Programming OpenSSL

The Server Perspective

by

Sean Walton
Host Addressing & Ports

- Hosts use addresses to interconnect.
- TCP/IP uses a 4-byte number for IDs.
  \[128.98.2.254\]
- TCP adds ports to addresses for services.
  - Ports can be between 1-65535
  - System services are between 1-1023
  - User services are between 1024-65535
  \[198.176.2.45:80 \quad \text{or} \quad 12.63.99.240.3246\]
Basic Client/Server

- Networked hosts connect through sockets.
- Servers publish services through ports.
- Clients connect to the ports.
Client Algorithm

The Basic Client:
• Sets up socket.
• Connects to server.
• Sends command.
• Retrieves reply.
• Closes socket.

int sd;
struct sockaddr_in addr;
sd = socket(PF_INET, SOCK_STREAM, 0);
bzero(&addr, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_port = htons(80);
aton("127.0.0.1", &addr.sin_addr);
connect(sd, &addr, sizeof(addr));
send(sd, msg, msglen, 0);
recv(sd, reply, replylen, 0);
close(sd);
Server Algorithm

The Basic Server:

- Sets up socket.
- Publishes port.
- Socket → listener.
- Awaits connection.
- Gets command.
- Sends reply.
- Closes connection.

```c
int sd, client;
struct sockaddr_in addr;
sd = socket(PF_INET, SOCK_STREAM, 0);
bzero(&addr, sizeof(addr));
addr.sin_family = AF_INET;
addr.sin_port = htons(80);
aton("127.0.0.1", &addr.sin_addr);
bind(sd, &addr, sizeof(addr));
listen(sd, 10);
client = accept(sd, 0, 0);
recv(client, cmd, cmdlen, 0);
send(client, reply, replylen, 0);
close(client);
```
Secure Sockets

- Building on top of TCP/IP.
- Using keys and ciphers.
- Offering certificates.
- Verifying message with a Message Digest.
- Extending the handshake.
OpenSSL Sits on Top of TCP/IP

• Simplifies interface with TCP/IP stack.
• Simplifies programming.
• Limits session recoverability.
• Keeps interface clean and direct.
Keys and Ciphers

- Ciphers are encryption algorithms.
- Keys are numbers within a special range.
- Private-key ciphers
  - Use the same key to encrypt & decrypt data
  - Have highest security
  - Are very fast.
- Public-key ciphers
  - Require two keys: encryption & decryption
  - Are <25% as secure as private keys
  - Are very slow.
Certificates

• Solves the “Man in the Middle” dilemma.
• Solve the problem of host identification with a trusted third party.
• Contain information about the server:
  – Who owns the certificate
  – Who issued the certificate
  – Where the owner is located
  – When the certificate will expire.
Message Digest

• “Summarizes” the message.
• Must be irreversible (real data cannot be recovered from digest value).
• Most message digests are hash functions.
• Combined with an encryption key yields the Message Authentication Code (MAC).
Different Handshakes

• TCP offers the “Three-Way Handshake”.
  – Client extends request (SYN)
  – Server accepts (ACK) and reciprocates (SYN)
  – Client accepts and begins communications
Extending the Handshake

- Client sends cipher list & random number.
- Server indicates cipher, sends certificate, public key & random number.
- Client verifies certificate and sends an encrypted private key with public key.
- Server accepts private key and sends own private key.
OpenSSL Initialization

• Build algorithm tables.
• Load error messages.
• Select interface methods.
• Create new context.

```c
SSL_METHOD *method;
SSL_CTX *ctx;
OpenSSL_add_all_algorithms();
SSL_load_error_strings();
method = SSLv2_server_method();
ctx = SSL_CTX_new(method);
```
Initialization (cont.)

- Load certificates file.
- Load private keys file.
- Verify private keys.

```c
SSL_CTX_use_certificate_file(ctx, CertFile, SSL_FILETYPE_PEM);
SSL_CTX_use_PrivateKey_file(ctx, KeyFile, SSL_FILETYPE_PEM);
if ( !SSL_CTX_check_private_key(ctx) )
    fprintf(stderr, "Files don't match!");
```
Set Up Server Socket

... Use the same algorithm for setting up server!
Attach Client to SSL

- Create SSL instance.
- Attach client to instance.
- Establish SSL handshake.
- Commence transactions.

```c
SSL *ssl;
ssl = SSL_new(ctx);
SSL_set_fd(ssl, client);
SSL_accept(ssl);
SSL_read(ssl, cmd, cmdlen);
SSL_write(ssl, reply, replylen);
```
OpenSSL Features

- Offers direct development path from sockets.
- Simplifies interfacing.
- Can create private certificates.
- Supports multi-threading.
- Interfaces directly with off-the-shelf browsers.
- Supports multiple platforms & OSs.
- Is GPL!