SSL Certificates

This section describes how to use SSL Certificates with SOA Gateway running on Linux.

Setup

To enable the use of SSL Certificates for your SOA Gateway, openssl must be installed.

SSL support is not automatically built into SOA Gateway due to export restrictions in certain countries. Please contact your SOA Gateway representative to get access to a SSL enabled version.

Introduction

The Apache module mod_ssl provides strong cryptography for the SOA Gateway via Secure Sockets Layer (SSL v2/v3) and the Transport Security Layer (TLS v1) protocols with the help of the SSL/TLS implementation library openssl. This section will help you to secure your SOA Gateway using mod_ssl. In order to run a secure server, you need a private key and a certificate for the server. In a commercial environment, it would be advisable to purchase a signed certificate from a well-know Certificate Authority (CA), such as Thawte or Verisign. For the purpose of this section, we will become the CA and generate our certificates using the openssl toolkit. Some terms used in this section are outside of the scope of the documentation, and will not be explained in detail. For more information on SSL, and corresponding keys or certificates, see here

Step 1: Set up your own CA

Firstly we will setup our own CA, and generate a certificate and a key that can be used to sign other certificates.

Generate the key, entering a password when prompted:

```
openssl genrsa -des3 -out myCa.key 2048
```

Generate the X.509 certificate:

```
openssl req -new -x509 -key myCa.key -out myCa.crt
```

Enter the password you added when creating the key (when prompted).

Enter the information would you like to appear on your CA certificate. You should now have your CA key, myCa.key, and a CA certificate, myCa.crt in the current directory.

Optionally you may view the certificate by typing the command

```
openssl x509 -in myCa.crt -text -noout
```

Step 2: SOA Gateway Server key and certificate

This step will create a key and certificate for the SOA Gateway server.

Rather than creating a certificate directly, we will create a certificate request, then use the CA key we made in Step 1 to sign the server certificate.

Generate the key, entering a password when prompted:

```
openssl genrsa -des3 -out asg-server.key 1024
```

Generate the server certificate request

```
openssl req -new -key asg-server.key -out asg-server.csr
```

Sign the certificate request with out CA information and generate our server certificate. Note: For this certificate, the “Common Name” should be the hostname of the server this certificate is going to be used on.

```
openssl x509 -req -in asg-server.csr -out asg-server.crt -sha1 -CA
myCa.crt -CAkey myCa.key -CAcreateserial -days 3650
```

Optionally, you can view the server certificate you’ve created with the command:

```
openssl x509 -in asg-server.crt -text -noout
```

You should see that the Certificate issuer is your CA Company.

Server Configuration

Before importing the key and certificate into the SOA Gateway server, we need to enable SSL support. Choose next section depending on your system.

SSL Configuration

The SSL configuration file is should be located in your SOA Gateway installation directory. Your SOA Gateway representative will provide you with a version of this file to suit your system. The following is an example:

```
<IfDefine SSL>
<IfDefine !NOSSL>
<IfModule mod_ssl.c>

AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

SSLPassPhraseDialog  builtin
```

```
SSLSessionCache         shmcb:/usr/local/soaGateway/apache2/logs/ssl_scache
SSLSessionCacheTimeout  600

SSLMutex  sem
#SSLMutex file:/usr/local/soaGateway/apache2/logs/ssl_mutex

SSLRandomSeed startup builtin
SSLRandomSeed connect builtin

<VirtualHost _default_:443>

        DocumentRoot "/srv/www/htdocs"
        ErrorLog /usr/local/soaGateway/apache2/logs/error_log
        TransferLog /usr/local/soaGateway/apache2/logs/access_log
        ServerName <<hostname>>

        SSLEngine on

        SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

        SSLCertificateFile /usr/local/soaGateway/apache2/certs/asg-server.crt
        SSLCertificateKeyFile /usr/local/soaGateway/apache2/keys/asg-server.key

        SSLCACertificateFile /usr/local/soaGateway/apache2/certs/myCa.crt

        <Files ~ "\.(cgi|shtml|phtml|php3?)$">
                SSLOptions +StdEnvVars
        </Files>
        <Directory "/srv/www/cgi-bin">
                SSLOptions +StdEnvVars
        </Directory>

          SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown downgrade-1.0 force-response-1.0

          CustomLog /usr/local/soaGateway/apache2/logs/ssl_request_log   ssl_combined

</VirtualHost>

</IfModule>
</IfDefine>
</IfDefine>
```

For the ServerName <<hostname>> directive ensure <<hostname>> is the hostname of your machine. This should match the “Common Name” of the asg-server.crt created earlier.

Take the “asg-server.crt” and copy it into /usr/local/soaGateway/apache2/certs/ directory Take the “asg-server.key” and copy it into the /usr/local/soaGateway/apache2/keys directory.

Stop Apache (apache2ctl stop). Start Apache with SSL support (apache2ctl sslstart). Enter the pass phrase for the server key.

Open a browser and enter “https://<hostname>:<port>/configurationService?WSDL” where hostname and port are location where your SOA Gateway is running.

You should be asked do you wish to accept the certificate. Click “Accept”. The WSDL will be downloaded to the browser over a secure connection.

To disable SSL support on this SOA Gateway, stop the server (apache2ctl stop) and start the server normally (apache2ctl start)

Troubleshooting

- Cannot connect to https:// page

- Ensure the “Include ssl.conf” directive has been added to httpd.conf
- Ensure that you have started apache with the “sslstart” parameter
- Check Apache logs for error (see /usr/local/soaGateway/apache2/logs/*)
- Ensure that you have connectivity to that particular hostname.

- Function not implemented: Cannot create SSLMutex
  - Change the SSLMutex directive to “file:/usr/local/soaGateway/apache2/logs/ssl_mutex”

Example

The following is a PHP program to connect to an SSL enabled web service provided by SOA Gateway. Note: You must have openssl support in your PHP installation. To check if you do, run the following PHP program.

<?php phpinfo(); ?>

You should check the “configure command” section. If there is no –with-openssl option, then you need to download PHP and build the requirements into it. See instructions here

This example uses an SSL enabled endpoint (https://) the user name and password set up earlier in the documentation.

If this username and password are not required, remove the array type from the soapClient constructor. E.g. $soapClient = new SoapClient(https://localhost:8080/adabas_QE_Employees?WSDL);

```
<?php

ini_set( "soap.wsdl_cache_enabled", 0);

$soapClient = new SoapClient(
        "https://localhost:8080/adabas_QE_Employees?WSDL",
        array('login'=>"asg", 'password'=>"boston1") );

$adabasEmployeeGetKey = array('personnel_id'=>50005000);

try{
        $results = $soapClient->get($adabasEmployeeGetKey);
}
catch( Exception $e){

        print "An exception occurred!\n";
        print "Code : ";
        print_r( $e->faultcode);

        print "\nString : ";
        print_r( $e->faultstring);

        print "\n ";

        exit;
}
```

```
print_r($results);

?>
```

Client verification using SSL

This section outlines how to create and use a SSL client certificate. This certificate must be digitally signed by the CA that the server trusts, and the user must import the certificate into their web service client program. We will use the OpenSSL toolkit to create this client certificate.

Step 1: Generate client key and certificate

Generate the client's key:

```
openssl genrsa -des3 -out asg-client.key 1024
```

Generate the client's certificate request:

```
openssl req -new -key asg-client.key -out asg-client.csr
```

Sign (using "our" CA) and generate the client's certificate. Note: For this certificate, the "Common Name" should be the hostname of the server this certificate is going to be used on.

```
openssl x509 -req -in asg-client.csr -out asg-client.crt -sha1 -CA
myCa.crt -CAkey myCa.key -CAcreateserial -days 3650
```

Step 2: Generate the PKCS12 cert

The industry standard in client certificates is the Public Key Cryptography Standard 12 (PKCS12) encoding. These are binary files which again can be generated using the OpenSSL toolkit.

Generate the PKCS12 encoded certificate. The "export password" that is prompted for here is the password that the user needs to know when they import this certificate into the program.

```
openssl pkcs12 -export -in asg-client.crt -inkey asg-client.key -name
"SOA Gateway Client" -out asg-client.p12
```

You can optionally view the created certificate with the command:

```
openssl pkcs12 -in asg-client.p12 -clcerts -nokeys -info
```

Step 3: Apache Configuration

Apache must be configured to only allow clients who have the correct certificate. For the purposes of this example, we will only all the resource "secure_adabas_employees" to be accessed by a client with the correct certificate.

Perform the following steps:

- Edit the SOA Gateway Apache configuration file

- Enter the following directives.

```
<IfModule mod_xmiddle.c>
          <Location /secure_adabas_employees>
                    SSLVerifyClient require
                    SSLVerifyDepth 1
          </Location>
</IfModule>
```

- Restart the server

To test this, attempt to access this resource's WSDL. Open a browser and enter the following: https://<host>:<port>/secure_adabas_employees?WSDL" where <host> and <port> (if required) are the hostname and port your SOA Gateway is running on. You should be rejected by the server, and see a validation error message in Apache's error_log.

Step 4: Import Client Certificate

Firstly we will import the certificate into a browser and access the WSDL.

- Firefox
  - Tools -> Options -> Advanced -> Security -> View Certificates -> Import
  - Choose you PKCS12 client certificate and enter the password.
- Internet Explorer
  - Tools -> Internet Options -> Privacy -> Certificates -> Import
  - Choose the PKCS12 client certificate and enter the password

Now when you attempt to get the WSDL for secure_adabas_employees you should be able to accept the certificate signed by "our" CA company, and then view the WSDL.

If there are any errors in doing this, check Apache's error_log for messages. Also ensure that the certificate import has worked, and you are accessing the correct URL. Finally ensure that the <Location> directive in httpd.conf is correct. Remember this is case sensitive!

PHP Example

The following PHP example accesses the "secure_adabas_employees" resource, which has been secured above.

PHP will not accept a PKCS certificate. Instead, it requires a file containing both the x509 client key and cert. To create this file, copy asg-client.crt to a new file and append the contents of asg-client.key to asg-client.crt. These files will have been created in 1.5.8. E.g.

```
cat asg-client.crt > asg-newCert.crt

cat asg-client.key >> asg-newCert.crt
```

Or, on Windows, use Notepad.exe to create asg-newCert.crt..

Important:
There is a bug in Apache version 2.0.x which prevents this PHP example from working properly. This bug has been fixed in Apache version 2.2

```
<?

ini_set("soap.wsdl_cache_enabled", "0"); // disabling WSDL cache

$soapClient = new SoapClient(
        "https://lxbre/secure_adabas_employees?WSDL",
        array( 'local_cert'=> "asg-newCert.crt" ) );

$adabasEmployeeGetKey = array('personnel_id'=>50005000);

try{
        $results = $soapClient->get($adabasEmployeeGetKey);
}
catch( Exception $e){

        print "An exception occurred!\n";
        print "Code : ";
        print_r( $e->faultcode);

        print "\nString : ";
        print_r( $e->faultstring);

        print "\n ";

        exit;
}

print_r($results);

?>
```