29-我应该迁移到HTTPS吗？

今天是“安全篇”的最后一讲，我们已经学完了HTTPS、TLS相关的大部分知识。不过，或许你心里还会有一些困惑：

“HTTPS这么复杂，我是否应该迁移到HTTPS呢？它能带来哪些好处呢？具体又应该怎么实施迁移呢？”

这些问题不单是你，也是其他很多人，还有当初的我的真实想法，所以今天我就来跟你聊聊这方面的事情。

迁移的必要性

如果你做移动应用开发的话，那么就一定知道，Apple、Android、某信等开发平台在2017年就相继发出通知，要求所有的应用必须使用HTTPS连接，禁止不安全的HTTP。

在台式机上，主流的浏览器Chrome、Firefox等也早就开始“强推”HTTPS，把HTTP站点打上“不安全”的标签，给用户以“心理压力”。

Google等搜索巨头还利用自身的“话语权”优势，降低HTTP站点的排名，而给HTTPS更大的权重，力图让网民只访问到HTTPS网站。

这些手段都逐渐“挤压”了纯明文HTTP的生存空间，“迁移到HTTPS”已经不是“要不要做”的问题，而是“要怎么做”的问题了。HTTPS的大潮无法阻挡，如果还是死守着HTTP，那么无疑会被冲刷到互联网的角落里。

目前国内外的许多知名大站都已经实现了“全站HTTPS”，打开常用的某宝、某东、某浪，

都可以在浏览器的地址栏里看到“小锁头”，如果你正在维护的网站还没有实施HTTPS，那可要抓点紧了。

迁移的顾虑

据我观察，阻碍HTTPS实施的因素还有一些这样、那样的顾虑，我总结出了三个比较流行的观点：“慢、贵、难”。

所谓“慢”，是指惯性思维，拿以前的数据来评估HTTPS的性能，认为HTTPS会增加服务器的成本，增加客户端的时延，影响用户体验。

其实现在服务器和客户端的运算能力都已经有了很大的提升，性能方面完全没有担心的必要，而且还可以应用很多的优化解决方案（参见第28讲）。根据Google等公司的评估，在经过适当优化之后，HTTPS的额外CPU成本小于1%，额外的网络成本小于2%，可以说是与无加密的HTTP相差无几。

所谓“贵”，主要是指证书申请和维护的成本太高，网站难以承担。

这也属于惯性思维，在早几年的确是个问题，向CA申请证书的过程不仅麻烦，而且价格昂贵，每年要交几千甚至几万元。

但现在就不一样了，为了推广HTTPS，很多云服务厂商都提供了一键申请、价格低廉的证书，而且还出现了专门颁发免费证书的CA，其中最著名的就是“Let’s Encrypt”。

所谓的“难”，是指HTTPS涉及的知识点太多、太复杂，有一定的技术门槛，不能很快上手。

这第三个顾虑比较现实，HTTPS背后关联到了密码学、TLS、PKI等许多领域，不是短短几周、几个月就能够精通的。但实施HTTPS也并不需要把这些完全掌握，只要抓住少数几个要点就好，下面我就来帮你逐个解决一些关键的“难点”。

申请证书

要把网站从HTTP切换到HTTPS，首先要做的就是为网站申请一张证书。

大型网站出于信誉、公司形象的考虑，通常会选择向传统的CA申请证书，例如DigiCert、GlobalSign，而中小型网站完全可以选择使用“Let’s Encrypt”这样的免费证书，效果也完全不输于那些收费的证书。

“Let’s Encrypt”一直在推动证书的自动化部署，为此还实现了专门的ACME协议（RFC8555）。有很多的客户端软件可以完成申请、验证、下载、更新的“一条龙”操作，比如Certbot、acme.sh等等，都可以在“Let’s Encrypt”网站上找到，用法很简单，相关的文档也很详细，几分钟就能完成申请，所以我在这里就不细说了。

不过我必须提醒你几个注意事项。

第一，申请证书时应当同时申请RSA和ECDSA两种证书，在Nginx里配置成双证书验证，这样服务器可以自动选择快速的椭圆曲线证书，同时也兼容只支持RSA的客户端。

第二，如果申请RSA证书，私钥至少要2048位，摘要算法应该选用SHA-2，例如SHA256、SHA384等。

第三，出于安全的考虑，“Let’s Encrypt”证书的有效期很短，只有90天，时间一到就会过期失效，所以必须要定期更新。你可以在crontab里加个每周或每月任务，发送更新请求，不过很多ACME客户端会自动添加这样的定期任务，完全不用你操心。

配置HTTPS

搞定了证书，接下来就是配置Web服务器，在443端口上开启HTTPS服务了。

这在Nginx上非常简单，只要在“listen”指令后面加上参数“ssl”，再配上刚才的证书文件就可以实现最基本的HTTPS。

```
listen                443 ssl;

ssl_certificate       xxx_rsa.crt;  #rsa2048 cert
ssl_certificate_key   xxx_rsa.key;  #rsa2048 private key

ssl_certificate       xxx_ecc.crt;  #ecdsa cert
ssl_certificate_key   xxx_ecc.key;  #ecdsa private ke
```

为了提高HTTPS的安全系数和性能，你还可以强制Nginx只支持TLS1.2以上的协议，打开“Session Ticket”会话复用：

```
ssl_protocols               TLSv1.2 TLSv1.3;

ssl_session_timeout         5m;
ssl_session_tickets         on;
ssl_session_ticket_key      ticket.key;
```

密码套件的选择方面，我给你的建议是以服务器的套件优先。这样可以避免恶意客户端故意选择较弱的套件、降低安全等级，然后密码套件向TLS1.3“看齐”，只使用ECDHE、AES和ChaCha20，支持“False Start”。

```
ssl_prefer_server_ciphers   on;


ssl_ciphers   ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-R
```

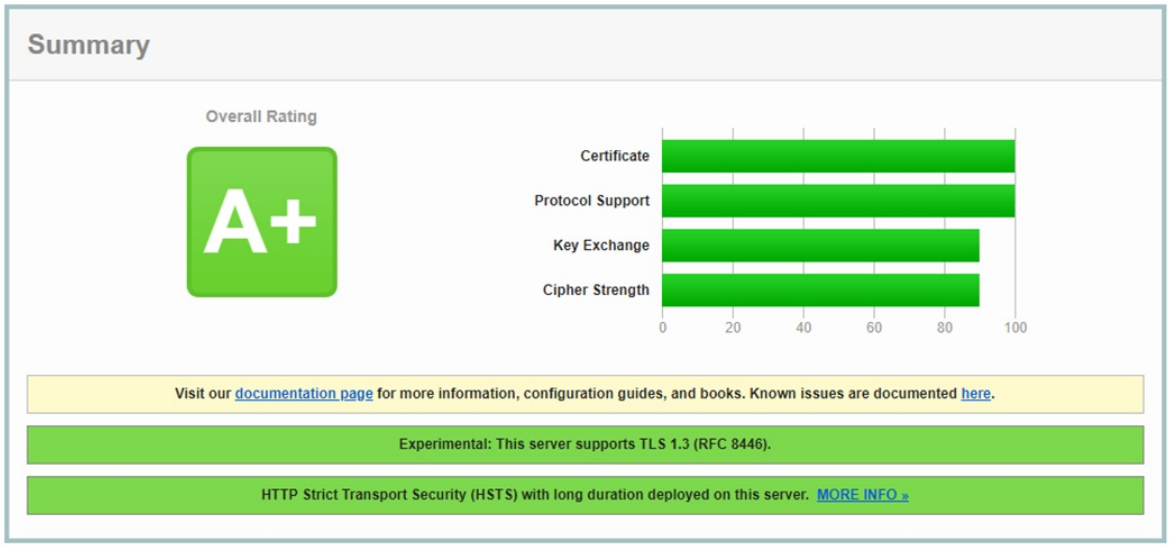
如果你的服务器上使用了OpenSSL的分支BorringSSL，那么还可以使用一个特殊的“等价密码组”（Equal preference cipher groups）特性，它可以让服务器配置一组“等价”的密码套件，在这些套件里允许客户端优先选择，比如这么配置：

```
ssl_ciphers
[ECDHE-ECDSA-AES128-GCM-SHA256|ECDHE-ECDSA-CHACHA20-POLY1305];
```

如果客户端硬件没有AES优化，服务器就会顺着客户端的意思，优先选择与AES“等价”的ChaCha20算法，让客户端能够快一点。

全部配置完成后，你可以访问“SSLLabs”网站，测试网站的安全程度，它会模拟多种客户端发起测试，打出一个综合的评分。

下图就是GitHub网站的评分结果：



服务器名称指示

配置HTTPS服务时还有一个“虚拟主机”的问题需要解决。

在HTTP协议里，多个域名可以同时在一个IP地址上运行，这就是“虚拟主机”，Web服务器会使用请求头里的Host字段（参见第9讲）来选择。

但在HTTPS里，因为请求头只有在TLS握手之后才能发送，在握手时就必须选择“虚拟主机”对应的证书，TLS无法得知域名的信息，就只能用IP地址来区分。所以，最早的时候每个HTTPS域名必须使用独立的IP地址，非常不方便。

那么怎么解决这个问题呢？

这还是得用到TLS的“扩展”，给协议加个**SNI**（Server Name Indication）的“补充条款”。它的作用和Host字段差不多，客户端会在“Client Hello”时带上域名信息，这样服务器就可以根据名字而不是IP地址来选择证书。

```
Extension: server_name (len=19)
    Server Name Indication extension
        Server Name Type: host_name (0)
        Server Name: www.chrono.com
```

Nginx很早就基于SNI特性支持了HTTPS的虚拟主机，但在OpenResty里可还以编写Lua脚本，利用Redis、MySQL等数据库更灵活快速地加载证书。

重定向跳转

现在有了HTTPS服务，但原来的HTTP站点也不能马上弃用，还是会有很多网民习惯在地址栏里直接敲域名（或者是旧的书签、超链接），默认使用HTTP协议访问。

所以，我们就需要用到第18讲里的“重定向跳转”技术了，把不安全的HTTP网址用301或302“重定向”到

新的HTTPS网站，这在Nginx里也很容易做到，使用“return”或“rewrite”都可以。

```
return 301 https://$host$request_uri;             #永久重定向
rewrite ^  https://$host$request_uri permanent;   #永久重定向
```

但这种方式有两个问题。一个是重定向增加了网络成本，多出了一次请求；另一个是存在安全隐患，重定向的响应可能会被“中间人”窜改，实现“会话劫持”，跳转到恶意网站。

不过有一种叫“**HSTS**”（HTTP严格传输安全，HTTP Strict Transport Security）的技术可以消除这种安全隐患。HTTPS服务器需要在发出的响应头里添加一个“**Strict-Transport-Security**”的字段，再设定一个有效期，例如：

```
Strict-Transport-Security: max-age=15768000; includeSubDomains
```

这相当于告诉浏览器：我这个网站必须严格使用HTTPS协议，在半年之内（182.5天）都不允许用HTTP，你以后就自己做转换吧，不要再来麻烦我了。

有了“HSTS”的指示，以后浏览器再访问同样的域名的时候就会自动把URI里的“http”改成“https”，直接访问安全的HTTPS网站。这样“中间人”就失去了攻击的机会，而且对于客户端来说也免去了一次跳转，加快了连接速度。

比如，如果在实验环境的配置文件里用“add_header”指令添加“HSTS”字段：

```
add_header Strict-Transport-Security max-age=15768000; #182.5days
```

那么Chrome浏览器只会在第一次连接时使用HTTP协议，之后就会都走HTTPS协议。

小结

今天我介绍了一些HTTPS迁移的技术要点，掌握了它们你就可以搭建出一个完整的HTTPS站点了。

但想要实现大型网站的“全站HTTPS”还是需要有很多的细枝末节的工作要做，比如使用CSP（Content Security Policy）的各种指令和标签来配置安全策略，使用反向代理来集中“卸载”SSL，话题太大，以后有机会再细谈吧。

简单小结一下今天的内容：

1. 从HTTP迁移到HTTPS是“大势所趋”，能做就应该尽早做；
2. 升级HTTPS首先要申请数字证书，可以选择免费好用的“Let’s Encrypt”；
3. 配置HTTPS时需要注意选择恰当的TLS版本和密码套件，强化安全；

4. 原有的HTTP站点可以保留作为过渡，使用301重定向到HTTPS。

课下作业

1. 结合你的实际工作，分析一下迁移HTTPS的难点有哪些，应该如何克服？
2. 参考上一讲，你觉得配置HTTPS时还应该加上哪些部分？

欢迎你把自己的学习体会写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，也欢迎把文章分享给你的朋友。

课外小贴士

01 也有少数知名网站仍然坚持使用 HTTP，例如 nginx.org、apache.org。

02 SNI 使用明文表示域名，也就提前暴露了一部分 HTTPS 的信息，有安全隐患，容易被“中间人”发起拒绝攻击，被认为是 TLS“盔甲上最后的一个缝隙”，目前正在起草 ESNI 规范。

03 “HSTS”无法防止黑客对第一次访问的攻击，所以 Chrome 等浏览器还内置了一个“HSTS preload”的列表（chrome://net-internals/#hsts），只要域名在这个列表里，无论何时都会强制使用 HTTPS 访问。

04 “HPKP”（HTTP Public Key Pinning）是另一种 HTTPS 安全技术，指示客户端固定网站使

用的公钥，防止 中间人 攻击，但因为接受程度过低，现在已经被放弃了。

05 如果要支持老的 WindowsXP 和 IE6，可以选择开启 SSLv3 和 RSA、RC4、SHA-1。

06 之前在实验环境访问 HTTP 协议时可以看到请求头里有“Upgrade-Insecure-Requests: 1”，它就是 CSP 的一种，表示浏览器支持升级到 HTTPS 协议。

精选留言：

- 许童童 2019-08-02 15:27:14
  老师你好，刚才搞了半天编译好了Nginx新版本With OpenSSL才开启TLSv1.3，不知道老师是怎么安装这些软件的，有什么好的建议吗？
  nginx version: nginx/1.16.0
  built by gcc 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC)
  built with OpenSSL 1.1.1 11 Sep 2018 [1赞]

作者回复2019-08-02 16:03:31
支持tls1.3主要就是OpenSSL要升级到1.11，Nginx的版本用较新的就好。

因为Nginx的ssl功能依赖于底层的OpenSSL，所以支持tls1.3比较简单，只要重新编译就好。

这方面好像没什么简单的方法，不过也不是很麻烦。

- 许童童 2019-08-02 11:45:44
个人博客网站很早就用上了https，但老师说的那些Nginx优化参数没有用上，我这就去加上。

作者回复2019-08-02 12:21:19
很好的实践机会。

- 阿锋 2019-08-02 11:40:42
上文提到的虚拟主机，跟正向代理，反向代理，有什么区别。

作者回复2019-08-02 12:23:41
虚拟主机与代理没有关系，是http服务器里的概念，在一个ip地址上存在多个域名（即主机），所以叫“虚拟主机”。

因为不能用ip地址区分，所以就要用host字段，区分不同的主机（域名、网站）。

- Geek_54edc1 2019-08-02 09:16:11
2、TLS1.3的pre-shared-key，实现0-RTT；OCSP Stapling；

作者回复2019-08-02 09:33:34
对，相应的Nginx指令是：

ssl_early_data on;
ssl_stapling on;