Let's Encrypt 给网站加 HTTPS 完全指南certbot

29 MAY 2016

前段时间在北京联通3G移动网络下，发现自己的站点被联通劫持注入恶心的话费充值广告，决定让我的网站强制使用 HTTPS，避免 ISP 劫持。

使用 HTTPS 前的一些疑惑

现在是 2016 年，使用 HTTPS 已经不像几年前是一件昂贵的事情。当然我也是自己了解了一圈才消除了自己的疑惑，主要是：

1. 我的网站（一个简单的博客）可能没必要使用 HTTPS
2. HTTPS 会不会让网站速度变慢？
3. HTTPS 需要花钱吧？证书好像不便宜
4. 配置和维护 HTTPS 代价很高

要回答这些问题，推荐大家去看一下 Google I/O 2016 的视频（Youtube）：[Mythbusting HTTPS: Squashing security's urban legends - Google I/O 2016]

视频里把所有问题都详细一一解答，强烈推家把视频看完。

我简单总结：

1. 每个网站都应该用 HTTPS，就算是全静态站点也同样如此，运营商劫持严重干扰访问者的体验
2. 有几项技术可以提高 HTTPS 的性能，包括 *Strict Transport Security*，*TLS False Start* 和 *HTTP/2*，这些技术让 HTTPS 速度不慢，某些情况下会甚至更快
3. HTTPS 针对个人单个（或者几个）域名的使用来说，已经是免费的
4. 配置和维护 HTTPS 异常简单，*Let's Encrypt* 这个项目通过自动化把事情简单化了

有哪些靠谱的免费 HTTPS 证书提供商？

选择证书提供商有3个主要考量：1. 浏览器和操作系统支持程度 2. 证书类型 3. 维护成本

1. 浏览器和操作系统支持程度

基本你能查到的热门证书提供商，支持程度都不会太差。例如 *Let's Encrypt* 的支持可以访问：Which browsers and operating systems support Let's Encrypt

可以看到，Android 2.3.6 以上，Firefox 2.0 以上，Windows Vista 以上，iOS 3.1 以上，Google Chrome全平台都是支持的。这一点就不用太担心了，看你你的网站受众情况来决定。对于我来说，我完全不在乎 Windows XP 的 IE 用户。

2. 证书类型

HTTPS 证书分为3类， 1. DV 域名验证证书 2. OV 组织机构验证证书 3. EV 增强的组织机构验证证书。每类证书在审核和验证方面要求严格程度不同，浏览器会在地址栏给予不同证书不一样的展现。

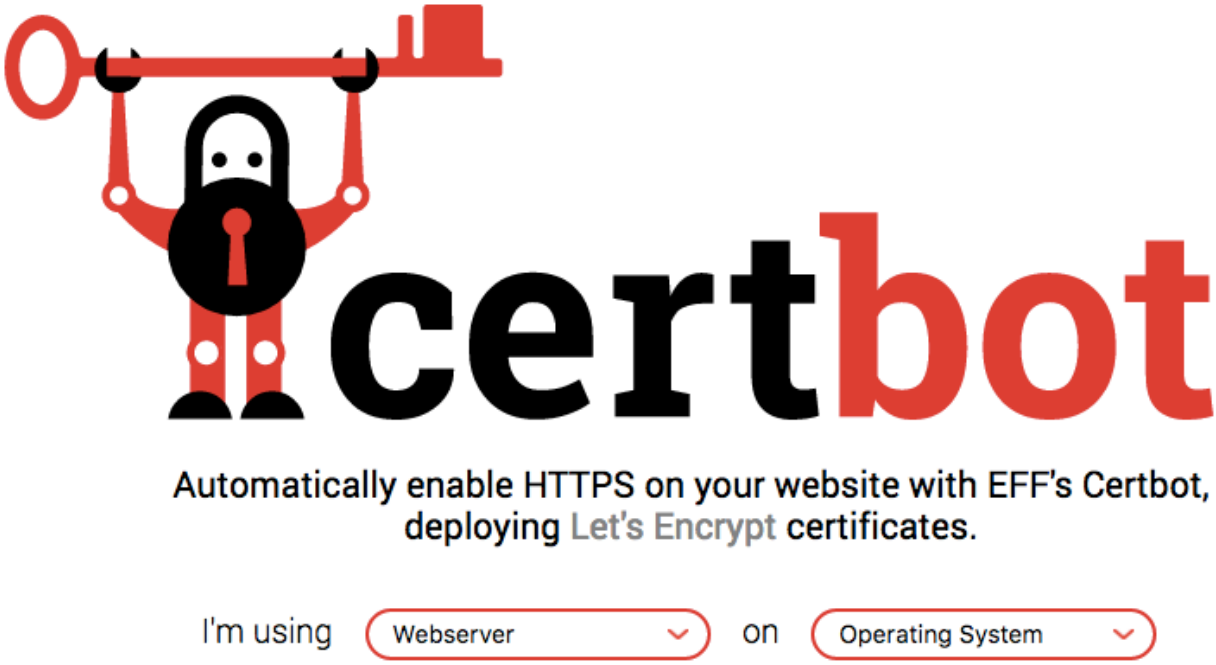
一般个人使用DV证书完全够了，浏览器表现为地址栏前会有绿色的小锁。下面聊到的免费证书都是 DV 域名验证证书。

3. 维护成本

我调研不多，使用过 *StartSSL*，现在用 *Let's Encrypt*。*StartSSL* 的免费证书有效期是1年，1年后需要手动更换。配置过程还挺麻烦的。

更推荐 *Let's Encrypt*，虽然有效期只有3个月，但可以用 certbot 自动续期，完全不受影响。而且 *Let's Encrypt* 因为有了 certbot 这样的自动化工具，配置管理起来非常容易。

生成 Let's Encrypt 证书



Let's Encrypt 证书生成不需要手动进行，官方推荐 certbot 这套自动化工具来实现。3步轻松搞定：

1. 下载安装 certbot (Let's Encrypt项目的自动化工具)
2. 创建配置文件
3. 执行证书自动化生成命令

下面的教程运行在 Arch Linux 上，其他操作系统也大同小异。你可以在 certbot 网站上，选择你的 Web Server 和 操作系统，就能看到对应的安装和配置教程。

1. 下载安装 certbot

在 Arch Linux 上，安装很简单：

```
$ sudo pacman -S letsencrypt
```

2. 创建配置文件

先创建存放配置文件的文件夹：

```
$ sudo mkdir /etc/letsencrypt/configs
```

编辑配置文件：

```
$ sudo vim /etc/letsencrypt/configs/example.com.conf
```

把 example.com 替换成自己的域名，配置文件内容：

```
email = your-email@example.com

# 把下面的路径修改为 example.com 的目录位置

authenticator = webroot

webroot-path = /var/www/example
```

这里需要解释一下，上面配置文件用了 webroot 的验证方法，这种方法适用于已经有一个 Web Server 运行中的情况。certbot 会自动在 `/var/www/example` 下面创建一个隐藏文件 `.well-known/acme-challenge`，通过请求这个文件来验证 example.com 确实属于你。外网服务器访问 http://www.example.com/.well-known/acme-challenge ，如果访问成功则验证OK。

我们不需要手动创建这个文件，certbot 会根据配置文件自动完成。

3. 执行证书自动化生成命令

一切就绪，我们现在可以运行 certbot 了。（注意：certbot 其实是一个 Python 脚本，在不同平台上的命令会不同，例如在 Arch Linux 上 certbot 的命令是 letsencrypt）。

```
$ sudo letsencrypt -c /etc/letsencrypt/configs/example.com.conf certonly

 - Congratulations! Your certificate and chain have been saved at
/etc/letsencrypt/live/example.com/fullchain.pem.
```

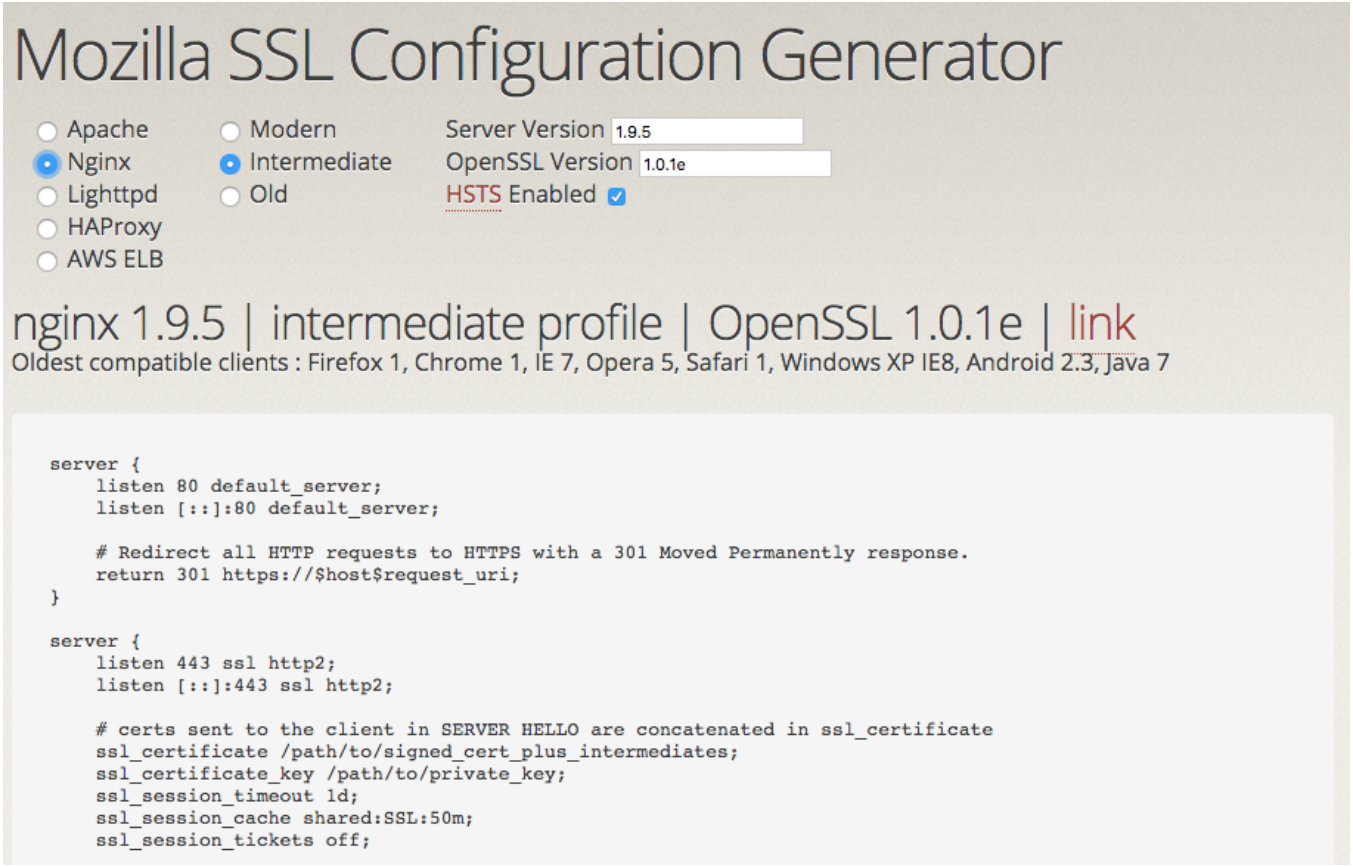
如果运行顺利，所有服务器所需要的证书就已经生成好了。他们被放在了 `/etc/letsencrypt/live/example.com/` 下：

```
$ ls /etc/letsencrypt/live/example.com/

cert.pem #server cert only
```

```
privkey.pem #private key

chain.pem #intermediates

fullchain.pem #server cert + intermediates
```

配置 Nginx 加入证书

到这里已经成功一大半了，只需要配置 Nginx 支持刚刚生成的证书。而且这个配置有最佳实践可以参考，访问：Mozilla SSL Configuration Generator，这是 Mozilla 搞得一个 HTTPS 配置文件自动生成器，支持 Apache，Nginx 等多种服务器。按照这个配置文件，选择 **Intermediate** 的兼容性。这里生成的配置文件是业界最佳实践和结果，让 Nginx 打开了各种增加安全性和性能的参数。



默认配置文件是这样的：

```
    listen 80 default_server;

    listen [::]:80 default_server;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently
response.
```

```
    return 301 https://$host$request_uri;

    listen 443 ssl http2;

    listen [::]:443 ssl http2;

    # certs sent to the client in SERVER HELLO are concatenated in
ssl_certificate

    ssl_certificate /path/to/signed_cert_plus_intermediates;

    ssl_certificate_key /path/to/private_key;

    ssl_session_timeout 1d;

    ssl_session_cache shared:SSL:50m;

    ssl_session_tickets off;

    # Diffie-Hellman parameter for DHE ciphersuites, recommended 2048 bits

    ssl_dhparam /path/to/dhparam.pem;

    # intermediate configuration. tweak to your needs.

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;

    ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-
AES128-SHA256:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-
AES128-SHA:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-
AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-
SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-
SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-CBC3-SHA:!DSS';

    ssl_prefer_server_ciphers on;

    # HSTS (ngx_http_headers_module is required) (15768000 seconds = 6
months)

    add_header Strict-Transport-Security max-age=15768000;

    # fetch OCSP records from URL in ssl_certificate and cache them

    ssl_stapling_verify on;

    ## verify chain of trust of OCSP response using Root CA and
Intermediate certs

    ssl_trusted_certificate /path/to/root_CA_cert_plus_intermediates;

    resolver <IP DNS resolver>;
```

请根据自己的服务配置修改和添加内容，重点只需要关注6行：

```
    listen 443 ssl http2;
    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;
    ssl_dhparam /etc/nginx/ssl/dhparam.pem;
    ssl_trusted_certificate
/etc/letsencrypt/live/example.com/root_ca_cert_plus_intermediates;
    resolver <IP DNS resolver>;
```

这6行中，部分文件还不存在，逐个说明。

首先是第一行 `listen 443 ssl http2;` 作用是启用 Nginx 的 ngx*http*v2_module 模块 支持 HTTP2，Nginx 版本需要高于 1.9.5，且编译时需要设置 `--with-http_v2_module` 。Arch Linux 的 Nginx 安装包中已经编译了这个模块，可以直接使用。如果你的 Linux 发行版本中的 Nginx 并不支持这个模块，可以自行 Google 如何加上。

`ssl_certificate` 和 `ssl_certificate_key` ，分别对应 `fullchain.pem` 和 `privkey.pem`，这2个文件是之前就生成好的证书和密钥。

`ssl_dhparam` 通过下面命令生成：

```
$ sudo mkdir /etc/nginx/ssl
$ sudo openssl dhparam -out /etc/nginx/ssl/dhparam.pem 2048
```

（可选）`ssl_trusted_certificate` 需要下载 *Let's Encrypt* 的 Root Certificates，不过根据Nginx 官方文档 所说，`ssl_certificate` 如果已经包含了 `intermediates` 就不再需要提供 ssl*trusted*certificate 了。这一步可以省略：

```
$ cd /etc/letsencrypt/live/example.com
```

```
$ sudo wget https://letsencrypt.org/certs/isrgrootx1.pem

$ sudo mv isrgrootx1.pem root.pem

$ sudo cat root.pem chain.pem > root_ca_cert_plus_intermediates
```

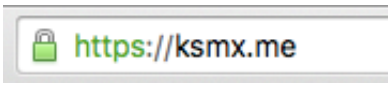
`resolver` 的作用是“resolve names of upstream servers into addresses”， 在這個配置中，resolver 是用來解析 OCSP 服務器的域名的，建议填写你的 VPS 提供商的 DNS 服务器，例如我的 VPN 在 Linode，DNS服务器填写：

```
resolver 106.187.90.5 106.187.93.5;
```

Nginx 配置完成后，重启后，用浏览器测试是否一切正常。

```
$ sudo systemctl restart nginx
```

这时候你的站点应该默认强制使用了 HTTPS，并且浏览器地址栏左边会有绿色的小锁：



自动化定期更新证书

Let's Encrypt 证书有效期是3个月，我们可以通过 certbot 来自动化续期。

在 Arch Linux 上，我们通过 systemd 来自动执行证书续期任务。

```
$ sudo vim /etc/systemd/system/letsencrypt.service

Description=Let's Encrypt renewal

ExecStart=/usr/bin/letsencrypt renew

ExecStartPost=/bin/systemctl reload nginx.service
```

然后增加一个 systemd timer 来触发这个服务：

```
$ sudo vim /etc/systemd/system/letsencrypt.timer
```

```
Description=Monthly renewal of Let's Encrypt's certificates
```

启用服务，开启 timer：

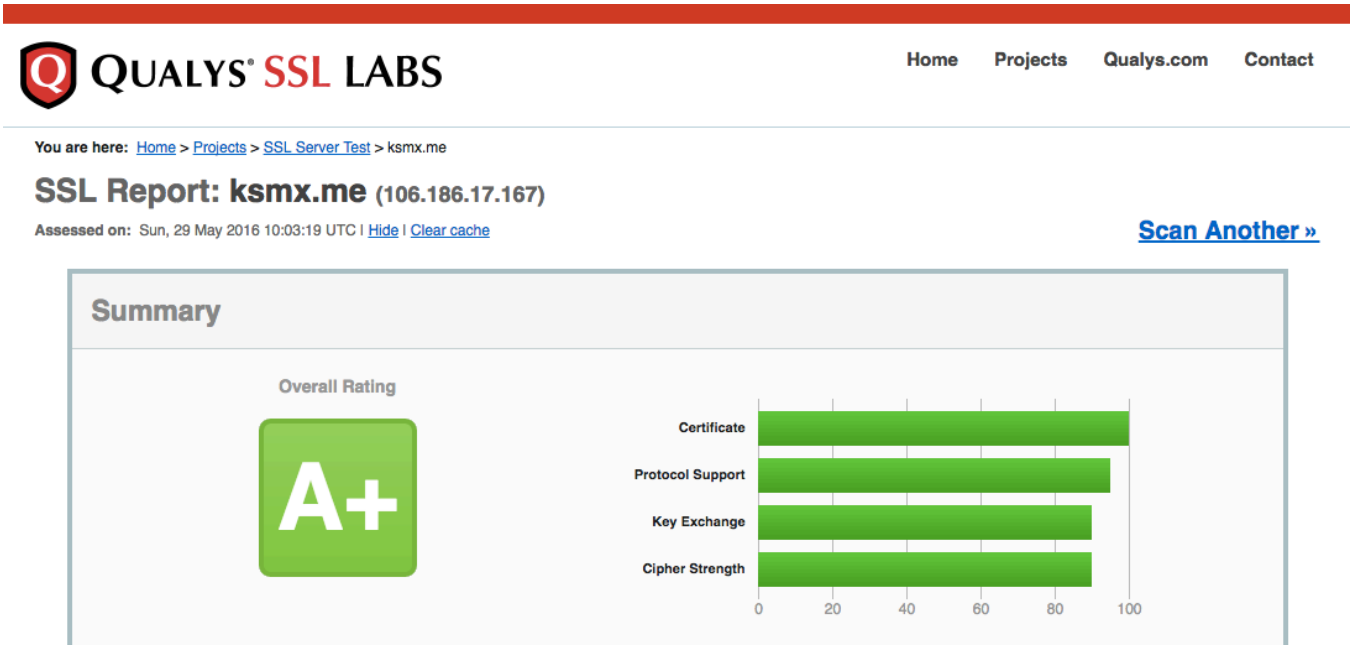
```
$ sudo systemctl enable letsencrypt.service

$ sudo systemctl start letsencrypt.timer
```

在其他 Linux 发行版本中，可以使用 crontab 来设定定时任务，自行 Google 吧。

用专业在线工具测试你的服务器 SSL 安全性

Qualys SSL Labs 提供了全面的 SSL 安全性测试，填写你的网站域名，给自己的 HTTPS 配置打个分。

如果你完全按照我上面教程配置，遵循了最佳实践，你应该和我一样得分是 A+



这意味着你启用了HTTPS，现在足够的安全，并且使用了最新技术，保证了性能。

为自己鼓个掌。 (๑•ㅂ•),✧