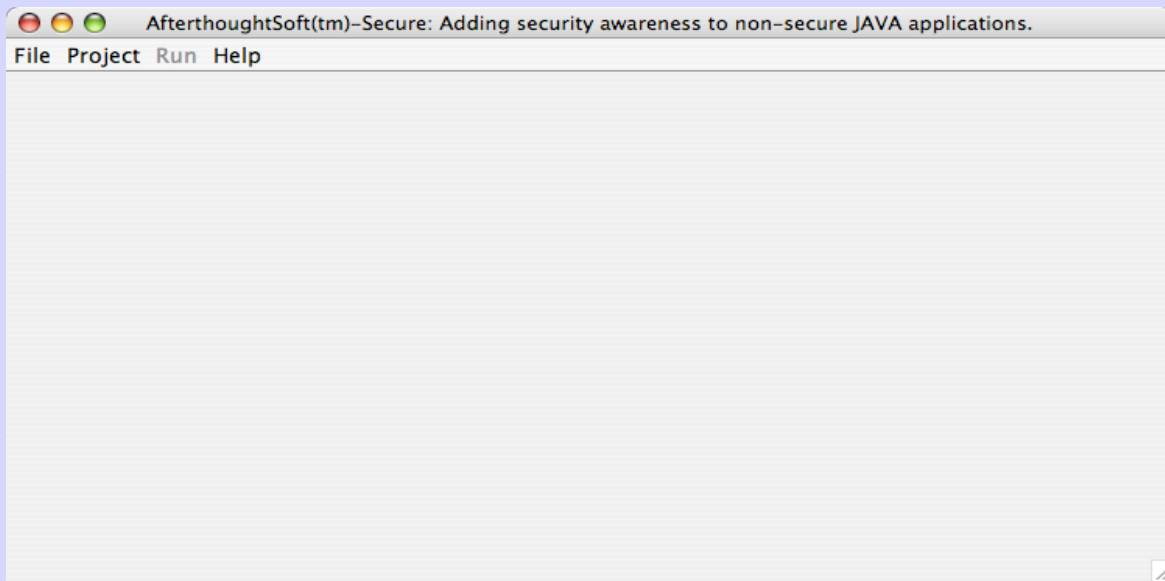
For example, imagine you had three employees in your company, Joe, Maria, and Riad:

Joe	5a105e8b9d40e1329780d62ea2265d8a	group1:group2
Maria	ad0234829205b9033196ba818f7a872b	group2:group3
Riad	8ad8757baa8564dc136c1e07507f4a98	group3

And one day you (the boss), decide that you need to control access to a particular JAVA application at your company. Let's pick the demo application "SwingSet2.jar" from Sun corporation as our example application that needs security. Here is what the SwingSet2 application looks like



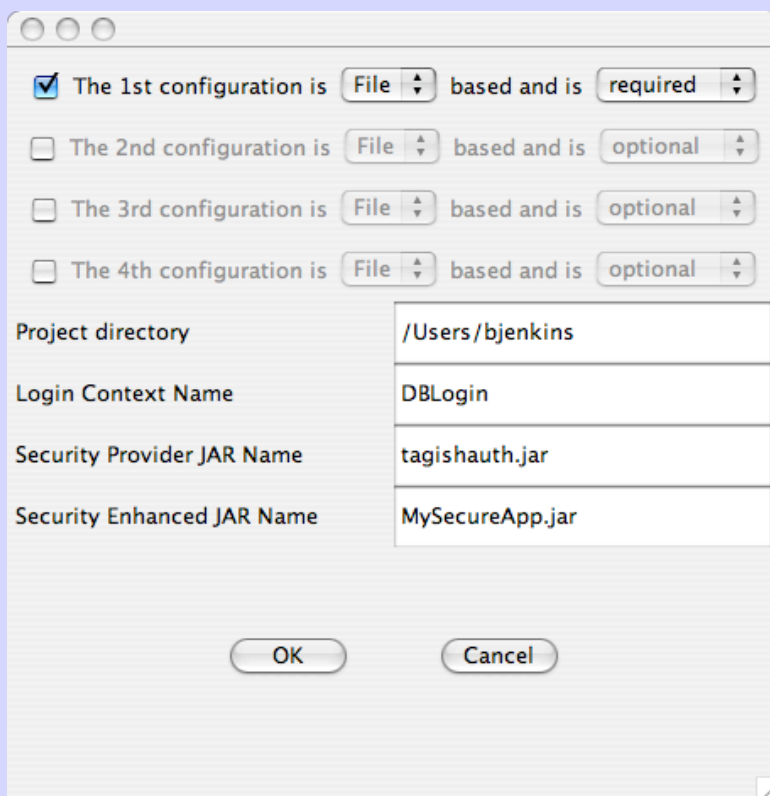
To add a user login to only allow those employees that are in "group2" from our list above we can use the AfterthoughtSoft-Secure application to add security to the SwingSet2 application. Launching the application brings you to the main screen:



AfterthoughtSoft-Secure works with the reference security realms provided by Sun. These range in sophistication from a simple file-based list of users and groups (as shown above) to users and groups information stored in relational databases (Oracle, MySql) accessible via JDBC all the way up to sophisticated enterprises using Kerberos V for authentication.

The first step in securing our SwingSet2 java application is to load it into the AfterthoughtSoft-Secure application. Simply click "File / Open", browse to the location of the SwingSet2.jar and click Open.

The second step in securing our JAVA app is to select a security realm. From the main application window, clicking Preferences allows you to select which type of security realm you wish to use.

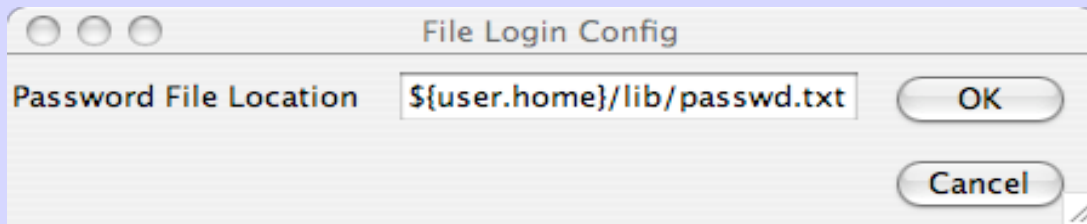


Java Authentication and Authorization (JAAS) is patterned after UNIX's PAM security mechanism (Pluggable Authentication Module) and allows one to have multiple security mechanisms that can "stack" to provide complete authentication / authorization security for your apps.

AfterthoughtSoft-Secure allows you to have up to 4 stacked security mechanisms, from simple text based security in ASCII text files to LDAP or Kerberos V authentication and authorization.

It even allows for custom security mechanisms, such as biometrics. Why multiple levels of authentication? Having several would allow you to have 3 factor authentication, for example. This might consist of a password (what you **know**), a security token (what you **have**), and a biometric (what you **are**).

The only security realm available in the community (free) edition is a file based, ascii text file with MD5 hashed passwords (what you know). So clicking on the combo box that has the default of "File" in it brings up the file based realm dialog.



The default storage location for this ASCII based text file is a file called passwd.txt located in the lib directory that is in the users home directory (indicated by the `{user.home}` JAVA configuration variable). Note: A default “passwd.txt” file is installed for you automatically in your home directory and has the same entries as the table shown earlier.

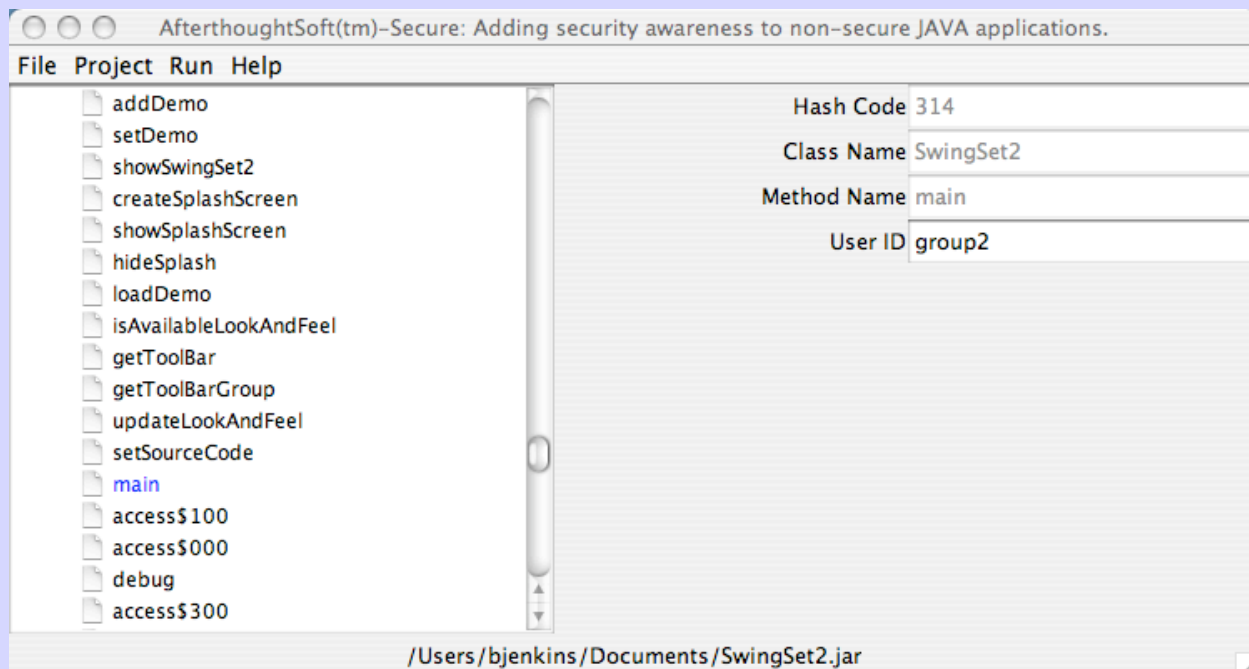
For any stackable security mechanism, you can choose which is **required**, **optional**, **requisite**, or **sufficient**. You typically will have at least one “required” entry (otherwise what is the point of having security for your application!)

The third step in the process is to choose which users / groups have access to your application. At a minimum you should protect the front door to the application.

## Front Door ?

Each and every JAVA application (and most every type of software application) has a main entry point from which it starts when launched. This is named “main” (gee, how appropriate). By guarding this function with security we effectively lock the “front door” of our application. Let’s say we only want to allow Joe and Maria access to our application (for whatever reason). Since they both belong to group2, by telling AfterthoughtSoft-Secure to only allow user “group2” access to the “main” method, we effectively restrict access for the entire application to Joe and Maria

So, lets at LEAST protect general access to usage of the SwingSet2 application by protecting the main method in the SwingSet2 class. This is shown in the next image.



Then simply selecting “Run” and then “Go”, you will get a copy of the application, renamed to include the string “Secure” in front of it. So, if we were still using the SwingSet.jar as our example, we would now have a copy, with security woven into it called “SecureSwingSet.jar”, that, when launched, would prompt for a UserID and Password.

And that’s all there is to it!