HTTP + SSL = HTTPS

XLsn0w (/u/eacc1a5d2fda) ＋关注
2016.12.19 11:08* 字数 10135 阅读 1286 评论 2 喜欢 30 阅读 1286 评论 2 喜欢 30
(/u/eacc1a5d2fda)

一、作用

不使用SSL/TLS的HTTP通信，就是不加密的通信。所有信息明文传播，带来了三大风险。

（1）窃听风险（eavesdropping）：第三方可以获知通信内容。

（2）篡改风险（tampering）：第三方可以修改通信内容。

（3）冒充风险（pretending）：第三方可以冒充他人身份参与通信。

SSL/TLS协议是为了解决这三大风险而设计的，希望达到：

（1） 所有信息都是加密传播，第三方无法窃听。

（2） 具有校验机制，一旦被篡改，通信双方会立刻发现。

（3） 配备身份证书，防止身份被冒充。

互联网是开放环境，通信双方都是未知身份，这为协议的设计带来了很大的难度。而且，协议还必须能够经受所有匪夷所思的攻击，这使得SSL/TLS协议变得异常复杂。

二、历史

互联网加密通信协议的历史，几乎与互联网一样长。

1994年，NetScape公司设计了SSL协议（Secure Sockets Layer）的1.0版，但是未发布。

1995年，NetScape公司发布SSL 2.0版，很快发现有严重漏洞。

1996年，SSL 3.0版问世，得到大规模应用。

1999年，互联网标准化组织ISOC接替NetScape公司，发布了SSL的升级版TLS (http://en.wikipedia.org/wiki/Secure_Sockets_Layer)1.0版。

2006年和2008年，TLS进行了两次升级，分别为TLS 1.1版和TLS 1.2版。最新的变动是2011年TLS 1.2的修订版 (http://tools.ietf.org/html/rfc6176)。

目前，应用最广泛的是TLS 1.0，接下来是SSL 3.0。但是，主流浏览器都已经实现了TLS 1.2的支持。

TLS 1.0通常被标示为SSL 3.1，TLS 1.1为SSL 3.2，TLS 1.2为SSL 3.3。

三、基本的运行过程

SSL/TLS协议的基本思路是采用公钥加密法 (http://en.wikipedia.org/wiki/Public-key_cryptography)，也就是说，客户端先向服务器端索要公钥，然后用公钥加密信息，服务器收到密文后，用自己的私钥解密。

但是，这里有两个问题。

（1）如何保证公钥不被篡改?

解决方法：将公钥放在数字证书 (http://en.wikipedia.org/wiki/Digital_certificate)中。只要证书是可信的，公钥就是可信的。

（2）公钥加密计算量太大，如何减少耗用的时间?

解决方法：每一次对话（session），客户端和服务器端都生成一个"对话密钥"（session key），用它来加密信息。由于"对话密钥"是对称加密，所以运算速度非常快，而服务器公钥(非对称加密)只用于加密"对话密钥"本身，这样就减少了加密运算的消耗时间。

因此，SSL/TLS协议的基本过程是这样的：

（1） 客户端向服务器端索要并验证公钥。

（2） 双方协商生成"对话密钥"。

（3） 双方采用"对话密钥"进行加密通信。

上面过程的前两步，又称为"握手阶段"（handshake）。

四、握手阶段的详细过程

"握手阶段"涉及四次通信，我们一个个来看。需要注意的是，"握手阶段"的所有通信都是明文的。

4.1 客户端发出请求（ClientHello）

首先，客户端（通常是浏览器）先向服务器发出加密通信的请求，这被叫做ClientHello请求。

在这一步，客户端主要向服务器提供以下信息。

（1） 支持的协议版本，比如TLS 1.0版。

（2） 一个客户端生成的随机数，稍后用于生成"对话密钥"。

（3） 支持的加密方法，比如RSA公钥加密。

（4） 支持的压缩方法。

这里需要注意的是，客户端发送的信息之中不包括服务器的域名。也就是说，理论上服务器只能包含一个网站，否则会分不清应该向客户端提供哪一个网站的数字证书。这就是为什么通常一台服务器只能有一张数字证书的原因。

对于虚拟主机的用户来说，这当然很不方便。2006年，TLS协议加入了一个Server Name Indication扩展 (http://tools.ietf.org/html/rfc4366)，允许客户端向服务器提供它所请求的域名。

4.2 服务器回应（SeverHello）

服务器收到客户端请求后，向客户端发出回应，这叫做SeverHello。服务器的回应包含以下内容。

（1） 确认使用的加密通信协议版本，比如TLS 1.0版本。如果浏览器与服务器支持的版本不一致，服务器关闭加密通信。

（2） 一个服务器生成的随机数，稍后用于生成"对话密钥"。

（3） 确认使用的加密方法，比如RSA公钥加密。

（4） 服务器证书。

除了上面这些信息，如果服务器需要确认客户端的身份，就会再包含一项请求，要求客户端提供"客户端证书"。比如，金融机构往往只允许认证客户连入自己的网络，就会向正式客户提供USB密钥，里面就包含了一张客户端证书。

4.3 客户端回应

客户端收到服务器回应以后，首先验证服务器证书。如果证书不是可信机构颁布、或者证书中的域名与实际域名不一致、或者证书已经过期，就会向访问者显示一个警告，由其选择是否还要继续通信。

如果证书没有问题，客户端就会从证书中取出服务器的公钥。然后，向服务器发送下面三项信息。

（1） 一个随机数。该随机数用服务器公钥加密，防止被窃听。

（2） 编码改变通知，表示随后的信息都将用双方商定的加密方法和密钥发送。

（3） 客户端握手结束通知，表示客户端的握手阶段已经结束。这一项同时也是前面发送的所有内容的hash值，用来供服务器校验。

上面第一项的随机数，是整个握手阶段出现的第三个随机数，又称"pre-master key"。有了它以后，客户端和服务器就同时有了三个随机数，接着双方就用事先商定的加密方法，各自生成本次会话所用的同一把"会话密钥"。

至于为什么一定要用三个随机数，来生成"会话密钥"，dog250(http://blog.csdn.net/dog250/article/details/5717162)解释得很好：

"不管是客户端还是服务器，都需要随机数，这样生成的密钥才不会每次都一样。由于SSL协议中证书是静态的，因此十分有必要引入一种随机因素来保证协商出来的密钥的随机性。

对于RSA密钥交换算法 (http://lib.csdn.net/base/datastructure)来说，pre-master-key本身就是一个随机数，再加上hello消息中的随机，三个随机数通过一个密钥导出器最终导出一个对称密钥。

pre master的存在在于SSL协议不信任每个主机都能产生完全随机的随机数，如果随机数不随机，那么pre master secret就有可能被猜出来，那么仅适用pre master secret作为密钥就不合适了，因此必须引入新的随机因素，那么客户端和服务器加上pre master secret三个随机数一同生成的密钥就不容易被猜出了，一个伪随机可能完全不随机，可是是三个伪随机就十分接近随机了，每增加一个自由度，随机性增加的可不是一。"

此外，如果前一步，服务器要求客户端证书，客户端会在这一步发送证书及相关信息。

4.4 服务器的最后回应

服务器收到客户端的第三个随机数pre-master key之后，计算生成本次会话所用的"会话密钥"。然后，向客户端最后发送下面信息。

（1）编码改变通知，表示随后的信息都将用双方商定的加密方法和密钥发送。

（2）服务器握手结束通知，表示服务器的握手阶段已经结束。这一项同时也是前面发送的所有内容的hash值，用来供客户端校验。

至此，整个握手阶段全部结束。接下来，客户端与服务器进入加密通信，就完全是使用普通的HTTP协议，只不过用"会话密钥"加密内容。

网址2：
http://blog.163.com/magicc_love/blog/static/185853662201321423527263/

1. HTTPS ()

HTTPS（全称：Hypertext Transfer Protocol over Secure Socket Layer），是以安全为目标的HTTP通道，简单讲是HTTP的安全版。即HTTP下加入SSL层，HTTPS的安全基础是SSL，因此加密的详细内容就需要SSL。 它是一个URI scheme（抽象标识符体系），句法类同http:体系。用于安全的HTTP数据传输。https:URL表明它使用了HTTPS，但HTTPS存在不同于HTTP的默认端口及一个加密/身份验证层（在HTTP与TCP之间）。这个系统的最初研发由网景公司进行，提供了身份验证与加密通讯方法，现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面

HTTPS和HTTP的区别 ()

一、https协议需要到ca申请证书，一般免费证书很少，需要交费。

二、http是超文本传输协议，信息是明文传输，https 则是具有安全性的ssl加密传输协议。

三、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

四、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

https的实现原理 ()

有两种基本的加解密算法类型 ()

1)对称加密：密钥只有一个，加密解密为同一个密码，且加解密速度快，典型的对称加密算法有DES、AES等；

2)非对称加密：密钥成对出现（且根据公钥无法推知私钥，根据私钥也无法推知公钥），加密解密使用不同密钥（公钥加密需要私钥解密，私钥加密需要公钥解密），相对对称加密速度较慢，典型的非对称加密算法有RSA、DSA等。

https的通信过程 ()

图2.1 https的通信过程

https通信的优点 ()

1）客户端产生的密钥只有客户端和服务器端能得到；

2）加密的数据只有客户端和服务器端才能得到明文；

3）客户端到服务端的通信是安全的。

HTTPS解决的问题 ()

()一、信任主机的问题. ()

采用https的服务器必须从CA （Certificate Authority）申请一个用于证明服务器用途类型的证书。该证书只有用于对应的服务器的时候，客户端才信任此主机。所以目前所有的银行系统网站，关键部分应用都是https 的。客户通过信任该证书，从而信任了该主机。其实这样做效率很低，但是银行更侧重安全。这一点对我们没有任何意义，我们的服务器，采用的证书不管是自己发布的还是从公众的地方发布的，其客户端都是自己人，所以我们也就肯定信任该服务器。

()二、通讯过程中的数据的泄密和被篡改 ()

1. 一般意义上的https，就是服务器有一个证书。

a) 主要目的是保证服务器就是他声称的服务器，这个跟第一点一样。

b) 服务端和客户端之间的所有通讯，都是加密的。

i. 具体讲，是客户端产生一个对称的密钥，通过服务器的证书来交换密钥，即一般意义上的握手过程。

ii. 接下来所有的信息往来就都是加密的。第三方即使截获，也没有任何意义，因为他没有密钥，当然篡改也就没有什么意义了。

2. 少许对客户端有要求的情况下，会要求客户端也必须有一个证书。

a) 这里客户端证书，其实就类似表示个人信息的时候，除了用户名/密码，还有一个 CA 认证过的身份。因为个人证书一般来说是别人无法模拟的，所有这样能够更深的确认自己的身份。

b) 目前少数个人银行的专业版是这种做法，具体证书可能是拿U盘（即U盾）作为一个备份的载体

理解误区 () ()

它的安全保护依赖浏览器的正确实现以及服务器软件、实际加密算法的支持.

一种常见的误解是“银行用户在线使用https:就能充分彻底保障他们的银行卡号不被偷窃。”实际上，与服务器的加密连接中能保护银行卡号的部分，只有用户到服务器之间的连接及服务器自身。并不能绝对确保服务器自己是安全的，这点甚至已被攻击者利用，常见例子是模仿银行域名的钓鱼攻击。少数罕见攻击在网站传输客户数据时发生，攻击者会尝试窃听传输中的数据。

商业网站被人们期望迅速尽早引入新的特殊处理程序到金融网关，仅保留传输码(transaction number)。不过他们常常存储银行卡号在同一个数据库(http://lib.csdn.net/base/mysql)里。那些数据库和服务器少数情况有可能被未授权用户攻击和损害。

()SSL ()

SSL介绍 ()

SSL安全套接层协议(Secure Socket Layer)

为Netscape所研发，用以保障在Internet上数据传输之安全，利用数据加密(Encryption)技术，可确保数据在网络上之传输过程中不会被截取及窃听。目前一般通用之规格为40 bit之安全标准，美国则已推出128 bit之更高安全标准，但限制出境。只要3.0版本以上之IE或Netscape浏览器即可支持SSL。

当前版本为3.0。它已被广泛地用于Web浏览器与服务器之间的身份认证和加密数据传输。

SSL协议位于TCP/IP协议与各种应用层协议之间，是一种国际标准的加密及身份认证通信协议,为TCP提供一个可靠的端到端的安全服务，为两个通讯个体之间提供保密性和完整性(身份鉴别)。SSL协议可分为两层：SSL记录协议（SSL Record Protocol）：它建立在可靠的传输协议（如TCP）之上，为高层协议提供数据封装、压缩、加密等基本功能的支持。SSL握手协议（SSL Handshake Protocol）：它建立在SSL记录协议之上，用于在实际的数据传输开始前，通讯双方进行身份认证、协商加密算法、交换加密密钥等。

()SSL协议特点 ()

1）SSL协议可用于保护正常运行于TCP之上的任何应用协议，如HTTP、FTP、SMTP或Telnet的通信，最常见的是用SSL来保护HTTP的通信。

2）SSL协议的优点在于它是与应用层协议无关的。高层的应用协议（如HTTP、FTP、Telnet等）能透明地建立于SSL协议之上。

3）SSL协议在应用层协议之前就已经完成加密算法、通信密钥的协商以及服务器的认证工作。在此之后应用层协议所传送的数据都会被加密，从而保证通信的安全性。

4）SSL协议使用通信双方的客户证书以及CA根证书，允许客户/服务器应用以一种不能被偷听的方式通信，在通信双方间建立起了一条安全的、可信任的通信通道。

5）该协议使用密钥对传送数据加密，许多网站都是通过这种协议从客户端接收信用卡编号等保密信息。常用于交易过程中。

SSL功能 ()

1)客户对服务器的身份认证:

SSL服务器允许客户的浏览器使用标准的公钥加密技术和一些可靠的认证中心（CA）的证书，来确认服务器的合法性。

2)服务器对客户的身份认证:

也可通过公钥技术和证书进行认证，也可通过用户名，password来认证。

3)建立服务器与客户之间安全的数据通道:

SSL要求客户与服务器之间的所有发送的数据都被发送端加密、接收端解密，同时还检查数据的完整性。

SSL协议工作的基本流程 ()

1）接通阶段：客户机通过网络向服务器打招呼，服务器回应

2）密码交换阶段：客户机与服务器之间交换双方认可的密码，一般选用RSA密码算法

3）会谈密码阶段：客户机器与服务器间产生彼此交谈的会谈密码

4）检验阶段：客户机检验服务器取得的密码

5）客户认证阶段：服务器验证客户机的可信度

6）结束阶段：客户机与服务器之间相互交换结束的信息

SSL安全实现原理 ()

SSL 提供了用于启动 TCP/IP 连接的安全性“信号交换”：

1. 这种信号交换导致客户和服务器同意将使用的安全性级别，并履行连接的任何身份验证要求.

2. 通过数字签名和数字证书可实现浏览器和Web服务器双方的身份验证。

3.在用数字证书对双方的身份验证后，双方就可以用保密密钥进行安全的会话了。

()SSL协议说明 ()

SSL协议具有两层结构:

1)其底层是SSL记录协议层（SSL Record Protocol Layer）

2)其高层是SSL握手协议层（SSL Handshake Protocol Layer）, 主要用来让客户端及服务器确认彼此的身分,为了保护SSL记录封包中传送的数据，Handshake协议还能协助双方选择连接时所会使用的加密算法、MAC算法、及相关密钥。

SSL协议定义了两个通信主体：客户（client）和服务器（server）。其中，客户是协议的发起者

在客户/服务器结构中，应用层从请求服务和提供服务的角度定义客户和服务器，而SSL协议则从建立加密参数的过程中所扮演的角色来定义客户和服务器。

SSL握手协议包含四个阶段:第一个阶段建立安全能力;第二个阶段服务器鉴别和密钥交换;第三个阶段客户鉴别(可选的)和密钥交换;第四个阶段完成握手协议。

SSL协议工作的基本流程 ()

服务器认证阶段：1）客户端向服务器发送一个开始信息“Hello”以便开始一个新的会话连接；2）服务器根据客户的信息确定是否需要生成新的主密钥，如需要则服务器在响应客户的“Hello”信息时将包含生成主密钥所需的信息；3）客户根据收到的服务器响应信息，产生一个主密钥，并用服务器的公开密钥加密后传给服务器；4）服务器恢复该主密钥，并返回给客户一个用主密钥认证的信息，以此让客户认证服务器。

用户认证阶段: ()

在此之前，服务器已经通过了客户认证，这一阶段主要完成对客户的认证。经认证的服务器发送一个提问给客户，客户则返回（数字）签名后的提问和其公开密钥，从而向服务器提供认证。

()SSL流程 ()

第一步：身份验证

第二步：发明密语规则

第三步：密语规则共享

第四步：进行安全通信

简要描述

1）客户端向服务器发送一个开始信息“Hello”以便开始一个新的会话连接

2）服务器根据客户的信息确定是否需要生成新的主密钥，如需要则服务器在响应客户的“Hello”信息时将包含生成主密钥所需的信息

3）客户根据收到的服务器响应信息，产生一个主密钥，并用服务器的公开密钥加密后传给服务器

4）服务器恢复该主密钥，并返回给客户一个用主密钥认证的信息，以此让客户认证服务器。(以上为服务端认证)

5）服务器已经通过了客户认证，这一阶段主要完成对客户对服务端的认证。经认证的服务器发送一个提问给客户，客户则返回（数字）签名后的提问和其公开密钥，从而向服务器提供认证。（客户端认证）

详细描述

SSL 协议既用到了公钥加密技术又用到了对称加密技术，对称加密技术虽然比公钥加密技术的速度快，可是公钥加密技术提供了更好的身份认证技术。SSL 的握手协议非常有效的让客户和服务器之间完成相互之间的身份认证，其主要过程如下：

1)客户端的浏览器向服务器传送客户端SSL 协议的版本号，加密算法的种类，产生的随机数，以及其他服务器和客户端之间通讯所需要的各种信息。

2)服务器向客户端传送SSL 协议的版本号，加密算法的种类，随机数以及其他相关信息，同时服务器还将向客户端传送自己的证书。

3)客户利用服务器传过来的信息验证服务器的合法性，服务器的合法性包括：证书是否过期，发行服务器证书的CA 是否可靠，发行者证书的公钥能否正确解开服务器证书的“发行者的数字签名”，服务器证书上的域名是否和服务器的实际域名相匹配。如果合法性验证没有通过，通讯将断开；如果合法性验证通过，将继续进行第四步。

4)用户端随机产生一个用于后面通讯的“对称密码”，然后用服务器的公钥（服务器的公钥从步骤②中的服务器的证书中获得）对其加密，然后将加密后的“预主密码”传给服务器。(服务端验证成功)

5)如果服务器要求客户的身份认证（在握手过程中为可选），用户可以建立一个随机数然后对其进行数据签名，将这个含有签名的随机数和客户自己的证书以及加密过的“预主密码”一起传给服务器。

6)如果服务器要求客户的身份认证，服务器必须检验客户证书和签名随机数的合法性，具体的合法性验证过程包括：客户的证书使用日期是否有效，为客户提供证书的CA 是否可靠，发行CA 的公钥能否正确解开客户证书的发行CA 的数字签名，检查客户的证书是否在证书废止列表（CRL）中。检验如果没有通过，通讯立刻中断；如果验证通过，服务器将用自己的私钥解开加密的“预主密码”，然后执行一系列步骤来产生主通讯密码（客户端也将通过同样的方法产生相同的主通讯密码）。

7)服务器和客户端用相同的主密码即“通话密码”，一个对称密钥用于SSL 协议的安全数据通讯的加解密通讯。同时在SSL 通讯过程中还要完成数据通讯的完整性，防止数据通讯中的任何变化。

8)客户端向服务器端发出信息，指明后面的数据通讯将使用的步骤7中的主密码为对称密钥，同时通知服务器客户端的握手过程结束。

9)服务器向客户端发出信息，指明后面的数据通讯将使用的步骤7中的主密码为对称密钥，同时通知客户端服务器端的握手过程结束。

10)SSL 的握手部分结束，SSL 安全通道的数据通讯开始，客户和服务器开始使用相同的对称密钥进行数据通讯，同时进行通讯完整性的检验。

SSL缺点 ()

1）SSL协议需要在握手之前建立TCP连接，因此不能对UDP应用进行保护。

2）为了不致于由于安全协议的使用而导致网络性能大幅下降SSL协议并不是默认地要求进行客户鉴别

用AFN框架实现https注意事项：（http://my.oschina.net/non6/blog/290175）

友情提示：本文使用的AFNetworking是最新Git (http://lib.csdn.net/base/git)pull的2.3.1版本，如果想确认你机器上的AFNetworking版本，请打git tag命令查看。

绝大部分iOS (http://lib.csdn.net/base/ios)程序的后台服务都是基于RESTful或者WebService的，不论在任何时候，你都应该将服务置于HTTPS上，因为它可以避免中间人攻击的问题，还自带了基于非对称密钥的加密通道！现实是这些年涌现了大量速成的移动端开发人员，这些人往往基础很差，完全不了解加解密为何物，使用HTTPS后，可以省去教育他们各种加解密技术，生活轻松多了。

使用HTTPS有个问题，就是CA证书。缺省情况下，iOS要求连接的HTTPS站点必须为CA签名过的合法证书，AFNetworking是个iOS上常用的HTTP访问库，由于它是基于iOS的HTTP网络通讯库，自然证书方面的要求和系统是一致的，也就是你需要有一张合法的站点证书。

正式的CA证书非常昂贵，很多人都知道，AFNetworking2只要通过下面的代码，你就可以使用**（方法1：自签证书）**来访问HTTPS

? (http://my.oschina.net/non6/blog/290175#)

1

2AFSecurityPolicy *securityPolicy = [AFSecurityPolicy defaultPolicy];

securityPolicy.allowInvalidCertificates = YES;

这么做有个问题，就是你无法验证证书是否是你的服务器后端的证书，给中间人攻击，即通过重定向路由来分析伪造你的服务器端打开了大门。

解决方法。AFNetworking2是允许**（方法2：内嵌证书）**的，通过内嵌证书，AFNetworking2就通过比对服务器端证书、内嵌的证书、站点域名是否一致来验证连接的服务器是否正确。由于CA证书验证是通过站点域名进行验证的，如果你的服务器后端有绑定的域名，这是最方便的。将你的服务器端证书，如果是pem格式的，用下面的命令转成cer格式

? (http://my.oschina.net/non6/blog/290175#)

1

openssl x509 -in<你的服务器证书>.pem -outform der -out server.cer

然后将生成的server.cer文件，如果有自建ca，再加上ca的cer格式证书，引入到app的bundle里，AFNetworking2在

? (http://my.oschina.net/non6/blog/290175#)

1

2

3

4AFSecurityPolicy *securityPolicy = [AFSecurityPolicy AFSSLPinningModeCertificate];

或者

AFSecurityPolicy *securityPolicy = [AFSecurityPolicy AFSSLPinningModePublicKey];

securityPolicy.allowInvalidCertificates = YES;//还是必须设成YES

情况下，会自动扫描bundle中.cer的文件，并引入，这样就可以通过自签证书来验证服务器唯一性了。

我前面说过，验证站点证书，是通过域名的，如果服务器端站点没有绑定域名（万恶的备案），仅靠IP地址上面的方法是绝对不行的。怎么办？答案是想通过设置是不可以的，你只能修改AFNetworking2的源代码！打开AFSecurityPolicy.m文件，找到方法：

? (http://my.oschina.net/non6/blog/290175#)

1

2- (BOOL)evaluateServerTrust:(SecTrustRef)serverTrust

forDomain:(NSString *)domain

将下面这部分注释掉

? (http://my.oschina.net/non6/blog/290175#)

1

2

3

4

5

6

7

8

9// SecTrustSetAnchorCertificates(serverTrust, (__bridge CFArrayRef)pinnedCertificates);

//

```
//          if (!AFServerTrustIsValid(serverTrust)) {
//              return NO;
//          }
//
//          if (!self.validatesCertificateChain) {
//              return YES;
//          }
```

这样，AFSecurityPolicy就只会比对服务器证书和内嵌证书是否一致，不会再验证证书是否和站点域名一致了。

这么做为什么是安全的？了解HTTPS的人都知道，整个验证体系中，最核心的实际上是服务器的私钥。私钥永远，永远也不会离开服务器，或者以任何形式向外传输。私钥和公钥是配对的，如果事先在客户端预留了公钥，只要服务器端的公钥和预留的公钥一致，实际上就已经可以排除中间人攻击了。

用AFN框架实现https注意事项：
(http://www.cnblogs.com/canghaixiaoyuer/p/4738453.html)

本篇说说安全相关的AFSecurityPolicy模块，AFSecurityPolicy用于验证HTTPS请求的证书，先来看看HTTPS的原理和证书相关的几个问题。

HTTPS ()

HTTPS 连接建立过程大致是，客户端和服务端建立一个连接，服务端返回一个证书，客户端里存有各个受信任的证书机构根证书，用这些根证书对服务端 返回的证书进行验证，经验证如果证书是可信任的，就生成一个pre-master secret，用这个证书的公钥加密后发送给服务端，服务端用私钥解密后得到pre-master secret，再根据某种算法生成master secret，客户端也同样根据这种算法从pre-master secret生成master secret，随后双方的通信都用这个master secret（客户端和服务端生成的一样，）对传输数据进行加密解密（对称加密）。

以上是简单过程，中间还有很多细节，详细过程和原理已经有很多文章阐述得很好，就不再复述，推荐一些相关文章：

关于非对称加密算法的原理：RSA算法原理<一>
(http://www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html)、<二>
(http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html)

关于整个流程：HTTPS那些事<一> (http://www.guokr.com/post/114121/)、<二>
(http://www.guokr.com/post/116169/)、<三> (http://www.guokr.com/blog/148613/)

关于数字证书：浅析数字证书
(http://www.cnblogs.com/hyddd/archive/2009/01/07/1371292.html)

这里说下一开始我比较费解的两个问题：

1.证书是怎样验证的？怎样保证中间人不能伪造证书?

首先要知道非对称加密算法的特点，非对称加密有一对公钥私钥，用公钥加密的数据只能通过对应的私钥解密，用私钥加密的数据只能通过对应的公钥解密。

我们来看最简单的情况：一个证书颁发机构(CA)，颁发了一个证书A，服务器用这个证书建立https连接。客户端在信任列表里有这个CA机构的根证书。

首先CA机构颁发的证书A里包含有证书内容F，以及证书加密内容F1，加密内容F1就是用这个证书机构的私钥对内容F加密的结果。（这中间还有一次hash算法，略过。）

建 立https连接时，服务端返回证书A给客户端，客户端的系统里的CA机构根证书有这个CA机构的公钥，用这个公钥对证书A的加密内容F1解密得 到F2，跟证书A里内容F对比，若相等就通过验证。整个流程大致是：F->CA私钥加密->F1->客户端CA公钥解密->F。 因为中间人不会有CA机构的私钥，客户端无法通过CA公钥解密，所以伪造的证书肯定无法通过验证。

2.什么是SSL Pinning?

可以理解为证书绑定，是指客户端直接保存服务端的证书，建立https连接时直接对比服务端返回的和客户端保存的两个证书是否一样，一样就表明证书 是真的，不再去系统的信任证书机构里寻找验证。这适用于非浏览器应用，因为浏览器跟很多未知服务端打交道，无法把每个服务端的证书都保存到本地，但CS架 构的像手机APP事先已经知道要进行通信的服务端，可以直接在客户端保存这个服务端的证书用于校验。

为什么直接对比就能保证证书没问题？ 如果中间人从客户端取出证书，再伪装成服务端跟其他客户端通信，它发送给客户端的这个证书不就能通过验证吗？确 实可以通过验证，但后续的流程走不下去，因为下一步客户端会用证书里的公钥加密，中间人没有这个证书的私钥就解不出内容，也就截获不到数据，这个证书的私 钥只有真正的服务端有，中间人伪造证书主要伪造的是公钥。

为什么要用SSL Pinning？正常的验证方式不够吗？如果服务端的证书是从受信任的的CA机构颁发的，验证是没问题的，但CA机构颁发证书比较昂贵，小企业或个人用 户可能会选择自己颁发证书，这样就无法通过系统受信任的CA机构列表验证这个证书的真伪了，所以需要SSL Pinning这样的方式去验证。

AFSecurityPolicy ()

NSURLConnection 已经封装了https连接的建立、数据的加密解密功能，我们直接使用NSURLConnection是可以访问 https网站的，但NSURLConnection并没有验证证书是否合法，无法避免中间人攻击。要做到真正安全通讯，需要我们手动去验证服务端返回的 证书，AFSecurityPolicy封装了证书验证的过程，让用户可以轻易使用，除了去系统信任CA机构列表验证，还支持SSL Pinning方式的验证。使用方法：

1

2

3

4

5

6

7//把服务端证书(需要转换成cer格式)放到APP项目资源里，AFSecurityPolicy会自动寻找根目录下所有cer文件

AFSecurityPolicy *securityPolicy = [AFSecurityPolicy policyWithPinningMode:AFSSLPinningModePublicKey];

securityPolicy.allowInvalidCertificates = YES;

[AFHTTPRequestOperationManager manager].securityPolicy = securityPolicy;

[manager GET:@"https://example.com/
(https://example.com/)"parameters:nil success:^(AFHTTPRequestOperation *operatio
n, id responseObject) {

} failure:^(AFHTTPRequestOperation *operation, NSError *error) {

}];

AFSecurityPolicy分三种验证模式：

AFSSLPinningModeNone

这个模式表示不做SSL pinning，只跟浏览器一样在系统的信任机构列表里验证服务端返回的证书。若证书是信任机构签发的就会通过，若是自己服务器生成的证书，这里是不会通过的。

AFSSLPinningModeCertificate

这个模式表示用证书绑定方式验证证书，需要客户端保存有服务端的证书拷贝，这里验证分两步，第一步验证证书的域名/有效期等信息，第二步是对比服务端返回的证书跟客户端返回的是否一致。

这里还没弄明白第一步的验证是怎么进行的，代码上跟去系统信任机构列表里验证一样调用了SecTrustEvaluate，只是这里的列表换成了客户端保存的那些证书列表。若要验证这个，是否应该把服务端证书的颁发机构根证书也放到客户端里？

AFSSLPinningModePublicKey

这个模式同样是用证书绑定方式验证，客户端要有服务端的证书拷贝，只是验证时只验证证书里的公钥，不验证证书的有效期等信息。只要公钥是正确的，就能保证通信不会被窃听，因为中间人没有私钥，无法解开通过公钥加密的数据。

整个AFSecurityPolicy就是实现这这几种验证方式，剩下的就是实现细节了，详见源码。（AFSecurity.m）

具体代码:(http://blog.csdn.net/woaifen3344/article/details/41145729)

下面说明一下使用AFNetworking网络库访问的方式：

[objc]view plain (http://blog.csdn.net/woaifen3344/article/details/41145729#)copy
(http://blog.csdn.net/woaifen3344/article/details/41145729#)
(https://code.csdn.net/snippets/520166)
(https://code.csdn.net/snippets/520166/fork)

- (void)testClientCertificate {

SecIdentityRef identity =NULL;

SecTrustRef trust =NULL;

NSString*p12= [[NSBundlemainBundle]pathForResource:@"testClient"ofType:@"p12"
];

NSData*PKCS12Data = [NSDatadataWithContentsOfFile:p12];

[[selfclass]extractIdentity:&identityandTrust:&trustfromPKCS12Data:PKCS12Data];

NSString*url =@"https://218.244.131.231/ManicureShop/api/order/pay/%@";

NSDictionary*dic = @{@"request": @{

@"orderNo":@"1409282102222110030643",

@"type": @(2)

}

};

_signString =nil;

NSData*postData = [NSJSONSerializationdataWithJSONObject:dicoptions:NSJSON
WritingPrettyPrintederror:nil];

NSString*sign = [selfsignWithSignKey:@"test"params:dic];

NSMutableData*body = [postDatamutableCopy];

NSLog(@"%@", [[NSStringalloc]initWithData:bodyencoding:NSUTF8StringEncoding])
;

url = [NSStringstringWithFormat:url,sign];

AFHTTPRequestOperationManager*manager = [AFHTTPRequestOperationManagerm
anager];

manager.requestSerializer= [AFJSONRequestSerializerserializer];

manager.responseSerializer= [AFJSONResponseSerializerserializer];

[manager.requestSerializersetValue:@"application/json"forHTTPHeaderField:@"Acce
pt"];

[manager.requestSerializersetValue:@"application/json"forHTTPHeaderField:@"Cont
ent-Type"];

manager.responseSerializer.acceptableContentTypes= [NSSetsetWithArray:@[@"app
lication/json",@"text/plain"]];

manager.securityPolicy= [selfcustomSecurityPolicy];

[managerPOST:urlparameters:dicsuccess:^(AFHTTPRequestOperation*operation,idre
sponseObject) {

NSLog(@"JSON: %@", responseObject);

}failure:^(AFHTTPRequestOperation*operation,NSError*error) {

NSLog(@"Error: %@", error);

}];

}

下面这段代码是处理SSL安全性问题的:

[**objc**]view plain (http://blog.csdn.net/woaifen3344/article/details/41145729#)copy (http://blog.csdn.net/woaifen3344/article/details/41145729#) (https://code.csdn.net/snippets/520166) (https://code.csdn.net/snippets/520166/fork)

```
/**** SSL Pinning ****/

- (AFSecurityPolicy*)customSecurityPolicy {

NSString*cerPath = [[NSBundlemainBundle]pathForResource:@"testClient"ofType:@
"cer"];

NSData*certData = [NSDatadataWithContentsOfFile:cerPath];

AFSecurityPolicy*securityPolicy = [AFSecurityPolicydefaultPolicy];

[securityPolicysetAllowInvalidCertificates:YES];

[securityPolicysetPinnedCertificates:@[certData]];

[securityPolicysetSSLPinningMode:AFSSLPinningModeCertificate];

/**** SSL Pinning ****/

returnsecurityPolicy;

}
```

用SSL Pinning报错:

Error Domain=NSURLErrorDomain Code=-1012 "The operation couldn't be completed. (NSURLErrorDomain error -1012.)" UserInfo=0x14dc29e0 {NSErrorFailingURLKey=https://user.nowifi.cn/app/reg?action=send, NSErrorFailingURLStringKey=https://user.nowifi.cn/app/reg?action=send}

解决办法:

(http://blog.csdn.net/u012409247/article/details/49852919#)
(http://blog.csdn.net/u012409247/article/details/49852919#)
(http://blog.csdn.net/u012409247/article/details/49852919#)
(http://blog.csdn.net/u012409247/article/details/49852919#)
(http://blog.csdn.net/u012409247/article/details/49852919#)

顶

1

小礼物走一走，来简书关注我

赞赏支持

重驹 (/u/2af7a3f33fc8)
2楼 · 2016.12.21 23:06
(/u/2af7a3f33fc8)
^_^

赞 回复

> XLsn0w (/u/eacc1a5d2fda)： @重驹 (/users/2af7a3f33fc8) 吃惊
> 2016.12.22 09:28 回复

添加新评论

被以下专题收入，发现更多相似内容

iOS Dev... (/c/3233d1a249ca?utm_source=desktop&utm_medium=notes-included-collection)

SSM+shiro等 (/c/f4c65dcb6f21?utm_source=desktop&utm_medium=notes-included-collection)

计算机网络 (/c/0394ef689745?utm_source=desktop&utm_medium=notes-included-collection)

我爱编程 (/c/7847442e0728?utm_source=desktop&utm_medium=notes-included-collection)

网络通讯 (/c/72d3db0a7da7?utm_source=desktop&utm_medium=notes-included-collection)

计算机基础 (/c/3628ab952a8e?utm_source=desktop&utm_medium=notes-included-collection)

推荐阅读 更多精彩内容 >(/)

写给齐帆齐老师的一封信 (/p/70a264c7e835?utm_c...
(/p/70a264c7e835?utm_campaign=maleskine&utm_content=note&utm_
亲爱的齐齐 你好！ 苏州一别，不觉快有两个月的时间了，虽然你是我的老师，名叫齐帆齐。但我还是习惯性的叫你齐齐，因为这样称呼我感觉很亲切，
春天里百花香 (/u/dd8cbb61a2bf?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

中年的某一个下午 (/p/14e29e25b737?utm_campaig...
(/p/14e29e25b737?utm_campaign=maleskine&utm_content=note&utm_
三原则：图原创，文原创，诗原创。 这个下午天气有点闷热 点了一杯热咖啡 貌似有点失策 等一个故意迟到的人却无可奈何 叹口气 他是我的客户这是我的
秋水饮马 (/u/a7f876850fa6?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

爱在大山深处 (/p/ee0269658440?utm_campaign=...
(/p/ee0269658440?utm_campaign=maleskine&utm_content=note&utm_
当海浪迷失了风帆， 当泪水模糊了双眼； 愤怒的澜沧江水， 淌不尽，沧海桑田。 巍巍的大山深处， 充斥着寒门的辛艰； 颓废的教室里， 聚集着学子的辛
奔奔的果果 (/u/7b4320588f6a?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

〈初试〉生死茫茫总断肠 (/p/7bb138e8899a?utm_c...
(/p/7bb138e8899a?utm_campaign=maleskine&utm_content=note&utm_
文/孤瞳 写在前面: 因柴胡大哥多番鼓励我可以尝试写一点诗词赏析，所以今日初试，分享一首大家耳熟能详的词——苏轼的《 江城子·乙卯正月二十日夜记

孤瞳 (/u/e328c061eed6?
utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

北上广这群“精神中产”，累到变坚强 (/p/8295f7d6ac...

(/p/8295f7d6acd3?
utm_campaign=maleskine&utm_content=note&utm_

“精神中产”，只能在精神上达到中产。在爬往物质中产这一阶梯上，他们有太多累与颠的故事。 他们是为微信微博的主流，可是注意力都在“点”上，看文

祖三三 (/u/117dc5364f62?
utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

---

ios中用AFN做https (/p/c9f980786f87?utm_campaign=maleskine&ut...

原文地址 http://blog.csdn.net/u012409247/article/details/49852919 SSL/TLS协议运行机制的概述
（http://www.ruanyifeng.com/blog/2014/02/ssl_tls.html） 一、作用 ...

123321123 (/u/0fbf551ff6fb?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/c2bf75485c15?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

浏览器、HTTP、SSL、HTTPS执行流程 (/p/c2bf75485c15?utm_campai...

浏览器、HTTP、SSL、HTTPS执行流程 当在浏览器中输入URL后，页面加载完成的过程中都发生了什么事情 1、解析URL 判断URL是否是合法的URL，如果URL不合法，则调用默认的搜索引擎，直接把输入的内容

凌巅 (/u/3af05d83ed3b?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/e093f1a11005?

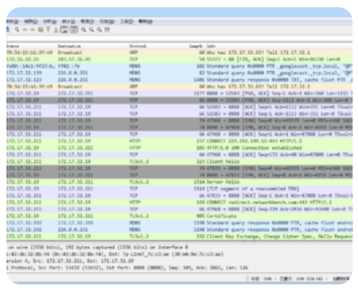
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

（转）SSL/TLS协议运行机制的概述 (/p/e093f1a11005?utm_campaign=...

互联网的通信安全，建立在SSL/TLS协议之上。 本文简要介绍SSL/TLS协议的运行机制。文章的重点是设计思想和运行过程，不涉及具体的实现细节。如果想了解这方面的内容，请参阅RFC文档。 一、作用 不使

拉肚 (/u/15514ffcefda?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/cf8c2f2cd18a?



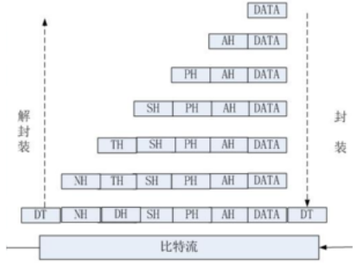
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

Wireshark 抓包理解 HTTPS 请求流程 (/p/cf8c2f2cd18a?utm_campaign...

准备 分析2.1. 三次握手2.2. 创建 HTTP 代理（非必要）2.3. TLS/SSL 握手2.4. 数据传输2.5. 四次挥手 扩展3.1. 摘要、MAC、数字签名到 CA 证书3.2. Session ID 和 Session Ticket3.3. SNI（Ser...

李狄青 (/u/5e87e5d9ab47?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/116ebf3034d9?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

OKHttp源码解析(二)："前戏"——HTTP的那些事 (/p/116ebf3034d9?ut...

1.OkHttp源码解析(一):OKHttp初阶2 OkHttp源码解析(二):OkHttp连接的"前戏"——HTTP的那些事3 OkHttp源码解析(三):OKHttp中阶之线程池和消息队列4 OkHttp源码解析(四):OKHttp中阶之拦截器及调用

 隔壁老李头 (/u/8b9c629f69dd?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

每日一句：重要不紧急 (/p/803ced8890f1?utm_campaign=maleskine&...

人的精力是有限的，你投入到了“紧急不重要”的事，就没办法兼顾“重要不紧急”的事。 而很多时候，正是“重要但不紧急”的事，拉开了人和人之间的差距。

 似是而飞 (/u/1e790d4fae51?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/205796d1414e?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

南航新版 (/p/205796d1414e?utm_campaign=maleskine&utm_content...

基友们又在撩着出去玩儿，于是先看看机票。 上次去杭州发现南航app有新版了，用了一下发现，预定机票的流程还不错呢。 首页 是不是有点无忧行的神髓，啊哈哈～入口清晰，第一印象不错。 信息输入 我们

 P_Simona (/u/807faa750249?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/5694a2d1af88?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

正念自我关怀 (/p/5694a2d1af88?utm_campaign=maleskine&utm_cont...

正念(Mindfulness)是对当下不评判的觉知，由美国麻省医学院的卡巴金博士在38年前创立，它将东方的古老智慧与西方的现代医学相结合，已被大量研究证明可以改变大脑结构和基因表达，体现在增强工作记

 海岳 (/u/310781beaeea?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

---

003 (/p/6b0d19ea5813?utm_campaign=maleskine&utm_content=not...

有些两句就说尽了 便流走 有些说一辈子也说不完 便成了诗

 吴宇良 (/u/ebfd26ab176a?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/64f26e57e5ec?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

201703回顾 (/p/64f26e57e5ec?utm_campaign=maleskine&utm_cont...

现在回顾第一季度的计划 计划一定要围绕目标来回顾，否则还定目标作甚？ 一、1-3月整体分析 1-3月的三个期待分别是围绕工作、健康和家庭三方面来做的。工作方面完成了拖了很久的课题一个，但是论文写


墨兰七七 (/u/63a5ac51897d?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)