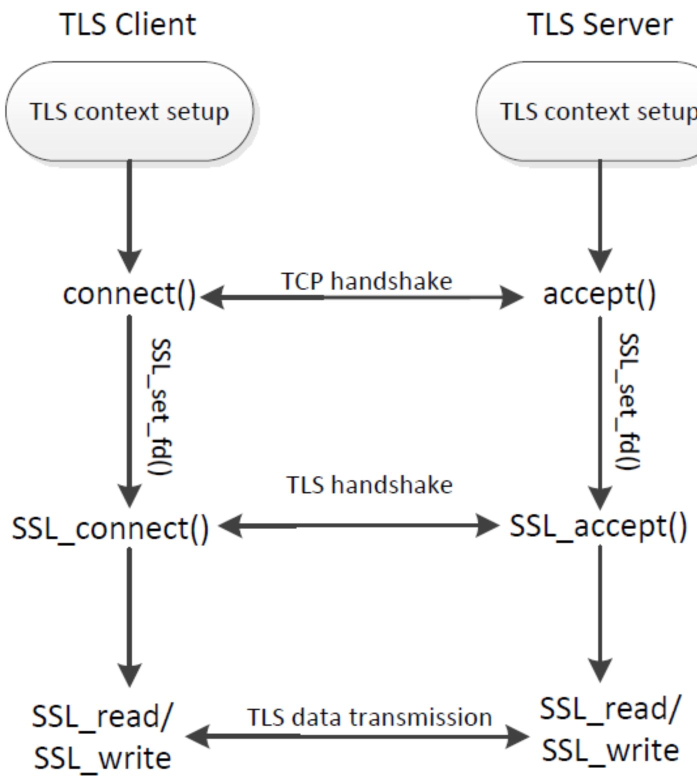
TLS Client Program

TLS Programming : Overall Picture



TLS Client Program: TLS Initialization

- TLS protocol is a stateful protocol
- Create a context data structure
- Create a SSL structure to hold state information

SSL Context: holding SSL configuration →

Holding SSL states →

```
// Step 1: SSL context initialization
SSL_METHOD *meth = (SSL_METHOD *)TLSv1_2_method();
SSL_CTX* ctx = SSL_CTX_new(meth);
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
SSL_CTX_load_verify_locations(ctx, NULL, "./cert");

// Step 2: Create a new SSL structure for a connection
SSL* ssl = SSL_new (ctx);
```

TLS Client Program: TLS Initialization (cont'd)

```
// Step 1: SSL context initialization
SSL_METHOD *meth = (SSL_METHOD *)TLSv1_2_method();
SSL_CTX* ctx = SSL_CTX_new(meth);
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, NULL);
SSL_CTX_load_verify_locations(ctx, NULL, "./cert");

// Step 2: Create a new SSL structure for a connection
SSL* ssl = SSL_new (ctx);
```

Should verify server's certificate

Folder containing trusted CA' certificates, such as root CA's certificates.

```
// Step 3: Enable the hostname check
X509_VERIFY_PARAM *vpm = SSL_get0_param(ssl);
X509_VERIFY_PARAM_set1_host(vpm, hostname, 0);
```

Check whether the certificate's subject field matches with hostname.

TLS Client Program: Set Up a TCP Connection

- TLS is primarily built on top of TCP.
- This part is standard.

```
int setupTCPClient(const char* hostname, int port)
{
   struct sockaddr_in server_addr;

   // Get the IP address from hostname
   struct hostent* hp = gethostbyname(hostname);

   // Create a TCP socket
   int sockfd= socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);

   // Fill in the destination information (IP, port #, and family)
   memset (&server_addr, '\0', sizeof(server_addr));
   memcpy(&(server_addr.sin_addr.s_addr), hp->h_addr, hp->h_length);
   server_addr.sin_port   = htons (port);
   server_addr.sin_family = AF_INET;

   // Connect to the destination
   connect(sockfd, (struct sockaddr*) &server_addr,
           sizeof(server_addr));

   return sockfd;
}
```

TLS Client Program: Initiate TLS Handshake

Establish the SSL session on top of an established TCP connection

```
SSL* ssl = setupTLSClient(hostname);
int  sockfd = setupTCPClient(hostname, port);

SSL_set_fd(ssl, sockfd);
int err = SSL_connect(ssl);
```

Initiate the TLS Handshake protocol

TLS Client Program: Send/Receive Data

- We construct a simple HTTP GET request, and print out the reply from the web server.

Send data

Send data

```
char buf[9000];
char sendBuf[200];

sprintf(sendBuf, "GET / HTTP/1.1\nHost: %s\n\n", hostname);
SSL_write(ssl, sendBuf, strlen(sendBuf));

int len;
do {
     len = SSL_read (ssl, buf, sizeof(buf) - 1);
     buf[len] = ’\0’;
     printf("%s\n",buf);
} while (len > 0);
```

TLS Server Program

Create a simple HTTPS server

TLS Server Program: Setup

```
// Step 1: SSL context initialization
meth = (SSL_METHOD *)TLSv1_2_method();
ctx = SSL_CTX_new(meth);
SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);

// Step 2: Set up the server certificate and private key
SSL_CTX_use_certificate_file(ctx, "./bank_cert.pem",
                             SSL_FILETYPE_PEM);
/* SSL_CTX_use_certificate_chain_file(ctx,
                             "./bank_chain_cert.pem"); */
SSL_CTX_use_PrivateKey_file(ctx, "./bank_key.pem",
                             SSL_FILETYPE_PEM);

// Step 3: Create a new SSL structure for a connection
ssl = SSL_new (ctx);
```

Will not verify the client's certificate

Server's certificate

Server's private key

TLS Server Program: TCP Setup

This program creates a TCP socket, binds it to a TCP port (4433) and marks the socket as a passive socket. This is quite standard.

```
int setupTCPServer()
{
    struct sockaddr_in sa_server;
    int listen_sock;

    listen_sock= socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    memset (&sa_server, '\0', sizeof(sa_server));
    sa_server.sin_family      = AF_INET;
    sa_server.sin_addr.s_addr = INADDR_ANY;
    sa_server.sin_port        = htons (4433);
    bind(listen_sock, (struct sockaddr*)&sa_server,
    sizeof(sa_server));
    listen(listen_sock, 5);
    return listen_sock;
}
```

TLS Server: Handshake & Data Communication

Conduct TLS handshake with the client

We can now use this established SSL session to conduct data communication

```
while (1) {
   int sock = accept(listen_sock, (struct sockaddr*)&sa_client,
  &client_len);
   if (fork() == 0) { // The child process
       close (listen_sock);

       SSL_set_fd (ssl, sock);
       int err = SSL_accept (ssl);
       CHK_SSL(err);
       printf ("SSL connection established!\n");

       processRequest(ssl, sock);
       close(socket);
       return 0;
   } else { // The parent process
        close(sock);
   }
}
```

TLS Server Program: Data Transmission

- Logic for sending/receiving data is the same as the client program.
- We simply send an HTTP reply message back to the client.

```
void processRequest(SSL* ssl, int sock)
{
    char buf[1024];
    int len = SSL_read (ssl, buf, sizeof(buf) - 1);
    buf[len] = '\0';
    printf("Received: %s\n",buf);

    // Construct and send the HTML page
    char *html = "... (omitted) ...";
    SSL_write(ssl, html, strlen(html));
    SSL_shutdown(ssl);  SSL_free(ssl);
}
```

Summary

- Write a simple TLS client program
- Write a simple TLS server program