<http://netsecurity.51cto.com/art/201505/476337.htm>

## 让SSL/TLS协议流行起来：深度解读SSL/TLS实现

SSL/TLS协议是网络安全通信的重要基石，本系列将简单介绍SSL/TLS协议，主要关注SSL/TLS协议的安全性，特别是SSL规范的正确实现。 本系列的文章大体分为3个部分...

作者：GoSSIP\_SJTU来源：GoSSIP\_SJTU|2015-05-13 09:45

[收藏](javascript:favorBox('open');)

[分享](javascript:;)

**一 前言**

SSL/TLS协议是网络安全通信的重要基石，本系列将简单介绍SSL/TLS协议，主要关注SSL/TLS协议的安全性，特别是SSL规范的正确实现。 本系列的文章大体分为3个部分：

SSL/TLS协议的基本流程

典型的针对SSL/TLS协议的攻击

SSL/TLS协议的安全加固措施

本文对SSL/TLS协议概况做基本介绍，包括SSL/TLS协议的版本变迁，协议的详细流程以及流行的SSL/TLS协议实现。文章的主要内容翻译自波鸿鲁尔大学Christopher Meyer的文章《20 Years of SSL/TLS Research An Analysis of the Internet’s Security Foundation》，同时也根据作者自己的理解增加了部分内容，以使对SSL/TLS协议的介绍更为完整。

**二 什么是SSL/TLS?**

SSL全称是Secure Sockets Layer，安全套接字层，它是由网景公司(Netscape)设计的主要用于Web的安全传输协议，目的是为网络通信提供机密性、认证性及数据完整性保障。如今，SSL已经成为互联网保密通信的工业标准。

SSL最初的几个版本(SSL 1.0、SSL2.0、SSL 3.0)由网景公司设计和维护，从3.1版本开始，SSL协议由因特网工程任务小组(IETF)正式接管，并更名为TLS(Transport Layer Security)，发展至今已有TLS 1.0、TLS1.1、TLS1.2这几个版本。

如TLS名字所说，SSL/TLS协议仅保障传输层安全。同时，由于协议自身特性(数字证书机制)，SSL/TLS不能被用于保护多跳(multi-hop)端到端通信，而只能保护点到点通信。

SSL/TLS协议能够提供的安全目标主要包括如下几个：

认证性——借助数字证书认证服务器端和客户端身份，防止身份伪造

机密性——借助加密防止第三方窃听

完整性——借助消息认证码(MAC)保障数据完整性，防止消息篡改

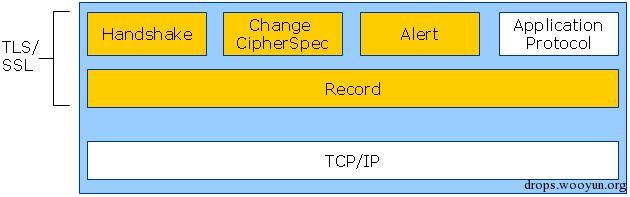
重放保护——通过使用隐式序列号防止重放攻击

为了实现这些安全目标，SSL/TLS协议被设计为一个两阶段协议，分为握手阶段和应用阶段：

握手阶段也称协商阶段，在这一阶段，客户端和服务器端会认证对方身份(依赖于PKI体系，利用数字证书进行身份认证)，并协商通信中使用的安全参数、密码套件以及MasterSecret。后续通信使用的所有密钥都是通过MasterSecret生成。

在握手阶段完成后，进入应用阶段。在应用阶段通信双方使用握手阶段协商好的密钥进行安全通信。

SSL/TLS协议有一个高度模块化的架构，分为很多子协议，如下图所示：



Handshake协议：包括协商安全参数和密码套件、服务器身份认证(客户端身份认证可选)、密钥交换;

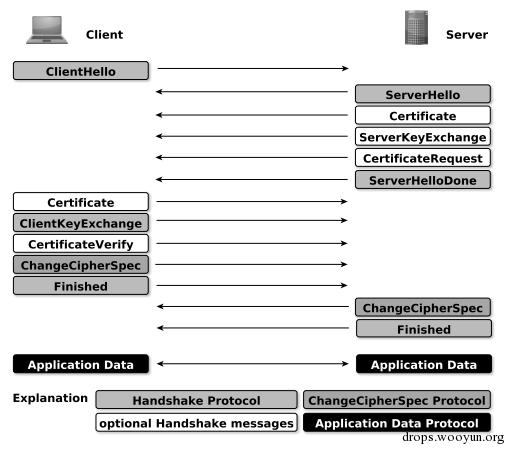
ChangeCipherSpec 协议：一条消息表明握手协议已经完成;

Alert 协议：对握手协议中一些异常的错误提醒，分为fatal和warning两个级别，fatal类型的错误会直接中断SSL链接，而warning级别的错误SSL链接仍可继续，只是会给出错误警告;

Record 协议：包括对消息的分段、压缩、消息认证和完整性保护、加密等。

**三 协议流程详解**

本节对SSL/TLS协议的流程进行详细介绍。一个典型的TLS 1.0协议交互流程如下图所示：



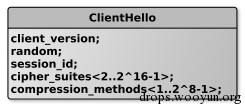
每一个SSL/TLS链接都是从握手开始的，握手过程包含一个消息序列，用以协商安全参数、密码套件，进行身份认证以及密钥交换。握手过程中的消息必须严格按照预先定义的顺序发生，否则就会带来潜在的安全威胁。今年顶级安全会议CCS 有文章提出了建立综合状态机来检查SSL链接中消息序列……

**3.1 握手过程中的消息序列**

ClientHello：ClientHello通常是握手过程中的第一条消息，用于告知服务器客户端所支持的密码套件种类、最高SSL/TLS协议版本以及压缩算法。

ClientHello中还包含一个随机数，这个随机数由4个字节的当前GMT UNIX时间以及28个随机选择的字节组成，共32字节。该随机数会在密钥生成过程中被使用。

另外，ClientHello中还可能包含客户端支持的TLS扩展。(TLS扩展可以被用来丰富TLS协议的功能或者增强协议的安全性)

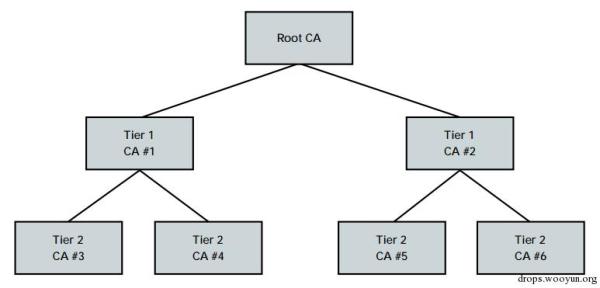


ServerHello：服务器接受到ClientHello后，会返回ServerHello。服务器从客户端在ClientHello中提供的密码套件、SSL/TLS版本、压缩算法列表里选择它所支持的项，并把它的选择包含在ServerHello中告知客户端。接下来SSL协议的建立就基于服务器选择的密码套件类型、SSL/TLS协议版本以及压缩算法。

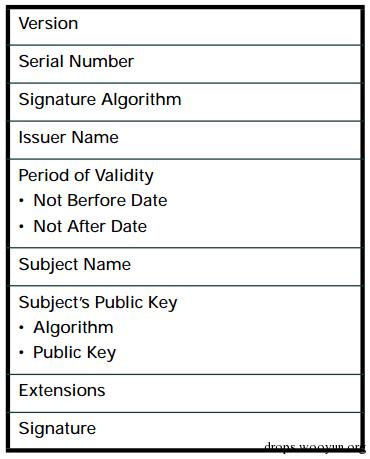
ServerHello中同样会包含一个随机数，同样4+28 字节类型，由服务器生成。



Certificate：客户端和服务器都可以发送证书消息来证明自己的身份，但是通常客户端证书不被使用。 服务器一般在ServerHello后会接一条Certificate消息，Certificate消息中会包含一条证书链，从服务器证书开始，到Certificate authority(CA)或者最新的自签名证书结束。下图形象地描述了证书链：

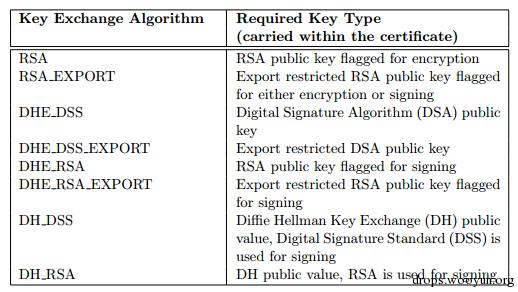


SSL中使用的证书通常是X.509类型证书，X.509证书的内容如下表所示：



在用的X.509证书包含Version 1和Version 3两种版本，其中v1版本的证书存在安全隐患，同时不支持TLS扩展，被逐渐弃用。现在大多数在用的SSL证书都是V3版本。

同时证书会附带与协商好的密钥交换算法对应的密钥。密钥交换算法以及它们所要求的密钥类型如下表所示。



ServerKeyExchange：该消息仅当以下密钥交换算法被使用时由服务器发出：

RSA\_EXPORT(仅当服务器的公钥大于512bit时)、DHE\_DSS、DHE\_DSS\_EXPORT、DHE\_RSA、DHE\_RSA\_EXPORT、DH\_anon 使用其它密钥交换算法时，服务器不能发送此消息。

ServerkeyExchange消息会携带这些密钥交换算法所需要的额外参数，以在后续步骤中协商PreMasterSecret。这些参数需要被签过名。



CertificateRequest：这个消息通常在要求认证客户端身份时才会有。消息中包含了证书类型以及可接受的CA列表。

ServerHelloDone：服务器发送这条消息表明服务器部分的密钥交换信息已经发送完了，等待客户端的消息以继续接下来的步骤。这条消息只用作提醒，不包含数据域。

ClientKeyExchange：这条消息包含的数据与所选用的密钥交换算法有关。

如果选择的密钥交换算法是RSA，那么消息包含的参数为用服务器RSA公钥(包含在之前证书中的或者是ServerKeyExchange中的)加密过的PreMasterSecret，它有48个字节，前2个字节表示客户端支持的最高协议版本，后46个字节是随机选择的。

如果选择的密钥交换算法是DH或者DHE，则可能有两种情况：

隐式DH公开值：包含在Certificate消息里;

显示DH公开值：公开值是本消息的一部分。

CertificateVerify：这条消息用来证明客户端拥有之前提交的客户端证书的私钥。

Finished：表明握手阶段结束。这是第一条用协商的算法和密钥保护的消息。

因为是用协商好的密钥加密的消息，它可以用来确认已经协商好的密钥。

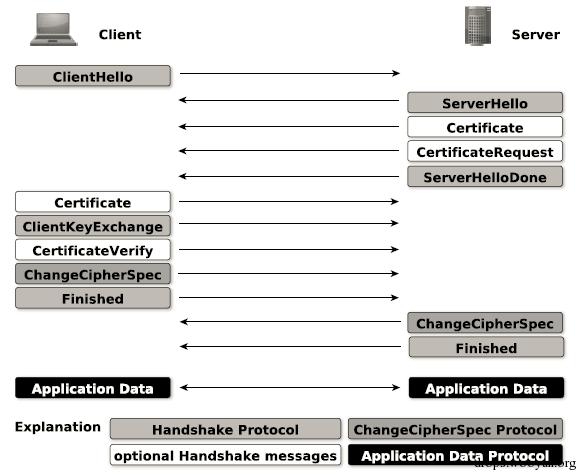
同时Finished消息包含一个verify\_data域，可以用来校验之前发送和接收的信息。

Verify\_data域是一个PRF函数的输出(pseudo-random function)。这个伪随机函数的输入为：(1)两个hash值：一个SHA-1，一个MD5，对之前握手过程中交换的所有消息做哈希;(2)the MasterSecret，由预备主密钥生成;(3)finished\_label，如果客户端发送的则是”client finished”，服务器发送的则是”server finished”。关于这个PRF的细节在3.3节中会具体描述。 此外，Finished 消息不能够在ChangeCipherSpec前发送。

**3.2 不同密钥交换算法对应的握手过程**

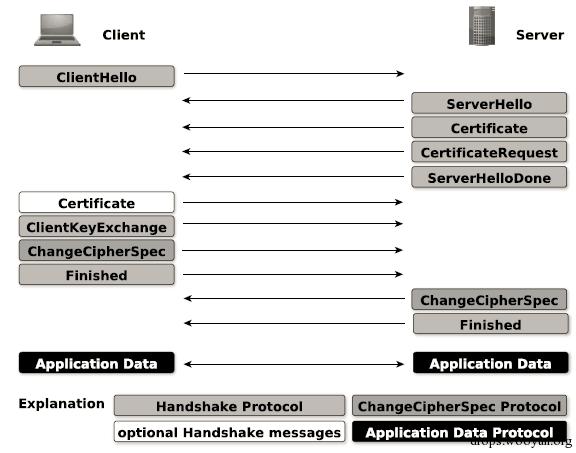
不同的密钥交换算法对应的握手过程中的消息序列是不同的，相应的实现方式也不同，本节介绍几个常见密钥交换算法对应的握手过程。

TLS-RSA：在这个场景下，PreMasterSecret是由客户端指定的，并用RSA公钥加密发送给服务器。服务器不影响PReMasterSecret的生成。

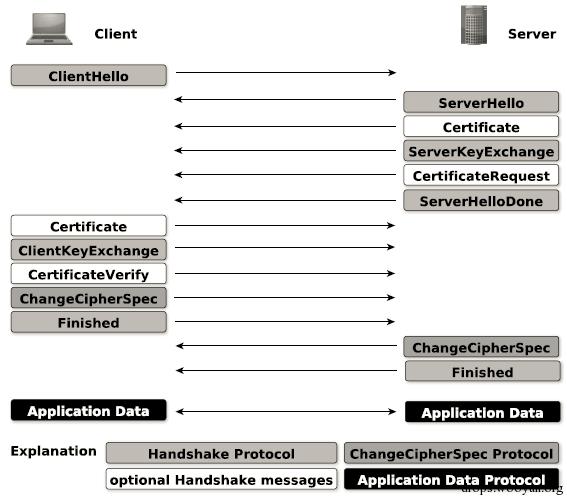


TLS-DH：基于DH的密钥交换也被称为静态Diffie-Hellman。在这种场景下，可能是双方各自提交一个证书包含DH公开值，或者服务器端提交证书包含DH公开值，客户端在每次会话中选择一个值。协商好的DH值被用作PreMasterSecret。显然证书中的参数是固定的，那么每次链接的PreMasterSecret也是相同的。

TLS-DH不能提供前向安全性。

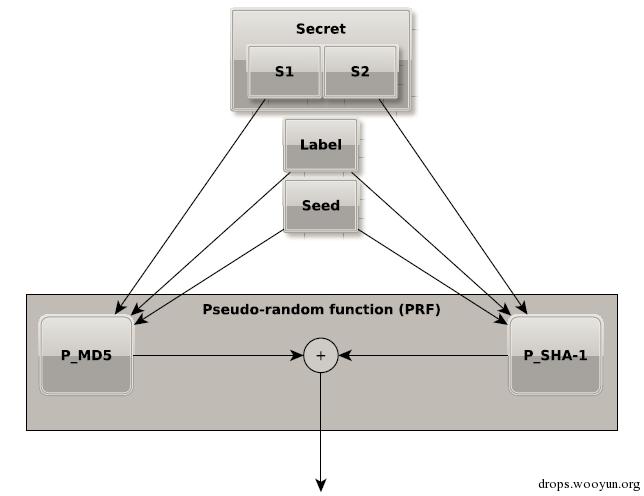


TLS-DHE：基于DHE的TLS握手中会有ServerKeyExchange消息。握手过程中交换参数的认证通过数字签名来实现，支持的签名算法包括RSA和DSS。DH参数会有它的数字签名一起被包含在ServerKeyExchange中被发送出去。客户端在ClientKeyExchange中返回它的公开DH参数，但没有签名保护。同样协商出来的DH密钥被用作PreMasterSecret。



**3.3 密钥生成**

Pseudo-random Function(PRF)：伪随机函数是SSL协议中的一个重要组成部分，它被用来秘密扩展以及生成密钥。在3.1节讲解Finished消息时已经简单提及PRF，在这里我们详细讨论PRF的工作原理。SSL/TLS协议中的PRF如下图所示：



这个PRF基于两个hash函数：MD5和SHA-1，它有3个输入，一个Secret(比如PreMasterSecret)，一个标志符(比如”client finished”, “server finished”)，还有一个种子值(比如客户端随机数+服务器端随机数)。

Secret在使用时被分为长度相同的两半：S1和S2，分别作为P\_MD5和P\_SHA-1的输入。

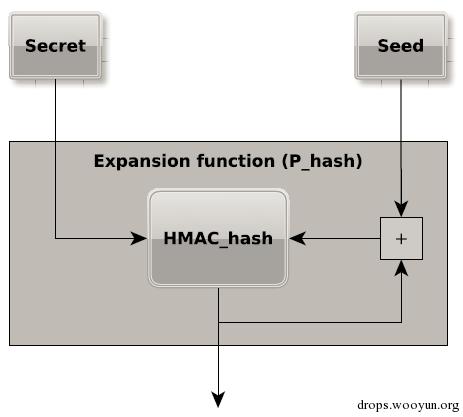
PRF的输出按如下方式处理得到：

PRF( secret , label , seed ) = P\_MD5( S1 , label + seed ) XOR P\_SHA−1(S2 , label + seed ) ;

P\_MD5和P\_SHA-1都是扩展函数，用来扩展秘密值以用于密钥生成，它们的计算方式如下：

P hash ( secret , seed ) = HMAC hash(secret , A( 1 ) + seed ) +HMAC hash(secret , A( 2 ) + seed ) +HMAC hash(secret , A( 3 ) + seed ) + . . .

其中A(0) = seed, A(i) = HMAC hash( secret, A( i −1) )



这个秘密扩展会一直进行直到得到足够多的扩展数据。 Key Derivation：主密钥(MasterSecret)是利用上述PRF从预备主密钥(PreMasterSecret)生成的。每个MasterSecret为48字节，生成方式如下：

mastersecret = PRF( pre mastersecret , ” mastersecret ” , ClientHello.random + ServerHello.random)

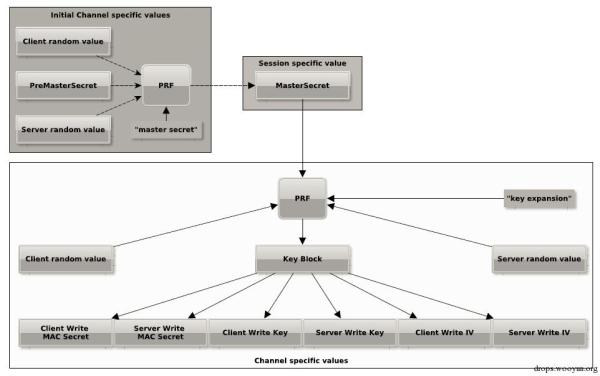
得到MasterSecret后，它会被进一步处理最后生成4个不同的密钥和2个初始向量(IV)。处理过程如下：

keyblock = PRF( SecurityParameters.mastersecret , ”key expansion ” , SecurityParameters.server random +SecurityParameters.client random ) ;

处理过程一直持续到足够多的输出被生成，然后把输出分为4个key和2个IV：

client\_write\_MAC\_secret，server\_write\_MAC\_secret, client\_wriete\_key, server\_write\_key, client\_write\_IV, server\_write\_IV.

下图完整阐述了SSL/TLS协议中的密钥生成过程。



**四 从SSL到TLS**

本节介绍SSL/TLS协议的版本变迁，不同版本的区别以及安全特性等。

SSL 1.0由于从来没有被公开过，并且存在严重安全漏洞，我们就不讨论了。

SSL 2.0：SSL 2.0于1995年4月被发布。SSL 2.0中主要存在的问题如下：

MAC不能覆盖填充长度域，攻击者可能利用这点破坏消息完整性;

缺乏握手认证，攻击者可以篡改密码套件列表，诱骗通信双方使用较弱的密码套件;

使用较弱的或有问题的密码算法(如MD5，RC4等)，或者使用不安全的分组模式(如CBC模式);

对于不同的密码学基元使用相同的密钥，违背基本安全常识。

由于以上安全问题，RFC 6176已经明确提出避免使用SSL 2.0，但是现实生活中还有少量客户端和服务器支持SSL 2.0.

SSL 3.0：SSL 3.0引入了一些新的特性和机制解决了很多之前版本存在的漏洞。此外，SSL 3.0中引入了ChangeCipherSpec子协议。SSL 3.0向后兼容SSL 2.0，相对于SSL 2.0，它的主要改变包括以下几点：

支持更多的密码套件(支持更多的密码算法如DSS，SHA-1)

在握手阶段支持密钥协商(DH和FORTEZZA)

支持密码学参数的重协商

增加了消息压缩选项

MAC能够覆盖填充长度域了，同时MAC可以使用MD5或者SHA-1

不同的密码学基元使用不同的key

Alert子协议能对任何错误给出两种提示：Warning和Fatal

中止链接的时候会用一个close\_notify警告通知通信双方

支持证书链，而非单个证书

通过Finished消息认证所有发送和接收的消息

加密了的PreMasterSecret包含当前使用的协议版本，防止协议回滚

TLS 1.0：TLS 1.0和SSL 3.0差别非常小。实际上，TLS 1.0是SSL 3.1，在IETF接手后改名为TLS。TLS 1.0版本是目前使用最广泛的SSL/TLS协议版本。

TLS 1.0不再支持使用FORTEZZA的密码套件。

TLS 1.0中MAC被替换成HMAC。

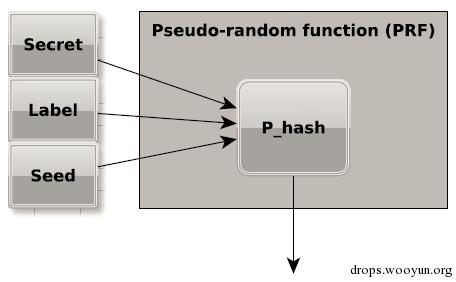
之前提到ChangeCipherSpec消息必须在Finished消息前发送，在TLS 1.0中，如果消息序列不符合这个要求，会产生FATAL警告并终止链接。

TLS 1.1：这个版本相比之前改动也很小。最重要的改动是预防了针对CBC分组模式的一些攻击。现在的填充错误变的和非法MAC错误不可区分了，防止攻击者利用可区分错误响应建立解密预言机对密文进行攻击。

在每次加密过程中，使用CBC分组模式时，都需要显示给出IV，而不用再密钥生成时使用PRF生成IV。

此外，TLS 1.1禁止为适应之前出口限制而使用弱化的密码套件。

TLS 1.2：这是最新的版本，部署的还比较少。这个版本禁用了PRF中的MD5和SHA-1，而用一个可配置的hash函数取代了它们，这样的修改简化了计算过程。修改后的PRF风格如下：



此外，TLS 1.2的一个重要变化是支持认证加密模式(支持GCM等)。但是由于一些AEAD(Authenticated Encryption with Associated Data)密码算法要求IV为隐式的，所以IV又恢复到由MasterSecret生成，即TLS 1.0以前的风格。

TLS 1.2支持使用GCM、CCM的新密码套件。

同时SSL 2.0被宣布放弃，不再向后兼容SSL 2.0.

下图

**五 SSL/TLS的流行实现**

本节简单介绍一下流行的SSL/TLS实现库，SSL协议非常复杂，由开发者自己实现常常会出错，开发者在具体实现SSL协议时通常会依赖于这些密码学库。

**5.1 常见的SSL/TLS 实现**

OpenSSL：这是非常流行的开源SSL/TLS实现。

OpenSSLim完全用C语言实现，支持SSL 2.0/3.0，TLS 1.0/1.1/1.2以及DTLS 1.0。

OpenSSL 近年来出现了很多的安全漏洞，比如2014年曝出的著名的Heartbleed漏洞等。

JSSE：这是使用Java实现的，支持SSL 3.0，TLS 1.0/1.1/1.2.

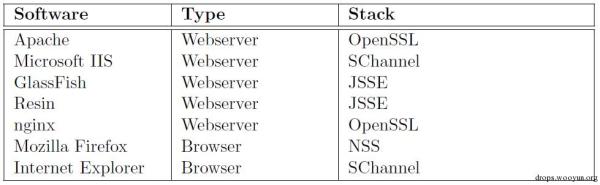
Bouncy Castle：它不仅仅支持SSL/TLS，它是一个完整的密码学库，支持各种密码学算法和协议。不过它仅仅支持TLS 1.0版本。

Android平台主要使用这个密码学库。

GnuTLS：这是另一个用C语言实现的库，支持SSL 3.0，TLS 1.0/1.1/1.2以及DTLS 1.0。主要在Unix世界被使用。同时以各种安全漏洞多而闻名。

NSS：这是最初由网景公司(Netscape)开发的库，支持SSL 2.0/3.0，TLS 1.0/1.1，现在主要被浏览器和客户端软件使用，比如Firefox使用的就是NSS库，Chrome使用的是一个NSS库的修正版。

下表是一些常见软件以及它们所使用的SSL/TLS实现库的情况：



其它还有一些常用的SSL实现库，如cryptlib、CyaSSL、MatrixSSL、PolarSSL等，由于市场占有率不高，我们这里就不多做介绍了。

**5.2 流行SSL/TLS实现库的安全研究**

最近几年曝出的高风险SSL安全漏洞大多跟SSL实现库有关，比如2014年4月曝出的“心脏滴血”漏洞，存在于OpenSSL 1.0.1-1.0.1f版本中，影响全球近17%的Web服务器;同样是2014年曝出的苹果公司iOS 7.0.6版本系统中存在的“gotofail”漏洞，因为程序员的疏忽导致SSL证书校验中的签名校验失效;包括今年曝出的SSL Freak攻击也是由于SSL实现库的安全漏洞导致的攻击，我们研究小组的同学对这个攻击有详细的分析，参见《SSL Freak来袭：如何实施一个具体的SSL Freak攻击》。同时我们还开发了一个基于python的中间人代理攻击框架“风声”对某国内知名电商的服务器进行具体的攻击，并上报了漏洞。

考虑到大量SSL/TLS实现库中存在安全问题，同时这些主流的SSL/TLS实现库对开发者而言使用难度较高，比如有些SSL/TLS实现库要求开发者自己进行随机数生成或密钥管理，让缺乏系统信息安全知识培训的开发者去使用这样高度复杂的密码学库容易产生很多安全问题。我们在这里推荐一些高级密码学库：Google keycazer、NaCl、Cryptlib、GPGME。这些密码学库存在的安全问题较少，同时封装了一些底层的密码学操作，降低了开发者的使用难度。

以上就是本次要介绍的SSL /TLS协议基本知识，后续的文章我们会对一些典型SSL/TLS攻击进行具体介绍。