Security in Networked Computer Systems
OpenSSL Lab Session #7

Pericle Perazzo

pericle.perazzo@iet.unipi.it

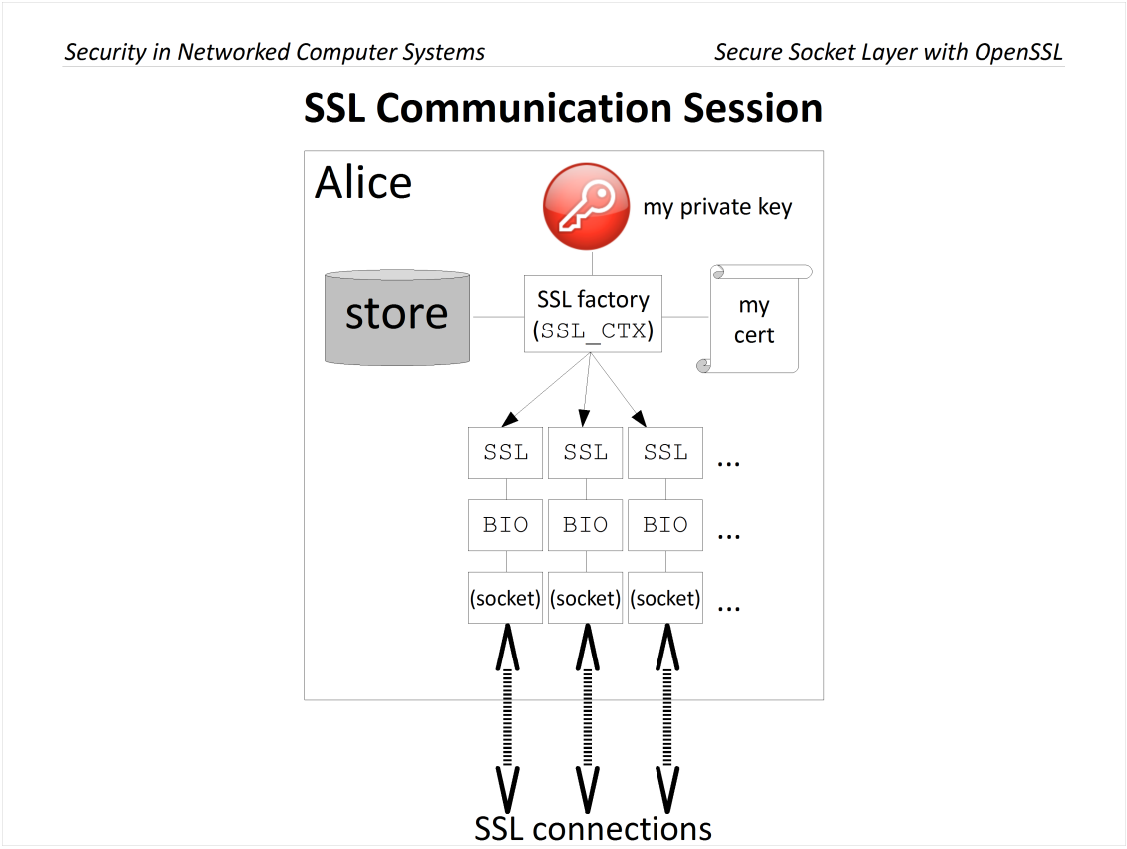
http://www.iet.unipi.it/p.perazzo/teaching/

29th April 2015

Secure Socket Layer

Lesson Outline

- Use BIO objects to create SSL connections.
- Create an SSL connection.
- Let the client authenticate the server and the server authenticate the client by means of certificates and CRL's.
- Use the SSL connection to send/receive a file on a secure channel.



In OpenSSL, an Secure Socket Layer connection is represented by an `SSL` object. An `SSL` object is created by a factory object called `SSL_CTX`. A factory holds a store to authenticate the peer, and a certificate plus a private key to authenticate itself. A single factory can create several SSL connections. Each SSL connection sends and reads bytes from a character stream, represented by a `BIO` object, which is in turn attached to a socket.

BIO Objects

- The concept of *character stream* is represented in OpenSSL by `BIO` objects.
- **`#include <openssl/bio.h>`**
- **`BIO`** (data structure)

  Represents a character stream.
- **`BIO* BIO_new_socket(int socket, BIO_NOCLOSE);`**

  Allocates a new socket `BIO`, i.e. a `BIO` sending to and receiving from the network.
  - **`socket`** → The socket which the BIO is associated to.
  - Returns the allocated `BIO` structure (or NULL if error).

  On the server, the BIO must be associated to the communication socket, not to the listening socket.
- **`void BIO_free(BIO* bio);`**

  Deallocates a `BIO` object.

SSL Factory

- **`#include <openssl/ssl.h>`**
- **`void SSL_library_init();`**

  Initializes the internal OpenSSL data structures for managing SSL connections.
- **`void SSL_load_error_strings();`**

  Initializes the internal OpenSSL table of error descriptions.
- **`SSL_CTX`** (data structure)

  Represents a factory of SSL objects.
- **`SSL_CTX* SSL_CTX_new(SSLv23_method());`**

  Allocates a new SSL factory implementing a given version of the SSL protocol. The parameter `SSLv23_method()` makes the peers negotiate the highest version supported by both (among SSLv3, TLSv1, TLSv1.1, TLSv1.2).

SSL Factory

- **`store = SSL_CTX_get_cert_store(ctx);`**

  Returns the store of the SSL factory. The store can be modified to add certificates, CRL's, and so on.
  - **`ctx`** → The SSL factory.
  - It returns the store (or NULL if error).
- **`int SSL_CTX_use_certificate(SSL_CTX* ctx, X509* x);`**

  Tells to the SSL factory which is my certificate.
  - **`ctx`** → The SSL factory.
  - **`x`** → My certificate.
  - It returns 1 on success, non-1 on error.
- **`int SSL_CTX_use_PrivateKey(SSL_CTX* ctx, EVP_PKEY* prvkey);`**

  Tells to the SSL factory which is my private key. If my certificate has been set, then it also checks the validity of the public key-private key coupling.
  - **`ctx`** → The SSL factory.
  - **`prvkey`** → My private key.
  - It returns 1 on success, non-1 on error.

SSL Factory

- **`void SSL_CTX_set_verify(SSL_CTX* ctx, int mode, NULL);`**

  Sets the flags to tell to the SSL factory whether to request and verify the other peer's certificate.
  - **`ctx`** → The SSL factory.
  - **`mode`** → A set of logically or'ed flags.
    The most common flags' configuration for the client is:
    - **`SSL_VERIFY_PEER`** → It receives and verifies the server's certificate.

    Those for the server are:
    - **`SSL_VERIFY_NONE`** → It does not request nor verify the client's certificate (one-way authentication).
    - **`SSL_VERIFY_PEER | SSL_VERIFY_FAIL_IF_NO_PEER_CERT`** → It requests and verifies the client's certificate (two-way authentication).
- **`void SSL_CTX_free(SSL_CTX* ctx);`**

  Deallocates an SSL factory.
  - **`ctx`** → The SSL factory.

SSL Connection

- **`SSL* SSL_new(SSL_CTX* ctx);`**

  Creates a new SSL session from the factory.
  - **`ctx`** → The SSL factory.
  - It returns the created SSL session.
- **`void SSL_set_bio(SSL *ssl, BIO *rbio, BIO *wbio);`**

  Sets the input and the outbut BIO's for an SSL connection. Usually the same socket BIO.
  - **`ssl`** → The SSL connection.
  - **`rbio`** → The input BIO.
  - **`wbio`** → The output BIO.

SSL Connection

- **`int SSL_connect(SSL* ssl);`**

  Initiates an SSL connection from the client side, and verifies the server's certificate. It is blocking if the underlying BIO is read-blocking (yes, by default). It must be invoked after the "classic" `connect()` function on the socket.
  - **`ssl`** → The SSL connection.
  - It returns 1 if the connection was successful, 0 if it was gracefully shut down by the peer, <0 if a fatal error has occurred.
- **`int SSL_accept(SSL* ssl);`**

  Initiates an SSL connection from the server side, and (eventually) verifies the client's certificate. It is blocking if the underlying BIO is read-blocking (yes, by default). It must be invoked after the "classic" `accept()` function on the socket.
  - **`ssl`** → The SSL connection.
  - It returns 1 if the connection was successful, 0 if it was gracefully shut down by the peer, <0 if a fatal error has occurred.

SSL Connection

- **`X509* SSL_get_peer_certificate(const SSL* ssl);`**

  Retrieves the peer's certificate.
  - **`ssl`** → The SSL connection.
- **`int SSL_write(SSL *ssl, const void *buf, int num);`**

  Sends num bytes from the buffer buf to the SSL connection.
  - **`ssl`** → The SSL connection.
  - It returns the number of bytes sent, or <=0 on error.
- **`int SSL_read(SSL *ssl, void *buf, int num);`**

  Receives num bytes from the SSL connection to the buffer buf. It is blocking if the underlying BIO is read-blocking (yes, by default).
  - **`ssl`** → The SSL connection.
  - It returns the number of bytes received, or <=0 on error.

SSL Connection

- **`int SSL_shutdown(SSL* ssl);`**

  Closes an SSL connection. It is blocking if the underlying BIO is read-blocking (yes, by default). It must be called before the "classic" `close()` function on the socket.
  - **`ssl`** → The SSL connection.
  - It returns 1 on success, non-1 on error.
- **`void SSL_free(SSL* ssl);`**

  Deallocates an SSL connection. It also frees the associated BIO's, so there is no need to invoke `BIO_free()`.
  - **`ssl`** → The SSL connection.

Compilation & Link

- When linking, you have to add also the “ssl” library:
  `gcc hello.c -lcrypto -lssl -o hello`

Final Exercise

- File exchange through SSL connection.
- The client:
  - Wants to upload some sensitive data to a server.
  - Creates an SSL connection with the server.
  - Authenticates the server, checking certificates and CRL.
- The server:
  - Creates an SSL connection with the client.
  - Authenticates the client, checking certificates and CRL.
- The client uses the SSL connection to send a file to the server.
- The server receives and stores it.