**实验二 基于OpenSSL的安全Web服务器程序**

1. **实验目的**

（1）理解HTTPS协议与SSL协议的基本工作原理；

（2）掌握使用OpenSSL编程的方法；

（3）掌握安全Web系统设计的基本设计与编程方法。

1. **实验要求**

（1）在Linux平台上利用OpenSSL库，编写一个Web Server程序；

（2）Server程序要能够并发处理多个请求，要求至少能够支持HTTPS协议下最基本的Get命令，可以增强Web Server的功能，如支持Head、Post以及Delete命令等；

（3）编写必要的客户端测试程序，用于发送HTTPS请求并显示返回结果。

1. **实验内容**

根据本章所学的知识，在Linux平台上利用OpenSSL实现安全的Web Server。Server能够并发处理多个请求，要求至少能支持Get请求。可以增强Web Server的功能，如支持Head、Post以及Delete命令等。

编写必要的客户端测试程序，用于发送HTTPS请求并显示返回结果，也可以使用一般的Web浏览器测试。

1. **实验原理**

实验程序可分为两个部分：初始化模块，即通过编译及函数调用对OpenSSL库进行初始化并且创建上下文环境；Web服务模块，即基于SSL机制利用OpenSSL库函数实现HTTPS的服务。程序整体流程如下图：



**（1）初始化模块**

在开启HTTPS服务之前，服务器端只需要初始化OpenSSL库和创建上下文环境。在此过程中，会用到如下函数：

SSL\_library\_init(); //加载OpenSSL将会用到的算法

SSL\_load\_error\_strings(); //加载错误字符串

SSL\_METHOD \*meth();

SSL\_CTX \*ctx; //SSL\_CTX对象

meth=SSLv23\_method(); //相应的SSL结构使用的是SL2.0、3.0但可以回到SSL2.0

ctx=SSL\_CTX\_new(mesh); //创建一个上下文环境

SSL\_CTX\_use\_certificate\_chain\_file(ctx, SERVERPEM); //指定所使用的证书文件

SSL\_CTX\_set\_default\_passwd\_cb(ctx, password\_cb); //设置密码回调函数

SSL\_CTX\_use\_PrivateKey\_file(ctx, SERVERKEYPEM, SSL\_FILETYPE\_PEM); //加载私钥文件

SSL\_CTX\_load\_verify\_locations(ctx, ROOTCERTPEM, 0); //加载受信任的CA证书

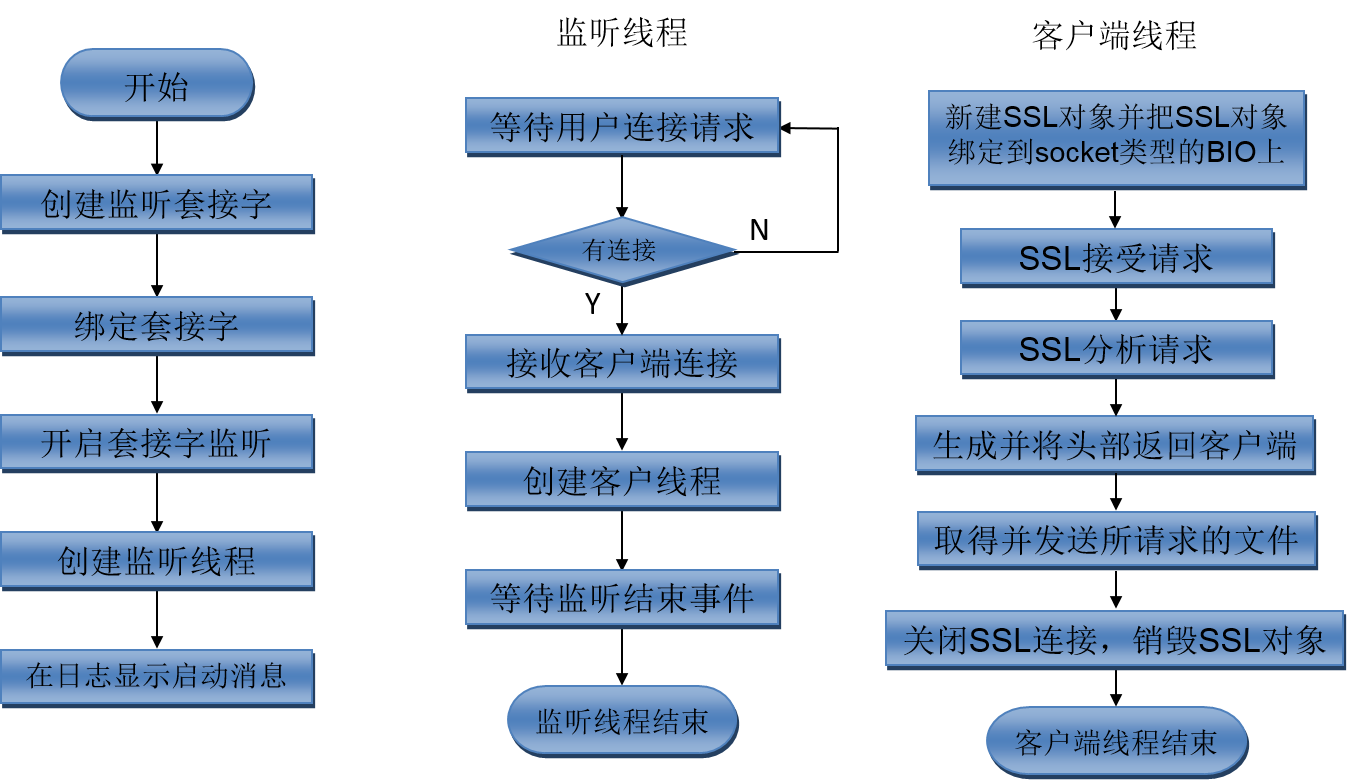
load\_dh\_params(ctx, ROOTKEYPEM);

//当使用RSA算法鉴别的时候，会有一个临时的DH密钥磋商发生。这样会话数据将用

//这个临时的秘钥加密，而证书中的密钥作为签名。

**（2）Web服务模块**

Web服务模块是本程序的核心部分，其执行流程如下图所示：



Web服务模块在类CHttpProtocol中实现。该类中封装了HTTPS中与本程序相关的操作，主要包括监听线程函数以及客户端函数中需要调用的子函数的定义及实现。其中，比较核心的子函数包括SSL分析请求，将响应头部返回给客户端以及将文件发送回客户端等。

1. **实验环境**

（1）Linux系统：Ubuntu .18.04 LTS

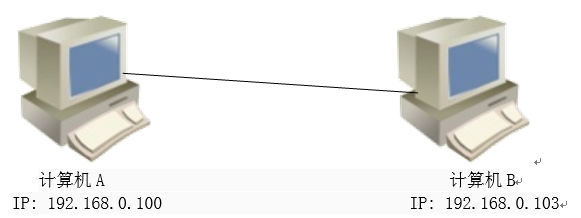
（2）Openssl版本：OpenSSL 1.1.1f

（3）g++ : 7.5.0

（4）浏览器: firefox

**6 实验组网**

本实验组网如下图所示：



**7 实验步骤**

**7.1 OpenSSL的安装配置实验**

**步骤1** 在计算机上安装OpenSSL库(实验配置的虚拟机已经安装好)

sudo apt-get install libssl-dev

**步骤2** 在计算机A上制作自己的WebServer目录，目录下应有index.html文件以及favicon.ico文件。在/home/ubuntu/Desktop/2024\_cyber\_security/SSL实验/SSL目录下已经定义好了一个WebServer目录。建议将该文件放到/home下

**步骤3** 在HttpProtocol.cpp文件中第9行处，将m\_strRootDir设置成自己的WebServer目录路径。例如，m\_strRootDir="/home/WebServer"（注意路径中不能含有中文字符）。

**步骤4** 在命令行模式下，切换到代码所在目录，输入make后回车，完成代码的编译。若未安装g++，使用以下命令进行安装。

sudo apt-get install g++

**步骤5** 在命令行中输入：

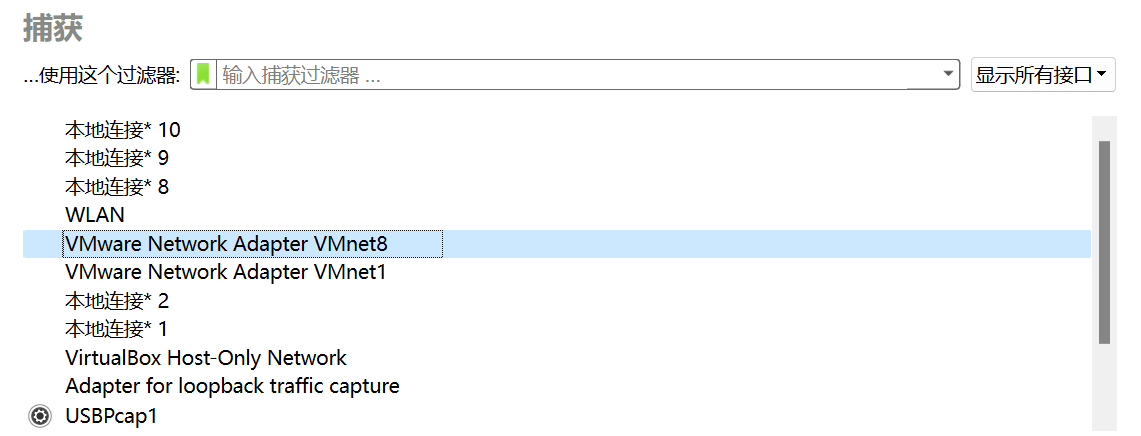
./MyWebServer

以运行服务器端程序。

**步骤6** 在主机B上打开浏览器，在地址栏输入https://192.168.0.100:8000(注意是https且IP地址为计算机A对应的IP地址)，即可登录服务器页面。**注意：**如果浏览器提示此链接不受信任，可添加例外后再进行访问。添加后，可以在firefox中查看添加的认证，位置为preferences-> privacy&security -> view certificates -> servers；或者直接选择高级，信任该链接并接受风险。

**7.2 OpenSSL的报文分析实验**

**步骤1** 在物理机上（电脑本机）上启动Wireshark软件，选择监听端口为VMware Network Adapter VMnet8（若电脑本机上没有Wireshark软件，需要下载安装）。



**步骤2**  在主机A上重新启动./MyWebServer程序，使用主机B访问 [https://192.168.0.100:8000](https://192.168.176.139:8000)

**步骤3**  在wireshark软件上选择过滤条件为：

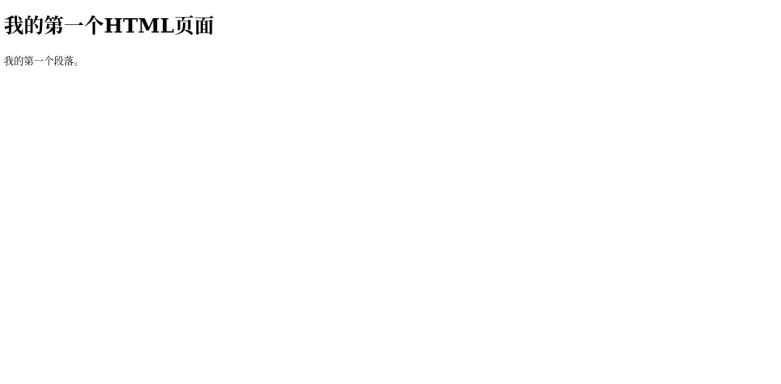
ssl && ip.addr == 192.168.0.100 && ip.addr == 192.168.0.103

这样就能查看在主机A、B之间的SSL报文交互情况。分析有几种SSL报文，它们分别有什么作用？

**8 预期实验结果**

**8.1 OpenSSL的安装配置实验**

1. 能够访问服务器页面。



（2）控制台中有如下记录：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Server starts\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Starting ListenThread...

Starting ClientThread...

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Starting ClientThread...

IF...

Request received!!

GET / HTTP/1.1

Host: 127.0.0.1:8000

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:37.0) Gecko/20100101 Firefox/37.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3

Accept-Encoding: gzip, deflate

Connection: keep-alive

Cache-Control: max-age=0

/home/WebServer/index.html

SSLSendHeader successfully!

Sending..............................

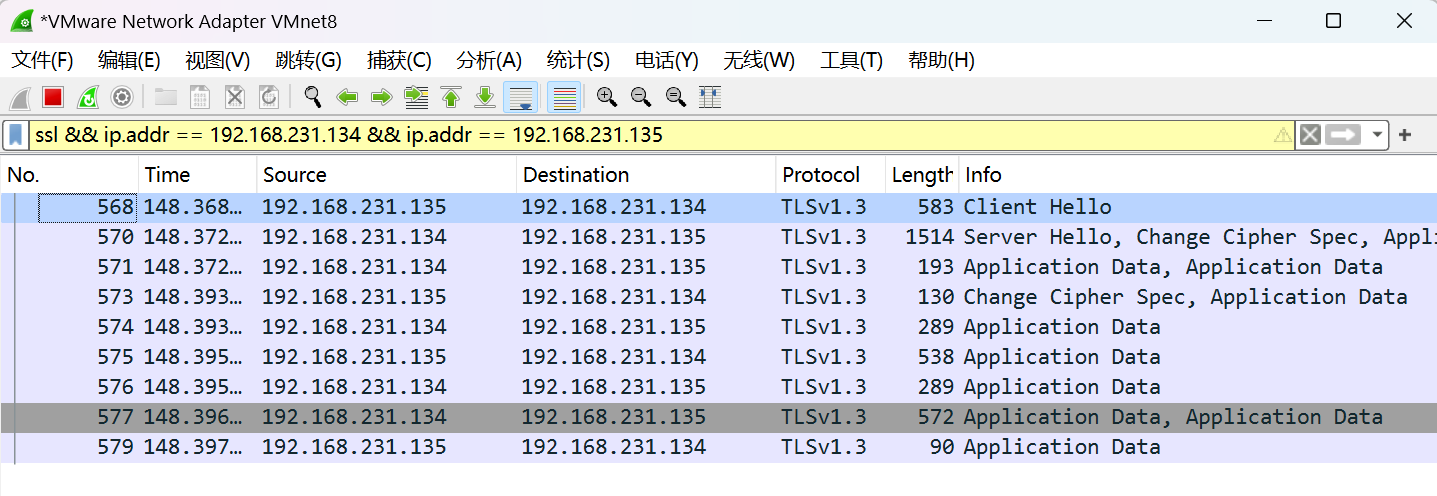
complete

File sent!!Closing socket!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**8.2 OpenSSL的报文分析实验**

SSL报文如下：



注：TLS是SSLv3的标准化版。

**9 尝试自己生成证书：**

证书的格式：

.key格式：私有的密钥

.csr格式：证书签名请求（证书请求文件），含有公钥信息，certificate signing request的缩写

.crt格式：证书文件，certificate的缩写

.crl格式：证书吊销列表，Certificate Revocation List的缩写

.pem格式：用于导出，导入证书时候的证书的格式，有证书开头，结尾的格式

生成CA根证书的步骤：

1. 生成CA的RSA私钥(.key)

openssl genrsa -des3 -out ca.key 1024

1. 生成根证书申请文件(.csr)

openssl req -new -key ca.key -out ca.csr

1. 生成自签名的crt根证书(.crt)

openssl x509 -req -extensions v3\_ca -days 3650 -signkey ca.key -in ca.csr -out ca.crt

1. 生成pem根证书

openssl req -x509 -new -nodes -key ca.key -sha256 -days 3650 -out ca\_cert.pem

生成服务器端用户证书

1. 生成服务器密钥

openssl req -new -sha256 -nodes -out server.csr -newkey rsa:1024 -keyout server\_private.key

1. 生成crt服务器证书

openssl x509 -req -in server.csr -CA rootCA\_cert.pem -CAkey rootCA\_private.key -CAcreateserial -out server\_cert.crt