Obtaining an SSL certificate for a web server

Obtaining an SSL certificate for a web server

> **Advanced Topic**
> This is an advanced topic, intended for use by experienced system administrators or developers who need to run their own web servers. If you want an SSL-enabled site without having to manage your own certificates, consider IS&T's DrupalCloud service (Requesting SSL Certificate on Drupal Cloud), or SIPB's Scripts service. If you are the customer of a hosting service, rather than self-hosting, the hosting service maintainers need to be resolving any certificate problems on your behalf. If you do not have access to the webserver configuration, you cannot proceed with this guide.

On this page:

An SSL certificate is required to serve web pages and content via HTTPS. You can issue your own self-signed certificate for testing purposes, but for public-facing services, your certificate must be signed by a trusted Certificate Authority. These certificate authorities form a "web of trust", which ultimately leads back to one of the certificate authorities that is pre-installed in your web browser or operating system.

> **Let's Encrypt**
> Many hosting services and websites use Let's Encrypt which is open source software that will handle fetching, installing and renewing short duration certificates. If you are administering your own web hosting stack, it's strongly recommended that you configure and use Let's Encrypt. It's expected that it will work fine for MIT hostnames, as long as the DNS configuration is already set up. If you are hosting on a third party hosting service like Pantheon or SquareSpace, it's very likely that the platform administration is already using Let's Encrypt to take care of this for you. See Let's Encrypt (org)

MIT, through a site license with the Internet2 InCommon federation, can issue signed certificates, valid for 1 year, which are ultimately trusted by the AddTrust CA. The rest of this guide covers this process, but it's expected that this should be little used.

Contents:

A note about pathnames
These instructions are tailored for a Red Hat Enterprise Linux server, running Apache 2.4, and the pathnames reflect that. The commands here should work on any modern Linux or UNIX-based operated system, including Mac OS X. The pathnames where the key and certificates are stored will vary across operating systems. Consult the documentation for your web server or operating system to determine the preferred place to store keys and certificates. • Ubuntu: Keys are typically stored in /etc/ssl/private, certs in /etc/ssl/certs, and requests in /etc/ssl/reqs It is essential that your private key be stored in a directory that is readable only by the web server software or the system administrator. This directory should never be readable by other system users. If your private key is ever inadvertently exposed, it should be considered compromised, and a new one should be generated and new certificates should be issued.

Generating the key

> **First Time Only**
> You only need to generate a key when you first request a certificate. If you are renewing a certificate, you can skip to the next section.

The `openssl genrsa` command is used to generate a key. The filename after `-out` is the file where the key will be stored, and `2048` is the number of bits in the key, which roughly corresponds to its strength. MIT will not sign certificates with a key less than 2048 bits. (NIST projects 2048-bit keys to be sufficient through 2030, so we don't recommend you change this value without fully understanding the implications.)

```
openssl genrsa -out /etc/httpd/conf/ssl.key/hostname.mit.edu.key 2048
```

(Replace hostname.mit.edu with the hostname of your machine. If you have multiple Virtual Hosts on the same server, you will need a separate key and certificate for each virtual host.)

which should generate the following output:

```
Generating RSA private key, 2048 bit long modulus
.....................................................+++
....................................+++
e is 65537 (0x10001)
```

You may see a message which says `unable to write 'random state'`. This message may be disregarded (it indicates that the random number generator was unable to save its random state for future use.)

Generate the certificate request

The `openssl req` command is used to generate a certificate request. `-new` indicates this is a new certificate request. The filename after `-key` specifies the RSA key to use (generated in the previous step). The filename after `-out` is the file where the request will be stored.

```
openssl req -new -key /etc/httpd/conf/ssl.key/hostname.mit.edu.key -out
/etc/httpd/conf/hostname-YYYY.req
```

(Replace YYYY with the current year to keep a record of your previous requests.)

You will be prompted for various pieces of information, and you should generally use the following answers:

- **Country Name**: `US`
- **State or Province Name**: `Massachusetts`
- **Locality Name**: `Cambridge`
- **Organization Name**: `Massachusetts Institute of Technology`
- **Organizational Unit Name**: *Your DLC name (e.g. "IS&T", "Math", "DSL")*
- **Common Name**: *The hostname of your server (e.g. hostname.mit.edu)*
  This **must** match the hostname by which people will access your service.
- **Email Address**: A contact address for the server (e.g. your e-mail address, or a mailing list)

- You will also be prompted for a **challenge password** and an **optional company name**. These may be left blank. If you choose to specify a **challenge password**, you must make a note of it. It can be a simple word or phrase, and should **NOT** be your Kerberos password or a password you use at any other site.

Submit the request for signing.

You must now e-mail the file you just created (in this example, `/etc/httpd/conf/hostname-YYYY.req`}) to mitcert@mit.edu. The contents of the file are plain text, and may be pasted into the body of an e-mail, including the `---`~~`-----BEGIN CERTIFICATE REQUEST-----`~~ and ~~`-----END CERTIFICATE REQUEST-----`~~`---` lines. This is preferable to including the file as an attachment.

You should immediately receive a reply from our ticket tracking system, letting you know your request has been received. Within a day or so, you should receive another e-mail from `support@cert-manager.com` letting you know that your certificate request has been submitted to InCommon for processing and signing.

Downloading and installing your certificate

Once your certificate has been signed, you should receive another e-mail from the InCommon Certificate Services Manager, letting you know your certificate has been approved, and providing you with links to download your signed certificate. There are several links in this email, which contain your certificate encoded in several different formats.

Unless directed otherwise, you should use the link that says **Certificate only, PEM encoded**. You may use your web browser to download the file and transfer it to your server, or you may use a utility like `wget` or `curl` to download the certificate directly to your server.

Tip: It's sometimes convenient to install this as hostname.mit.edu-YYYY.crt and symlink that to hostname.mit.edu.crt, depending on your server configuration. Path names and naming conventions vary; for this example, the certificate will be installed at `/etc/httpd/conf/ssl.crt/hostname.mit.edu.crt`

Intermediate certificates

In most cases, you will also need to download the intermediate certificates (the certificates that form the web of trust between your certificate, and the one that is ultimately installed in the web browser or operating system), referred to as a "Certificate Chain". The chain is available via the last two links in the e-mail, and the order of certificates in the chain is important.

Apache users should use the link that says **Intermediate(s)/Root only, PEM encoded** and save the file as `/etc/httpd/conf/ssl.ca/InCommon-chain.crt` (For other server software, consult your documentation as to the correct order of the certificates in the intermediate chain.)

The InCommon chain is also available at https://web.mit.edu/apache-ssl/certificates/InCommon-chain.crt.txt (you will need to rename the file).

Configure your server software

Important: Verify that your certificate, your private key, and the certificate chain are only readable by the system administrator and the web server software, and are not writable. On Linux systems, this typically means the file has mode `0400` and is owned by `root`, but consult your operating system documentation.

To configure apache, you will need the following configuration values:

```
SSLCertificateChainFile    /etc/httpd/conf/ssl.ca/InCommon-chain.crt
SSLCertificateKeyFile      /etc/httpd/conf/ssl.key/hostname.mit.edu.key
SSLCertificateFile         /etc/httpd/conf/ssl.crt/hostname.mit.edu.crt
```

Apache must be restarted after updating the configuration. Consult your operating system documentation for the correct configuration file in which to specify these values, and also for the correct method to restart Apache.

Test the new certificate

Test your site in multiple browsers (Firefox, Chrome), ideally with a clean browser profile, to verify that it works and that no security warnings are generated.

Verifying your certificates

If you run into problems, it may be helpful to inspect the certificate and key files, and to verify the certificate chain. You can use the following commands (note that the pathnames and filenames may vary for your specific server).

Inspect the certificate:

```
openssl x509 -in /etc/httpd/conf/ssl.crt/hostname.mit.edu.crt -text
```

Inspect the key:

```
openssl rsa -in /etc/httpd/conf/ssl.key/hostname.mit.edu.key -text
```

Verify the certificate chain:

```
openssl verify -verbose -purpose sslserver -CAfile /etc/httpd/conf/ssl.ca/InCommon-chain.crt
/etc/httpd/conf/hostname-YYYY.crt
```