

Introduction

In order to have a website run on HTTPS the necessary certificates need to be created. Although one can create its own certificates, usually such a certificate is generated by a so called Certification Authority(CA). Although an own created certificate can also be used for secure communication, these certificates are not being recognized by browsers, resulting in a severe warning when such a page is being opened.

A single certificate can be valid for a single domain or subdomain or multiple domains. Some certificates (wildcard) can also be valid for each subdomain of a specific domain.

Though (official) certificates used to be quite expensive (the more secure they are/were, the more expensive they are/were), since a while there is a CA called 'Let's encrypt'.

They provide free to use certificates that are recognized by all modern browsers. The validation process for Let's encrypt is quite simple and though they may not be as secure as other commercial certificates they are still far better to be used than just run your server on http or use an own created certificate.

Let's encrypts certificates are only 90 days valid after their creation time and thus they should be renewed within this period. The process to renew a certificate is quite similar as the process to create a new certificate.

In the following text 'CA' will refer to any 'Certification Authorization'. 'LE' will specifically refer to Let's encrypt.

Challenging

In order to verify whether a certificate effectively belongs to the person/company which is referenced all CAs have mechanisms in place to do this validation.

This process boils down to the following (simplified):

- a (browser) certificate is requested for one or multiple domains at the CA.
- for such a request the CA sends one or multiple challenging codes
- the user should then put these codes somewhere that can be accessed by the CA and where the CA can determine the domains in the certificates are being managed by the user/company requesting the certificates in the first place.

LE has 2 mechanisms

HTTP challenging

The challenging codes as received from LE are to be put in a specific location on your webserver. The validation process will exist in LE hitting the exact location on your webserver using a plain HTTP call. The fact that the codes can be obtained from this URL on your domain is an indication for LE that you have control on the domains in the certificate (or at least the webserver running on this domain).

A certificate that was challenged in this way can be valid for multiple domains or subdomains (the challenging codes should be accessible by the webserver on the respective domains and subdomains). This challenging method does NOT allow to create wildcard certificates.

DNS challenging

The challenging codes as received from the LE need to be put in specific TXT records on the respective domains (“_acme-challenge”). As soon as this DNS record has been set on your domain(s), LE will check the existence. Again: the fact that this code can be read from your domain(s) indicate you have control on the domain and thus you are a rightful owner of the certificate. DNS challenging does allow to create wildcard certificates.

Tool

The attached tool allow for a fully automated certification process using DNS challenging. The full process consists of different steps:

- local generation of private/public key pairs (for an account and a CSR file – Certificate Signing Request). This done by openssl
- Communication with LE to obtain the challenging codes (using the standardized ACME protocol, implemented in the acme_client.jar file)
- Communication with your DNS provider (the one managing the nameservers of your domain(s)). As this communication can be done using an API of your provider and since this API is not standardized, currently only CloudFlare as our DNS provider is supported. Even if you purchased your domain by another registrar, you can still have your domains being hosted (for free) by Cloudflare by changing its nameservers (this communication is done in the CloudFlareAPI.jar file)
- Obtaining your certificates after the challenging process using (acme_client.jar)
- Converting the received certificates to the Java standard (JKS) to be used in the B4X webserver Jetty.

The automatic DNS challenging on Cloudflare might not work if you have 2-step validation on your Cloudflare’s account.

The renewal of certificates can also be done automatically using the tool: the process is the same as the one described above but starts at the second bullet point.

Prerequisites

- The latest version of Java 8 or any later version. Make sure you are using the latest version of Java 8 as previous versions do not have LE installed in its Certificate Store.
- The latest version of openssl: see <https://wiki.openssl.org/index.php/Binaries>
 - For most Linux distributions: openssl is probably already preinstalled, but make sure to update to the latest version (eg: apt-get update) or install it (eg: apt-get install openssl)
 - For Windows you can use the version of Indy: see <https://indy.fulgan.com/SSL/>
- The acme_client jar: see https://github.com/porunov/acme_client/releases
- The Java “keytool” tool. This is mostly part of your Java distribution (bin folder of java). If not it can be installed (eg: apt) get install keytool)

How to use the tool

It is advised to have the java binary/exe and keytool in our searchpath.

Run the main tool with

```
java -jar AutoGenerator.jar -mode:create -config:<full path of your config file>
```

No worries if you can't find your config file. When you run the Autogenerator and it fails to detect the config file it will generate a default one on the given location and then quit. Make sure you review and modify the settings in your config file before restarting the above command.

The default config file contains following sections:

```
openssl=<full path to your openssl binary/exe>
acme_client=<full path to the location of the acme_client.jar>
dns_client= <full path to the location of the CloudFlareAPI.jar file>
java=<full path to the java binary/exe or just 'java' when java is in your searchpath>
keytool=<full path to the keytool binary/exe or just 'java' when java is in your searchpath>
keyfolder=<main folder that will contain all necessary / generated files>
logfile= <filename of the logfile(s)>
domain=<main domain for which the certificate is being generated>
alternatives=<other domainnames to be included in the certificate>
email=<email address of the owner of the certificate>
certificat_c= <2 digit ISO code of the you/your company>
certificat_st=<your or your companies state>
certificat_l=<your or your companies city>
certificat_o=<your or your companies name>
certificat_ou=<department within your company>
password=<password of your certificate>
cf_email=<your cloudflare's email login>
cf_key = <your Cloudflare's global authorization code>
```

Note that, even for a Windows setup, using the slash (/) instead of the backslash to divide your subfolders. All keys should be present in the config file.

Notes:

keyfolder: this folder will contain all files created in the process of generating your certificate. Apart from the final .JKS file it, specifically, also contains:

- account.key: a unique private/public key linked to your person and with which authentication on LE is being done. Make sure you backup this key. It will be used when renewing your certificate.
- domain.csr: the certificate signing request. Make sure you backup this key. It will be used when renewing your certificate.
- PEM files: the generated certificates
- JKS: the final certificate to be used in the webserver
- digests folder: containing the challenging codes

Make sure this folder is created in advance and with the necessary permissions.

logfile: contains the location where the logfiles are being created. The system starts a new logfile for each day: therefore the logfile should contain a full a path + filename. The % section in the filename will be replaced by the current date (yyyyMMdd).

Eg: /logs/autogenerator_%.log

Make sure the folder of the logfile(s) is created in advance and with the necessary permissions
References

domain: the main domain the certificate is being created for. Generally this does not include a subdomain (eg: example.com instead of www.example.com)

alternatives: a comma-separated list of domains also to be supported by the certificate. Note: it should **not** contain the domain of the previous setting !. Some examples:
www.example.com,*example.com (for a wildcard certificate)

certificate_c/st/l/o/ou: these fields provide some information on you/your company. Any value should do but it is advised to avoid any special character (or space)

password: the generated .jks certificate will be protected by the given password. When you load the certificate in Jetty this password should be provided.

Renewing a certificate:

Run the following command

```
java -jar AutoGenerator.jar -mode:renew -config:<full path of your config file>
```

Make sure you use the same config file as the one used to create the original certificate (or at least use the same keyfolder, as it contains the previously generated account.key and csr file, which are being reused when renew a certificate)

Troubleshooting

In case of issues; have a look at the logfile: the location is the one specified in the logfile setting of the config file.

acme_client.jar also has its own log file, which can be found in `c:\var\log\acme` on Windows PC's and `/var/log/acme` on Linux machines.