**DELIVERABLE 08 ITEM 06: IMPLEMENTING HTTPS**

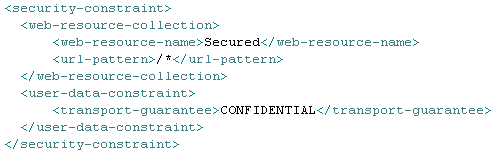
In this report we will explain how we can configure our project and the pre-production configuration to run applications using HTTPS, the version of HTTP which uses secure communications by using SSL. The steps to follow are very simple, but also very important.

The first thing we need to do is to create a certificate. We could also buy a signed certificate on the Internet, but for the purpose of this project it is better to create our own, self-signed one. To create the certificate we will use a tool included in Java called Keytool. It is a command line utility to work with certificates. If it is not available globally, it is necessary to run the commands from the installation directory; however, in the provided virtual machines it can be used wherever the command prompt is opened. Press the Windows key and R to open the “Execute” window, and then type “cmd.exe” and press Enter to open the command prompt. Write this command to create the certificate:

keytool -genkey -alias tomcat -keysize 2048 -keyalg RSA -keystore keystore.jks -validity 3650

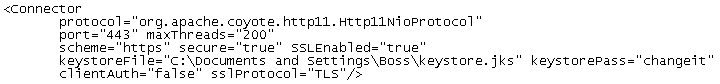
After pressing Enter, a password for the keystore (where the certificate will reside) is requested. Type “changeit” to set that as the password and press Enter. After confirming the password, several configuration parameters can be set. Just keep pressing Enter until a message asking if everything is correct appears. Type “y” or “yes” and press Enter, and then once more to set the certificate password (it will be the same as the keystore one). After these steps, we can find the newly created certificate inside a keystore named “keystore.jks” inside our personal folder (C:\Documents and Settings\{user}).

Before we configure the Tomcat server to allow HTTPS connections, we should change a file inside our web project to redirect every HTTP request to HTTPS. This file is called “web.xml”, and it contains different configuration parameters for our application. At the end of the file, but before closing the “web-app” tag, insert the following snippet of code:



By setting “transport-guarantee” to “CONFIDENTIAL” and specifying “/\*” as the URL pattern, every request will be forwarded by Tomcat to the corresponding HTTPS port. This is the only file that we need to change in the project.

Now we will move to the Tomcat server. First, locate a configuration file called “server.xml”, found inside a “conf” folder within the Tomcat installation folder. Open it, and add the following lines of code:



This will tell Tomcat to use the keystore that we created previously, with the password we set before. This also sets the HTTPS port to 443, which is the default port for this protocol. In the developer configuration, however, it is recommended that we use another port, such as 8443. By using 443, the port in the URL of the browser will be hidden, just like when using 80 for HTTP. Note that you can move the keystore file to wherever you want, as long as you remember to change the path to the file in this configuration.

There is one more thing that we need to change in this file. Find this line and change “redirectPort” to 443, so that Tomcat knows where to redirect HTTP requests:



After making these two changes, we need to restart Tomcat. Find the corresponding icon in the bottom right corner of the screen, double click it, and a Tomcat configuration program will open. There, click “Stop” to stop the server, and then “Start” to start it again. After that, the server will be ready to serve applications using HTTPS. Just go to the Tomcat manager using the browser, deploy an application, and try it. In Chrome, the URL bar will look like this:



Note that “https” is in red. That is because our certificate is self-signed, so Chrome does not trust it. In fact, the first time that we attempt to access the website a warning will be displayed; we can just ignore it.

By following these simple steps, we have made our applications more secure by using the HTTPS protocol.