LetsEncrypt

Acme.sh client installation

Automatic installation

Automatic installation can be done by either of the two following commands:

```
curl https://get.acme.sh | sh -s email=my@example.com
wget -O -  https://get.acme.sh | sh -s email=my@example.com
```

The automatic installer will perform 3 actions:

1. Create a acme.sh copy in home dir ($HOME): ~/.acme.sh/. All certs will be placed in this folder too.
2. Create alias for: acme.sh=~/.acme.sh/acme.sh.
3. Create daily cron job to check and renew the certs if needed.

Advanced installation

Advanced Installation see also: How-to-install. Download the installation script:

```
wget https://github.com/acmesh-official/acme.sh/archive/master.zip
unzip master.zip
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 :

1. –home is a customized dir to install acme.sh in. By default, it installs into ~/.acme.sh
2. –config-home is a writable folder, acme.sh will write all the files(including cert/keys, configs) there. By default, it's in –home
3. –cert-home is a customized dir to save the certs you issue. By default, it's saved in –config-home.
4. –accountemail is the email used to register an account to Let's Encrypt, you will receive a renewal notice email here.
5. –accountkey is the file saving your account private key. By default, it's saved in –config-home.
6. –user-agent is the user-agent header value used to send to Let's Encrypt.

My preferred home location is: /home/www-data/acme

Create Diffie-Hellmann Parameters

Diffie-Hellman is an algorithm used to establish a shared secret between two parties. It is primarily used as a method of exchanging cryptography keys for use in symmetric encryption algorithms like AES. The algorithm in itself is very simple. Diffie-Hellman is a way of generating a shared secret between two people in such a way that the secret can't be seen by observing the communication. That's an important distinction: You're not sharing information during the key exchange, you're creating a key together.

```
mkdir /path-to/acme/cert/dhparam
openssl dhparam -out /path-to/acme/cert/dhparam/dhparam.pem 4096
```

Issue a Certificate - Webroot Mode

Example: Single domain

```
acme.sh --issue -d example.com -w /home/wwwroot/example.com
```

Example: Multiple domains with the same cert.

```
acme.sh --issue -d example.com -d www.example.com -d cp.example.com -w
/home/wwwroot/example.com
```

The second argument “example.com” is the main domain you want to issue the cert for. You must have at least one domain there. The third argument /home/wwwroot/example.com or /home/username/public_html or /var/www/html is the web root folder where you host your website files. You **MUST** have **write access** to this folder.

You must point and bind all the domains to the same webroot dir: /home/wwwroot/example.com.

The certs will be placed in ~/.acme.sh/example.com/ The certs will be renewed automatically every 60 days.

More examples: How-to-issue-a-cert

Install certs to Nginx Manual

Edit the config files in /etc/nginx/sites-available:

```
ssl_certificate         /path-to/acme/cert/domain.com/domain.com.cer;
ssl_certificate_key     /path-to/acme/cert/domain.[Unit]
ssl_dhparam             /path-to/acme/cert/dhparam/dhparam.pem;
```

Install cert to Nginx Script

After the cert is generated, you probably want to install/copy the cert to your Nginx server. You MUST use this command to copy the certs to the target files, DO NOT use the certs files in ~/.acme.sh/ folder, they are for internal use only, the folder structure may change in the future.

Nginx example:

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

Only the domain is required, all the other parameters are optional.

The ownership and permission info of existing files are preserved. You can pre-create the files to define the ownership and permission.

Install/copy the cert/key to the production Apache or Nginx path.

The cert will be renewed every 60 days by default (which is configurable). Once the cert is renewed, the Apache/Nginx service will be reloaded automatically by the command: service apache2 force-reload or service nginx force-reload.

Please take care: The reloadcmd is very important. The cert can be automatically renewed, but, without a correct 'reloadcmd' the cert may not be flushed to your server(like nginx or apache), then your website will not be able to show renewed cert in 60 days. /lib/systemd/system

Issue a Certificate - Nginx Mode

If you are running a web server, it is recommended to use the Webroot mode. **(requires you to be root/sudoer, since it is required to interact with Nginx server)**

Particularly, if you are running an nginx server, you can use nginx mode instead. This mode doesn't write any files to your web root folder.

Just set string “nginx” as the second argument. It will configure nginx server automatically to verify the domain and then restore the nginx config to the original version.

So, the config is not changed.

```
acme.sh --issue --nginx -d example.com -d www.example.com -d cp.example.com
```

This nginx mode is only to issue the cert, it will not change your nginx config files. You will need to configure your website config files to use the cert by yourself. We don't want to mess with your nginx server, don't worry.

Issue Wildcard certificates

It's simple, just give a wildcard domain as the -d parameter.

```
acme.sh  --issue -d example.com  -d '*.example.com'  --dns dns_cf
```

How to renew the certs

No, you don't need to renew the certs manually. All the certs need to be renewed within 90 days. The renew command will take a look at all active certificates and renew those who are close to expiring - which is currently defined as 30 days before the expiration date. If your certificates aren’t due for renewal yet, the client won’t renew them.

```
acme.sh --renew -d example.com
```

You can also force to renew a cert:

```
acme.sh --renew -d example.com --force
```

How to stop cert renewal

To stop renewal of a cert, you can execute the following to remove the cert from the renewal list:

```
acme.sh --remove -d example.com [--ecc]
```

The cert/key file is not removed from the disk. You can remove the respective directory (e.g. ~/.acme.sh/example.com) by yourself.

Renewal with Systemd Timer

Certificates expire after 90 days, so we need to set up the auto-renewal. The renew command will take a look at all active certificates and renew those who are close to expiring - which is currently defined as 30 days before the expiration date. If your certificates aren’t due for renewal yet, the client won’t renew them. The reason why a daily cronjob is recommended is in order to avoid issues caused by service downtime on Let’s Encrypt’s end, or any issues your server might have. If you, for example, run the cronjob just once every month or every two months, and the service just happens to be down during those times, you’ll end up with an expired certificate eventually. By doing it daily instead, Let’s Encrypt would have to be down for 30 consecutive days for that to happen, which is rather unlikely.

There are two parts to this. The first is to create the systemd service file which when started will perform the renewal. Create this file at /lib/systemd/system/letsencrypt-renew.service and add the following content:

```
#nano /lib/systemd/system/letsencrypt-renew.service
-------------------------------------------------
[Unit]
Description=Renew LetsEncrypt certificates

[Service]
Type=oneshot
WorkingDirectory=/home/www-data/acme

#Make sure that we run the renewal script as user www-data
ExecStart=/bin/su -s /bin/bash -c '/home/www-data/acme/my-certs-renew.sh'
www-data
ExecStartPost=/bin/systemctl restart nginx.service
```

The second part of the auto-renewal is the systemd timer. Create the file at /lib/systemd/system/letsencrypt-renew.timer and add the following content:

```
#nano /lib/systemd/system/letsencrypt-renew.timer

[Unit]
Description=Periodic check for cert renewal

[Timer]
OnCalendar=*-*-* 06,18:09:00   #Every day at 06:09 and 18:09
OnCalendar=*-*-1,15 4:00:00    #Approximately every 2 weeks (1st and 15th of
the month) at 04:00
# Be kind to the Let's Encrypt servers: add a random delay of 0–3600 seconds
RandomizedDelaySec=3600
Persistent=true

[Install]
WantedBy=timers.target
```

The OnCalendar section determines how frequently this will run. It takes the format:

```
DayOfWeek Year-Month-Day Hour:Minute:Second
```

So in our example above we have it running every day of the week, every month, every year at both 6:09am and 6:09pm.

Next, we need to start and enable our timer:

```
# systemctl enable letsencrypt-renew.timer
# systemctl start letsencrypt-renew.timer
```

You can validate that your timer is running by running:

```
# systemctl list-timers
-----------------------
NEXT                         LEFT          LAST
```

```
PASSED    UNIT                         ACTIVATES
Fri 2018-02-16 18:00:00 UTC  2h 29min left Fri 2018-02-16 06:00:07 UTC  9h
ago    renew-letsencrypt.timer      renew-letsencrypt.service
```

Sudoers

The timer/service need to restart nginx, in order to load the new certificates. However, the update script is run as user 'www-data'. To allow www-data to run systemctl restart as root, add the following file to /etc/sudoers.d:

```
visudo /etc/sudoers.d/www-data
------------------------------
www-data ALL=(root) NOPASSWD: /bin/systemctl restart nginx
```

Links

- Letsencrypt How It Works
- ACME.sh client