使用OpenSSL加载证书文件的过程分析与代码示例

一般说来，当前主流网站都走的单项认证的路子，即只有服务器需向客户端发送证书，客户端不需向服务器发送证书。在这种情况下，加载证书是服务端需要做的事情。所以下面给个基于openssl的SSL服务端例程，内含加载证书的代码（有特别标注）：

```
    // 前面省略了服务端socket套接字的创建过程
    // 接受客户端的socket连接
    m_nConversion = accept(nListen, (sockaddr *)&addr_client, &size);
    if (m_nConversion == -1)
    {
        printf("accept failed!\n");
        continue;
    }

    // 创建服务端SSL会话环境
    m_pServerCtx = SSL_CTX_new(SSLv23_server_method());
    if (m_pServerCtx == 0)
    {
        printf("SSL_CTX_new for Server failed!\n");
        return -1;
    }

/*-------------------------Begin of:服务端公私钥加载-------------------------*/
    // 为服务端指定SSL连接所用公钥证书
    //参数 m_pServerCtx，服务端SSL会话环境
    //参数 pCertPath，你存放公钥证书的路径
    //参数 SSL_FILETYPE_PEM，指定你所要加载的公钥证书的文件编码类型为 Base64
    if (SSL_CTX_use_certificate_file(m_pServerCtx, pCertPath, SSL_FILETYPE_PEM) != 1)
    {
        printf("SSL_CTX_use_certificate_file failed!\n");
        return -1;
    }
    // 为服务端指定SSL连接所用私钥
    //参数 m_pServerCtx，服务端SSL会话环境
    //参数 pKeyPath，你存放对应私钥文件的路径
    //参数 SSL_FILETYPE_PEM，指定你所要加载的私钥文件的文件编码类型为 Base64
    if (SSL_CTX_use_PrivateKey_file(m_pServerCtx, pKeyPath, SSL_FILETYPE_PEM) != 1)
    {
        printf("SSL_CTX_use_PrivateKey_file failed!\n");
        return -1;
    }
    // 检查SSL连接 所用的私钥与证书是否匹配【所以你仅有公钥证书是不够的】
    if (!SSL_CTX_check_private_key(m_pServerCtx))
    {
```

目录

分类专栏

```
        printf("Private key does not match the certificate public key\n");
        return -1;
    }
/*-------------------------End of:服务端公私钥加载-------------------------*/

    // 创建一个与客户端通信的SSL套接字
    m_pServerSSL = SSL_new(m_pServerCtx);
    if (m_pServerSSL == 0)
    {
        printf("SSL_new for Server failed!\n");
        return -1;
    }
    // 将与客户端通信的 SSL套接字&&socket套接字 进行可读写地绑定
    SSL_set_fd(m_pServerSSL, m_nConversion);
    // 接受客户端的SSL连接
    if (SSL_accept(m_pServerSSL) == -1)
    {
        printf("SSL_set_fd for Server failed!\n");
        return -1;
    }
    //后面省略的是基于SSL_read()与SSL_write()的SSL通信过程
```

一、关于证书文件的编码类型

这是我需要特别补充的第一点，也就是加载公私钥文件API函数的第三个参数。当前证书文件有两种编码类型，即：二进制编码【宏定义为SSL_FILETYPE_ASN1】与ASCII(Base64)【宏定义为SSL_FILETYPE_PEM】编码。对于公钥证书的加载，它两种类型都支持，但函数本身并不能自动同时识别处理两种类型，必须由用户在调用的时候根据自己所用文件的类型自行指定；对于私钥文件的加载，则仅支持SSL_FILETYPE_PEM。

第三个参数的取值务必和你要加载的公钥证书文件的编码类型相匹配。若是你第三个参数设为SSL_FILETYPE_PEM，实际加载的却是一个二进制编码的证书文件，加载就会出错。假设你要加载的公钥证书是一个通过浏览器的导出的cer文件，那么仅凭cer这个扩展名还无法断定文件编码类型。其实在你导出文件的时候，是有指定编码类型的，不知你是否还记得下图：

如果你当时按照默认一路点下去，那么就该使用SSL_FILETYPE_ASN1类型加载；反之，如果你选定了第二项，那么就该使用SSL_FILETYPE_PEM类型加载。如果你不记得当时怎么选的，那么就用记事本打开证书文件，若有乱码，则说明是二进制编码，该使用SSL_FILETYPE_ASN1类型加载；反之，若皆可识别，则说明经过了Base64编码，该使用SSL_FILETYPE_PEM类型加载。

通过抓包从数据包里获取的证书一般都属于二进制编码，使用SSL_FILETYPE_ASN1类型加载即可。

如果想要了解证书文件的扩展名与其编码类型的关系，请参考这个：《**电子证书 DER vs. CRT vs. CER vs. PEM .**》。

二、如何加载证书链

前面我所举的例子中，只加载了应用本身的公钥证书，如果你想加载其完整的证书链，又该怎么做呢？你可以使用下面这两个函数：

```
int SSL_CTX_use_certificate_chain_file(SSL_CTX *ctx, const char *file)  //加载完整的证书链
long SSL_CTX_add_extra_chain_cert(SSL_CTX ctx, X509 *x509)              //向证书链上附加证书
```

2.1 SSL_CTX_use_certificate_chain_file

对比前文用到的SSL_CTX_use_certificate_file函数，前两个参数的意义和用法基本是类似的，只是缺少了第三个参数，文件编码类型。这是因为本函数要求证书链文件的格式必须为PEM格式，使用Base64编码，对应前文的SSL_FILETYPE_PEM。

在调用本函数之前，你需要把应用证书和对其签名的CA的证书合并到一个文件中。合并后的文件，内容大致如下：

```
-----BEGIN CERTIFICATE-----
MIIDHDCCAoWgAwIBAgIBAjANBgkqhkiG9w0BAQUFADBcMQswCQYDVQQGEwJVUzES
MBAGA1UECBMJQmVya3NoaXJlMRAwDgYDVQQHEwdOZXdidXJ5MRIwEAYDVQQKEwlD
QSBDZW50ZXIxEzARBgNVBAMTCnd3dy5jYS5jb20wHhcNMTQwMTIyMDkxMDI3WhcN
MTUwMTIyMDkxMDI3WjBLMQswCQYDVQQGEwJVUzESMBAGA1UECBMJQmVya3NoaXJl
MRIwEAYDVQQKEwlDQSBDZW50ZXIxFDASBgNVBAMMCyoudm94ZXIuY29tMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA5LnD7D2YNrSfKj73Gdq3JW/m2CTl
BbxqhIoM7YsYGf26bbp6fjsmrM8zf3JQim6Aaq6mhUCdFLj6yZnPDmfZXbzcLXfo
ubfiqBH/VjMx5yxOrHzIJhtOLiQ2bIvqKxO/neBgpZLcawYrs7FWm53aDKWeAQ3u
4IV22tkGwW8W7uprjDp4G33pvEdXt6QgwXlmKet9VsyDKgafh2r+gi9TpL9YFBEg
9qnZuIWy0mpeBmS1n5VYVNt4H4d4neReEq2WH2R9TGbAxp0FjFvMg0G4GkWKJMpl
c3AbYfakDlijFjwoXIVLhAIzMm0cTD2fy57LQXl1tkxWMO+XLWSfkKGrDwIDAQAB
o3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRl
ZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUKCIYrBq0IJiveHlT/1e3+CBU0x4wHwYD
VR0jBBgwFoAUKuqqlybSKZ/df8XQ01jUBDfHwJwwDQYJKoZIhvcNAQEFBQADgYEA
pCBtDPpWWJJNB+EylG3E2uEPEUf3QQNxCim6dZpYBvndrrdjsQF6rl03kDeCbyuG
Y6MFi8MXTkMceRdKEOGxNhEIPxNM3SBax823GCMNO30K/fYsQgD/lf2LhzC2FHEG
RNp8Bwr1me+4rS6S5qXO2pAYwWF5yYmtOmIRkqsws1k=
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIC+DCCAmGgAwIBAgIJAPVmFw/BUcffMA0GCSqGSIb3DQEBBQUAMFwxCzAJBgNV
BAYTAlVTMRIwEAYDVQQIEwlCZXJrc2hpcmUxEDAOBgNVBAcTB05ld2J1cnkxEjAQ
BgNVBAoTCUNBIENlbnRlcjETMBEGA1UEAxMKd3d3LmNhLmNvbTAeFw0xNDAxMjIw
OTA4MzlaFw0yNDAxMjAwOTA4MzlaMFwxCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlC
ZXJrc2hpcmUxEDAOBgNVBAcTB05ld2J1cnkxEjAQBgNVBAoTCUNBIENlbnRlcjET
MBEGA1UEAxMKd3d3LmNhLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA
5jzdgxYG2WucyVWp+e5n9szWVSchzok0vFkOT+3AVQshulTjdFtxzKMmM/WnizQ5
qcPQ/GbNb5FiihPbeFJxKtAslhhsF8s0zj/1lZmnsmSQpWUakuJmZblX9TyT5sUy
Wz9zIh01ctp90EmituryR2MCKPLtrgqn1CLQKnX6zZkCAwEAAaOBwTCBvjAdBgNV
HQ4EFgQUKuqqlybSKZ/df8XQ01jUBDfHwJwwgY4GA1UdIwSBhjCBg4AUKuqqlybS
KZ/df8XQ01jUBDfHwJyhYKReMFwxCzAJBgNVBAYTAlVTMRIwEAYDVQQIEwlCZXJr
c2hpcmUxEDAOBgNVBAcTB05ld2J1cnkxEjAQBgNVBAoTCUNBIENlbnRlcjETMBEG
A1UEAxMKd3d3LmNhLmNvbYIJAPVmFw/BUcffMAwGA1UdEwQFMAMBAf8wDQYJKoZI
hvcNAQEFBQADgYEA1Q0wATn2y3ZXQln4i0CstQTOGujjtM3keCkDFUTN6vrHOllt
YbIJ6ugvbuP2akzPx7YQUPATTXm05UsX/rJsuzcHFnkmPz2xxGrbS8J/vm3MDwK9
g5U2wMnjS/hPJ3yXf2CZz3e8Qx/AYicWwwdCv2UxX/qhe/Nn8PI99lTSibM=
-----END CERTIFICATE-----
```

每个证书文件以"-----BEGIN CERTIFICATE-----"标志开始，以"-----END CERTIFICATE-----"标志结束。证书两两之间不空行。如果发送的是完整的证书链，那最后一个就一定是自签名证书。如上例，是一个两层的证书链，放在最前面的是您自己的应用证书，紧邻其后的是对您应用证书签名的CA的证书（这已经是一个自签名证书）。调用SSL_CTX_use_certificate_chain_file函数的时候，其第二个参数就是指向这样一个文件的指针。

使用SSL_CTX_use_certificate_chain_file()代替SSL_CTX_use_certificate_file()，改写证书加载部分代码如下：

```
// 前面不变，略

/*-------------------------Begin of:服务端公私钥加载-------------------------*/
    // 为服务端指定SSL连接所用公钥证书的完整证书链
    //参数 m_pServerCtx，服务端SSL会话环境
    //参数 pCertPath，你存放证书链文件的路径
    if (SSL_CTX_use_certificate_chain_file(m_pServerCtx, pCertPath) != 1)
    {
        printf("SSL_CTX_use_certificate_chain_file failed!\n");
        return -1;
    }
    // 为服务端指定SSL连接所用私钥
```

```
    //参数 m_pServerCtx，服务端SSL会话环境
    //参数 pKeyPath，你存放对应私钥文件的路径
    //参数 SSL_FILETYPE_PEM，指定你所要加载的私钥文件的文件编码类型为 Base64
    if (SSL_CTX_use_PrivateKey_file(m_pServerCtx, pKeyPath, SSL_FILETYPE_PEM) != 1)
    {
        printf("SSL_CTX_use_PrivateKey_file failed!\n");
        return -1;
    }
    // 检查SSL连接 所用的私钥与证书是否匹配【所以你仅有公钥证书是不够的】
    if (!SSL_CTX_check_private_key(m_pServerCtx))
    {
        printf("Private key does not match the certificate public key\n");
        return -1;
    }
/*-------------------------End of:服务端公私钥加载-------------------------*/

// 后面不变，略
```

2.2 SSL_CTX_add_extra_chain_cert

今天时间不够了，详细的介绍等下次有空再写吧。

三、写在结尾

本文主要内容节选自作者本人在CSDN论坛上的回复帖，原帖链接：http://bbs.csdn.net/topics/390467536

私信 关注

搜博主文章

热门文章

使用OpenSSL工具制作X.509证书的方法及其注意事项总结 22896

使用WireShark分析HTTP协议时几种常见的汉字编码及其解码方法小结 21567

Wireshark入门与进阶---Capture Options各项的含义与设定 18688

王垠：完全用Linux工作 16058

王垠：清华梦的粉碎—写给清华大学的退学申请 2005.9.22 15959

最新评论

OpenSSL编程初探2 --- 关于证书文件的…
技术大白: SSL_CTX_add_extra_chain_cert，这个你咋不讲了，就用到这个

OpenSSL编程初探2 --- 关于证书文件的…
钛白先生: 可以，解释挺详细。

基于 tcpdump for Android 的智能移动终…
YuRi_Kwon: 请问您如何解决的问题呢？我的小米抓到的包也是空的

详谈为何两台主机网络掩码不一致可能导…
guogengcai 回复 sunhecool: 网段不应该是32位的吗，自己的IP地址和子网掩码做位…

在不同版本的Ubuntu系统中开启root账户…
1_____1: 楼主比较用心，写的可以

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

几张趣图助你理解HTTP状态码

如何知道一台Linux服务器使用的是千兆网卡还是万兆网卡

基于人工分析的HTTP-POST请求报文特征获取一般方法

2018年 1篇 2016年 1篇
2015年 4篇 2014年 36篇
2013年 41篇 2012年 11篇

调用OpenSSL实现数字签名功能例程（二） 求索 5935
// PKCS7Sign.cpp : Defines the entry point for the console application. // #include "stdafx.h" #include #include #include #include #include #include …

openssl基本原理 + 生成证书 + 使用实例_huang714的专栏 9-22
密钥文件的格式用OpenSSL生成的就只有PEM和DER两种格式,PEM的是将密钥用base64编码表示出来的,直接打开你能看到一串的英文字母,DER格式是…

调用OpenSSL实现数字签名功能例程(二)_wl_haanel的专栏 10-12
如何测试新证书? ... 浏览器打开 OpenSSL编程初探2 --- 关于证书文件的加载 热门推荐 使用OpenSSL加载证书文件的过程分析与代码示例 本文由CSDN-…

finereport 07-06
是用于报表设计的好工具。听说是比水晶报表功能还强大的报表设计工具。

机器视觉手眼标定详解，有用 03-14
目录 1. 相机固定不动，上往下看 引导 机器人 移动 2. 相机固定不动，下往上看 3. 相机固定在器人上 相机固定在器人上 ，离旋转中心较近 离旋转中心较…

OpenSSL编程初探3 --- 根据给定的域名自动伪造应用证书 10-29
在实现证书的自动生成前,必须先弄清楚使用OpenSSL命令手工制作证书的方法与步骤。以生成一个二级证书链为例,将会用到以下命令: // 生成顶级CA的…

利用OpenSSL生成证书文件的总结_larryliuqing的专栏-xxx 10-21
三.生成CA证书文件 server.csr与client.csr文件必须有CA的签名才可形成证书. 1.首先生成CA的key文件: openssl genrsa -des3 -out ca.key 1024 2.生成C…

Certificate Chain (证书链) 简述 O_o 1703
涉及 Certificate Chain.

OpenSSL之SSL_CTX_use_certificate_file分析 最新发布 u012023606的博客 2483
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…

openssl生成证书+安装+使用实例 (二)_zhangji 10-1
将证书导出成浏览器支持的.p12文件(在此输入的密码为证书安装时的密码) OpenSSL>pkcs12-export-clcerts-inca-cert.pem-inkeyca-key.pem-outca.p12 …

OpenSSL编程初探 huang714的专栏 109
1 --- 使用OpenSSL API建立SSL通信的一般流程简介 OpenSSL是一套开放源代码的SSL套件，其函数库是以C语言所写成，实现了基本的传输层数据加密…

openssl master源码目录 明潮的BLOG 865
├─apps | | apps.c | | apps.h | | app_rand.c | | asn1pars.c | | build.info | | ca-cert.srl | | ca-key.pem | | ca-req.pem | | ca.c | | CA.pl.in | | …

OpenSSL 解析P12格式证书文件 04-06
NULL 博文链接：https://jacky-dai.iteye.com/blog/1545241

FB15K 数据集 03-02
用在此处 https://github.com/thunlp/OpenKE For training, datasets contain three files: train2id.txt: training file, the first line is the number of triples for tr…

通过OpenSSL解码X509证书文件 热门推荐 密码开发者 1万+
通过OpenSSL解码X509证书文件，包括*.cer/*.p7b/*.pfx格式文件。

OpenSSL证书操作 火雨(Nick) 1844
OpenSSL证书操作

openssl程序设计详解 yxyhack's blog 2323
作者:Eric Rescorla on Sat, 2001-09-01 01:0如果你急切的想构建一个简单的Web客户端和服务器对,这时你就需要使用SSL了..SSL是一种保护基于TCP协…

GmSSL编程实现gmtls协议C/S通信(BIO版本) xiejianjun417的专栏 2703
GmSSL实现gmtls协议时，服务端必须设置双证书(签名证书和加密证书)才能正常通信。如果服务端只使用单证书(加密证书)，会出现如下错误：SSL routi…

使用openssl API编写client和server 当今明月的专栏 796
使用openssl api编写的client程序和server程序，其中第一个client采用了BIO的方式，第二个client采用了ssl接口，第一个server程序基本没有使用BIO方式…