Chapter 3

<<Securing Console Access>>

As shipped, the Controller's web server — which presents the Scalent Console through your web browser — listens on both an insecure port (http-alt, port 8080) and a secure port (https, port 8443). You may choose to require all communications go through the secure port. This will prevent eavesdropping when connecting from outside your corporate network, such as in a hotel or cafe hot-spot.

The Controller asserts its identity to your web browser by presenting credentials. You may choose to replace the default credentials with ones more appropriate to your enterprise.

Requiring secure communications

The Controller is set to communicate with your web browser over http-alt (port 8080), the alternative web traffic port; URLs will start with

```
http://the.controller.machine:8080/
```

Instead of being broadcast as cleartext, where it may be intercepted, you may choose to have the communications encrypted with Secure Sockets Layer (SSL) over https (port 8443).

Edit the file `/opt/scalent/etc/controller.properties`, adding a line

```
com.scalent.webserver.constraintLevel=2
```

This will direct the Controller's behavior from the default — accepting communication on both the http-alt (8080) and https (8443) ports — to forcing all traffic onto the secure port. (The value `0` restores the default behavior.)

Once you restart the Controller

```
# reboot
```

you'll access the Controller with a modified URL

```
https://the.controller.machine:8443/
```

Replacing the web server's credentials

The Controller asserts its identity by presenting security credentials — also known as digital certificates — to your web browser. You may choose to replace the default credentials with ones more appropriate to your enterprise. This may prevent both spoofing and man-in-the-middle security attacks.

Between two or more companies which network their computers a well-known, trusted, third-party Certificate Authority (CA) signs the digital certificates. Within a company, however, you may act as your own CA.

Generating & Signing Certificates with OpenSSL

You will generate, certify, and distribute security credentials to the web server (and, in general, to any other computers in your network of trust). You may do with with OpenSSL, a robust, commercial-grade, full-featured, open-source implementation of the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols along with a full-strength general-purpose cryptography library. The `CA.sh` script greatly simplifies the traditional steps required to create a Certificate Authority and user certificates. OpenSSL may be obtained from http://www.openssl.org/.

Creating a root certificate for your CA

Once you've installed OpenSSL, generate a root Certificate Authority. This CA will be the guarantor for the identities of the user certificates you make.

```
# export PATH=/usr/local/ssl/bin:/usr/local/ssl/misc:$PATH
# cd /usr/local/ssl
# CA.sh -newca
```

The root certificate of this CA — which will verify the certificates signed by this CA — is stored in the file `./demoCA/cacert.pem`.

Generating a user certificate

Now that the root CA is done, generate a user certificate. First make a user certificate request.

```
# CA.sh -newreq
```

Then sign the user certificate request with the root CA's private key, certifying it.

```
# CA.sh -sign
```

The signed certificate is in the file `./newcert.pem`.

Adding the certificates to the web server's keystore

[WARNING: There's a digital certificate disconnect between the OpenSSL way of doing things and the Java way. After trying to make the leap from the former to the latter I've uncovered a Java-only way which should (hopefully) work. I'll email it around tomorrow and document it here.]

The web server needs both the signed user certificate and the root CA certificate added to it's keystore.

```
# /opt/scalent/jre/bin/keytool -import -keystore /opt/scalent/etc/WebServer.jks
  -alias webserver
  -file ./demoCA/cacert.pem
```

You must also add the CA's own root authority certificate to the keystore:

```
# keytool -import -keystore -keystore /opt/scalent/etc/WebServer.jks
  -alias webserver
  -file  /etc/certs/webserverCert.pem
```

One final security note: the keystore's default password is `test123` and should be changed with the command

```
# keytool -storepasswd -keystore /opt/scalent/etc/WebServer.jks
```

Edit the file `/opt/scalent/etc/controller.properties` and change the property to reflect the new password:

```
scalent.messaging.keystore.password=theNewPassword
```

Verifying the new certificates are being used

[What's the accepted Scalent way of connecting and seeing that a certificate is being used? Being prompted to view and accept it upon first cornnection? Should we provide a screen capture of this? Just a textual description? --Mickey]