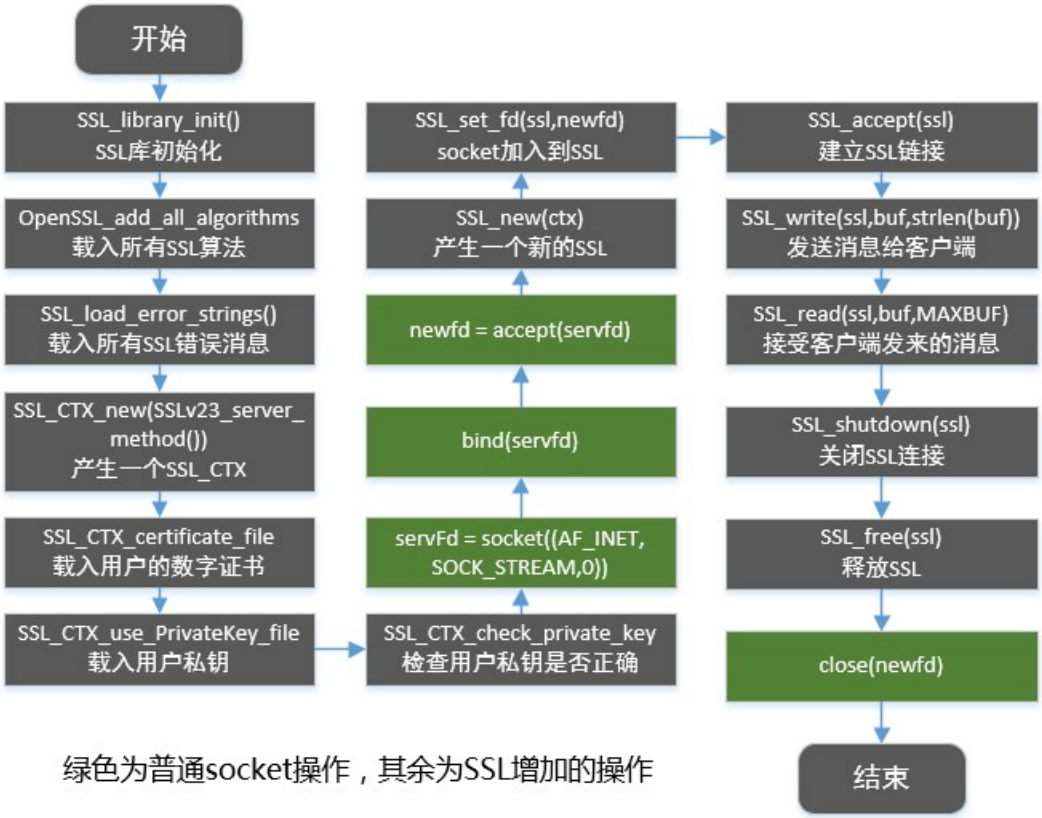
魄魄熊

在二十几岁的年纪里做件到八十岁想起来仍然会微笑的事

ssl客户端与服务端通信的demo

服务端程序流程




```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>
4  #include <string.h>
5  #include <sys/types.h>
6  #include <netinet/in.h>
7  #include <sys/socket.h>
8  #include <sys/wait.h>
9  #include <unistd.h>
10 #include <arpa/inet.h>
11 #include <openssl/ssl.h>
12 #include <openssl/err.h>
```

```
13
14 #define MAXBUF 1024
15 /************关于本文档***********************************************
16 *filename: ssl-server.c
17 *purpose: 演示利用 OpenSSL 库进行基于 IP层的 SSL 加密通讯的方法，这是服务器端例子
18 *wrote by: zhoulifa(zhoulifa@163.com) 周立发(http://zhoulifa.bokee.com)
19 Linux爱好者 Linux知识传播者 SOHO族 开发者 最擅长C语言
20 *date time:2007-02-02 19:40
21 *Note: 任何人可以任意复制代码并运用这些文档，当然包括你的商业用途
22 * 但请遵循GPL
23 *Thanks to:Google
24 *Hope:希望越来越多的人贡献自己的力量，为科学技术发展出力
25 * 科技站在巨人的肩膀上进步更快！感谢有开源前辈的贡献！
26 ********************************************************************/
27 int main(int argc, char **argv)
28 {
29     int sockfd, new_fd;
30     socklen_t len;
31     struct sockaddr_in my_addr, their_addr;
32     unsigned int myport, lisnum;
33     char buf[MAXBUF + 1];
34     SSL_CTX *ctx;
35
36     if (argv[1])
37         myport = atoi(argv[1]);
38     else
39         myport = 7838;
40
41     if (argv[2])
42         lisnum = atoi(argv[2]);
43     else
44         lisnum = 2;
45
46     /* SSL 库初始化 */
47     SSL_library_init();
48     /* 载入所有 SSL 算法 */
49     OpenSSL_add_all_algorithms();
50     /* 载入所有 SSL 错误消息 */
51     SSL_load_error_strings();
52     /* 以 SSL V2 和 V3 标准兼容方式产生一个 SSL_CTX ，即 SSL Content Text */
53     ctx = SSL_CTX_new(SSLv23_server_method());
54     /* 也可以用 SSLv2_server_method() 或 SSLv3_server_method() 单独表示 V2 或 V3标准 */
55     if (ctx == NULL) {
56         ERR_print_errors_fp(stdout);
57         exit(1);
58     }
59     /* 载入用户的数字证书， 此证书用来发送给客户端。 证书里包含有公钥 */
60     if (SSL_CTX_use_certificate_file(ctx, argv[4], SSL_FILETYPE_PEM) <= 0) {
61         ERR_print_errors_fp(stdout);
62         exit(1);
63     }
64     /* 载入用户私钥 */
65     if (SSL_CTX_use_PrivateKey_file(ctx, argv[5], SSL_FILETYPE_PEM) <= 0) {
66         ERR_print_errors_fp(stdout);
67         exit(1);
68     }
69     /* 检查用户私钥是否正确 */
70     if (!SSL_CTX_check_private_key(ctx)) {
```

2016年12月 (2)
2016年7月 (1)
2016年6月 (2)
2016年3月 (1)
2015年9月 (3)

最新评论

1. Re:ssl客户端与服务端通信
很好，关注啦，可以交流

阅读排行榜

1. C++点滴----关于类常成员i
2. ssl客户端与服务端通信的
3. jetBrain系列软件激活试用
4. 在Ubuntu Server 14.04中
器(VMWare)(238)
5. 字符串匹配算法--Brute-Fo

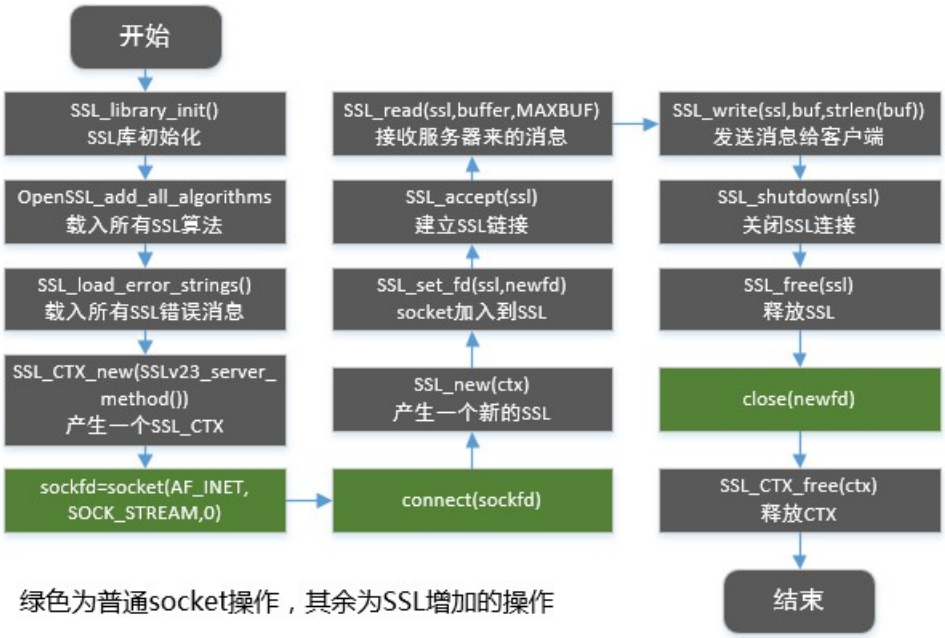
评论排行榜

1. ssl客户端与服务端通信的

```
71          ERR_print_errors_fp(stdout);
72          exit(1);
73      }
74
75      /* 开启一个 socket 监听 */
76      if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
77          perror("socket");
78          exit(1);
79      } else
80          printf("socket created\n");
81
82      bzero(&my_addr, sizeof(my_addr));
83      my_addr.sin_family = PF_INET;
84      my_addr.sin_port = htons(myport);
85      if (argv[3])
86          my_addr.sin_addr.s_addr = inet_addr(argv[3]);
87      else
88          my_addr.sin_addr.s_addr = INADDR_ANY;
89
90      if (bind(sockfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr))
91          == -1) {
92          perror("bind");
93          exit(1);
94      } else
95          printf("binded\n");
96
97      if (listen(sockfd, lisnum) == -1) {
98          perror("listen");
99          exit(1);
100     } else
101         printf("begin listen\n");
102
103     while (1) {
104         SSL *ssl;
105         len = sizeof(struct sockaddr);
106         /* 等待客户端连上来 */
107         if ((new_fd =
108              accept(sockfd, (struct sockaddr *) &their_addr,
109                     &len)) == -1) {
110             perror("accept");
111             exit(errno);
112         } else
113             printf("server: got connection from %s, port %d, socket %d\n",
114                    inet_ntoa(their_addr.sin_addr),
115                    ntohs(their_addr.sin_port), new_fd);
116
117         /* 基于 ctx 产生一个新的 SSL */
118         ssl = SSL_new(ctx);
119         /* 将连接用户的 socket 加入到 SSL */
120         SSL_set_fd(ssl, new_fd);
121         /* 建立 SSL 连接 */
122         if (SSL_accept(ssl) == -1) {
123             perror("accept");
124             close(new_fd);
125             break;
126         }
127
128         /* 开始处理每个新连接上的数据收发 */
```

```
129         bzero(buf, MAXBUF + 1);
130         strcpy(buf, "server->client");
131         /* 发消息给客户端 */
132         len = SSL_write(ssl, buf, strlen(buf));
133 
134         if (len <= 0) {
135             printf
136                 ("消息'%s'发送失败！错误代码是%d，错误信息是'%s'\n",
137                  buf, errno, strerror(errno));
138             goto finish;
139         } else
140             printf("消息'%s'发送成功，共发送了%d个字节！\n",
141                    buf, len);
142 
143         bzero(buf, MAXBUF + 1);
144         /* 接收客户端的消息 */
145         len = SSL_read(ssl, buf, MAXBUF);
146         if (len > 0)
147             printf("接收消息成功:'%s'，共%d个字节的数据\n",
148                    buf, len);
149         else
150             printf
151                 ("消息接收失败！错误代码是%d，错误信息是'%s'\n",
152                  errno, strerror(errno));
153         /* 处理每个新连接上的数据收发结束 */
154       finish:
155         /* 关闭 SSL 连接 */
156         SSL_shutdown(ssl);
157         /* 释放 SSL */
158         SSL_free(ssl);
159         /* 关闭 socket */
160         close(new_fd);
161     }
162 
163     /* 关闭监听的 socket */
164     close(sockfd);
165     /* 释放 CTX */
166     SSL_CTX_free(ctx);
167     return 0;
168 }
```

客户端编写流程




```c
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <resolv.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <openssl/ssl.h>
#include <openssl/err.h>

#define MAXBUF 1024

void ShowCerts(SSL * ssl)
{
    X509 *cert;
    char *line;

    cert = SSL_get_peer_certificate(ssl);
    if (cert != NULL) {
        printf("数字证书信息:\n");
        line = X509_NAME_oneline(X509_get_subject_name(cert), 0, 0);
        printf("证书: %s\n", line);
        free(line);
        line = X509_NAME_oneline(X509_get_issuer_name(cert), 0, 0);
        printf("颁发者: %s\n", line);
        free(line);
        X509_free(cert);
    } else
        printf("无证书信息! \n");
}
/************关于本文档***********************************************
*filename: ssl-client.c
*purpose: 演示利用 OpenSSL 库进行基于 IP层的 SSL 加密通讯的方法，这是客户端例子
```

```
*wrote by: zhoulifa(zhoulifa@163.com) 周立发(http://zhoulifa.bokee.com)
Linux爱好者 Linux知识传播者 SOHO族 开发者 最擅长C语言
*date time:2007-02-02 20:10
*Note: 任何人可以任意复制代码并运用这些文档，当然包括你的商业用途
* 但请遵循GPL
*Thanks to:Google
*Hope:希望越来越多的人贡献自己的力量，为科学技术发展出力
* 科技站在巨人的肩膀上进步更快！感谢有开源前辈的贡献！
********************************************************************/
int main(int argc, char **argv)
{
    int sockfd, len;
    struct sockaddr_in dest;
    char buffer[MAXBUF + 1];
    SSL_CTX *ctx;
    SSL *ssl;

    if (argc != 3) {
        printf
            ("参数格式错误！正确用法如下: \n\t\t%s IP地址 端口\n\t比如:\t%s 127.0.0.1 80
            \n此程序用来从某个 IP 地址的服务器某个端口接收最多 MAXBUF 个字节的消息",
             argv[0], argv[0]);
        exit(0);
    }

    /* SSL 库初始化，参看 ssl-server.c 代码 */
    SSL_library_init();
    OpenSSL_add_all_algorithms();
    SSL_load_error_strings();
    ctx = SSL_CTX_new(SSLv23_client_method());
    if (ctx == NULL) {
        ERR_print_errors_fp(stdout);
        exit(1);
    }

    /* 创建一个 socket 用于 tcp 通信 */
    if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("Socket");
        exit(errno);
    }
    printf("socket created\n");

    /* 初始化服务器端（对方）的地址和端口信息 */
    bzero(&dest, sizeof(dest));
    dest.sin_family = AF_INET;
    dest.sin_port = htons(atoi(argv[2]));
    if (inet_aton(argv[1], (struct in_addr *) &dest.sin_addr.s_addr) == 0) {
        perror(argv[1]);
        exit(errno);
    }
    printf("address created\n");

    /* 连接服务器 */
    if (connect(sockfd, (struct sockaddr *) &dest, sizeof(dest)) != 0) {
        perror("Connect ");
        exit(errno);
    }
    printf("server connected\n");
```

```
 93
 94     /* 基于 ctx 产生一个新的 SSL */
 95     ssl = SSL_new(ctx);
 96     SSL_set_fd(ssl, sockfd);
 97     /* 建立 SSL 连接 */
 98     if (SSL_connect(ssl) == -1)
 99         ERR_print_errors_fp(stderr);
100     else {
101         printf("Connected with %s encryption\n", SSL_get_cipher(ssl));
102         ShowCerts(ssl);
103     }
104 
105     /* 接收对方发过来的消息，最多接收 MAXBUF 个字节 */
106     bzero(buffer, MAXBUF + 1);
107     /* 接收服务器来的消息 */
108     len = SSL_read(ssl, buffer, MAXBUF);
109     if (len > 0)
110         printf("接收消息成功:'%s'，共%d个字节的数据\n",
111                buffer, len);
112     else {
113         printf
114             ("消息接收失败！错误代码是%d，错误信息是'%s'\n",
115              errno, strerror(errno));
116         goto finish;
117     }
118     bzero(buffer, MAXBUF + 1);
119     strcpy(buffer, "from client->server");
120     /* 发消息给服务器 */
121     len = SSL_write(ssl, buffer, strlen(buffer));
122     if (len < 0)
123         printf
124             ("消息'%s'发送失败！错误代码是%d，错误信息是'%s'\n",
125              buffer, errno, strerror(errno));
126     else
127         printf("消息'%s'发送成功，共发送了%d个字节！\n",
128                buffer, len);
129 
130   finish:
131     /* 关闭连接 */
132     SSL_shutdown(ssl);
133     SSL_free(ssl);
134     close(sockfd);
135     SSL_CTX_free(ctx);
136     return 0;
137 }
```

编译程序用如下命令：

```
gcc -Wall ssl-client.c -o client -lssl -lcrypto
gcc -Wall ssl-server.c -o server -lssl -lcrypto
```

证书 privkey.pem 和 cacert.pem 生成使用如下命令(具体请参考 “OpenSSL体系下使用密钥数字证书等”)：

```
openssl genrsa -out privkey.pem 2048
openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095
```

运行程序使用如下命令：

```
./server 7838 1 127.0.0.1 cacert.pem privkey.pem
./client 127.0.0.1 7838
```

运行截图如下:

服务端:

```
calvin@Lenovo:~/Desktop/ssl$ ./server 7838 1 127.0.0.1 cacert.pem privkey.pem
socket created
binded
begin listen
server: got connection from 127.0.0.1, port 49382, socket 4
消息'server->client'发送成功，共发送了14个字节！
接收消息成功:'from client->server'，共19个字节的数据
```

客户端:

```
calvin@Lenovo:~/Desktop/ssl$ ./client 127.0.0.1 7838
socket created
address created
server connected
Connected with AES256-GCM-SHA384 encryption
数字证书信息:
证书: /C=AU/ST=anhui/L=hefei/O=company/OU=name/CN=name/emailAddress=name@163.com
颁发者: /C=AU/ST=anhui/L=hefei/O=company/OU=name/CN=name/emailAddress=name@163.com
接收消息成功:'server->client'，共14个字节的数据
消息'from client->server'发送成功，共发送了19个字节！
```

参考资料:

1、http://blog.csdn.net/thq0201/article/details/6766449#

2、http://blog.chinaunix.net/uid-20682147-id-76392.html

3、http://zhoulifa.bokee.com

魄魄熊
关注 - 4
粉丝 - 2
+加关注

0 0

« 上一篇: 字符串匹配算法--Brute-Force算法
» 下一篇: jetBrain系列软件激活试用

posted @ 2016-12-25 16:13 魄魄熊 阅读(1801) 评论(1) 编辑 收藏

评论列表

#1楼 2016-12-28 23:52 c_sun_boke

很好，关注啦，可以交流

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。