

原创

rtoax


2019-08-10 20:46:44

1246

收藏

版权

分类专栏：[【计算机网络】](#)



【计算机网络】 专栏收录该内容

8 订阅

170 篇文章

订阅专栏

Table of Contents

- HOWTO: Using Openssl C library
- [Installing Openssl library](#)
  - [Compiling your C program with the Openssl library](#)
  - [A few pointers on the do\\_crypt function](#)

Libssl API

- [How to get libssl API to compile with it\[edit\]](#)

例：Diffie-Hellman parameters

- [Contents](#)
- [Diffie-Hellman\[edit\]](#)
- [Validating Parameters\[edit\]](#)
- [Elliptic curve Diffie-Hellman\[edit\]](#)
- [RFC 3526 PEM Encoded Groups\[edit\]](#)

例：Hostname validation

- [Example Usage\[edit\]](#)
- [SSL Conservatory and cURL code\[edit\]](#)

相关博客文章

## HOWTO: Using Openssl C library

<http://theshybulb.com/2015/10/10/use-openssl-c-library.html>

Oct 10, 2015

For one of the [Matasano crypto challenges](#), I had to decrypt the text which was encrypted using AES in ECB mode. Everything about AES is actually documented by the [National Institute of Standards and Technology](#). You can get all the algorithms behind AES encryption. It is probably not a good idea to implement it from scratch. Openssl has a well tested and widely used library which works.

This [Openssl library page](#) gives a complete example of how to use them. There are a few preparatory steps before you can use the instructions though. These instructions are for Ubuntu like Linux distributions. These worked well on my Raspberry Pi too.

### Installing Openssl library

Following command installs all the C libraries needed to use Openssl with your C code.

目录

- HOWTO: Using Openssl C library
- [Installing Openssl library](#)
  - [Compiling your C program with the ...](#)
  - [A few pointers on the do\\_crypt func...](#)
- Libssl API
- [How to get libssl API to compile with...](#)
- 例：Diffie-Hellman parameters
- [Contents](#)
  - [Diffie-Hellman\[edit\]](#)
  - [Validating Parameters\[edit\]](#)
  - [Elliptic curve Diffie-Hellman\[edit\]](#)

广告 关闭

PARATERA 并行®

5000核时CPU 或 500元卡时GPU

计算资源任性送!

免费申领

分类专栏		
	笔记	32篇
	【DPDK生态】	120篇
	【DPDK】	68篇
	【FD.io VPP】	39篇

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

For example, you will want to include the following header files:

```
1 #include <openssl/evp.h>
2 #include <openssl/ssl.h>
3 #include <openssl/rsa.h>
4 #include <openssl/x509.h>
```

## Compiling your C program with the Openssl library

Next, you can follow the instructions from the [Openssl crypto library page](#) to create your C program. I have an example program in my [Cryptopals](#) Github repository. While linking the program you need to provide the ssl and crypto library names. Following command should do it:

```
gcc yourprogram.c -lssl -lcrypto
```

## A few pointers on the `do_crypt` function

- If you are going to use the `do_crypt` function for decrypting a text encrypted using electronic code book (ECB) mode, you should remove the following assert line since there is no Initialization Vector for ECB.

```
OPENSSL_assert(EVP_CIPHER_CTX_iv_length(&ctx) == 16);
```

- The example code operates on the raw data. So, if you are trying to decrypt the data which is base64 encoded, your first step should be to convert it into raw data.

## Libssl API

[https://wiki.openssl.org/index.php/Libssl\\_API](https://wiki.openssl.org/index.php/Libssl_API)

libssl is the portion of OpenSSL which supports TLS ( [SSL and TLS Protocols](#) ), and depends on [libcrypto](#).

This is a **C** api. To use it you need to include (at least) **openssl/ssl.h** and to link your program with libssl library.

- [Diffie-Hellman parameters](#)
- [Hostname validation](#)

## How to get libssl API to compile with it[\[edit\]](#)

on Debian base ( debian, ubuntu, ... ) you would need libssl-dev : apt-get install libssl-dev.

on Redhat base ( RedHat, Fedora, ... ) you would need openssl-devel : yum install openssl-devel

You can get sources directly too to compile statically over it.

## 例 : Diffie-Hellman parameters

To use [perfect forward secrecy](#) cipher suites, you must set up [Diffie-Hellman](#) parameters ([on the server side](#)), or the PFS cipher suites will be silently ignored.

	【F-Stack】	15篇
	【Hyperscan.io】	2篇
	【OVS】	4篇
	【SPDK】	2篇
	【分布式】	16篇
	【服务器与硬件】	29篇
	【高性能计算】	49篇
	【CUDA】	16篇
	【MPICH】	7篇
	【OpenMP】	4篇
	【基础知识】	415篇
	【计算机网络】	170篇
	【Nginx】	9篇
	【计算机语言】	343篇
	【GNU/GCC】	36篇
	【社区】	41篇
	【Linux内核】	383篇
	【Linux基础】	92篇
	【eBPF】	94篇
	【进程管理】	38篇
	【驱动/模块】	28篇
	【文件系统】	25篇
	【内存管理】	110篇
	【容器虚拟化】	11篇
	【网络协议栈】	45篇
中断	【中断子系统】	26篇
Shell	【Shell脚本】	59篇
UI UX	【人机交互】	134篇

# Contents











[hide]

- 1Diffie-Hellman
- 2Validating Parameters
- 3Elliptic curve Diffie-Hellman
- 4RFC 3526 PEM Encoded Groups

## Diffie-Hellman[edit]

SSL\_CTX\_set\_tmp\_dh is used to set the Diffie-Hellman parameters for a context. One of the easiest ways to get Diffie-Hellman parameters to use with this function is to generate random Diffie-Hellman parameters with the [dhparam](#) command-line program with the -c option, and embed the resulting code fragment in your program. For example, openssl dhparam -C 2236 might result in:

```
1  #ifndef HEADER_DH_H
2  #include <openssl/dh.h>
3  #endif
4  DH *get_dh2236()
5  {
6      static unsigned char dh2236_p[]={
7          0x0F,0x52,0xE5,0x24,0xF5,0xFA,0x9D,0xDC,0xC6,0xAB,0xE6,0x04,
8          0xE4,0x20,0x89,0x8A,0xB4,0xBF,0x27,0xB5,0x4A,0x95,0x57,0xA1,
9          0x06,0xE7,0x30,0x73,0x83,0x5E,0xC9,0x23,0x11,0xED,0x42,0x45,
10         0xAC,0x49,0xD3,0xE3,0xF3,0x34,0x73,0xC5,0x7D,0x00,0x3C,0x86,
11         0x63,0x74,0xE0,0x75,0x97,0x84,0x1D,0x0B,0x11,0xDA,0x04,0xD0,
12         0xFE,0x4F,0xB0,0x37,0xDF,0x57,0x22,0x2E,0x96,0x42,0xE0,0x7C,
13         0xD7,0x5E,0x46,0x29,0xAF,0xB1,0xF4,0x81,0xAF,0xFC,0x9A,0xEF,
14         0xFA,0x89,0x9E,0x0A,0xFB,0x16,0xE3,0x8F,0x01,0xA2,0xC8,0xDD,
15         0xB4,0x47,0x12,0xF8,0x29,0x09,0x13,0x6E,0x9D,0xA8,0xF9,0x5D,
16         0x08,0x00,0x3A,0x8C,0xA7,0xFF,0x6C,0xCF,0xE3,0x7C,0x3B,0x6B,
17         0xB4,0x26,0xCC,0xDA,0x89,0x93,0x01,0x73,0xA8,0x55,0x3E,0x5B,
18         0x77,0x25,0x8F,0x27,0xA3,0xF1,0xBF,0x7A,0x73,0x1F,0x85,0x96,
19         0x0C,0x45,0x14,0xC1,0x06,0xB7,0x1C,0x75,0xAA,0x10,0xBC,0x86,
20         0x98,0x75,0x44,0x70,0xD1,0x0F,0x20,0xF4,0xAC,0x4C,0xB3,0x88,
21         0x16,0x1C,0x7E,0xA3,0x27,0xE4,0xAD,0xE1,0xA1,0x85,0x4F,0x1A,
22         0x22,0x0D,0x05,0x42,0x73,0x69,0x45,0xC9,0x2F,0xF7,0xC2,0x48,
23         0xE3,0xCE,0x9D,0x74,0x58,0x53,0xE7,0xA7,0x82,0x18,0xD9,0x3D,
24         0xAF,0xAB,0x40,0x9F,0xAA,0x4C,0x78,0x0A,0xC3,0x24,0x2D,0xDB,
25         0x12,0xA9,0x54,0xE5,0x47,0x87,0xAC,0x52,0xFE,0xE8,0x3D,0x0B,
26         0x56,0xED,0x9C,0x9F,0xFF,0x39,0xE5,0xE5,0xBF,0x62,0x32,0x42,
27         0x08,0xAE,0x6A,0xED,0x88,0x0E,0xB3,0x1A,0x4C,0xD3,0x08,0xE4,
28         0xC4,0xAA,0x2C,0xCC,0xB1,0x37,0xA5,0xC1,0xA9,0x64,0x7E,0xEB,
29         0xF9,0xD3,0xF5,0x15,0x2B,0xFE,0x2E,0xE2,0x7F,0xFE,0xD9,0xB9,
30         0x38,0x42,0x57,0x03,
31     } ;
32     static unsigned char dh2236_g[]={
33         0x02,
34     } ;
35     DH *dh ;
36
37     if ((dh=DH_new()) == NULL) return(NULL);
38     dh->p =BN_bin2bn(dh2236_p,sizeof(dh2236_p),NULL);
39     dh->g =BN_bin2bn(dh2236_g,sizeof(dh2236_g),NULL);
40     if ((dh->p == NULL) || (dh->g == NULL))
41     { DH_free(dh); return(NULL); }
42     return(dh);
43 }
```

	【算法与数学】	114篇
	【数据库与(No)SQL】	46篇
	【MySQL】	13篇
	【Redis】	21篇
	【通信技术】	95篇
	【虚拟化与云】	54篇
	【Docker】	15篇
	【KVM】	34篇
	【OpenStack】	7篇
	【Qemu】	9篇
	【Xen】	3篇

which can then be used like this:

```
1 DH *dh = get_dh2236 ();
2 if (1 != SSL_CTX_set_tmp_dh (ctx, dh))
3     error ();
4 DH_free (dh);
```

Be sure to choose a bit length [appropriate to the security level you want to achieve](#), although keep in mind that Diffie-Hellman parameters longer than 2236 bits may be [incompatible with older versions of NSS](#). Even worse, it appears that [versions of Java prior to 1.7 don't support Diffie-Hellman parameters longer than 1024 bits!](#)

## Validating Parameters[\[edit\]](#)

The Diffie-Hellman parameters should be validated after loading. To perform paramter validation, you call `DH_check`. `DH_check` returns 0 or a bitmask values of the following:

- `DH_CHECK_P_NOT_PRIME` (0x01)
- `DH_CHECK_P_NOT_SAFE_PRIME` (0x02)
- `DH_UNABLE_TO_CHECK_GENERATOR` (0x04)
- `DH_NOT_SUITABLE_GENERATOR` (0x08)

The validation code might look as follows (error checking omitted for clarity):

```
1 BIO* bio = ...;
2 DH* dh = PEM_read_bio_DHparams(bio, NULL, NULL, NULL);
3
4 int rc, codes = 0;
5 rc = DH_check(dh, &codes);
6 assert(rc == 1);
7
8 if(BN_is_word(dh->g, DH_GENERATOR_2))
9 {
10     long residue = BN_mod_word(dh->p, 24);
11     if(residue == 11 || residue == 23) {
12         codes &= ~DH_NOT_SUITABLE_GENERATOR;
13     }
14 }
15
16 if (codes & DH_UNABLE_TO_CHECK_GENERATOR)
17     printf("DH_check: failed to test generator\n");
18
19 if (codes & DH_NOT_SUITABLE_GENERATOR)
20     printf("DH_check: g is not a suitable generator\n");
21
22 if (codes & DH_CHECK_P_NOT_PRIME)
23     printf("DH_check: p is not prime\n");
24
25 if (codes & DH_CHECK_P_NOT_SAFE_PRIME)
26     printf("DH_check: p is not a safe prime\n");
```

The additional call to `BN_mod_word(dh->p, 24)` (and unmasking of `DH_NOT_SUITABLE_GENERATOR`) is performed to ensure your program accepts IETF group parameters. OpenSSL checks the prime is congruent to 11 when  $g = 2$ ; while the IETF's primes are congruent to 23 when  $g = 2$ . Without the test, the IETF parameters would fail validation. For details, see [Diffie-Hellman Parameter Check \(when  \$g = 2\$ , must  \$p \bmod 24 == 11\$ ?\)](#).

## Elliptic curve Diffie-Hellman[[edit](#)]

For [elliptic curve Diffie-Hellman](#), you can do something like this:

```
1 EC_KEY *ecdh = EC_KEY_new_by_curve_name (NID_X9_62_prime256v1);
2 if (! ecdh)
3     error ();
4 if (1 != SSL_CTX_set_tmp_ecdh (ctx, ecdh))
5     error ();
6 EC_KEY_free (ecdh);
```

Or, in OpenSSL 1.0.2 (not yet released, as of Feb 2013) and higher, you should be able to do:

```
SSL_CTX_set_ecdh_auto (ctx, 1)
```

For more information, see [Elliptic Curve Diffie Hellman](#) and [Elliptic Curve Cryptography](#).

## RFC 3526 PEM Encoded Groups[[edit](#)]

Below are three Diffie-Hellman MODP groups specified in [RFC 3526](#), [More Modular Exponential \(MODP\) Diffie-Hellman groups for Internet Key Exchange \(IKE\)](#) (the 1024-bit parameter is from [RFC 2409](#)). They can be used with `PEM_read_bio_DHparams` and a memory BIO. [RFC 3526](#) also offers 1536-bit, 6144-bit and 8192-bit primes.

```
1 static const char g_dh1024_sz[] =
2     "-----BEGIN DH PARAMETERS-----\n"
3     "MIGHAogBAP//yQ/aoiFowjTExmKLgNwc0SkCTgiKZ8x0Agu+pjsTmyJR\n"
4     "Sgh5jjQE3e+VGbPNOKMbMCsKbfJfFDdP4TVtbVHCReSftXZiXn7G9ExC6aY37WsL\n"
5     "/1y29Aa37e44a/taiZ+lrp8kEXxLH+ZJKGZR7OZTgf//AgEC\n"
6     "-----END DH PARAMETERS-----";
7
8 static const char g_dh1536_sz[] = "-----BEGIN DH PARAMETERS-----\n"
9     "MIHHAoHBAP//yQ/aoiFowjTExmKLgNwc0SkCTgiKZ8x0Agu+pjsTmyJR\n"
10    "Sgh5jjQE3e+VGbPNOKMbMCsKbfJfFDdP4TVtbVHCReSftXZiXn7G9ExC6aY37WsL\n"
11    "/1y29Aa37e44a/taiZ+lrp8kEXxLH+ZJKGZR7ORbPcIAfLihY78FmNpINhxV05pp\n"
12    "Fj+o/STPX4N1XSPco62WHGLzViCFUrue1SkHcJaWbWcMNU5KvJgE8XRScMoJcyf/\n"
13    "/wIBAg==\n"
14    "-----END DH PARAMETERS-----";
15
16 static const char g_dh2048_sz[] =
17     "-----BEGIN DH PARAMETERS-----\n"
18     "MIIBCAKCAQEA//JD9qiIWjCNMTGYouA3BzRKQJOcIpnzHQCC76mOx0b\n"
19     "I1FKChmONATd75UZs806QxswKwpt818UN0/hNW1tUcJF5IW1dmJefsb0TELppjft\n"
20     "awv/XLb0Brft7jhr+1qJn6WunyQRfEsf5kkoZlHs5Fs9wgB8uKFjvwWY2kg2HFXT\n"
21     "mmkWp6j9JM9fg2VdI9yjrZYcYvNWIIVSu57VKQdwlpZtZww1Tkq8mATxdGwIyhgh\n"
22     "fDKQXkYuNs474553LBgOhgObJ40i7Aeij7XFXfBvTFLJ3ivL9pVYFvg51U186pVq\n"
23     "5RXSJhiY+gUQFXKOWoqsqmj//wIBAg==\n"
24     "-----END DH PARAMETERS-----";
25
26 static const char g_dh3072_sz[] =
27     "-----BEGIN DH PARAMETERS-----\n"
28     "MIIBiAKCAYEA//JD9qiIWjCNMTGYouA3BzRKQJOcIpnzHQCC76mOx0b\n"
29     "I1FKChmONATd75UZs806QxswKwpt818UN0/hNW1tUcJF5IW1dmJefsb0TELppjft\n"
30     "awv/XLb0Brft7jhr+1qJn6WunyQRfEsf5kkoZlHs5Fs9wgB8uKFjvwWY2kg2HFXT\n"
31     "mmkWp6j9JM9fg2VdI9yjrZYcYvNWIIVSu57VKQdwlpZtZww1Tkq8mATxdGwIyhgh\n"
32     "fDKQXkYuNs474553LBgOhgObJ40i7Aeij7XFXfBvTFLJ3ivL9pVYFvg51U186pVq\n"
33     "5RXSJhiY+gUQFXKOWoqqxC2tMxcNBFB6M6hVIavfHLpk7PuFBFjb7wqK6nFXXQYM\n"
34     "fbOXD4Wm4eTHq/WuJNsJM9cejJTgSiVhnc7j0iYa0u5r8S/6BtmKCGTdgJzPshq\n"
```

```

35 "ZFIfKxgXeyAMu+EXV3phXwx3CYjAutlG4gjiT6B05asxQ9tb/OD9EI5LgtEgqTrS\n"
36 "yv//AgEC\n"
37 "-----END DH PARAMETERS-----";
38
39 static const char g_dh4096_sz[] =
40 "-----BEGIN DH PARAMETERS-----\n"
41 "MIICCAKCAgEA//JD9qiIWjCNMTGYouA3BzRKQJOCipnzHQCC76mOx0b\n"
42 "I1FKChMONATd75UZs806QxswKwpt8l8UN0/hNW1tUcJF5IW1dmJefsb0TElppjft\n"
43 "awv/XLb0Brrft7jhr+1qJn6WunyQRfEsf5kkoZlHs5Fs9wgB8uKFjvwWY2kg2HFXT\n"
44 "mmkWP6j9JM9fg2VdI9yjrZYcYvNWIIVSu57VKQdwlpZtZww1Tkq8mATxdGwIyhgh\n"
45 "fDKQXkYuNs474553LBgOhg0bJ40i7Aeij7XFXfBvTFLJ3ivL9pVYFvg5lU186pVq\n"
46 "5RXSJhiY+gUQFXKOWoqqxC2tMxcNBFB6M6hVIavfHLPk7PuFBFjb7wqK6nFXXQYM\n"
47 "fbOXD4Wm4eTHq/WujNsJM9cejJTgSiVhnc7j0iYa0u5r8S/6BtmKCGTYdgJzPshq\n"
48 "ZFIfKxgXeyAMu+EXV3phXwx3CYjAutlG4gjiT6B05asxQ9tb/OD9EI5LgtEgqSEI\n"
49 "ARpyPBKnh+bXiHGaeL26WyaZwycYavTiPBqUaDS2FQvaJYPpyirUT0jbu8LbBN60\n"
50 "+S6O/BQfvsqmKHxZR05rwF2ZspZPoJDDoiM7oYZRW+ftH2EpcM7i16+4G912IXBI\n"
51 "HNAGkSFvsFqpk7TqmI2P3cGG/7fcKkAj030Nck0BjGZ//8CAQI=\n"
52 "-----END DH PARAMETERS-----";

```

## 例 : Hostname validation

OpenSSL 1.1.0 provides built-in functionality for hostname checking and validation. Viktor Dukhovni provided the implementation in January, 2015. Its been available in Master since that time. The code is beginning to see widespread testing as the release of OpenSSL 1.1.0 approaches.

One [common mistake](#) made by users of OpenSSL is to assume that OpenSSL will validate the hostname in the server's certificate. Versions prior to 1.0.2 did not perform hostname validation. Version 1.0.2 and up contain support for hostname validation, but they still require the user to call a few functions to set it up.

A man page on hostname validation has been available since 1.0.2. Also see the [X509\\_check\\_host\(\)](#).

## Example Usage[\[edit\]](#)

The following is from [Hostname validation](#) and shows how you could use OpenSSL's built-in hostname validation.

```

1  const char servername[] = "www.example.com";
2  SSL *ssl = NULL;
3  X509_VERIFY_PARAM *param = NULL;
4  ...
5
6  servername = "www.example.com";
7  ssl = SSL_new(...);
8  param = SSL_get0_param(ssl);
9
10 /* Enable automatic hostname checks */
11 X509_VERIFY_PARAM_set_hostflags(param, X509_CHECK_FLAG_NO_PARTIAL_WILDCARDS);
12 if (!X509_VERIFY_PARAM_set1_host(param, servername, sizeof(servername) - 1)) {
13     // handle error
14     return 0;
15 }
16
17 /* Enable peer verification, (with a non-null callback if desired) */
18 SSL_set_verify(ssl, SSL_VERIFY_PEER, NULL);
19
20 /*
21  * Establish SSL connection, hostname should be checked
22  * automatically test with a hostname that should not match,
23  * the connection will fail (unless you specify a callback

```

```

24  * that returns despite the verification failure. In that
25  * case SSL_get_verify_status() can expose the problem after
26  * connection completion.
27  */
28  ...

```

The above works starting with OpenSSL 1.0.2. A simpler interface is available starting with OpenSSL 1.1.0:

```

1  ...
2  SSL_set_hostflags(ssl, X509_CHECK_FLAG_NO_PARTIAL_WILDCARDS);
3  if (!SSL_set1_host(ssl, "www.example.com")) {
4      /* handle error */
5  }
6  /* Enable peer verification (with a non-null callback if desired) */
7  SSL_set_verify(ssl, SSL_VERIFY_PEER, NULL);
8  ...

```

documentation at [SSL\\_set1\\_host](#)

Wildcard support is configured via the flags documented for `X509_check_host()`, the two most frequently useful are:

- **X509\_CHECK\_FLAG\_NO\_WILDCARDS**
- **X509\_CHECK\_FLAG\_NO\_PARTIAL\_WILDCARDS**

populate the `X509_VERIFY_PARAMS` with the desired hostname, and let the OpenSSL code call `X509_check_host` automatically.

This makes it easier to some day enable DANE TLSA support, because with DANE, name checks need to be skipped for DANE-EE(3) TLSA records, as the DNSSEC TLSA records provides the requisite name binding instead.

Also with the `X509_VERIFY_PARAM` approach, name checks happen early, and for applications that don't continue handshakes with unauthenticated peers, terminate as early as possible.

There is an associated new X509 error code: **X509\_V\_ERR\_HOSTNAME\_MISMATCH**

## SSL Conservatory and cURL code[\[edit\]](#)

This was the original information, might still be valid for < 1.0.2 openssl versions :

The [ssl-conservatory repository](#) shows how validating the hostname can be done. However, the ssl-conservatory code does not handle wildcard certificates, so [borrowing some code from cURL](#) might be one way to go instead. [This commit](#) shows how to graft the wildcard-matching code from cURL into the ssl-conservatory code.

Below is a copy of [openssl\\_hostname\\_validation.c](#), although to compile it also needs the files [hostcheck.h](#), [hostcheck.c](#), and [openssl\\_hostname\\_validation.h](#).

```

1  /* Obtained from: https://github.com/iSECPartners/ssl-conservatory */
2
3  /*
4  Copyright (C) 2012, iSEC Partners.
5
6  Permission is hereby granted, free of charge, to any person obtaining a copy of
7  this software and associated documentation files (the "Software"), to deal in
8  the Software without restriction, including without limitation the rights to
9  use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
10 of the Software, and to permit persons to whom the Software is furnished to do
11 so, subject to the following conditions:
12

```

```
13 The above copyright notice and this permission notice shall be included in all
14 copies or substantial portions of the Software.
15
16 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
17 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
18 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
19 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
20 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
21 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
22 SOFTWARE.
23 */
24
25 /*
26  * Helper functions to perform basic hostname validation using OpenSSL.
27  *
28  * Please read "everything-you-wanted-to-know-about-openssl.pdf" before
29  * attempting to use this code. This whitepaper describes how the code works,
30  * how it should be used, and what its limitations are.
31  *
32  * Author:  Alban Diquet
33  * License: See LICENSE
34  *
35  */
36
37 // Get rid of OSX 10.7 and greater deprecation warnings.
38 #if defined(__APPLE__) && defined(__clang__)
39 #pragma clang diagnostic ignored "-Wdeprecated-declarations"
40 #endif
41
42 #include <openssl/x509v3.h>
43 #include <openssl/ssl.h>
44
45 #include "openssl_hostname_validation.h"
46 #include "hostcheck.h"
47
48 #define HOSTNAME_MAX_SIZE 255
49
50 /**
51  * Tries to find a match for hostname in the certificate's Common Name field.
52  *
53  * Returns MatchFound if a match was found.
54  * Returns MatchNotFound if no matches were found.
55  * Returns MalformedCertificate if the Common Name had a NUL character embedded in it.
56  * Returns Error if the Common Name could not be extracted.
57  */
58 static HostnameValidationResult matches_common_name(const char *hostname, const X509 *server_cert) {
59     int common_name_loc = -1;
60     X509_NAME_ENTRY *common_name_entry = NULL;
61     ASN1_STRING *common_name_asn1 = NULL;
62     char *common_name_str = NULL;
63
64     // Find the position of the CN field in the Subject field of the certificate
65     common_name_loc = X509_NAME_get_index_by_NID(X509_get_subject_name((X509 *) server_cert), NID_commonName);
66     if (common_name_loc < 0) {
67         return Error;
68     }
69
70     // Extract the CN field
71     common_name_entry = X509_NAME_get_entry(X509_get_subject_name((X509 *) server_cert), common_name_loc);
72     if (common_name_entry == NULL) {
73         return Error;
```



```

14     }
15 }
16
17 // Convert the CN field to a C string
18 common_name_asn1 = X509_NAME_ENTRY_get_data(common_name_entry);
19 if (common_name_asn1 == NULL) {
20     return Error;
21 }
22 common_name_str = (char *) ASN1_STRING_data(common_name_asn1);
23
24 // Make sure there isn't an embedded NUL character in the CN
25 if ((size_t)ASN1_STRING_length(common_name_asn1) != strlen(common_name_str)) {
26     return MalformedCertificate;
27 }
28
29 // Compare expected hostname with the CN
30 if (Curl_cert_hostcheck(common_name_str, hostname) == CURL_HOST_MATCH) {
31     return MatchFound;
32 }
33 else {
34     return MatchNotFound;
35 }
36 }
37
38 /**
39  * Tries to find a match for hostname in the certificate's Subject Alternative Name extension.
40  *
41  * Returns MatchFound if a match was found.
42  * Returns MatchNotFound if no matches were found.
43  * Returns MalformedCertificate if any of the hostnames had a NUL character embedded in it.
44  * Returns NoSANPresent if the SAN extension was not present in the certificate.
45  */
46 static HostnameValidationResult matches_subject_alternative_name(const char *hostname, const X509 *server_cert) {
47     HostnameValidationResult result = MatchNotFound;
48     int i;
49     int san_names_nb = -1;
50     STACK_OF(GENERAL_NAME) *san_names = NULL;
51
52     // Try to extract the names within the SAN extension from the certificate
53     san_names = X509_get_ext_d2i((X509 *) server_cert, NID_subject_alt_name, NULL, NULL);
54     if (san_names == NULL) {
55         return NoSANPresent;
56     }
57     san_names_nb = sk_GENERAL_NAME_num(san_names);
58
59     // Check each name within the extension
60     for (i=0; i<san_names_nb; i++) {
61         const GENERAL_NAME *current_name = sk_GENERAL_NAME_value(san_names, i);
62
63         if (current_name->type == GEN_DNS) {
64             // Current name is a DNS name, Let's check it
65             char *dns_name = (char *) ASN1_STRING_data(current_name->d.dnsName);
66
67             // Make sure there isn't an embedded NUL character in the DNS name
68             if ((size_t)ASN1_STRING_length(current_name->d.dnsName) != strlen(dns_name)) {
69                 result = MalformedCertificate;
70                 break;
71             }
72             else { // Compare expected hostname with the DNS name
73                 if (Curl_cert_hostcheck(dns_name, hostname)
74                     == CURL_HOST_MATCH) {
75

```

```
135         result = MatchFound;
136         break;
137     }
138 }
139 }
140 }
141 sk_GENERAL_NAME_pop_free(san_names, GENERAL_NAME_free);
142
143 return result;
144 }
145
146
147 /**
148  * Validates the server's identity by looking for the expected hostname in the
149  * server's certificate. As described in RFC 6125, it first tries to find a match
150  * in the Subject Alternative Name extension. If the extension is not present in
151  * the certificate, it checks the Common Name instead.
152  *
153  * Returns MatchFound if a match was found.
154  * Returns MatchNotFound if no matches were found.
155  * Returns MalformedCertificate if any of the hostnames had a NUL character embedded in it.
156  * Returns Error if there was an error.
157  */
158 HostnameValidationResult validate_hostname(const char *hostname, const X509 *server_cert) {
159     HostnameValidationResult result;
160
161     if((hostname == NULL) || (server_cert == NULL))
162         return Error;
163
164     // First try the Subject Alternative Names extension
165     result = matches_subject_alternative_name(hostname, server_cert);
166     if (result == NoSANPresent) {
167         // Extension was not found: try the Common Name
168         result = matches_common_name(hostname, server_cert);
169     }
170
171     return result;
172 }
```

## 相关博客文章

《OpenSSL API》<https://blog.csdn.net/sunweixiang1002/article/details/83029232>

《openssl api 进行c语言编程的问题》<https://bbs.csdn.net/topics/392171145?page=1>

《使用 OpenSSL API 进行安全编程》<https://www.ibm.com/developerworks/cn/linux/l-openssl.html>

《OpenSSL使用之API指南》[http://www.360doc.com/content/16/1210/00/17203858\\_613428107.shtml](http://www.360doc.com/content/16/1210/00/17203858_613428107.shtml)

openssl c调用例子

05-06

openssl c调用例子 openssl c调用例子 openssl c调用例子

OpenSSL的实用思考

孤独的糖三角 1751

最近由于在做苹果的一个Push服务器，需要用到SSL的库，决定选择网上比较热炒的Openssl，在此之前，也听说了这个库比较难处理，因为资料并不是...

抢沙发



rtoax

码龄4年

暂无认证

1163 2774 335 175万+  
原创 周排名 总排名 访问 等级

2万+ 1544 1379 2153 4540  
积分 粉丝 获赞 评论 收藏



私信

关注

搜博主文章

热门文章

轻松解决远程链接的“Gtk-WARNING \*\*: cannot open display;”或“Cannot connect to display:”问题

61741

5G基站君的进化之路 — CU和DU分离

34438

windows下安装gcc编译器（c/c++/fortran）

30002

PostgreSQL的登录、创建用户、数据库并赋权

24066

gtk学习总结：GTK从入门到放弃，三天包教包会

23465

最新评论

Linux Jump Label/static-key机制详解  
rtoax: 内核模块测试例可参见：https://gitee.com/rtoax/test-linux/tree/main/jump-label

libcare Hello World测试例  
rtoax: 嗯呐，也在用

libcare Hello World测试例  
MDSCQYHM: 当前licareplus已经支持x86\_64和aarch64了哟，可以去用用看

Linux环境中的网络分段卸载技术 GSO/T...  
wanwanxia1997: 楼主，GRO的图放错了吧放成GSO了

Linux I/O原理和零拷贝Zero-copy技术全...  
Nintendo\_Nerd: 中断控制i/o实际应用是啥

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

ISOLINUX: A bootloader for Linux using ISO 9660/EI Torito CD-ROMs

The Syslinux Project: A suite of bootloaders for Linux

isolinux.cfg

2021

11月

8篇

10月

26篇

09月

53篇

08月

28篇

请发表有价值的评论， 博客评论不欢迎灌水，良好的社区氛围需大家一起维护。

评论

How to compile openssl library\_vptr\_grid的专栏

10-30

OpenSSL C API [HOWTO: Using Openssl C library] Table of Contents HOWTO: Using Openssl C library Installing Openssl library Compiling your C pro...

使用OpenSSL API 进行安全编程\_章志强的专栏

11-9

如果打算使用目录存储可信信任库,那么必须要以特定的方式命名文件。 OpenSSL 文档清楚 地说明了应该何去何从,不过,OpenSSL 附带了一个名为c\_rehash...

openssl教程（c语言）

openssl开发的c语言教程，是我找到的最好的教程，包括阻塞io和非阻塞io的开发指导。文档是英文版的，附示例源码。

12-20

openssl库C语言完整版

openssl库C语言完整版，包含dll和lib

05-08

OpenSSL 有关密钥的那些事儿(HOWTO keys)\_weixin\_33895...

12-21

&lt;DRAFT!&gt; OpenSSL 有关密钥的那些事儿(HOWTO keys)1. 介绍(Introduction)Keys are the basis of public key algorithms and PKI. Keys usuallyco...

OpenSSL Configure选项说明\_shb8845369的专栏

9-25

For example “--api=1.1.0” willremove support for all APIs that were deprecated in OpenSSLversion 1.1.0 or below. This is a rather specialized optionfor...

OpenSSL编程的基本步骤

u011285208的博客

229

OpenSSL编程的基本步骤 ===== 启用加密 ===== 客户端必备过程： ...

C语言使用openssl库进行加密

热门推荐

明潮的BLOG

1万+

使用MD5加密 我们以一个字符串为例，新建一个文件filename.txt，在文件内写入hello ，然后在Linux下可以使用命令md5sum filename.txt计算md5值 ...

Linux C 下使用openssl 进行SHA1加密\_weixin\_34026276...

9-26

Code snippet to calculate SHA1sum using openssl libs. Copyright 2005 Junichi Uekawa, given to public domain. \$ gcc openssltest.c -lssl \$ ./a.out < ./a....

openssl使用\_weixin\_34220623的博客

10-9

在当今互联网的时代,密码学是提供安全的最主要工具之一。密码学的主要目的是通过数据机密性、数据完整性、认证、不可抵赖性来挫败大部分的网络\*\*\*,...

openssl命令行和API

且行且珍惜

306

当安装openssl轻量版时（ OpenSSL v1.1.1gLight），可以使用openssl命令进行算法的验证。若需要使用API进行算法验证，则需要安装完整版，这两个...

OpenSSL API

刚刚起步的菜鸟

3070

/\* \* @file cert\_openssl.c \* @brief 利用openssl api处理证书 \* @author zy \* @date 2014-10-11 modify \*/ #include &lt;stdio.h&gt; #include &lt;unistd.h...

使用c语言实现在linux下的openssl客户端和服务端编程

weixin\_30687587的博客

759

使用c语言实现在linux下的openssl客户端和服务端编程 摘自：https://www.cnblogs.com/etangyushan/p/3679457.html 前几天组长让我实现一个使...

openssl API 函数库

11-18

详细介绍了openssl的API，帮助用户利用openssl库函数快速开发网络安全程序

使用 OpenSSL API 进行安全编程

linux\_chen的专栏

736

<br />http://www.ibm.com/developerworks/cn/linux/l-openssl.html<br /><br />OpenSSL API 的文档有些含糊不清。因为还没有多少关于 OpenSSL 使用的...

使用 OpenSSL API 建立安全连接 - 双向认证

bberdong的专栏

2039

使用 OpenSSL API 进行安全编程 一、概念： 1.什么是 SSL？ SSL 是一个缩写，全称是 Secure Sockets Layer。 它是支持在 Internet 上进行安全通信...

openssl-1.0.1c.tar

openssl-1.0.1c.tar

09-12

OpenSSL 有关密钥的那些事儿（ HOWTO keys ）

太阳火神的美丽人生

1275

OpenSSL 有关密钥的那些事儿（ HOWTO keys ）

Openssl C++ API

最新发布

一只菜鸟的练级路

101

Openssl C++ API Base64 将8位二进制信息编码为ASCII码。 openssl命令 base64 or-enc base64 EVP API EVP\_EncodeBlock((unsigned char \*)encode...

OpenSSL开发——利用SSL写的简单的C/S程序

oj847935591的博客

6233

先使用 openssl 生成服务器证书、私钥 openssl req -newkey rsa:2048 -nodes -keyout rsa\_keyServer.pem -x509 -days 365 -out certServer.cer -subj "/...

linux c openssl aes 加解密

whatday的专栏

6919

1.OpenSSL提供了AES加解密算法的API const char \*AES\_options(void); AES算法状态，是所有支持或者是部分支持。 返回值：“aes(full)”或者“aes(parti...

07月 49篇	06月 80篇	05月 70篇	04月 51篇
03月 103篇	02月 36篇	01月 19篇	
2020年 709篇	2019年 507篇	2018年 222篇	2017年 94篇

pem格式证书编码 x509\_SSL证书编码格式

Certificates and EncodingsAt its core an X.509 certificate is a digital document that has been encoded and/or digitally signed according to RFC 5280.I...

weixin\_39999859的博客 276

©2021 CSDN 皮肤主题: 代码科技 设计师:Amelia\_0503 返回首页

关于我们 招贤纳士 广告服务 开发助手 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00



rtoax

关注

👍 0

💬 0

🌟 0



🔗

专栏目录

0 0

举报

⬆