Using Secure Socket Layer with Tethys

Tethys, Antioch mosaic, 3rd century from Baltimore Museum of Art

Marie A. Roch, San Diego State University/Scripps Institution of Oceanography

Contents

Introduction

Tethys can be run over secure socket layer (SSL) which provides increased security by encrypting the data stream and by verifying that the Tethys server a client is communicating with is the correct one. Setting up Tethys for secure socket layer transport requires a certificate that is stored in the database directory. Certificates are used to establish trust between computers in a network. Your system administrator can help you obtain a certificate or you can create one on your own.

By far the easiest way to use a certificate is to have one issued by a certificate authority. If your organization does not have a relationship with a certificate authority, you can purchase a certificate from many vendors, but these vendors typically charge an annual sum. Consequently, many people choose to issue self-signed certificates. These are certificates that an individual creates. The problem with such certificates is that there is no way for other machines to automatically trust them. Steps must then be taken to inform Tethys clients that the self-signed certificate is valid. We outline methods for setting up Tethys to work with both types of certificates. Note that starting Tethys in secure socket mode without having installed a certificate will cause the server to fail.

Certificates rely on a public-private key system. One party holds the private key which must be kept secret and anyone else can have the public key. The pair of keys are used together to provide secure encryption.

Certificate authority issued certificate

Obtain a certificate from your certificate issuing authority. This may be your organization which may have a certificate authority for its domain, or could be an external certificate issuing authority that will typically charge a service fee.

Copy your certificate and private/public keys files to the database with which they are to be used, e.g. c:/Users/tethys/metadata. Start the server in secure socket mode. They must be named as follows:

private/public keys → `host.key`
certificate → `host.crt`

That is it – you are done!

Self-Signed certificate

There are a number of tools that can be used to generate self-signed certificates, we will use OpenSSL (http://www.openssl.org/), a freely available secure socket library and management tool. OpenSSL is bundled with most linux systems and can be downloaded as source code or a Windows binary at the OpenSSL web site. The steps to execute in this section are assumed to be done at a command line prompt and use a `console font`.

Note that there can be trust issues with self-signed certificates. Clients connecting to servers with self-signed certificates usually either prompt users with warnings or simply deny service. Most operating systems provide a mechanism for granting trust to such certificates, but you will likely need your information technology staff's assistance if you do not have administrative privileges on the machines involved.

Generating key files and certificates

Our self-signed certificate will rely on a public/private key pair. We will be generating a 2048 bit key using the RSA algorithm. These keys are stored in the privacy enhanced mail (PEM) format, a standard for public key representation also used for secure electronic mail.

```
openssl genrsa -out host.key 2048
```

This generates the key file host.key that contains both the private and public keys.

Next we issue the certificate using a configuration file host.cnf that can be found in the server/docs/examples folder relative to Tethys's base install folder (usually c:\Program Files\Tethys or c:\Program Files (x86)\Tethys for Windows systems). The host.cnf will have to be copied to the same folder as your key file or the entire path to the configuration file should be specified.

```
$ openssl req -new -x509 -key host.key -out host.crt -days 1500 –config
      host.cnf


      You are about to be asked to enter information that will be incorporated
      into your certificate request.
      What you are about to enter is what is called a Distinguished Name or a DN.
      There are quite a few fields but you can leave some blank
      For some fields there will be a default value,
      If you enter '.', the field will be left blank.
```

The following are prompts that should be answered with information relevant to your organization

Country Name (2 letter code) [US]: <provide country code>
State or Province Name (full name) [CA]:<provide state/province>
Locality Name (eg, city) [San Diego]:<provide city>
Organization Name (eg, company) [San Diego State University]:<provide organization>
Organizational Unit Name (eg, section) []:<provide unit name>
localhost []:<provide host name, e.g. localhost, tethys.mydomain.org>

Installing certificates

The host.crt is your certificate, and host.key contains both your private and public keys. Both of these should be copied to the base folder of your database, e.g. c:\Users\Tethys\metadata. The server will look for both of these files when starting.

When the server accesses this certificate, the client needs to verify that the certificate is trusted. This can be done in a variety of ways. Some clients may access the operating system's certificate trust, in which case the certificate can be stored there. Others may manage a private certificate trust, and the certificate must be installed in the private trust.

Certificates for Matlab

Matlab comes with a version of Java. Even if you have Java already installed, Matlab will install its own version of Java with the exception of certain versions of Apple's MacOS. Once Tethys is installed and the server is running in secure socket layer mode, the certificate can be installed with the following steps typed at a Matlab prompt assuming that the environment has been configured for Tethys and you have executed dbInit() (see Tethys documentation or the Matlab cookbook for setting up Matlab as a Tethys client):

```
import dbxml.InstallCert  % Make the install certificate class available
```

Request the server's certificate. In this case, we will assume that the client and server are running on the same machine, but in general we would want to substitute a more general server name (e.g. tethys.your.org)

```
c = InstallCert('localhost:9779')
```

If the certificate has not yet been installed (it only need be installed once for as long as the certificate is valid), you will see an error indicating that the client was unable to find a valid certification path. If not, you will see a message that the certificate is trusted and you need not do anything more.

To install the certificate (if there was an error), request a list of valid certificates:

```
c.ShowChain()
```

This will display all the certificates associated with the server (only one in our case). Although the data will be different as this is for a specific host and private key, it should have the following format:

```
Server sent 1 certificate(s):
 0 Subject CN=localhost, OU=San Diego State University, L=San Diego, ST=CA, C=US
   Issuer  CN=localhost, OU=San Diego State University, L=San Diego, ST=CA, C=US
   sha1    d6 69 f0 c1 c8 4e c5 50 be 1f 90 b6 08 9c 44 8e b6 d6 aa b6
   md5     42 31 4e 83 22 74 b0 38 6f b3 f4 ea 67 b4 95 c7
```

The first number associated with each certificate (0 in this case) is an index into the list of certificates. To install the certificate 0, execute:

```
c.InstallIndex(0)
```

This will result in the creation of a file called `jssecacerts` in current folder. This contains all of the certificates that were in the Java certificate trust and the newly installed certificate. We next have to place this file in a location that Matlab will find it.

The Matlab function `matlabroot` will indicate where Matlab is installed. Matlab's Java certificates are located relative to this folder in `sys\java\jre\win64\jre\lib\security` (win64-->win32 for 32 bit systems). Begin my making a copy of the cacerts file. Then copy jssecacerts into this folder and rename it cacerts.

Certificates for Java

Adding certificates for Java also uses the CertInst class, but has few steps. In this example, we will assume that the current directory is writable, and that the Java client has been installed.

Open a command line window. Begin by setting a variable that references the root folder of the Tethys Java client (modify for your installation, e.g. Program Files (x86) for 32 bit installs):

```
REM   Do not use quotes around the path
set JCLIENT=c:\Program Files\Tethys\client-java\classes
```

Make sure that your current folder is one that is writable and that the Java virtual machine (java.exe) is in your path. Then execute the following, substituting localhost:9779 with the name of your server and the port it is running on (default 9779):

```
java -classpath %JCLIENT% dbxml/InstallCert localhost:9779
```

If the certificate is installed, you will see a message indicating that this is the case and there is nothing to do. Otherwise an error will occur and you will be shown the certificates that were found in the same format as outlined in the section for Matlab clients:

```
Server sent 1 certificate(s):
 0 Subject CN=localhost, OU=San Diego State University, L=San Diego, ST=CA, C=US
   Issuer  CN=localhost, OU=San Diego State University, L=San Diego, ST=CA, C=US
```

```
sha1    d6 69 f0 c1 c8 4e c5 50 be 1f 90 b6 08 9c 44 8e b6 d6 aa b6
md5     42 31 4e 83 22 74 b0 38 6f b3 f4 ea 67 b4 95 c7
```

Next, you will be asked which certificate to import. In most cases, there will only be a single certificate (certificate 0), so enter 0. A `jssecacerts` file will be created.

For the next step, you will need administrator rights on your machine or need to have an administrator assist you. Navigate to the directory where Java stores it certifications. On Windows installations, this is typically C:\Program Files\Java\jre7\lib\security (possibly Program Files (x86)). Rename the `cacerts` file. Copy `jssecacerts` created earlier to `cacerts`.

Certificates for other clients

Many other clients are more forgiving, for example typing https://localhost:9779//Detections into Internet Explorer gives you a warning that the certificate cannot be verified and then allows you to proceed at your own risk, or even install the certificate.