SSL Sockets Programming Guide on Linux

1 Document Revision

Date	Version	Change Description
08/07/2014	1.0	Initial Version

2 Table of Contents

3 Introduction

A getting started guide to those who are new to SSL, Certificates and how it can be used to create a secure socket communication. For a secure socket communication a certificate with Public key and a private Key is needed. This app note will guide you to create a self-signed certificate, private key to create a SSL server-socket example in C.

3.1 Audience

Programmers who are familiar with sockets and wanted to learn how to make it secure.

4 How SSL works?

Secure Socket Layer protocol encrypts data before it is sent and decrypts at destination end using a public-private key asymmetric encryption algorithm. The data encrypted using a Public Key can be decrypted using its Private Key and vice versa. The protocol uses a third party, a Certificate Authority (CA), to identify one end or both end of the transactions.

The certificate contains the reference to the issuer, the public key of the owner of this certificate, the dates of validity of this certificate and the signature of the certificate to ensure this certificate hasn't been tampered with.

Usually this is how a SSL server-client communication takes place:-

- Client requests server for a new connection
- Server responds with its certificate having Public key.
- Client checks whether the certificate is issues by a trusted party (this process can be skipped)
- Client then uses public key to encrypt data & random symmetric encryption key to send it to server
- Server decrypts the symmetric encryption key using its private key and then uses symmetric key to decrypt the data.
- Server then responds with data as required which will be encrypted using this symmetric key
- This symmetric key will be used for all transactions during that session
- Once connection is restarted a new symmetric key will be generated and used.

5 Procedure

- Create a self-signed certificate
- Write SSL client and server sockets code that loads the above self-signed certificate to encrypt and decrypt data.

5.1 Preparing Host

To create a self-signed certificate we need to have OpenSSL installed.
Execute below command to check whether it is installed already or not:

```
which openssl
/usr/bin/openssl
```

If not present install by executing (on an ubuntu host)
$sudo apt-get install openssl
$sudo apt-get install libssl-dev (for .h header files)

5.2 Create a self-signed certificate.

Use OpenSSL commands to create a self-signed certificate.
Execute **$man openssl** to understand the below commands.

- openssl genrsa -des3 -passout pass:x -out serverprivate.key 2048
- openssl rsa -passin pass:x -in serverprivate.key -out server.key
- rm serverprivate.key
- openssl req -new -key server.key -out server.csr

  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.

  Country Name (2 letter code) [AU]:**IN**
  State or Province Name (full name) [Some-State]:**Karnataka**
  Locality Name (eg, city) []:**Bangalore**
  Organization Name (eg, company) [Internet Widgits Pty Ltd]:**My Own Company.**
  Organizational Unit Name (eg, section) []:**Research & Development**
  Common Name (e.g. server FQDN or YOUR name) []:**Bob Thomas**
  Email Address []:**bobthomas13@users.noreply.github.com**

  Please enter the following 'extra' attributes
  to be sent with your certificate request
  A challenge password []:**trialpass123**
  An optional company name []:**Myself**

- To generate certificate execute
  openssl x509 -req -days 365 -in server.csr -signkey server.key –out server.crt

- The self-signed SSL certificate is generated from the **server.key** private key and **server.csr** files.

- The server.crt file is your certificate and server.key private key.

5.3 SSL Server & Client Example

Refer https://github.com/bobthomas13/workrepo.git

5.4 Compile and execute

- gcc server.c -lssl
- ./a.out
- Now you should be able to build and run SSL Server application.

- In a different terminal execute
- gcc client.c -lssl
- ./a.out
- Now you should be able to see messages sent and received by server and client.

6 Conclusion

Congratulations, you have learnt how SSL works and how to communicate between two hosts via SSL sockets.

7 References

- http://www.tldp.org/HOWTO/SSL-Certificates-HOWTO/x64.html
- https://devcenter.heroku.com/articles/ssl-certificate-self
- https://www.cs.utah.edu/~swalton/listings/articles/ssl_server.c
- https://www.cs.utah.edu/~swalton/listings/articles/ssl_client.c