MindWorks UK

Technology Transfer Partners

Let's Encrypt

Let's Encrypt

Free SSL/TLS certificates and automatic renewals

A brief introduction to the subject

Joe Doupnik
Prof (ret.), Univ of Oxford
jrd@netlab1.net
jdoupnik@microfocus.com
MindworksUK and Micro Focus

Caution: many words within

UNIVERSITY OF OXFORD

Updated 31 Aug 2021

MICRO FOCUS

Home base https://letsencrypt.org

"Let's Encrypt is **a free, automated**, and open certificate authority (CA), run for the public's benefit.
It is a service provided by the Internet Security Research Group (ISRG)."

Enticements are *Free* & *Automation*

An advert

Let the user be wary.
All may not be so simple.

- No programming or command line knowledge required.
- No root access required.
- It works with shared hosting.
- Install and configure the app using a browser.
- Admin dashboard is mobile responsive.
- Completely FREE of Cost!!

Introduction

- Appealing aspects in this matter are **free** certificates plus **automated** installation and renewal. The price is learning and internal complexity.

- Certificate issuance requires the vendor check that we control the DNS name, else bad guys could easily abuse the DNS system. Certs hold host name(s), public key, issuer's signature and more. Verification is important.
- Let's Encrypt employs automated steps for host name checking and cert issuance, and we do the local work to make the scheme function.
- Let's Encrypt certs are valid for short intervals, say 60-90 days, due to the simple user verification. Renewals can be automated, though we may need to step in. Thus we need to be attentive about renewals.

- Let's Encrypt created a rules based process for this, named **ACME**. It is defined in RFC 8555. On the wire it uses JSON Web Signature (JWS) holding nested structures of *name:value* expressions. Plus other details.

What is ACME?

Automated **C**ertificate **M**anagement **E**nvironment

Created by Let's Encrypt, defined in RFC 8555 (95 dense pages)

1. User installs an ACME ***agent*** to prepare a Certificate Signing Request (CSR) and generate its PKI public and private keys. User supplies answers.

1. Agent sends the Cert Signing Request to the Vendor (say Let's Encrypt).

2. Vendor issues a ***token*** to the Agent. User or Agent installs it publicly.

3. Agent then requests a certificate from the Vendor.

4. Vendor checks user's control of the DNS domain via presence of the *token* in either a) host's DNS record or b) host's TCP port 80 web server.

5. After a successful check the Agent receives the certificate from the Vendor.

6. Agent usually attempts automatic certificate installation and tends renewals.

ACME protocol two methods of token exposure

1. The *token* can be stored at this particular URI in client's web server which **must** remain web accessible using TCP port 80:

   http://example.com/**.well-known/acme-challenge**/*token*

2. The token can be stated in the host's DNS record as item:

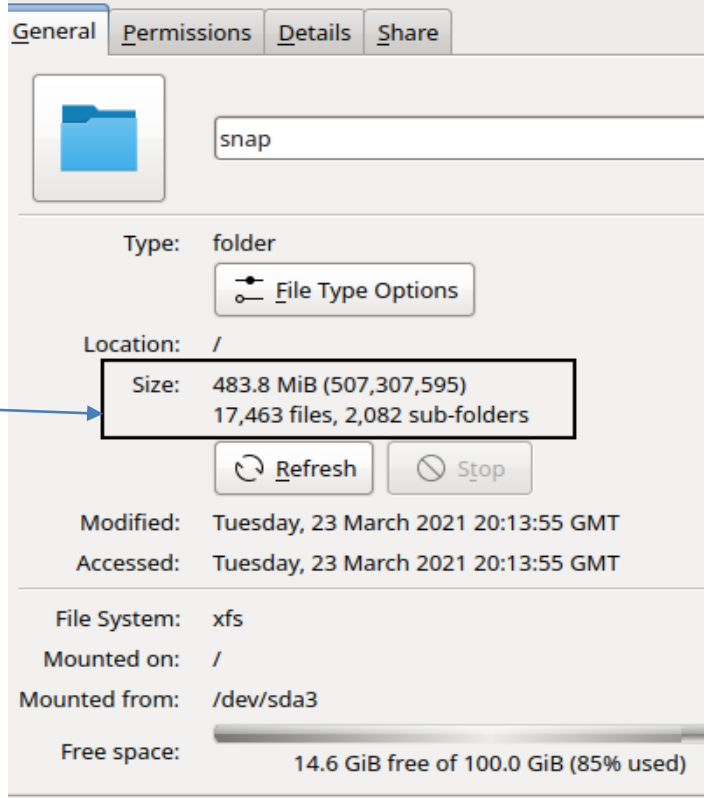
   _acme-challenge.example.com 3600 IN TXT "*token*"

- These are the *HTTP-01 Challenge* and *DNS-01 Challenge* methods.
- *.well-known/acme-challenge/* and *_acme-challenge* are fixed spellings.
- Wild card certificates typically require the *DNS-01 Challenge* method.
- Token form is -87YKoj3sQZB4rVCMZTiifl9QJKYm2eYYymAkpE0zBo
- The vendor (and world) can retrieve the token. No shared secret.

ACME Agents

"For Your Eyes Only"
I cannot watch

- There are dozens of them, mostly on github. See list at https://letsencrypt.org/docs/client-options/
- Creating them produced a programmer's bazaar, goals and quality vary.
- Let's Encrypt prefers the ***certbot*** agent from https://certbot.eff.org/. It is a 483MB diffuse Python app, in /snap and elsewhere.
- Most are small (a few MB), including shell script and PHP agents such as *acme.sh*, *getssl*, *bacme, freessl.tech* etc.
- None is actually simple nor yet complete.



Let's Encrypt documentation's advice

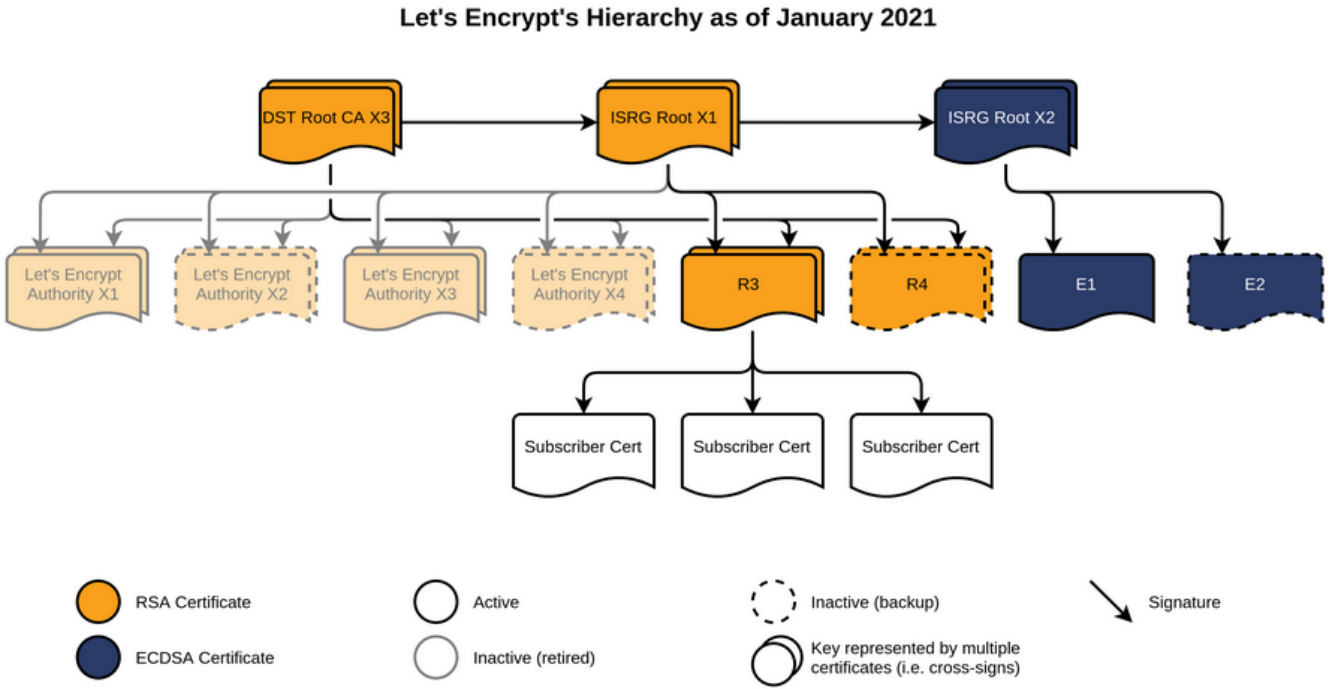
- If Certbot does not meet your needs, or you'd like to try something else, there are many more ACME clients to choose from. Once you've chosen ACME client software, see the documentation for that client to proceed.

- If you're experimenting with different ACME clients, use our staging environment to avoid hitting rate limits.

- The ACME URL for our ACME v2 staging environment is: https://acme-staging-v02.api.letsencrypt.org/directory

- List of CA root and intermediary certificates is in https://letsencrypt.org/certificates/

Let's Encrypt root CA and intermediaries



From https://letsencrypt.org/images/isrg-hierarchy.png

Linux Trusted Root area /etc/ssl/certs ships with file ISRG_ROOT_**X1**.pem
Certificate issuance supplies file *ca.cer* for chain intermediary **R3**.

ACME agent job description

- After account registration, Certificate Signing Request creation and certificate acquisition the next challenge is post-renewal style restarting of applications which use the certificate. There is often more than just a web server.

- Agent may try to communicate with its list of DNS servers to add the TXT record. Ours may not be on its list, plus its use of our credentials.

- Agent may interact with many applications, such as web server, mail (Postfix, Dovecot), SSH, FTP, Micro Focus apps, and more. A needed step is place certificates where apps expect them and restart the apps. Agents do try to help, but this remains a tricky matter.

- Agent activation to test for renewals and revocation can be via a cron entry or be a constantly running job. *Certbot* uses constantly running jobs, via systemd. Many agents use cron to awaken briefly once daily.

What we enter into a Certificate Signing Request, one way or another

From Wikipedia.org:

DN[1]	Information	Description	Sample
CN	Common Name	This is fully qualified domain name that you wish to secure	*.wikipedia.org
O	Organization Name	Usually the legal name of a company or entity and should include any suffixes such as Ltd., Inc., or Corp.	Wikimedia Foundation, Inc.
OU	Organizational Unit	Internal organization department/division name	IT
L	Locality	Town, city, village, etc. name	San Francisco
ST	State	Province, region, county or state. This should not be abbreviated (e.g. West Sussex, Normandy, New Jersey).	California
C	Country	The two-letter ISO code for the country where your organization is located	US
EMAIL	Email Address	The organization contact, usually of the certificate administrator or IT department	

An Openssl command which generates a CSR and keys:

openssl req -newkey rsa:2048 -keyout private.key -out mycsr.csr

See also https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs

Generating a CSR manually, using openssl

shhh, optional secret

prompts for our 7 details

none today, thanks

```
# openssl req -newkey rsa:2048 -keyout private.key -out mycsr.csr
Generating a 2048 bit RSA private key
.....................................+++++
....+++++
writing new private key to 'private.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:GB
State or Province Name (full name) [Some-State]:Oxfordshire
Locality Name (eg, city) []:Oxford
Organization Name (eg, company) [Internet Widgits Pty Ltd]:example.com
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:www.example.com
Email Address []:postmaster@example.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Viewing decoded contents of the CSR

```
# openssl req -text -noout -verify -in mycsr.csr
verify OK
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=GB, ST=Oxfordshire, L=Oxford, O=example.com, OU=IT, CN=www.example.com/emailAddress=post
master@example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:ba:22:ae:8b:0a:55:55:98:4e:99:94:6f:f6:72:
                    01:01:24:a9:32:50:84:69:c8:cf:e8:9a:84:9c:fe:
                    61:19:8b:7f:c1:23:31:3a:8d:b0:09:16:3c:e8:a0:
                    b4:c6:7d:40:70:16:2d:83:ce:e1:c5:00:43:c2:a4:
                    5f:ba:ae:90:09:4b:57:d0:dc:3b:4e:df:51:75:68:
                    82:fd:0d:b9:87:3b:2e:8c:75:26:42:b8:63:5d:74:
                    5e:89:68:05:44:75:f3:5a:9a:63:03:d5:66:ac:7a:
                    f1:df:01:54:b5:bf:21:9c:1b:23:f5:25:bc:7e:0e:
                    f6:70:3f:a7:a1:44:40:94:d7:31:01:0e:81:22:b1:
                    92:39:4e:e2:25:2d:3e:7c:7c:e0:89:73:c7:11:2f:
                    e4:0c:75:e2:31:27:f7:62:f0:83:cf:0b:9d:0c:c9:
                    92:7b:a1:ad:05:f1:cd:bf:53:63:f9:e4:3f:06:df:
                    fc:c5:d7:eb:44:c3:c1:78:77:14:85:3e:3d:22:44:
                    bf:df:22:05:06:43:10:53:e5:db:cb:00:5a:fa:47:
                    12:46:9c:d2:21:dc:1a:ec:11:e1:11:3a:f6:f1:2a:
                    42:cb:b6:ab:eb:8e:2a:bb:d6:de:b2:9d:fe:a1:b8:
                    45:54:94:55:81:21:aa:ce:58:52:9d:b0:d0:3d:10:
                    d1:55
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha256WithRSAEncryption
         31:bf:2a:69:29:2d:e6:77:d3:ff:34:26:e5:ad:f0:d2:7a:b6:
         65:90:86:93:c7:89:67:4f:6a:1b:0b:e7:c5:85:02:35:d7:91:
```

Our 7 details

This is our public key

Issuer‘s signature

ACME agents a preface to my experiments

There are tens of agents, written by a wide range of people, in various programming languages, with various goals.

They fetch certs using protocol ACME, then do more in their own ways. This is admirable work. Not simple, yet free.

Wanting automation to configure our machines reveals how complicated that goal can be.

What follows is choosing a general and interesting agent, Bash script *acme.sh*, followed by a few of its operations to illustrate the kinds of detailed work we encounter with many agents.

Note that many agents restrict themselves to web server applications, while acme.sh can deal with more kinds.

ACME agent acme.sh (bash script, 1MB)

Quoting from https://github.com/acmesh-official/acme.sh:

- An ACME protocol client written purely in Shell (Unix shell) language.
- Full ACME protocol implementation.
- Support ACME v1 and ACME v2
- Support ACME v2 wildcard certs
- Simple, powerful and very easy to use. You only need 3 minutes to learn it.
- Bash, dash and sh compatible.
- Purely written in Shell with no dependencies on python or the official Let's Encrypt client.
- Just one script to issue, renew and install your certificates automatically.
- DOES NOT require root/sudoer access.
- Docker friendly
- IPv6 support
- Cron job notifications for renewal or error etc.

It's probably the easiest & smartest shell script to automatically issue & renew the free certificates from Let's Encrypt.

See also: https://www.howtoforge.com/getting-started-with-acmesh-lets-encrypt-client

Before plunging in

- *acme.sh* and most agents have long lists of commands and parameters, meaning complexity and needing situational context.

  We need to study the agent's documentation to know what to do.

  Docs are often too brief about concepts, details, what is really occurring, and what is changed in our system. Informing vs robotics.

  Thus we have puzzles, and later we may need to repeat this exercise if troubles ensue.

- *Certbot* requires we add a new repo to fetch packager *snapd* (like rpm packaging, see *snapd* in wikipedia.org) plus fetch more files. For us that is project *snappy* from https://download.openSUSE.org/repositories/system:/snappy/
- ACME.sh has a problem: https://community.letsencrypt.org/t/the-acme-sh-will-change-default-ca-to-zerossl-on-august-1st-2021/144052/36

- Ensure DNS lookups for the host name(s) yield a correct IP number.
- Let's Encrypt provides a test/staging service to help us experiment.

Agent acme.sh commands & parameters

From https://github.com/acmesh-official/acme.sh/wiki/Options-and-Params

```
Usage: acme.sh <command> ... [parameters ...]
Commands:
  -h, --help               Show this help message.
  -v, --version            Show version info.
  --install                Install acme.sh to your system.
  --uninstall              Uninstall acme.sh, and uninstall the cron job.
  --upgrade                Upgrade acme.sh to the latest code from https://github.com/acmesh-official/a
  --issue                  Issue a cert.
  --deploy                 Deploy the cert to your server.
  -i, --install-cert       Install the issued cert to apache/nginx or any other server.
  -r, --renew              Renew a cert.
  --renew-all              Renew all the certs.
  --revoke                 Revoke a cert.
  --remove                 Remove the cert from list of certs known to acme.sh.
  --list                   List all the certs.
  --to-pkcs12              Export the certificate and key to a pfx file.
  --to-pkcs8               Convert to pkcs8 format.
  --sign-csr               Issue a cert from an existing csr.
  --show-csr               Show the content of a csr.
  -ccr, --create-csr       Create CSR, professional use.
  --create-domain-key      Create an domain private key, professional use.
  --update-account         Update account info.
  --register-account       Register account key.
  --deactivate-account     Deactivate the account.
  --create-account-key     Create an account private key, professional use.
  --install-cronjob        Install the cron job to renew certs, you don't need to call this.
                           The 'install' command can automatically install the cron job.
  --uninstall-cronjob      Uninstall the cron job. The 'uninstall' command can do this automatically.
  --cron                   Run cron job to renew all the certs.
  --set-notify             Set the cron notification hook, level or mode.
  --deactivate             Deactivate the domain authz, professional use.
  --set-default-ca         Used with '--server', to set the default CA to use to use.

Parameters:
  -d, --domain <domain.tld>         Specifies a domain, used to issue, renew or revoke etc.
  --challenge-alias <domain.tld>    The challenge domain alias for DNS alias mode.
                                    See: https://github.com/acmesh-official/acme.sh/wiki/DNS-alias-mode
```

And so on for ~120 lines. Many interesting items.

Missing: which parameters go with which commands, and workflow control.

Best to review the entire list to gather tips.

Agent acme.sh acquisition plus installation

Fetch acme.sh from https://github.com/acmesh-official/acme.sh

```
cd acme.sh
./acme.sh --install  \
    --home ~/myacme \
    --config-home ~/myacme/data \
    --cert-home  ~/mycerts \
    --accountemail  "my@example.com" \
    --accountkey  ~/myaccount.key \
    --accountconf ~/myaccount.conf \
    --useragent  "this is my client.“
```

“You don't need to set them all, just set those ones you care about.”

Explanations are shown in the next slide.

acme.sh installation command line options

--home is a customized dir to install acme.sh into.

default is ~/.acme.sh

--config-home is a writable folder, acme.sh will write all the files

(including cert/keys, configs) there. default is --home

--cert-home is a customized ***dir to save the certs***.

default is --config-home

--accountemail is the email used to register an account to Let's Encrypt,

you will receive a ***renewal notice email*** here.

--accountkey is the file saving your account ***private key***.

default is --config-home

--user-agent is the user-agent header value sent to Let's Encrypt.

Items above are important for using acme.sh. For others please review the wiki Options & Parameters section and see output of acme.sh -h

acme.sh about supported ACME servers

From its wiki page:

- For the **--server** parameter, you can specify an ACME server URL, and you can also use a short friendly name for known CAs.
- The supported short names are:

Short Name	ACME server URL
letsencrypt	https://acme-v02.api.letsencrypt.org/directory
letsencrypt_test	https://acme-staging-v02.api.letsencrypt.org/directory
buypass	https://api.buypass.com/acme/directory
buypass_test	https://api.test4.buypass.no/acme/directory
zerossl	https://acme.zerossl.com/v2/DV90

The short name will be treated as the same as the URL.

For now, the default CA is letsencrypt. If you want to use another CA, you need to specify **--server** for each command.

For example, if your want to use zerossl CA :

acme.sh **--register-account --server zerossl** -m myemail@example.com

acme.sh exposing the token via DNS

This example cites three names in the certificate, thus uses **SAN** Server Alternative Name style. Do remember SAN. From its documentation Readme.md file:

If your dns provider doesn't support any api access, you can add the Txt record by hand.

bash acme.sh --issue --dns -d example.com -d www.example.com -d cp.example.com

You should get an output like below

Add the following txt record:
Domain:_acme-challenge.example.com
Txt value:9ihDbjYfTExAYeDs4DBUeuTo18KBzwvTEjUnSwd32-c
etc

A proper TXT record looks like this, with required TTL & Internet class:

_acme-challenge.example.com 3600 IN TXT “9ihDbjYfTE…wd32-c”

acme.sh deploy cert files to applications

From its documentation Readme.md file, and wiki about *deployhooks*

Apache example:

```
bash acme.sh --install-cert -d example.com \
--cert-file      /path/to/certfile/in/apache/cert.pem  \
--key-file       /path/to/keyfile/in/apache/key.pem  \
--fullchain-file /path/to/fullchain/certfile/apache/fullchain.pem \
--reloadcmd     "service apache2 force-reload“
```

my notes**:**

-d is certificate host DNS name

--*-file tell the agent (not the app) where to store a cert after --**issue** or --**renew**

Templates are in subdir *deploy,* including shell scripts

acme.sh deploy cert files to applications

A complicated situation about using that previous application “hook”.

From https://github.com/acmesh-official/acme.sh/wiki/deployhooks:

> “Those hooks are only accepted by the **--issue** command, but will be saved and apply to **--renew** or **--cron** commands as well.”

Saved where? May I edit that list? This is puzzling. Is there a prize?

Hmmm, **reloadcmd** can do anything we wish, which is our route to sanity. Such as have it run a simple shell script which then tends to all applications one by one.

Many agents seem to avoid stating that users need to manually configure each app about the place of its certs. Fundamentally the hooks inform acme.sh which apps need automatic reloading.

See also: https://github.com/acmesh-official/acme.sh/wiki/Using-pre-hook-post-hook-renew-hook-reloadcmd

acme.sh certificate manual issuance

Two approaches (again three names in the certificate), from its wiki doc
https://github.com/acmesh-official/acme.sh/wiki/How-to-issue-a-cert.
Here acme.sh is internally using curl or wget to speak to the vendor.

1. Requires TCP port 80 be free:
   bash acme.sh --issue --standalone -d example.com -d www.example.com -d cp.example.com

or

2. Requires TCP port 443 be free: (note this uses the TLS-ALPN-01 Challenge)
   bash acme.sh --issue --alpn -d example.com -d www.example.com -d cp.example.com

Certs will be placed in ~/.acme.sh/example.com/ (see installation options for **another location**)
Certs will be **renewed** automatically every **60** days.
The --**install-cert** command is needed to tell acme.sh which app to restart and where to place it's certs, one such command for each application or an equivalent script.
Cron entry is in **/var/spool/cron/tabs/*username*** such as root. Check its timing value for your site's convenience.

acme.sh “stateless” mode experiment

This test required little effort. Stateless = steps are independent, no shared info. Described in the acme.sh wiki (best to review, scroll to Stateless). Steps are only summarized here.

1. Register an account and obtain a *thumbprint* string reply

   bash acme.sh --register-account

2. Create sub-directory **.well-known/acme-challenge** in server’s DocRoot.
3. Create file *index.php* in that acme-challenge subdir, 5 lines including these

   $token = array_pop(explode(‘/’, $_SERVER[‘REQUEST_URI’]));

   echo “$token.*thumbprint*”;
4. Add *index.php* to the server’s configuration statement DirectoryIndex.
5. Add clause <Directory “YourDocRoot/.well-known/acme-challenge”> holding

   RewriteEngine on

   RewriteRule “[-_a-zA-Z0-9]+$” “index.php”

   This ought to have IP restrictions, but what are the Vendor’s numbers?
6. Finally execute command

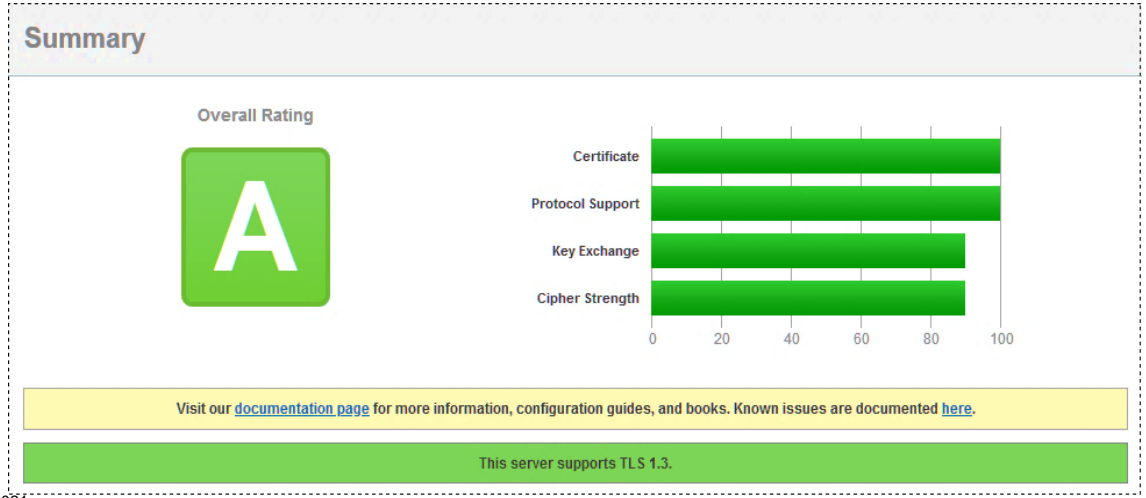
   bash acme.sh --issue -d example.com --stateless (no app handling)

acme.sh “stateless” mode experiment

- The result will be a handful of files: **ca.cer**, **fullchain.cer**, plus host.name with extensions **.cer**, **.conf**, **.csr**, **.csr.conf, .key**, all placed in directory /root/.acme.sh or in your home directory or that given by installation item **--cert-home**.
- By hand I modified file /etc/apache2/vhosts.d/vhost-ssl.conf to use:

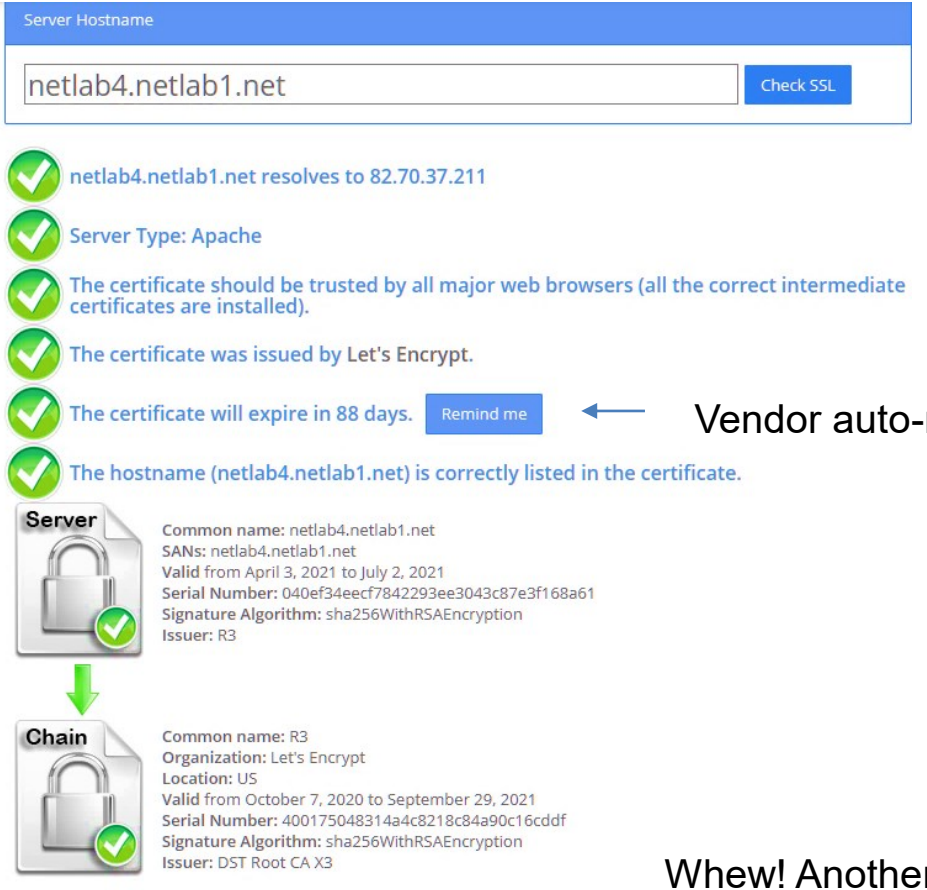
  SSLCertificateFile=path/**host.name.cer** SSLCertificateChainFile=path/**ca.cer**

  SSLCertificateKeyFile=path/**host.name.key** SSLCACertificatePath=/etc/ssl/certs
- Restart Apache and test via https://ssllabs.com. The report (top part) is



I think we passed this test.
Done on SuSE Leap 15.2.

Pop quiz from https://www.sslshopper.com/



Vendor auto-reminder feature

Whew! Another test has been passed.

Test Postfix with the new certificate

In file /etc/postfix/main.cf use these certificate settings:

smtpd_tls_cert = pathto/**fullchain.cer**
smtpd_tls_key = pathto/**host.name.key**
smtpd_tls_CApath = /etc/ssl/certs (trusted root store, normal setting)

Note: file *fullchain.cer* turns out to be the concatenation of the main cert file *host.name.cer* followed by the intermediary's file *ca.cer*. It is part of the bundle fetched by acme.sh.

Test via https://checktls.com:

"If you email protected information, such as information subject to HIPAA, GDPR, or PCI, you must make sure the email is encrypted."

MX Server	Pref	Answer	Connect	HELO	TLS	Cert	Secure	From	MTA-STS	DANE	Score
netlab4.netlab1.net [82.70.37.211:25]	10	OK (87ms)	OK (87ms)	OK (87ms)	OK (86ms)	OK (620ms)	OK (89ms)	OK (134ms)	no policy	no TLSA	114.00
Average		100%	100%	100%	100%	100%	100%	100%			114

See also: https://netlab1.net/long-term/POSIX-email-TLSv1.2.pdf for discussion about configuring TLS support for Postfix and Dovecot.

Test Dovecot (IMAP4 support)

Dovecot configuration file 10-ssl.conf changed to hold
ssl_crt = <pathto/**fullchain.cer**
ssl_key = <pathto/**host.name.key**

```
# openssl s_client -connect netlab4.netlab1.net:143 -starttls imap
CONNECTED(00000003)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = R3
verify return:1
depth=0 CN = netlab4.netlab1.net
verify return:1
---
Certificate chain
 0 s:/CN=netlab4.netlab1.net
   i:/C=US/O=Let's Encrypt/CN=R3
 1 s:/C=US/O=Let's Encrypt/CN=R3
   i:/O=Digital Signature Trust Co./CN=DST Root CA X3
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFLTCCBBWgAwIBAgISBA7zTuz3hCKT7jBDyH4/FophMA0GCSqGSIb3DQEBCwUA
MDIxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MQswCQYDVQQD
EwJSMzAeFw0yMTA0MDMxMjI5NDBaFw0yMTA3MDIxMjI5NDBaMB4xHDAaBgNVBAMT
E25ldGxhYjQubmV0bGFiMS5uZXQwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDAKm+yycZJhYa8tnHVhlwfeW0rjLpkoFb6aJDjAJEkxAu73rW9jPj0SDgE
```

Looks good, the new certs were found and used correctly.

Test conclusions

- The stateless test was fun. However, once we become serious then we ask what occurs for renewal events.

- There ought to be an explicit list of certificate-using applications which need restarting after a renewal. This part is a bit confusing.
- There is knowing when app restarting will/did occur. No surprises. Email can be helpful, if it is still working afterward.
- Emphasis is on Apache and Nginx web servers, yet the acme.sh source code subdirectory “deploy” lists many other applications. We do have many applications, automating them is somewhat confusing; note my sanity hint.
- Cerbot documentation is not much better about this. It focuses on web servers, and certbot is a truly huge assembly (483MB, 17K files).

- Documentation of many ACME Agents needs much more work because we can become lost in a dark forest of complicated details.

Apache web server new module mod_md

Description:	Managing domains across virtual hosts, certificate provisioning via the ACME protocol
Status:	Experimental
Module Identifier:	md_module
Source File:	mod_md.c
Compatibility:	Available in version 2.4.30 and later

OES 2018 uses v2.4.23
Leap 15.2 uses v2.4.43

"This module manages common properties of domains for one or more virtual hosts. Its serves two main purposes: for one, supervise/renew https: certificates via the ACME protocol (RFC 8555). Certificates will be renewed by the module ahead of their expiration to account for disruption in internet services. There are ways to monitor the status of all certificates managed this way and configurations that will run your own notification commands on renewal, expiration and errors."

"The default ACME Authority for managing certificates is Let's Encrypt, but it is possible to configure another CA that supports the protocol.

Wildcard certificates are possible, but not straight-forward to use out of the box. Let's Encrypt requires the `dns-01` challenge verification for those. No other is considered good enough."

Thoughtful design is apparent

Apache web server mod_md doc snippets

As with the HTTP/2 protocol, to allow this, you configure:

```
Protocols h2 http/1.1 acme-tls/1
```

And the `tls-alpn-01` challenge type is available.

TLS in a VirtualHost context

```
MDomain example.org

<VirtualHost *:443>
    ServerName example.org
    DocumentRoot htdocs/a

    SSLEngine on
    # no certificates specification
</VirtualHost>
```

Example

```
MDomain example.org

<VirtualHost *:443>
    ServerName example.org
    ServerAlias www.example.org
    DocumentRoot htdocs/root

    SSLEngine on
</VirtualHost>

MDomain example2.org auto

<VirtualHost *:443>
    ServerName example2.org
    ServerAlias www.example2.org
    ...
</VirtualHost>
```

If the validity of the certificate falls below duration, mod_md will get a new signed certificate.

Normally, certificates are valid for around 90 days and mod_md will renew them the earliest 33% of their complete lifetime before they expire (so for 90 days validity, 30 days before it expires). If you think this is not what you need, you can specify either the exact time, as in:

Example

```
# 21 days before expiry
MDRenewWindow 21d
# 30 seconds (might be close)
MDRenewWindow 30s
# 10% of the cert lifetime
MDRenewWindow 10%
```

When in auto drive mode, the module will check every 12 hours at least what the status of the managed domains is and if it needs to do something. On errors, for example when the CA is unreachable, it will initially retry after some seconds. Should that continue to fail, it will back off to a maximum interval of hourly checks.

Notice control of the renewal grace time

My conclusions

- The many agent projects spawned by Let's Encrypt need to mature. Their ACME parts are fine, but their systems integration varies greatly. Thus we have puzzles about choice of agent, its configuration and whether it works at renewal times.
- Our time and effort exploring and testing agents reveals significant complexity and many unknowns. This does not build confidence that automation will be robust.
- Agent documentation in general needs major improvement.
- The typical 60 day cert expiry interval means six renewals per year, thus many chances for application failure.

- Free this is, yet the automation puts us to work. Siri, Alexa: Help!

- An alternative is purchase ($ £ € ₹ R) a commercial certificate, store it in say /etc/ssl/servercerts/vendor-name, point application configs there, then renew it every year or two. The vendor will likely solicit renewals. No agent, no public token. This is simpler and clearer than present automation techniques. Do recall SAN, many names.

MindWorks Inc. Ltd
210 Burnley Road
Weir
Bacup
OL13 8QE UK

Telephone: +44 (0) 170 687 1900
Fax: +44 (0) 170 687 8203
Web: www.mindworksuk.com
Email: training@mindworksuk.com