码龄14年 暂无认证

867 原创 | 2129 周排名 | 963 总排名 | 74万+ 访问 | 等级

1万+ 积分 | 5708 粉丝 | 336 获赞 | 76 评论 | 2140 收藏

私信 关注

搜博主文章

热门文章

Mysql中用SQL增加、删除字段，修改字段名、字段类型、注释，调整字段顺序总结 56088

Raid0、 Raid1、 Raid5、 Raid10的原理、特点、性能区别 47878

数据库连接不上的几种情况 35200

Nginx跨域配置 25385

人工智能-OpenCV+Python实现人脸识别（视频人脸检测） 25142

最新评论

讲解MySQL8.0备份与还原工具(mysqlba...
老哥讲数据库: hello,转载我的文章还标注原创，不太好吧?

vc转换bcb lib
子正: hi, Mr. huang. Thanks a lot, my bro. This articles may save me about 3 hour...

六级病历对门诊电子病历系统功能要求的...
weixin_40471909: 没有新的更新了吗？想继续学习

c#接口和抽象类的区别
网络收藏家: 总算明白了，一点啥叫接口和抽像类，那个比喻非常好，看了上面一大...

Raid0、 Raid1、 Raid5、 Raid10的原理...
木棉脚: 3

您愿意向朋友推荐“博客详情页”吗?

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

经营目标量化考核

net xml获取固定节点及属性

使用nginx部署多个前端项目

2022年 35篇 2021年 83篇
2020年 321篇 2019年 445篇

原创 huang714 于 2020-03-10 08:54:47 发布 2246 收藏 25
分类专栏： OpenSSL 版权

OpenSSL 专栏收录该内容 2 订阅 24 篇文章 订阅专栏

一、背景说明

1.1 面临问题

最近一份产品检测报告建议使用基于pki的认证方式，由于产品已实现https，商量之下认为其意思是使用双向认证以处理中间人形式攻击。

《信息安全工程》中接触过双向认证，但有两个问题。

第一个是当时最终的课程设计客户端是浏览器，服务端是tomcat双向认证只需要对两者进行配置并不需要自己真的实现代码。

第二个是虽然课程也有接近双向认证的实现代码，但当时是Java+JCE环境现在要用C++ OpenSSL环境，总体意思确实还是差不多但具体函数和参数差别还是不少。

所以眼下有的是：证书生成的思想+双向认证实现的思想。对读者而言，即要假定已对证书、 SSL/TSL、socket编程等几个概念有基本的了解，本文不做详细介绍。

基于此本文要解决的问题是：openssl具体如何生成证书+openssl如何实现双向认证。

1.2 解决办法

1.2.1 openssl具体生成证书解决办法

参考https://blog.csdn.net/gengxiaoming7/article/details/78505107再加其他一些文章总结而成

1.2.2 openssl实现双向认证解决办法

使用https://blog.csdn.net/sardden/article/details/42705897代码实SSL，在其基础上实现双向认证。

双向认证的关键点在以下几个函数（服务端和客户端都一样），其他就不细说参看代码注释：

SSL_CTX_set_verify----配置启用双向认证

SSL_CTX_load_verify_locations----加载信任的根证书

SSL_CTX_use_certificate_file----加载自己的证书

SSL_CTX_use_PrivateKey_file----加载自己的私钥

SSL_get_verify_result----真正进行验证，一定要调用这个函数不然前面四个光配置而已并不会进行双向验证

二、双向认证程序实现

2.1 安装openssl及开发api

```
apt-get install libssl-dev
```

2.2 服务端代码

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>
#include <arpa/inet.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);
    // SSL_get_verify_result()是重点，SSL_CTX_set_verify()只是配置启不启用并没有执行认证，调用该函数才会真证进行证书认证
    // 如果验证不通过，那么程序抛出异常中止连接
    if(SSL_get_verify_result(ssl) == X509_V_OK){
        printf("证书验证通过\n");
    }
    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");
}

int main(int argc, char **argv) {
    int sockfd, new_fd;
    socklen_t len;
    struct sockaddr_in my_addr, their_addr;
    unsigned int myport, lisnum;
    char buf[MAXBUF + 1];
    SSL_CTX *ctx;

    if (argv[1])
```


分类专栏

考核 1篇
JAVA 2篇
医院信息资源 13篇
医学知识 8篇
MYSQL 7篇
Liunx 11篇
VM 2篇
SQL优化总结 34篇
nginx 1篇
DOS 2篇
Nginx 1篇
TCP/IOCP/Socket 5篇
API 11篇
办公资源 1篇
CEF4_CefSharp_delphi 4篇
初学者学习SQL SERVER 287篇
医疗软件 1篇
SQL Server 数据库性能优化 5篇
财务资料 2篇
PB系统函数介绍 22篇
VUE 1篇
项目管理 5篇
消息队列MQ对比 2篇
软件质量
程序员的工具 1篇
XML 1篇
算法 4篇
思科交换机命令 1篇
MySQL数据库常用命令 1篇
C++ 实践讲述 38篇
OpenSSL 24篇
生成随机码的几种方法 1篇
MSMQ消息队列 1篇
架构设计基础 6篇
数据源管理 8篇
SYBASE 1篇
ERP 1篇
Sqlite 4篇
WPS 1篇
HTTP 3篇
EXCEL 2篇
笔记 3篇
FastReport 1篇
SQL Server 16篇
SQL Server 语句实现 295篇
C++ builder 63篇
Delphi 44篇
VC++ 13篇
MFC 4篇
C# 31篇
Net 10篇
python 41篇
STL 3篇
C 25篇
C++ 56篇
Powber Builder 31篇

```
    myport = atoi(argv[1]);
else
    myport = 7838;

if (argv[2])
    lisnum = atoi(argv[2]);
else
    lisnum = 2;

/* SSL 库初始化 */
SSL_library_init();
/* 载入所有 SSL 算法 */
OpenSSL_add_all_algorithms();
/* 载入所有 SSL 错误消息 */
SSL_load_error_strings();
/* 以 SSL V2 和 V3 标准兼容方式产生一个 SSL_CTX ，即 SSL Content Text */
ctx = SSL_CTX_new(SSLv23_server_method());
/* 也可以用 SSLv2_server_method() 或 SSLv3_server_method() 单独表示 V2 或 V3标准 */
if (ctx == NULL) {
    ERR_print_errors_fp(stdout);
    exit(1);
}

// 双向验证
// SSL_VERIFY_PEER---要求对证书进行认证，没有证书也会放行
// SSL_VERIFY_FAIL_IF_NO_PEER_CERT---要求客户端需要提供证书，但验证发现单独使用没有证书也会放行
SSL_CTX_set_verify(ctx, SSL_VERIFY_PEER|SSL_VERIFY_FAIL_IF_NO_PEER_CERT, NULL);
// 设置信任根证书
if (SSL_CTX_load_verify_locations(ctx, "ca.crt",NULL)<=0){
    ERR_print_errors_fp(stdout);
    exit(1);
}

/* 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 */
if (SSL_CTX_use_certificate_file(ctx, argv[3], SSL_FILETYPE_PEM) <= 0) {
    ERR_print_errors_fp(stdout);
    exit(1);
}
/* 载入用户私钥 */
if (SSL_CTX_use_PrivateKey_file(ctx, argv[4], SSL_FILETYPE_PEM) <= 0) {
    ERR_print_errors_fp(stdout);
    exit(1);
}
/* 检查用户私钥是否正确 */
if (!SSL_CTX_check_private_key(ctx)) {
    ERR_print_errors_fp(stdout);
    exit(1);
}

/* 开启一个 socket 监听 */
if ((sockfd = socket(PF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket");
    exit(1);
} else
    printf("socket created\n");

bzero(&my_addr, sizeof(my_addr));
my_addr.sin_family = PF_INET;
my_addr.sin_port = htons(myport);
my_addr.sin_addr.s_addr = INADDR_ANY;

if (bind(sockfd, (struct sockaddr *) &my_addr, sizeof(struct sockaddr))
        == -1) {
    perror("bind");
    exit(1);
} else
    printf("binded\n");

if (listen(sockfd, lisnum) == -1) {
    perror("listen");
    exit(1);
} else
    printf("begin listen\n");

while (1) {
    SSL *ssl;
    len = sizeof(struct sockaddr);
    /* 等待客户端连上来 */
    if ((new_fd = accept(sockfd, (struct sockaddr *) &their_addr, &len))
            == -1) {
        perror("accept");
        exit(errno);
    } else
        printf("server: got connection from %s, port %d, socket %d\n",
                inet_ntoa(their_addr.sin_addr), ntohs(their_addr.sin_port),
                new_fd);

    /* 基于 ctx 产生一个新的 SSL */
    ssl = SSL_new(ctx);
    /* 将连接用户的 socket 加入到 SSL */
    SSL_set_fd(ssl, new_fd);
    /* 建立 SSL 连接 */
    if (SSL_accept(ssl) == -1) {
        perror("accept");
        close(new_fd);
        break;
    }
    ShowCerts(ssl);

    /* 开始处理每个新连接上的数据收发 */
    bzero(buf, MAXBUF + 1);
    strcpy(buf, "server->client");
    /* 发消息给客户端 */
    len = SSL_write(ssl, buf, strlen(buf));

    if (len <= 0) {
        printf("消息'%s'发送失败！错误代码是%d，错误信息是'%s'\n", buf, errno,
                strerror(errno));
        goto finish;
    } else
        printf("消息'%s'发送成功，共发送了%d个字节！\n", buf, len);

    bzero(buf, MAXBUF + 1);
    /* 接收客户端的消息 */
    len = SSL_read(ssl, buf, MAXBUF);
    if (len > 0)
        printf("接收消息成功:'%s'，共%d个字节的数据\n", buf, len);
    else
        printf("消息接收失败！错误代码是%d，错误信息是'%s'\n",
        errno, strerror(errno));
    /* 处理每个新连接上的数据收发结束 */
    finish:
    /* 关闭 SSL 连接 */
```

```
        SSL_shutdown(ssl);
        /* 释放 SSL */
        SSL_free(ssl);
        /* 关闭 socket */
        close(new_fd);
    }
    /* 关闭监听的 socket */
    close(sockfd);
    /* 释放 CTX */
    SSL_CTX_free(ctx);
    return 0;
}
```

2.3 客户端代码

```
#include <stdio.h>
#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
```

2.4 证书生成

注意三点

第一点，注意将其中的私钥加密密码（-passout参数）修改成自己的密码；下边都是以带-passout参数生成私钥，如果使用-nodes参数，则最后一步"将加密的RSA密钥转成未加密的RSA密钥"不需要执行。

第二点，证书和密钥给出了直接一步生成和分步生成两种形式，两种形式是等价的，这里使用直接生成形式（分步生成形式被注释）

第三点，注意将其中的证书信息改成自己的组织信息的。其中证数各参数含义如下：

C-----国家（Country Name）

ST----省份（State or Province Name）

L----城市（Locality Name）

O----公司（Organization Name）

OU----部门（Organizational Unit Name）

CN----产品名（Common Name）

emailAddress----邮箱（Email Address）

```
# CA证书及密钥生成方法一----直接生成CA密钥及其自签名证书
# 如果想以后读取私钥文件ca_rsa_private.pem时不需要输入密码，亦即不对私钥进行加密存储，那么将-passout pass:123456替换成-node
openssl req -newkey rsa:2048 -passout pass:123456 -keyout ca_rsa_private.pem -x509 -days 365 -out ca.crt -subj "/
# CA证书及密钥生成方法二----分步生成CA密钥及其自签名证书：
# openssl genrsa -aes256 -passout pass:123456 -out ca_rsa_private.pem 2048
# openssl req -new -x509 -days 365 -key ca_rsa_private.pem -passin pass:123456 -out ca.crt -subj "/C=CN/ST=GD/L=S

# 服务器证书及密钥生成方法一----直接生成服务器密钥及待签名证书
# 如果想以后读取私钥文件server_rsa_private.pem时不需要输入密码，亦即不对私钥进行加密存储，那么将-passout pass:server替换成-
openssl req -newkey rsa:2048 -passout pass:server -keyout server_rsa_private.pem  -out server.csr -subj "/C=CN/ST
# 服务器证书及密钥生成方法二----分步生成服务器密钥及待签名证书
# openssl genrsa -aes256 -passout pass:server -out server_rsa_private.pem 2048
# openssl req -new -key server_rsa_private.pem -passin pass:server -out server.csr -subj "/C=CN/ST=GD/L=SZ/O=COM/
# 使用CA证书及密钥对服务器证书进行签名：
openssl x509 -req -days 365 -in server.csr -CA ca.crt -CAkey ca_rsa_private.pem -passin pass:123456 -CAcreateseri
# 将加密的RSA密钥转成未加密的RSA密钥，避免每次读取都要求输入解密密码
# 密码就是生成私钥文件时设置的passout、读取私钥文件时要输入的passin，比如这里要输入"server"
openssl rsa -in server_rsa_private.pem -out server_rsa_private.pem.unsecure

# 客户端证书及密钥生成方法一----直接生成客户端密钥及待签名证书
# 如果想以后读取私钥文件client_rsa_private.pem时不需要输入密码，亦即不对私钥进行加密存储，那么将-passout pass:client替换成-
openssl req -newkey rsa:2048 -passout pass:client -keyout client_rsa_private.pem -out client.csr -subj "/C=CN/ST=
# 客户端证书及密钥生成方法二----分步生成客户端密钥及待签名证书：
# openssl genrsa -aes256 -passout pass:client -out client_rsa_private.pem 2048
# openssl req -new -key client_rsa_private.pem -passin pass:client -out client.csr -subj "/C=CN/ST=GD/L=SZ/O=COM/
# 使用CA证书及密钥对客户端证书进行签名：
openssl x509 -req -days 365 -in client.csr -CA ca.crt -CAkey ca_rsa_private.pem -passin pass:123456 -CAcreateseri
# 将加密的RSA密钥转成未加密的RSA密钥，避免每次读取都要求输入解密密码
# 密码就是生成私钥文件时设置的passout、读取私钥文件时要输入的passin，比如这里要输入"client"
openssl rsa -in client_rsa_private.pem -out client_rsa_private.pem.unsecure
```

2.5 开发环境配置

操作系统----kali-roaling。为了使用现在虚拟机而已，使用ubuntu、centos等等应该都是没差别的。

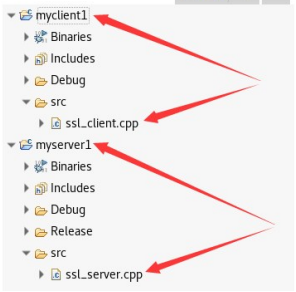
IDE----eclipse。直接在终端中编译不通过没深究，放eclipse编译没问题就直接使用eclipse。

2.5.1 在同一活动目录下建了两个project

myclient1----建src文件夹放客户端代码

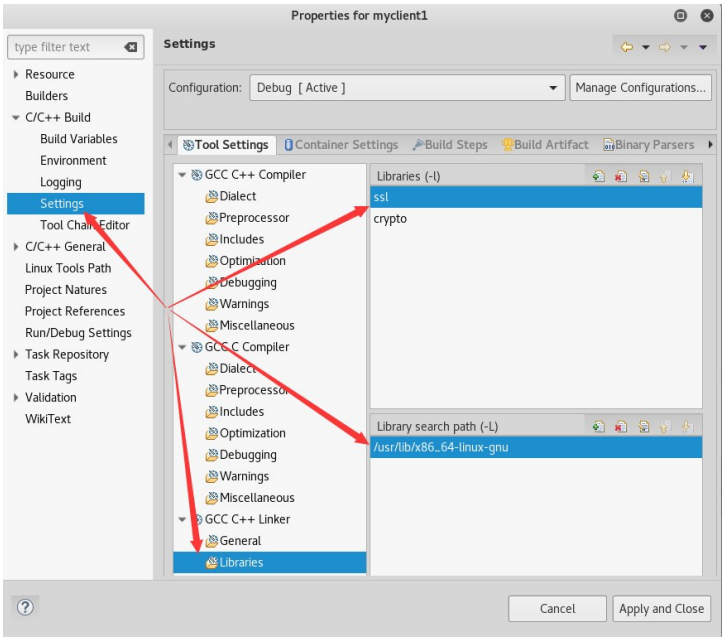
myserver1----建src文件夹放服务端代码

（其他目录要么是自动生成的，要么是编译后自动生成的，不用管；如果项目有报错试试多重启几次eclipse甚操作系统）



2.5.2 指定ssl和crypto

在项目文件夹上右键----Properties----指定ssl库和crypto库目录不然编译找不到ssl。两个project都要配置



2.5.3 编译

使用Ctrl+B快捷键进行编译，eclipse会编译所有project

2.5.4 证书复制

将前边生成的ca证书（ca.crt）、客户端证书（client.crt）、客户端未加密私钥文件（）复制到myclient1项目的Debug目录下

将前边生成的ca证书、服务端证书、服务端端未加密私钥文件复制到myclient1项目的Debug目录下

2.5.5 运行程序

先运行服务端后运行客户端

```
./myserver1 7838 1 server.crt server_rsa_private.pem.unsecure
./myclient1 127.0.0.1 7838 client.crt client_rsa_private.pem.unsecure
```

运行结果如下，服务端：

客户端：

```
root@kali:~/Desktop/eclipse/eclipse-workspace/myclient1/Debug# ls
ca.crt  client.crt  client_rsa_private.pem.unsecure  makefile  myclient1  objects.mk  sources.mk  src
root@kali:~/Desktop/eclipse/eclipse-workspace/myclient1/Debug#
root@kali:~/Desktop/eclipse/eclipse-workspace/myclient1/Debug# ./myclient1 127.0.0.1 7838 client.crt client_rsa_private.pem.unsecure
socket created
address created
server connected
Connected with ECDHE-RSA-AES256-GCM-SHA384 encryption
证书验证
数字证书(
证书: /C=CN/ST=GD/L=SZ/O=TVT/OU=NSP/CN=SERVER/emailAddress=who602@qq.cc
颁发者: /C=CN/ST=GD/L=SZ/O=TVT/OU=NSP/CN=CA/emailAddress=who602@qq.c
接收消息成功:'server->client'，共14个
消息'from client->server'发送成功，共发送
root@kali:~/Desktop/eclipse/eclipse-workspace/myclient1/Debug#
```

OpenSSL中调用OpenSSL_add_all_algorithms内存泄漏之问题调查 pony12的专栏 8198
目前，基于OpenSSL开发，初始化阶段调用了OpenSSL_add_all_algorithms();，循环测试后，发现有内存泄漏。 后来根据http://www.cnblogs.com/moon…

SSL双向认证的认证模式设置问题 enjoy.day 7355
今天介绍一下SSL双向认证服务器端认证模式的设置，很关键。 初始化ssl时，要使用SSL_CTX_set_verify函数设置验证模式，下面介绍下集中模式： Op…

ssl双向认证-openssl生成证书_长江空自流的博客 9-8
ssl双向认证-openssl生成证书 一、前言 已有教程大部分是使用jdk自带keytool生成自签名证书,自己测试可以的,但是用于实际使用时,在ssl双向认证中服务…

openssl实现双向认证教程(服务端代码+客户端代码+证书... 9-22
openssl实现双向认证教程(服务端代码+客户端代码+证书生成) 参考链接 注意事项 openssl版本差异很可能导致程序编译与运行出现问题 本程序在OpenS…

OpenSSL之SSL_CTX_use_certificate_file分析 u012023606的博客 5146
OpenSSL之SSL_CTX_use_certificate_file分析 本系列OpenSSL使用的代码版本为：1.0.2o 文章目录 系列文章目录 前言 一、pandas是什么？ 二、使用…

在OpenSSL中添加自定义加密算法 热门推荐 奋斗中拥有 1万+
在OpenSSL中添加自定义加密算法 1.加密算法的加载... 12.密码算法接口的定义... 43.示例... 8 1.加密算法的加载在调用加密算法之前，通过调用OpenSS…

SSL 服务器与客户端样本代码 最新发布 白袍小将的博客 327
本文为了方便进行SSL编程参考而整理，SSL编程调用的大多数流程就如如下样本代码，通过本样本代码可以比较快的测试SSL相关API。

Nginx启动，证书报错SSL_CTX_use_PrivateKey_file..... weixin_30254435的博客 1万+
报错nginx: [emerg] SSL_CTX_use_PrivateKey_file("/etc/nginx/ssl/myxxxxgame201904.key") failed (SSL: error:0906D06C:PEM routines:PEM_read_bio:…

OpenSSL创建生成CA证书、服务器、客户端证书及密钥 雍正不秃头 4301
使用OpenSSL创建生成CA证书、服务器、客户端证书及密钥 目录使用OpenSSL创建生成CA证书、服务器、客户端证书及密钥（一）生成CA证书（二）…

openssl双向认证 （自签发证书实例） chen55bo的专栏 9172
基础知识 SSL：Secure Socket Layer，安全套接字层，它位于TCP层与Application层之间。提供对Application数据的加密保护（密文），完整性保护（不…

基于OpenSSL 生成Nginx双向证书校验 风信子的专栏 126
1、创建Root CA私钥 openssl genrsa -out root-ca.key 1024 2、创建Root CA证书请求 openssl req -new -out root-ca.csr -key root-ca.key 3、签发Root …

openssl自签名证书生成与单双向验证 zhulianhai0927的专栏 531
https://www.csdn.net/gather_22/MtzaIgysNDgtYmxvZw0O0OO0O0O.html

用openssl创建双向证书 sking777的专栏 541
1. 创建根证书密钥文件(自己做CA)root.key： openssl genrsa -des3 -out root.key 输出内容为： [lenin@archer ~]$ openssl genrsa -des3 -out root.key …

OpenSSL之SSL_CTX_use_PrivateKey_file分析 u012023606的博客 4373
OpenSSL之SSL_CTX_use_PrivateKey_file分析 本系列OpenSSL使用的代码版本为：1.0.2o 前言 本篇文章纯属个人学习的一点经验分享，若有不对之处…

openssl生成证书及简化身份验证 louObaichu的专栏 1432
概述 本文将给出数个用于生成SSL证书的bat脚本，并探讨当不需要身份验证时可行的简化方案。 本文面向openssl初学者，但需要一定的密码学基础知识…

OpenSSL 双向认证 wangsifu2009的专栏 1万+
在使用OpenSSL进行SSL双向认证时，需要在服务器和客户端配置如下接口函数： SSL_CTX_set_verify(SSL_CTX* ctx,int mode,int (*verify_callback)(int,…

使用openssl进行RSA加密解密 fang437385323的专栏 7880
原文：http://blog.csdn.net/zzj806683450/article/details/17426193 我使用openssl 1.0.1e，过程中遇到一些问题。 #include "stdafx.h" #include #include …

“相关推荐”对你有帮助么？
非常没帮助 没帮助 一般 有帮助 非常有帮助