Installing Libresonic in a Secured Tomcat 8 Environment

Assumptions: This document describes how to install Libresonic in a FreeBSD 11.0 Release P7 using a .WAR package under Tomcat 8. It also illustrates steps necessary to secure Tomcat in order to limit Tomcat vulnerabilities and to prevent interception of web traffic. The setup under a Linux distribution should be similar with the most obvious differences being the FreeBSD package manager "pkg", the default FreeBSD non-root Tomcat user "www", and service/daemon intialization differences between various platforms. There may also be some differences in Tomcat directory paths. Finally, the use of the "sudo" command may also be required for distros that do not allow a user to become root.

1. Install Libresonic

Obtain root privledges and install Libresonic dependancies. Tomcat version 8 is assumed for Libresonic 6.x. Tomcat will also install or update the current version of OpenJDK.

```
$ su
# pkg update     ;this will also install pkg if it has not been used yet
# pkg install tomcat8 wget
```

NOTE: For Java, fdesc and proc must be mounted. Follow the instructions to check for and mount these resources if they are not mount already.

Install a text editor: A text editor will be required to edit the Tomcat config files. There are many such editors, but Nano is popular, straightforward, and easy to use. A different editor may be substituted if desired.

```
# pkg install nano
```

Optional -- install root certificates: The official Libresonic repository resides on an HTTPS server. You can either install the root certificates or use the `--no-check-certificate` switch when downloading with wget.

```
# pkg install ca_root_nss
```

Next download the Libresonic .WAR and integrate it with Tomcat. Example below downloads the latest version from the Libresonic repository on GitHub, which is 6.2 beta 1.

```
# cd /usr/local/apache-tomcat-8.0/webapps
# wget https://libresonic.org/release/libresonic-v6.2.beta1.war
# mv libresonic-v6.2.beta1.war libresonic.war
```

Create the libresonic directory and assign ownership to the Tomcat system user (www by default on FreeBSD)

```
# mkdir /var/libresonic
# chown www:www /var/libresonic
```

Set Tomcat to autorun at system service at startup.

```
# echo tomcat8_enable="YES" >> /etc/rc.conf
```

Now start Tomcat manually. On the first run it will build out the .WAR packages which can take up to 30 seconds.

```
# service tomcat8 start
```

When you open a browser and navigate to http://192.168.1.212:8080, you should see a Tomcat welcome screen. Assuming this was successful, try Libresonic at http://192.168.1.212:8080/libresonic. You should see the LIbresonic login screen. If this was successful, stop Tomcat.

```
# service tomcat8 stop
```

2. Secure Tomcat

Provided Libresonic will be accessed through the internet, secure Tomcat. There are many features we don't need and there are some changes that can be made to help defend against malicious attacks. This includes disabling built in apps and setting up https connections. All changes assume a fresh Tomcat install and that none of the disabled features are required for any other web applications (i.e. Tomcat serves Libresonic only). If your setup is different, these changes may not be appropriate in some cases, and additional changes may be required for a non-fresh Tomcat8 install. The following instructions are based on those found at: http://security-24-7.com/hardening-guide-for-tomcat-8-on-redhat-6-5-64bit-edition/

see also https://www.upguard.com/articles/15-ways-to-secure-apache-tomcat-8

```
# cd /usr/local/apache-tomcat-8.0   ; All commands below assume you are at this
                                    ; location...do not cd from this location!
```

Steps 1-8, 10: These lines install Tomcat & Java as a non-root user. If installed from the FreeBSD package manager or ports Tomcat should be configured automatically as user "www". Other OS's may be different. To verify you can restart the Tomcat8 server to reveal what Tomcat is running as:

```
root@SubEtha:/ # sockstat
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN ADDRESS
www      jsvc       39766 49 tcp4   192.168.1.7:8080      *:*
www      jsvc       39766 50 stream (not connected)
www      jsvc       39766 54 tcp4   192.168.1.7:8009      *:*
```

Step 9: These lines remove the default applicaitons bundled with Tomcat. Most are harmless but the manager in particular can allow an outside user to reconfigure the Tomcat server remotely. Otherwise disabling these apps simply prevents vulnerabilities, known or unknown, from being exploited. If any of these apps are absolutely required, at the minimum they should be secured with a strong password (not the scope of this document). For Libresonic, none are required, so remove everything from the webapps and work directories. Incase we need them later, we'll simply move them to a location outside of Tomcat's scope.

```
# mkdir ../tomcat8webapps
# mv webapps/* ../tomcat8webapps
# mv ../tomcat8webapps/libresonic webapps/libresonic      ;get Libresonic back...

# mkdir ../tomcat8work
# mv work/Catalina/localhost/* ../tomcat8work
# mv ../tomcat8work/libresonic work/Catalina/localhost/libresonic
```

Steps 11 - 16: These lines remove the version string from HTTP error messages by repacking `CATALINA_HOME/server/lib/catalina.jar` with an updated `ServerInfo.properties` file. This will make it harder to know what vulnerabilities may exist in Tomcat since it will not advertise itself or it's version number. The file that needs to be modified is inside a jar file that must be unpacked, modified, then repacked.

```
# jar xf lib/catalina.jar org/apache/catalina/util/ServerInfo.properties
  ;Extract to org directory
# nano org/apache/catalina/util/ServerInfo.properties
```

Change From:

```
server.info=Apache Tomcat/8.0.39
server.number=8.0.39.0
server.built=Nov 9 2016 08:48:39 UTC
```

Change To (something generic but plausible):

```
server.info=Secure Web Server
server.number=1.0.0.0
server.built=Jan 1 2015 14:13:12 UTC
```

Save the file and exit nano. Now repackage the jar file and remove the `org` working directory:

```
# jar uf lib/catalina.jar org/apache/catalina/util/ServerInfo.properties
# rm -R org
```

Step 17: These lines edit the connector (identifies ports that Tomcat "listens" to) to disable TRACE requests (a known attack vector) and XPOWEREDBY headers (identifies server application and version).

Change From:

```
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

To:

```
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               xpoweredBy="false"
               allowTrace="false"
               redirectPort="8443" />
```

(Note there is also a Connector Executor block immediately below this, however it is commented out and so can be ignored)

This step also disables the ability to remotely shutdown the Tomcat server by disabling the shutdown port:

Change From: `<Server port="8005" shutdown="SHUTDOWN">`

To: `<Server port="-1" shutdown="SHUTDOWN">`

Finally, this change disables the ability to run an injected conext.xml which can result in inceased priveleges to a

web application allowing an attacker to potentially gain control of the applicaiton or host system.

Change From: `autoDeploy="true">`
To: `autoDeploy="false">`

*** I am not certain but this may need to be temporarilly changed back to true when a new .WAR is downloaded ***

Save the updated configuration file and exit.

Step 18-19: These steps disable the built in error page which due to formating makes it somewhat obvious the server probably Tomcat, even if other details are obsured. This further obfuscates the underlying system under the theory that a server that cannot be identified will not have easilly identified attack vectors.

```
# mkdir webapps/ROOT
# nano webapps/ROOT/error.jsp
```

Create the new, generic error page in HTML within nano:

```
<html>
   <head>
      <title>404-Page Not Found</title>
   </head>
   <body> The requested URL was not found on this server. </body>
</html>
```

Save the error file and exit. Change the owner & group to the Tomcat user (www:www on FreeBSD):

```
# chown -R www:www webapps/ROOT
```

Now change the following to call the custom error page. Place these lines within the web-app tag (after the welcome-file-list tag at the end of the file is fine). HINT: CTRL-V in nano is page down.

```
# nano conf/web.xml
```

Change From:

```
    </welcome-file-list>
</web-app>
```

To:

```
    </welcome-file-list>

  <error-page>
    <error-code>400</error-code>
    <location>/error.jsp</location>
  </error-page>
  <error-page>
    <error-code>401</error-code>
```

```
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>403</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>404</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>405</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>410</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>411</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>412</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>413</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>408</error-code>
  <location>/error.jsp</location>
</error-page>
<error-page>
  <error-code>500</error-code>
  <location>/error.jsp</location>
</error-page>

<security-constraint>
  <web-resource-collection>
    <web-resource-name>HTMLManger and Manager command</web-resource-name>
    <url-pattern>/jmxproxy/*</url-pattern>
    <url-pattern>/html/*</url-pattern>
    <url-pattern>/list</url-pattern>
    <url-pattern>/sessions</url-pattern>
    <url-pattern>/start</url-pattern>
    <url-pattern>/stop</url-pattern>
    <url-pattern>/install</url-pattern>
```

```
      <url-pattern>/remove</url-pattern>
      <url-pattern>/deploy</url-pattern>
      <url-pattern>/undeploy</url-pattern>
      <url-pattern>/reload</url-pattern>
      <url-pattern>/save</url-pattern>
      <url-pattern>/serverinfo</url-pattern>
      <url-pattern>/status/*</url-pattern>
      <url-pattern>/roles</url-pattern>
      <url-pattern>/resources</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>manager</role-name>
    </auth-constraint>
  </security-constraint>

</web-app>
```

Step 20-23: These steps appear to be configuring Tomcat to run as a service. The equivalent step for FreeBSD was previously performed.

Step 24-27: These steps appear to configure the system's firewall (IPTables in Red Hat Linux) for ports 22 and 8080. For Libresonic we will configure in the next section to use the subsonic "legacy" ports of 4040 and 4443. Be sure these ports are open in your system's firewall software. DO NOT open 22 and 8080 unless they are actually needed.

SSL Configuration: ****If Libresonic is configured on a system that provides multiple web services, it is probably better to setup a central HTTPS proxy server as documented elsewhere. This allows a single SSL/TLS certificate to be maintained and has some other advantages. The following is best applied to a single purpose server that only serves Libresonic.****

For this phase we will only set up a self signed certificate. Certificates signed by a certificate authority are a good idea, especially if your server is accessed by multple users. Provided you can trust yourself though, a self signed certificate is probably sufficient.

Enable HTTPS connections by creating a self signed certificate and enabling HTTPS in the Tomcat server.xml file:

```
# ../bin/keytool -genkey -alias tomcat -keyalg RSA -sigalg SHA256withRSA -keysize
2048 -keystore /usr/local/apache-tomcat-8.0/.keystore
```

You will first be prompted to create a keystore password. Next, you will be prompted for general information about this certificate, such as company, contact name, and so on. This information will be displayed to users who attempt to access a secure page in your application, so make sure that the information provided here matches what they will expect.

Finally, you will be prompted for the key password, which is the password specifically for this Certificate (as opposed to any other Certificates stored in the same keystore file). The keytool prompt will tell you that pressing

the ENTER key automatically uses the same password for the key as the keystore. You are free to use the same password or to select a custom one.

If everything was successful, you now have a keystore file with a self signed Certificate that can be used by Tomcat. Now, lets configure Tomcat to enable and use "legacy" Subsonic ports of 4040 for HTTP and 4443 for HTTPS (this also incoprates step 17 into the HTTPS connector statement):

```
# nano conf/server.xml
```

Activate the HTTPS protocol and the HTTPS port to 4443:

Change From:

```
<!--
      <Connector port="8443"  protocol="org.apache.coyote.http11.Http11NioProtocol"
      maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
      clientAuth="false" sslProtocol="TLS" />
-->
```

To (Note that the comment block has been deleted to activate this section):

```
      <Connector port="4443"  protocol="org.apache.coyote.http11.Http11NioProtocol"
      maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
      xpoweredBy="false" allowTrace="false"
      keystoreFile=".keystore" keystorePass="<<password from keystore generation>>"
      clientAuth="false" sslProtocol="TLS" />
```

Also change the HTTP port from the default 8080 to port 4040. Set the HTTPS redirect port to 4443

Change From:

```
    <Connector port="8080"
               < ... >
               redirectPort="8443" />
```

To:

```
    <Connector port="4040" protocol="HTTP/1.1"
               < ... >
               redirectPort="4443" />
```

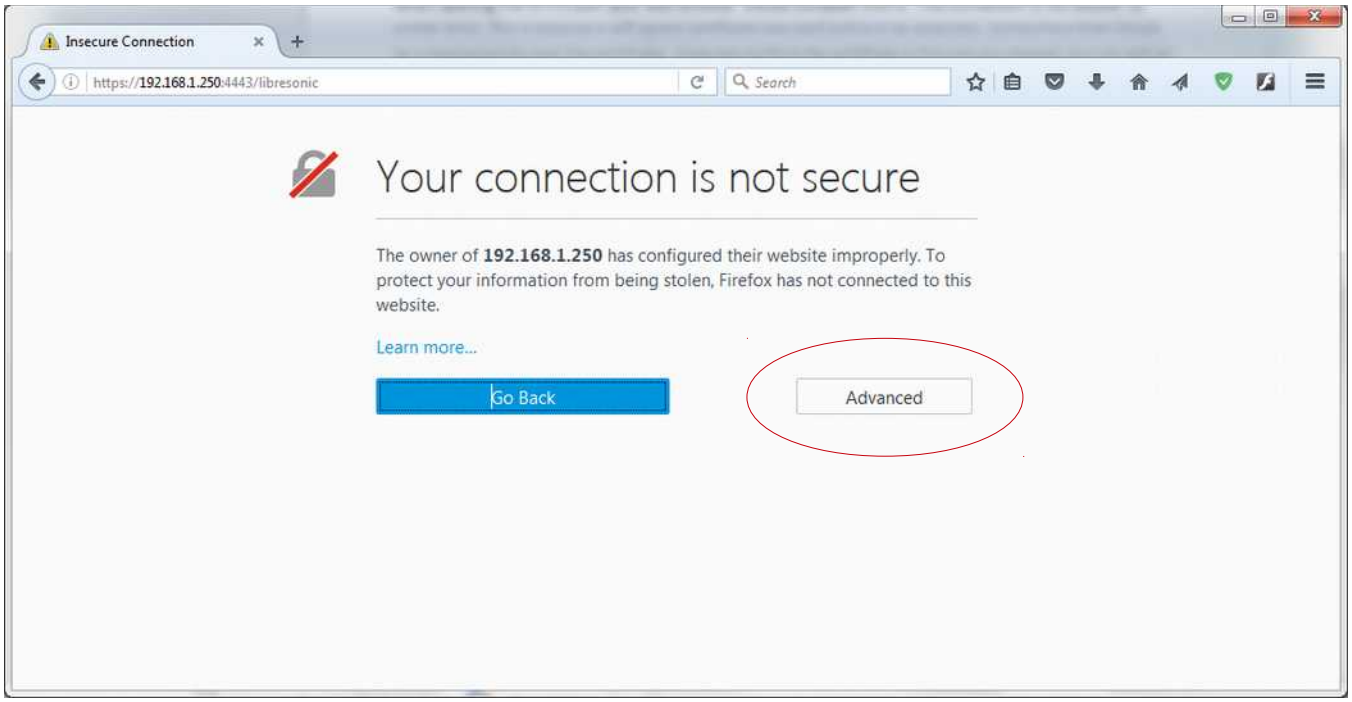
Finally, start Tomcat and test with a web brower. If the web browser is on the same computer as Tomcat, then substutute 127.0.0.1 as the IP address in the browser address bar. Both HTTP (port 4040) and HTTPS (4443) ports should be functional at this point.

http://127.0.0.1:4040/libresonic

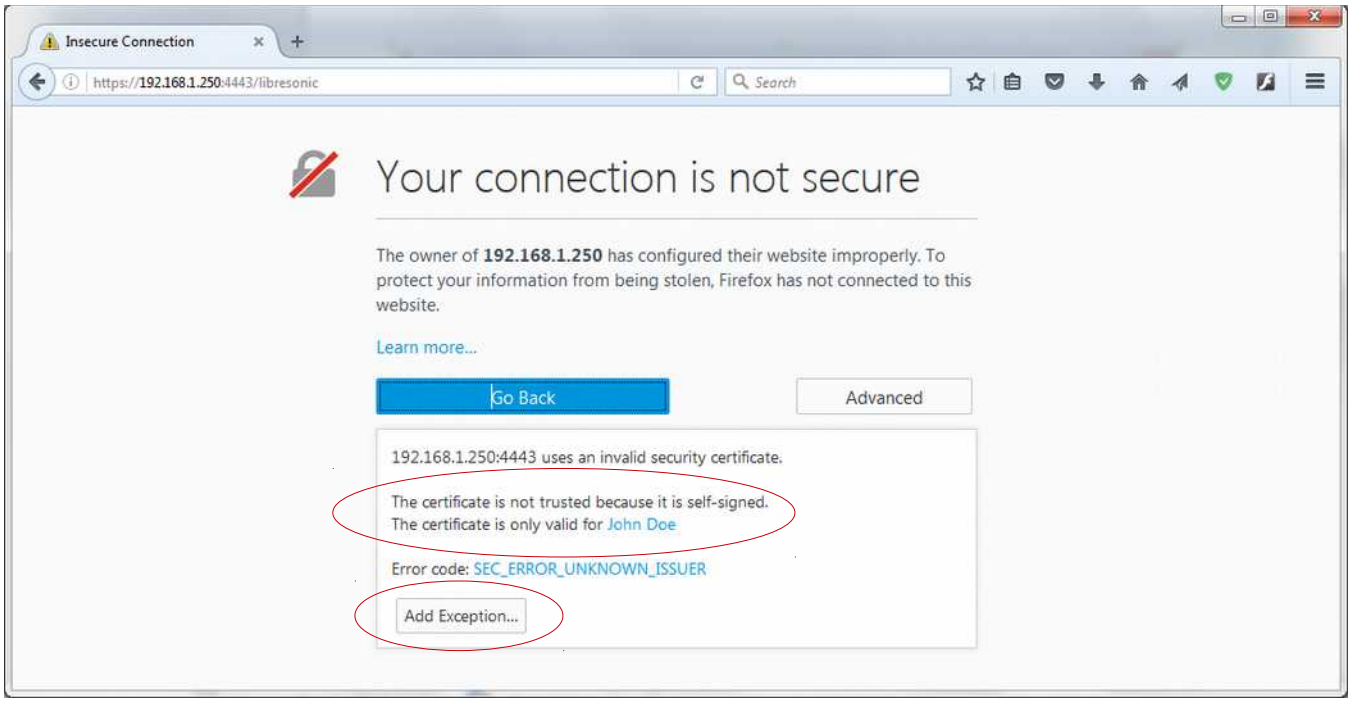
When opening the HTTPS port your web browser should complain with a "This connection is not secure" or similar error. This is because a self signed certificate was used and is to be expected. Somewhere their should

be a mechanism to view the certificate. Once you confirm the certificate is the one you created, you can add an exception to the browser to allow the conneciton. On Firefox, the sequence will go something like this:

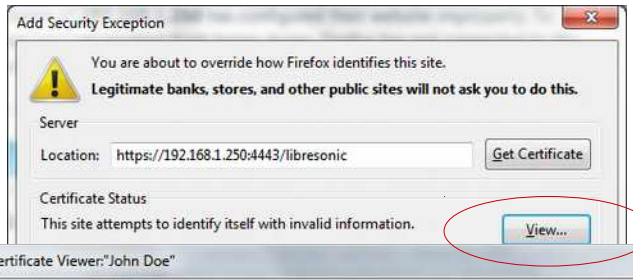
https://127.0.0.1:4443/libresonic



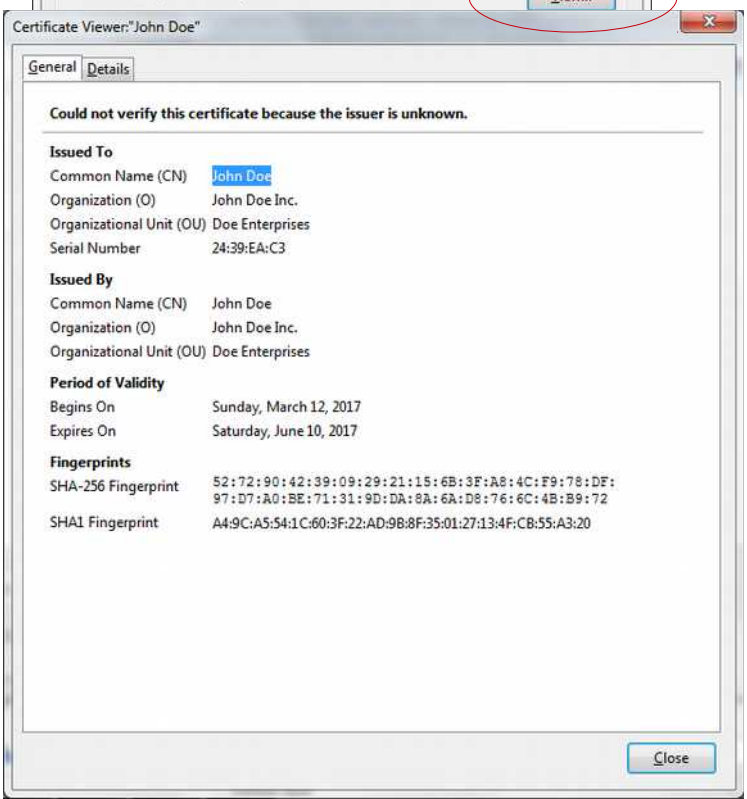
Click advanced



You will now see some certificate details, where you can see the certificate is self signed as well as who signed it. If this information is legitimate, you can start to add an exception to avoid this error on future visits to your libresonic server. Click “Add Exception”



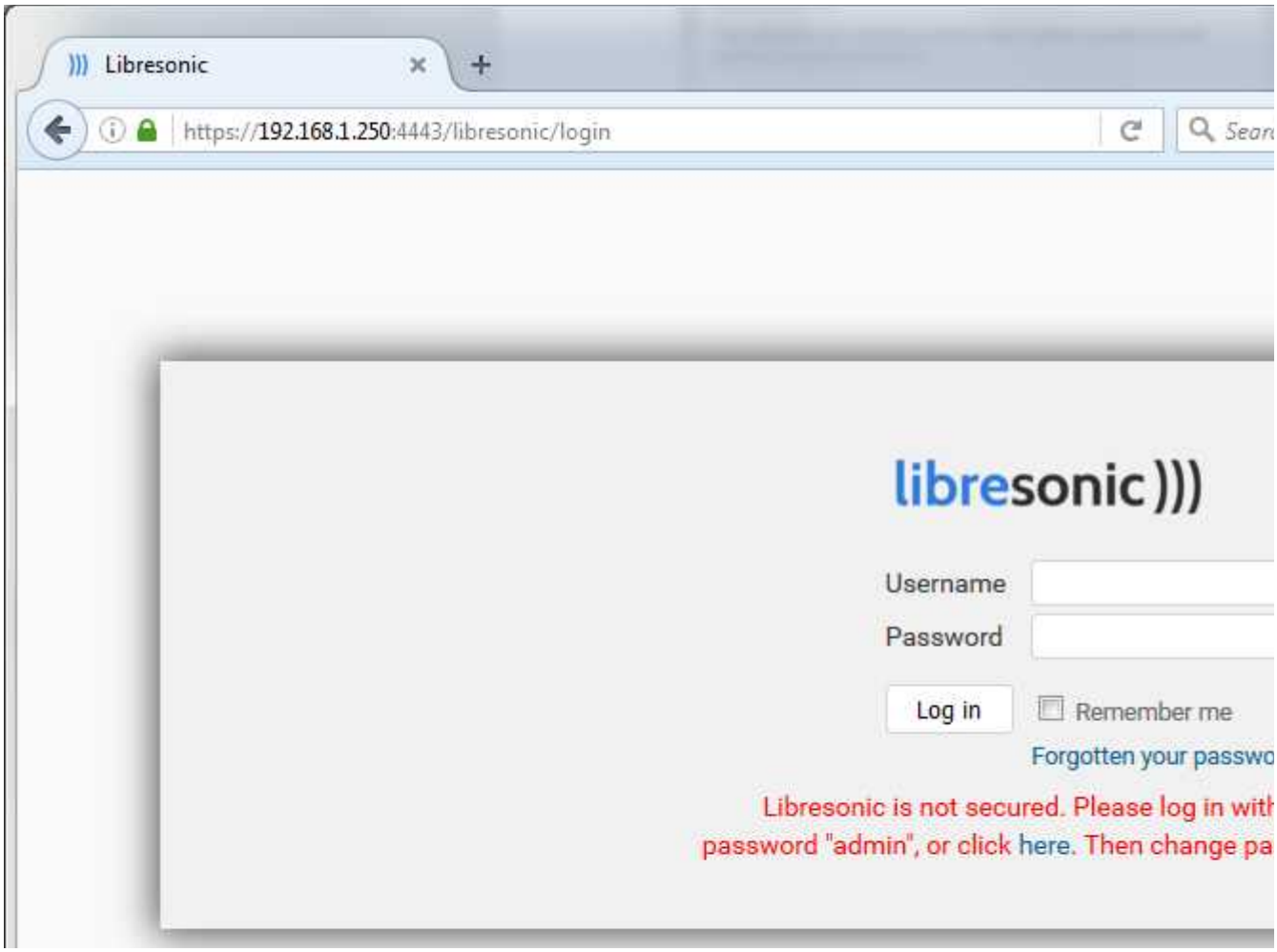
Additional details of the certificate can be viewed by clicking “View”...



The data shown here should match what was entered when the certificate was created. If it does not, DO NOT PROCEED ANY FURTHER.

Assuming what is shown matches your expectations, this is your self signed certificate. Close this window and click “Confirm Security Exception” in the window above.

The Libresonic login page now appears in https:


Libresonic
https://192.168.1.250:4443/libresonic/login
libresonic)))
Username
Password
Log in
Remember me
Forgotten your passwo
Libresonic is not secured. Please log in with
password "admin", or click here. Then change pa