



网络安全技术第 3 作业

--openssl 实现 https-web 服务器

姓 名:	柳致远
学 号:	2113683
专 业:	密码科学技术
指导老师:	贾岩

目录

一、	实验内容说明.....	3
1、	实验背景	3
2、	实验目的	3
3、	实验要求	3
4、	实验准备	3
二、	实验代码分析.....	4
1、	创建 SSL 上下文	4
2、	配置 SSL 上下文	4
3、	配置 TCP 套接字	5
4、	处理客户端请求	6
三、	实验过程及截图.....	6
	客户端.....	6
	服务器端.....	7
四、	实验结果分析.....	7

一、实验内容说明

1、实验背景

SSL (Secure Sockets Layer) 是一种用于保护网络通信安全性的协议，它的发展历史可以追溯到 1990 年代初。

SSL/TLS 协议通常会维护会话状态，以便在同一连接上的后续通信可以复用相同的会话密钥，提高效率。

总的来说，SSL/TLS 协议的工作原理通过数字证书验证、密钥协商和对称密钥加密等步骤，确保了网络通信的安全性和隐私保护。这使得敏感数据在互联网上的传输变得更加安全，同时防止了窃听、篡改和伪装攻击。SSL/TLS 协议在安全性和加密通信方面的重要性不可忽视，因此被广泛应用于 Web 浏览器、电子邮件、文件传输等各种网络应用中。

2、实验目的

- 理解 HTTPS 协议及 SSL 协议的工作原理。
- 掌握使用 OpenSSL 编程的方法。
- 提高网络安全系统设计的能力。

3、实验要求

- ✓ 运行于 Linux 平台
- ✓ 利用 OpenSSL 库编写一个 Web 服务器
- ✓ 实现 HTTPS 协议下最基本的 GET 命令功能

4、实验准备

在 Ubuntu 环境下安装 OpenSSL。编程语言为 C++ 语言，并安装实验所需的库文件。

二、实验代码分析

1、创建 SSL 上下文

```
SSL_CTX* create_context() {  
    const SSL_METHOD* method;  
    SSL_CTX* ctx;  
  
    method = SSLv23_server_method();  
  
    ctx = SSL_CTX_new(method);  
    if (!ctx) {  
        perror("Unable to create SSL context");  
        ERR_print_errors_fp(stderr);  
        exit(EXIT_FAILURE);  
    }  
  
    return ctx;  
}
```

该函数返回一个指向 `SSL_CTX` 结构的指针，该结构用于管理 SSL/TLS 会话的上下文信息。声明指向 `SSL_METHOD` 结构的指针 `method`，和指向 `SSL_CTX` 结构的指针 `ctx`，并在 `method` 获取 ssl 方法后创建 ssl 上下文。经过错误检查后返回上下文 `ctx`。

2、配置 SSL 上下文

```
void configure_context(SSL_CTX* ctx) {  
    if (SSL_CTX_use_certificate_file(ctx, "server.crt", SSL_FILETYPE_PEM) <= 0) {  
        ERR_print_errors_fp(stderr);  
        exit(EXIT_FAILURE);  
    }  
  
    if (SSL_CTX_use_PrivateKey_file(ctx, "server.key", SSL_FILETYPE_PEM) <= 0) {  
        ERR_print_errors_fp(stderr);  
        exit(EXIT_FAILURE);  
    }  
}
```

给传入的 SSL 上下文配置证书文件 `server.crt`，以及私钥文件 `server.key`。并确保在加载证书或私钥失败时，进行适当的错误处理，并终止程序。

3、配置 TCP 套接字

```
int create_socket(int port) {
    int sock;
    struct sockaddr_in addr;

    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0) {
        perror("Unable to create socket");
        exit(EXIT_FAILURE);
    }

    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(sock, (struct sockaddr*)&addr, sizeof(addr)) < 0) {
        perror("Unable to bind");
        exit(EXIT_FAILURE);
    }

    if (listen(sock, 1) < 0) {
        perror("Unable to listen");
        exit(EXIT_FAILURE);
    }

    return sock;
}
```

利用 `sock` 套接字配置 TCP 套接字，绑定套接字后监听所有传入的连接。

4、处理客户端请求

```
void handle_client(SSL* ssl) {
    char reply[] =
        "HTTP/1.1 200 OK\r\n"
        "Content-Type: text/plain\r\n"
        "Content-Length: 34\r\n"
        "\r\n"
        "Hello,i am lzy,nice to meet you!"
        "\r\n";

    char buffer[1024] = { 0 };
    int bytes = SSL_read(ssl, buffer, sizeof(buffer));

    if (bytes > 0) {
        buffer[bytes] = 0;
        std::cout << "Client msg: " << buffer << std::endl;

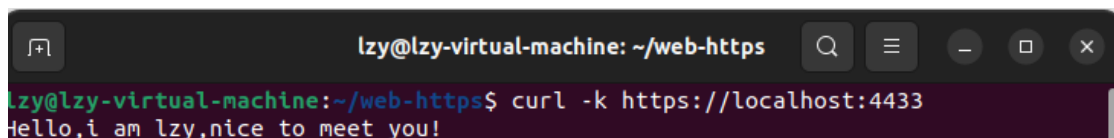
        if (strncmp(buffer, "GET ", 4) == 0) {
            SSL_write(ssl, reply, strlen(reply));
        }
    }
}
```

定义服务端返回给客户端的 **https** 响应报文，其中 **Content-Length: 34** 由我想让服务器相应给服务端的字符串字数反复调试得到。定义用于接收客户端请求的缓冲区并从 **SSL** 连接中读取数据，服务端输出从客户端读取的信息并检查读取的数据若为 **GET** 请求就返回给客户端预先准备的 **reply** 相应报文。

实验结束后注意 **ssl** 连接的关闭以及资源的释放。

三、实验过程及截图

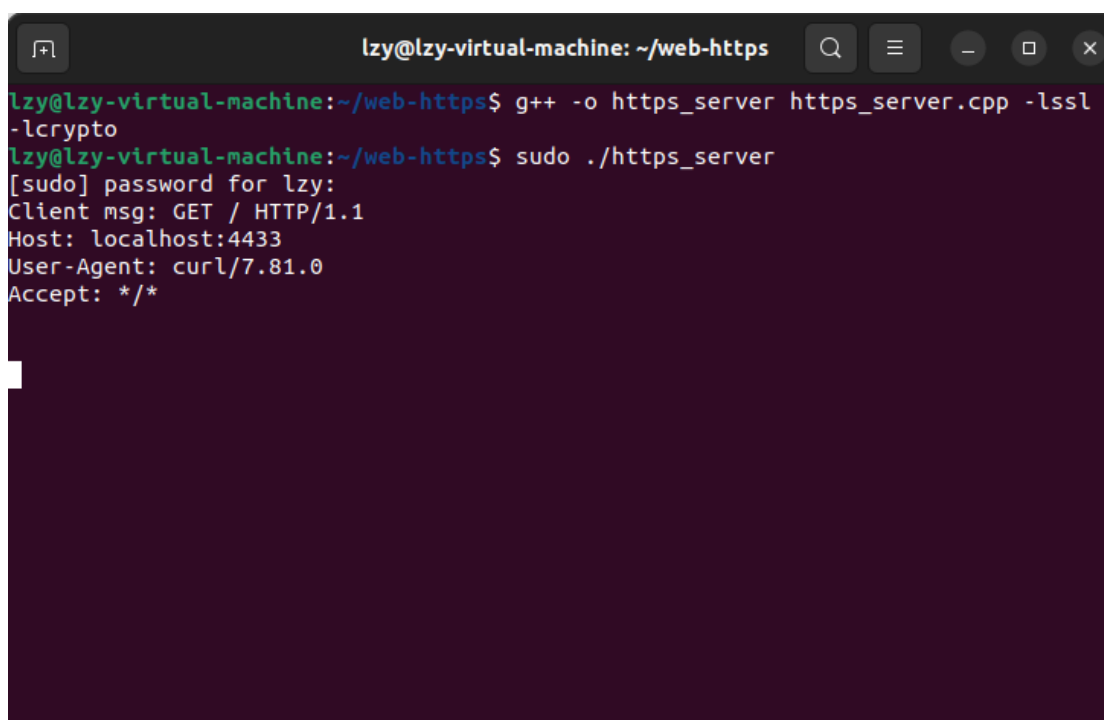
客户端



```
lzy@lzy-virtual-machine: ~/web-https
lzy@lzy-virtual-machine:~/web-https$ curl -k https://localhost:4433
Hello,i am lzy,nice to meet you!
```

curl 命令会给服务器发送一个 GET 命令，需要指出的是由于证书文件是自己生成的，故所以加上 ‘-k’ 参数。-k 参数是 curl 命令的一个选项，用于忽略 SSL 连接时的证书验证，即忽略对服务器证书的校验。在使用 -k 选项时，curl 将不会验证服务器的 SSL 证书是否有效，包括证书的合法性、签发者等，这样可以避免因证书问题而导致的连接失败或警告。

服务器端

A terminal window titled 'lzy@lzy-virtual-machine: ~/web-https'. The user enters the command 'g++ -o https_server https_server.cpp -lssl -lcrypto'. Then, they enter 'sudo ./https_server'. The terminal shows the output: '[sudo] password for lzy:', 'Client msg: GET / HTTP/1.1', 'Host: localhost:4433', 'User-Agent: curl/7.81.0', and 'Accept: */*'.

```
lzy@lzy-virtual-machine: ~/web-https
lzy@lzy-virtual-machine:~/web-https$ g++ -o https_server https_server.cpp -lssl
-lcrypto
lzy@lzy-virtual-machine:~/web-https$ sudo ./https_server
[sudo] password for lzy:
Client msg: GET / HTTP/1.1
Host: localhost:4433
User-Agent: curl/7.81.0
Accept: */*
```

四、实验结果分析

实验结果表明服务器与客户端之间 ssl 连接成功，并成功接收由客户端发来的 GET 请求，成功读取客户端发来的信息，并输出读取到的信息，客户端也成功接收服务端发回的 https 响应报文。