Let’s Encrypt! Free certificates for everyone!

Nicola Corti

Gruppo Utenti Linux Pisa

03 February 2016

Let’s Encrypt! Free certificates for everyone!

Nicola Corti

Introduction

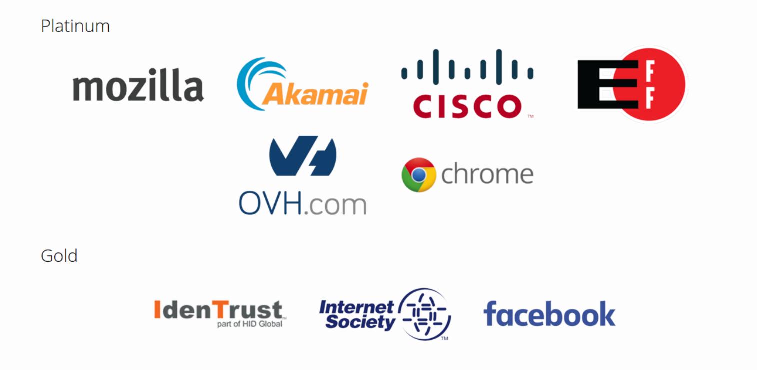
What is Let’s Encrypt?

Let’s Encrypt is a **Certification Authority (CA)** that issue **free** SSL/TLS certificates

- From 5 December 2015 L.E. is available in **Public Beta**
- L.E. has released more than **480 k** certificates
- It’s major focus is **automation** of processes.
- **https://letsencrypt.org/**

Sponsors

Who's behind the project?



History

2012	Project begins inside **Mozilla**
11-2014	Let’s Encrypt **publicly announced**
01-2015	The **ACME** protocol submitted to **IETF** for standardization
04-2015	**ISRG** (Internet Security Research Group) and **Linux Foundation** join the project
09-2015	Issued the first certificate for **helloworld.letsencrypt.org**
10-2015	L.E. intermediate certificate becomes ***cross-signed*** by IdenTrust. (*Let’s Encrypt is trusted!*)
12-2015	**Public beta!**

Why?

- Free
- Automated
- Open
- Secure
- Transparent

Server

Boulder

Architecture

The Let’s Encrypt system is based on **3 components**: a **server**, a **client** and the **protocol** that defines the communication rules between server and client

The server is called **Boulder** and it’s completely written in **Go**. It’s responsible of handling all the procedures for **issuing**, **renewal** and **revocation** of certificates.

It’s basically an HTTPS server that exposes a **RESTful** interface.

https://github.com/letsencrypt/boulder

Client

Let’s Encrypt! Free certificates for everyone!

Nicola Corti

Intro

- Sponsors
- History
- Why?

Server

Client

Protocol

- Security
- Challenges
- The protocol
- Cert revocation

Certs

- DV
- Cross Signing

Setup

- Requirements
- Download
- Run
- Plugins
- Revoke
- Renewal
- Update
- Screens

Conclusions

- Drawbacks
- Resources

letsencrypt

The client is called (obviously) **letsencrypt** and it's completely written in **Python**. It's responsible for interaction with the remote server and it **handles your certificates**.

- Download through .deb package **letsencrypt** (only on debian *sid/stretch*).
- Clone the **git repository**.

https://github.com/letsencrypt/letsencrypt

Plugins

The client’s main purpose is to **simplify and automate** the whole process of authentication and creation of the certificate.

For this reason the client comes with several plugins, useful to automatically setup the new certificates on popular web servers: **apache** and **nginx**.

Protocol

Let's Encrypt! Free certificates for everyone!

Nicola Corti

Intro

- Sponsors
- History
- Why?

Server

Client

Protocol

- Security
- Challenges
- The protocol
- Cert revocation

Certs

- DV
- Cross Signing

Setup

- Requirements
- Download
- Run
- Plugins
- Revoke
- Renewal
- Update
- Screens

Conclusions

- Drawbacks
- Resources

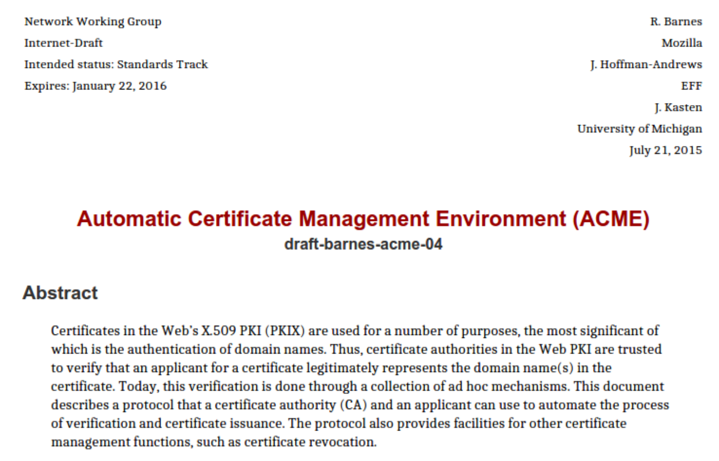
ACME

The procol use by Let’s Encrypt is called **Automated Certificate Management Environment (ACME)**.

ACME is based on exchanges of signed **JSON** files (a.k.a. **JWS, Json Web Signature**). These documents contains all the requests and the responses between the client and the server.

These documents **must** be exchanged over **HTTPS**.

ACME



Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 22, 2016

R. Barnes
Mozilla
J. Hoffman-Andrews
EFF
J. Kasten
University of Michigan
July 21, 2015

Automatic Certificate Management Environment (ACME)

draft-barnes-acme-04

Abstract

Certificates in the Web's X.509 PKI (PKIX) are used for a number of purposes, the most significant of which is the authentication of domain names. Thus, certificate authorities in the Web PKI are trusted to verify that an applicant for a certificate legitimately represents the domain name(s) in the certificate. Today, this verification is done through a collection of ad hoc mechanisms. This document describes a protocol that a certificate authority (CA) and an applicant can use to automate the process of verification and certificate issuance. The protocol also provides facilities for other certificate management functions, such as certificate revocation.

https://github.com/letsencrypt/acme-spec

ACME

The ACME protocol is aimed to:

1. **Prove** that we are the owners of a specific domain, say *example.com*
2. **Obtain** a new certificate for the domain *example.com*
3. **Revoke** or **Renew** a certificate for the domain *example.com*

Security

All the interactions between client and servers are encrypted with a **public/private key pair** generated during the first execution of the client.

In order to **prove** that we are the owners of the domain, the server sends us a set of **challenges** that we must solve.

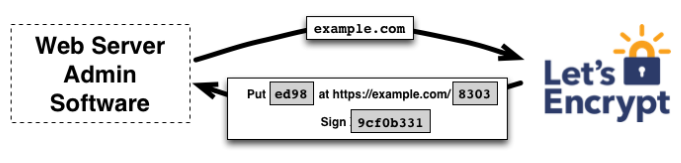
Every interaction with the server is marked with a **nonce** number that allows to avoid **Replay** attacks.

Challenges

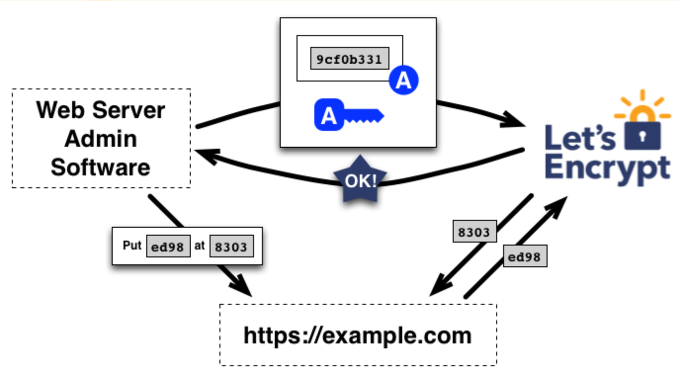
The server can decide to send one or more challenge from the followings:

Type	Description
Simple HTTP	You must place a **token file** inside your webserver root folder. Both HTTP and HTTPS are accepted
DNS	You must provide a token inside a **TXT record** of your DNS server
Proof of possession	You must **sign a document** using a keypair that the server already consider yours
Domain Validation with Server Name Indication	You must configure a **TLS server** on a specific IP address (through an A record inside the DNS).

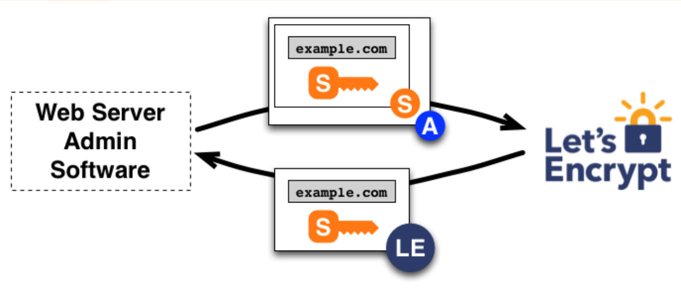
Domain Validation


Web Server Admin Software
example.com
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt

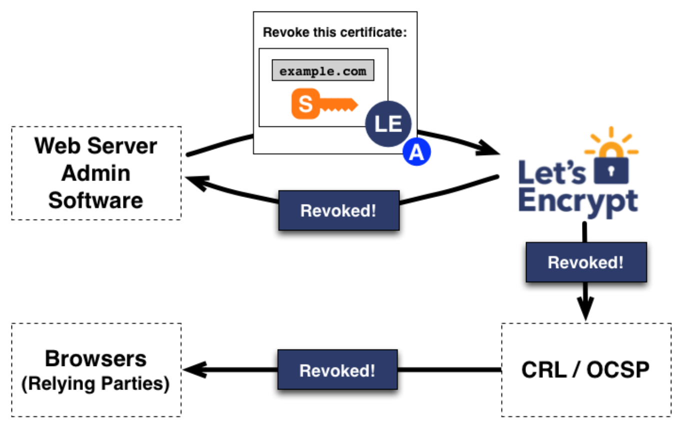
Domain Validation



Certificate Issuance



Certificate Revocation



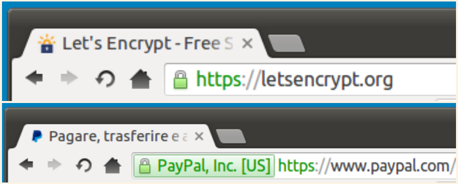
Certificates

DV Certificates

DV

All the certificates issue from L.E. are **Domain Validated** certificates. They basically prove that you are the owner of a specific domain, nothing more.

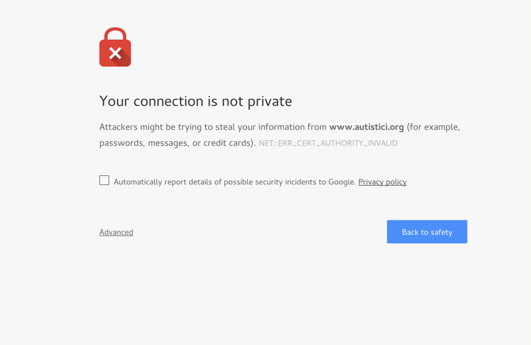
Organization Validation and **Extended Validation** certificates requires to explicit verify the identity of the subject that is requesting a certificate.



Cross Signing

All the issued certificates are *Cross-signed* by **IdenTrust**. In this way, all the L.E. certificates are trusted by major browsers.

We can avoid browser errors such as:

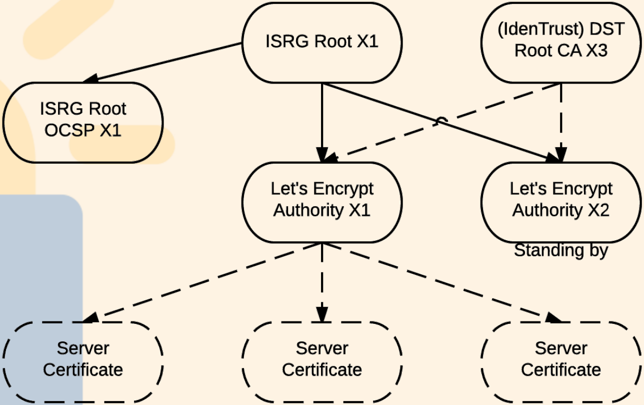


Your connection is not private

Attackers might be trying to steal your information from **www.autistici.org** (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

□ Automatically report details of possible security incidents to Google. Privacy policy

Cross Signing



Validity

All the certificates have a **90 days** validity. After the expiry, the certificates are not valid anymore and the browsers will raise security errors.

90-days validity

This is nothing new on the web. Having certificates with a reduced validity could help to limit damage from key compromise and mis-issuance.

You will receive *remind emails* whenever a certificate is near to expire. Certificate renewal can be automated with a **cron** task.

Setup

Requirements

The client minimal requirements are:

- **Unix like** system.
- **Python 2.6** or **2.7**.
- **root** rights on the system.

The **apache** setup plugins works only on Debian based system: **Ubuntu 12.04+** and **Debian 7+**

Download

First, let's download the **letsencrypt** client.

On Debian *sid/stretch*:

```
$ sudo apt-get install letsencrypt
$ letsencrypt --help
```

On other OS:

```
$ git clone https://github.com/letsencrypt/letsencrypt
$ cd letsencrypt
$ ./letsencrypt-auto --help
```

From now on we will use **letsencrypt-auto** for all the commands, assuming to proceed with clone of the repository.

Run

To execute the client you simply have to invoke:

```
$ ./letsencrypt-auto
```

We will be guided through the issuing process

Apache

If you want to automatically configure **Apache** with the generated certificates you can invoke:

```
$ ./letsencrypt-auto --apache -d example.com -d www.example.com
```

With **--apache** we are enabling the apache plugin, and with **-d** we are giving the list of involved domains.

Contacts

During the first run, the client will ask for our **mail address** and it will request to accept the **Terms of service**.

You can skip these steps using these flags:

```
$ ./letsencrypt-auto --email admin@example.com --agree-tos
```

Plugins

Plugin	A	I	Descrizione
Apache	Y	Y	Obtain and setup automatically the certs. on Apache 2.4 (Debian based)
Standalone	Y	N	Obtain the cert with a **standalone** web server on ports 80/443
Webroot	Y	N	Obtain a certificate *touching* a token inside the root folder of an already existing webserver
Manual	Y	N	Prints the commands to manually obtain the certs from a different client
Nginx	Y	Y	Obtain and setup automatically the certs. on nginx (experimental)

certonly

You can use the plugins with **authentication** support (A column) just to obtain the certificates **without installation**.

Simply add the option **certonly** to the command line.

Standalone example

```
$ ./letsencrypt-auto --standalone-supported-challenges \
 http-01 certonly -d example.com
```

This command will start a standalone webserver on port 80 and it will obtain the certificate for example.com

/etc/letsencrypt

All the certificates and the auth keys will be saved into **/etc/letsencrypt**.

Inside this folder you will find all the **certificates** and all the **public/private keys**. It’s **extremely** recommended to make a **backup** of this folder to a secure place.

Inside **/etc/letsencrypt/live/example.com/** you will find symlinks that will be updated after every renewal.

/etc/letsencrypt

You will find the following files:

privkey.pem Private key of the certificate. DO NOT SHARE IT!

cert.pem Webserver certificate (sent to the browser).

chain.pem List of all the intermediate certificates connected to this certificate.

fullchain.pem cert.pem + chain.pem

Revoke

To revoke a certificate you can simply use the option **revoke**.

```
$ /.letsencrypt-auto revoke --cert-path example-cert.pem
```

Renewal

The renewal process it’s extremely easy, you can simply invoke **letsencrypt** without parameters.

You can also use the **--renew-by-default** to perform the automatic renewal of the certificate without user interaction.

In this way, it’s possible to schedule a **cron** task to automatically renew the certificates before the expiry.

Update

Since Let’s Encrypt it’s a public beta, it’s fundamental to keep the client up to date.

On Debian *sid/stretch*:

```
$ apt-get update && apt-get upgrade
```

On other OS:

```
$ cd letsencrypt
$ git pull
```

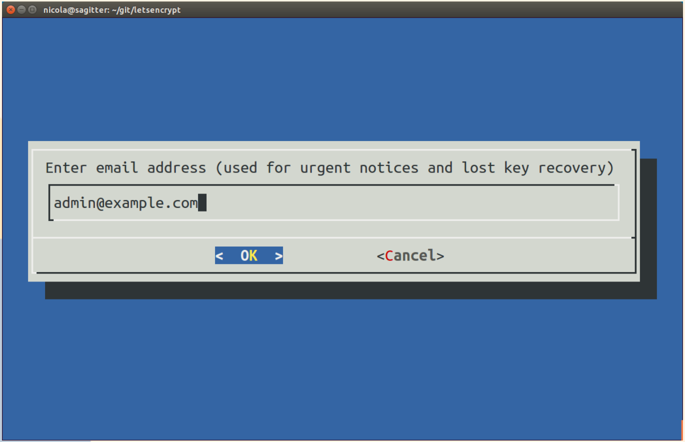
General usage

During everyday usage, you can simply invoke letsencrypt without parameters. The *terminal UI* will guide you through the desidered process.

You simply have to answer to the client questions.

Having only **too much parameters** could be hard to remember. The UI will help on this, but the parameter still give the flexibility to embed the client inside **scripts**.

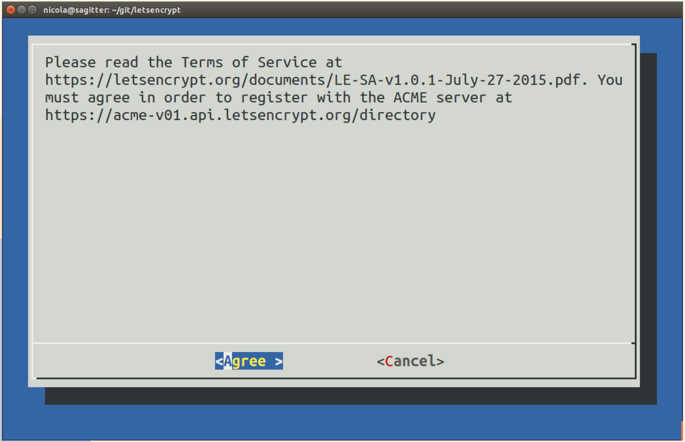
Screens



Enter email address (used for urgent notices and lost key recovery)

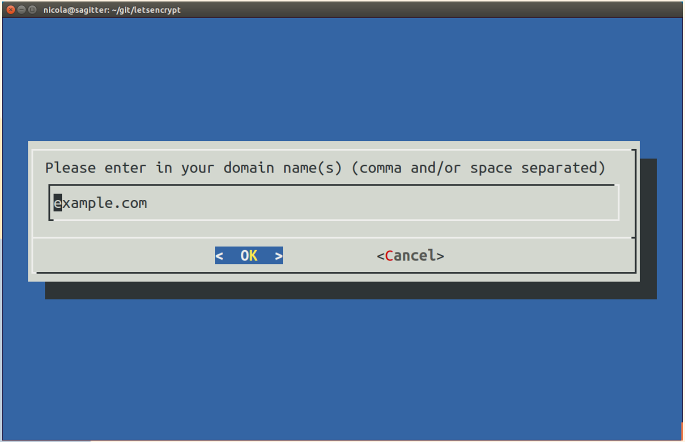
```
admin@example.com
```

Screens



```
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.0.1-July-27-2015.pdf. You
must agree in order to register with the ACME server at
https://acme-v01.api.letsencrypt.org/directory
```

Screens

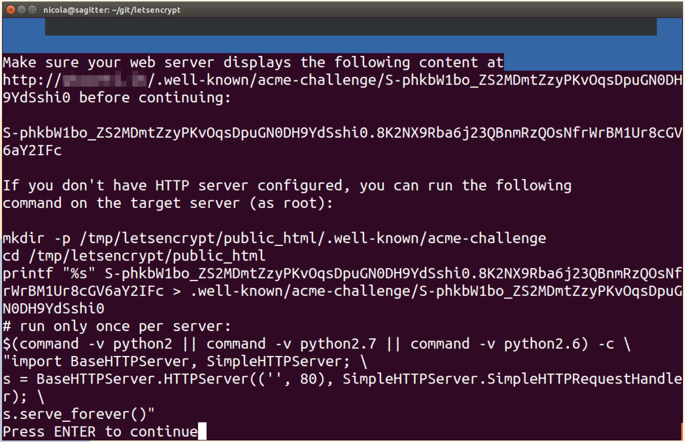


```
Please enter in your domain name(s) (comma and/or space separated)

example.com

<  OK  >          <Cancel>
```

Screens



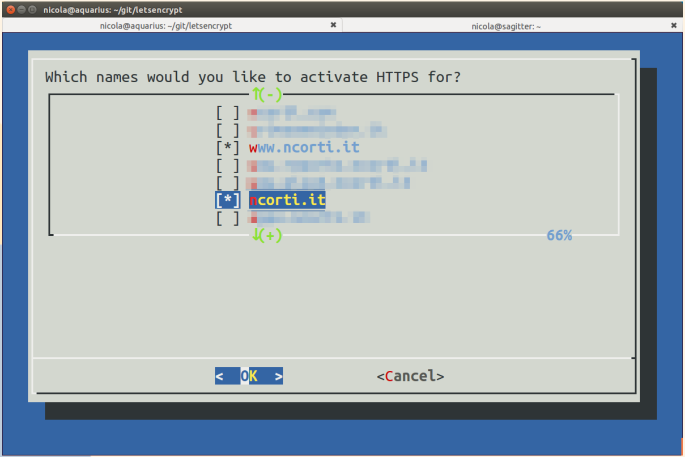
```
Make sure your web server displays the following content at
http://[redacted]/.well-known/acme-challenge/S-phkbW1bo_ZS2MDmtZzyPKvOqsDpuGN0DH
9YdSshi0 before continuing:

S-phkbW1bo_ZS2MDmtZzyPKvOqsDpuGN0DH9YdSshi0.8K2NX9Rba6j23QBnmRzQOsNfrWrBM1Ur8cGV
6aY2IFc

If you don't have HTTP server configured, you can run the following
command on the target server (as root):

mkdir -p /tmp/letsencrypt/public_html/.well-known/acme-challenge
cd /tmp/letsencrypt/public_html
printf "%s" S-phkbW1bo_ZS2MDmtZzyPKvOqsDpuGN0DH9YdSshi0.8K2NX9Rba6j23QBnmRzQOsNf
rWrBM1Ur8cGV6aY2IFc > .well-known/acme-challenge/S-phkbW1bo_ZS2MDmtZzyPKvOqsDpuG
N0DH9YdSshi0
# run only once per server:
$(command -v python2 || command -v python2.7 || command -v python2.6) -c \
"import BaseHTTPServer, SimpleHTTPServer; \
s = BaseHTTPServer.HTTPServer(('', 80), SimpleHTTPServer.SimpleHTTPRequestHandle
r); \
s.serve_forever()"
Press ENTER to continue
```

Screens

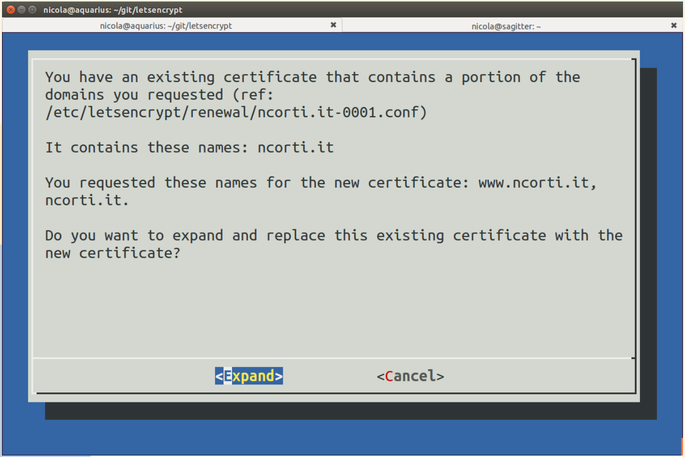


nicola@aquarius: ~/git/letsencrypt
nicola@aquarius: ~/git/letsencrypt
nicola@sagitter: ~

```
Which names would you like to activate HTTPS for?
                       ↑(-)
    [ ] [redacted]
    [*] www.ncorti.it
    [ ] [redacted]
    [*] ncorti.it
    [ ] [redacted]
                       ↓(+)                        66%

          <  OK  >               <Cancel>
```

Screens



nicola@aquarius: ~/git/letsencrypt
nicola@aquarius: ~/git/letsencrypt
nicola@sagitter: ~

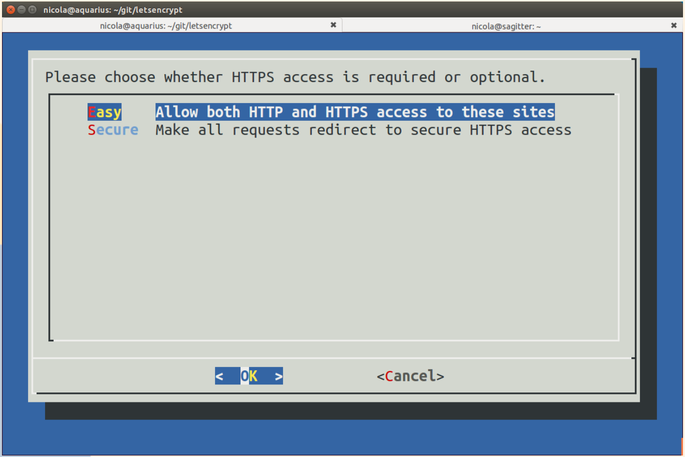
```
You have an existing certificate that contains a portion of the
domains you requested (ref:
/etc/letsencrypt/renewal/ncorti.it-0001.conf)

It contains these names: ncorti.it

You requested these names for the new certificate: www.ncorti.it,
ncorti.it.

Do you want to expand and replace this existing certificate with the
new certificate?
```

Screens

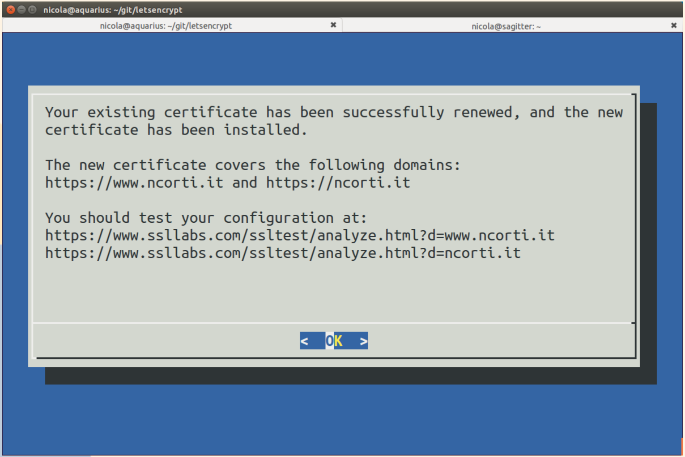


nicola@aquarius: ~/git/letsencrypt
nicola@aquarius: ~/git/letsencrypt
nicola@sagitter: ~

```
Please choose whether HTTPS access is required or optional.

    Easy      Allow both HTTP and HTTPS access to these sites
    Secure    Make all requests redirect to secure HTTPS access
```

Screens


nicola@aquarius: ~/git/letsencrypt
nicola@aquarius: ~/git/letsencrypt
nicola@sagitter: ~
< OK >

```
Your existing certificate has been successfully renewed, and the new
certificate has been installed.

The new certificate covers the following domains:
https://www.ncorti.it and https://ncorti.it

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=www.ncorti.it
https://www.ssllabs.com/ssltest/analyze.html?d=ncorti.it
```

Conclusions

Drawbacks

- No support for **Organization Validation** or **Extended Validation**, to hard to automate.
- No support for **wildcards** (*.example.com), maybe in the future.
- Only HTTP challenge is supported (public beta), DNS challenge support is not already available.

Resources

- How it works **https://letsencrypt.org/howitworks/**
- Tech details **https://letsencrypt.org/howitworks/technology/**
- Read the docs **https://letsencrypt.readthedocs.org/**
- Community board **https://community.letsencrypt.org/**
- Code **https://github.com/letsencrypt/**
- Mailing lists
  - Client **https://groups.google.com/a/letsencrypt.org/forum/#!forum/client-dev**
  - Server **https://groups.google.com/a/letsencrypt.org/forum/#!forum/ca-dev**
  - ACME (IETF) **https://www.ietf.org/mailman/listinfo/acme**

Questions...?

ncorti.com
@cortinico
@cortinico

Made with LaTeX Beamer.

Sources at **https://github.com/cortinico/gulp-letsencrypt**