计算机网络安全实验

华中科技大学

实验3 SSLVPN

SSL编程简介

步骤1：初始化SSL库


```
SSL_library_init();
SSL_load_error_strings();
```

```
SSL_library_init();
SSL_load_error_strings();
```

步骤2：创建SSL上下文接口(SSL_CTX)


```
const SSL_METHOD *meth;
SSL_CTX *ctx;

meth = SSLv23_client_method();
ctx = SSL_CTX_new(meth);
```

```
const SSL_METHOD *meth;
SSL_CTX *ctx;

meth = SSLv23_server_method();
ctx = SSL_CTX_new(meth);
```

协议号	通用接口	服务端专用	客户端专用
SSLv2	SSLv2_method()	SSLv2_server_method()	SSLv2_client_method()
SSLv3	SSLv3_method()	SSLv3_server_method()	SSLv3_client_method()
TLSv1	TLSv1_method()	TLSv1_server_method()	TLSv1_client_method()
SSLv23	SSLv23_method()	SSLv23_server_method()	SSLv23_client_method()

步骤3：设置证书及验证方式


```
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER, verify_callback);
SSL_CTX_load_verify_locations(ctx, CACERT, NULL);
SSL_CTX_use_certificate_file(ctx, CERTF, SSL_FILETYPE_PEM;
SSL_CTX_use_PrivateKey_file(ctx, KEYF, SSL_FILETYPE_PEM;
SSL_CTX_check_private_key(ctx);
```

认证方式	说明
SSL_VERIFY_NONE	不验证证书
SSL_VERIFY_PEER	验证对方证书
SSL_VERIFY_FAIL_IF_NO_PEER_CERT	或选项，无证书验证失败
SSL_VERIFY_CLIENT_ONCE	对客户端验证一次

```
SSL_CTX_set_verify(ctx, SSL_VERIFY_NONE, NULL);
SSL_CTX_load_verify_locations(ctx, CACERT, NULL);
SSL_CTX_use_certificate_file(ctx, CERTF, SSL_FILETYPE_PEM);
SSL_CTX_use_PrivateKey_file(ctx, KEYF, SSL_FILETYPE_PEM);
SSL_CTX_check_private_key(ctx);
```

步骤4：创建SSL套接字


```
SSL *ssl;

ssl = SSL_new(ctx);
```

```
SSL *ssl;

ssl = SSL_new(ctx);
```

步骤5：创建TCP套接字


```
struct sockaddr_in server_addr;
struct hostent *hp = gethostbyna
me(hostname);

int sockfd = socket(AF_INET, SOC
K_STREAM, IPPROTO_TCP);
memset(&server_addr, '\0', sizeo
f(server_addr));
memcpy(&(server_addr.sin_addr.s_
addr), hp->h_addr, hp->h_length);
server_addr.sin_port = htons(por
t);
server_addr.sin_family = AF_INET;

connect(sockfd, (struct sockaddr
*) &server_addr, sizeof(server_a
ddr));
```

SYN=1,ACK=0

SYN=1,ACK=1

SYN=0,ACK=1

```
struct sockaddr_in sa_server;
int listen_sock;

listen_sock = socket(PF_INET, SO
CK_STREAM, IPPROTO_TCP);
memset(&sa_server, '\0', sizeof
(sa_server));
sa_server.sin_family = AF_INET;
sa_server.sin_addr.s_addr = INAD
DR_ANY;
sa_server.sin_port = htons(4433);
int err = bind(listen_sock, (str
uct sockaddr *) &sa_server, size
of(sa_server));
err = listen(listen_sock, 5);


int sock = accept(listen_sock,
(struct sockaddr *) &sa_client,
&client_len);
```

步骤6：SSL关联及握手


```
SSL_set_fd(ssl, sockfd);

int err = SSL_connect(ssl);
```

```
SSL_set_fd(ssl, sock);

int err = SSL_accept(ssl);
```

客户端支持的最高版本，加密套件列表，压缩算法列表，客户端随机数，会话ID=0

ClientHello →

← 服务器同意的版本，加密套件，压缩算法，会话ID，服务器端随机数 ServerHello

← 服务器的证书 ServerCertificate*

← 服务器端密钥交换的附加信息 ServerKeyExchange*

← 通知对方服务器端握手消息发完 ServerHelloDone*

ClientKeyExchange → 客户端产生的PreMasterKey密钥参数

[ChangeCipherSpec] → 通知对方本端开始启用加密参数

Finished → 用新的算法、密钥计算校验消息，确认握手完成

← 通知对方本端开始启用加密参数 [ChangeCipherSpec]

← 用新的算法、密钥计算校验消息，确认握手完成 Finished

步骤7：读写数据


```
err = SSL_write(ssl, wbuf, strle
n(wbuf));

err = SSL_read(ssl, rbuf, sizeof
(rbuf)-1);
```

```
err = SSL_write(ssl, wbuf, strle
n(wbuf));

err = SSL_read(ssl, rbuf, sizeof
(rbuf)-1);
```

Application Data

传送应用层数据

Application Data

步骤8：关闭


```
SSL_shutdown(ssl);
SSL_free(ssl);
close(sockfd);
```

```
SSL_shutdown(ssl);
SSL_free(ssl);
close(sock);
```