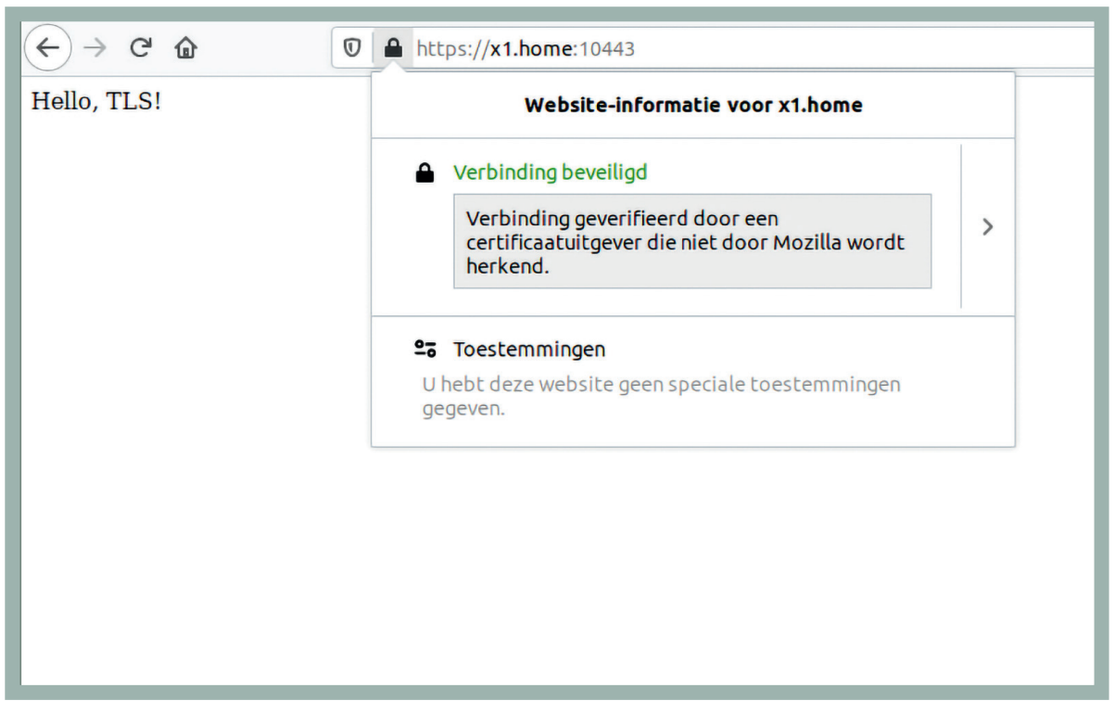
```
koan@x1:~$ sudo step ca certificate x1.home x1.crt x1.key --ca-url https://pow.home:8443 --root ~/.step/certs/root_ca.crt
✔ Provisioner: acme (ACME)
Using Standalone Mode HTTP challenge to validate x1.home .. done!
Waiting for Order to be 'ready' for finalization .. done!
Finalizing Order .. done!
✔ Certificate: x1.crt
✔ Private Key: x1.key
koan@x1:~$
```

Een certificaat aan je ACME-server aanvragen gaat met één opdracht.



Bezoek je in je webbrowser de webserver met als url https://x1.home:10443, dan zou je de boodschap "Hello, TLS!" moeten zien.

Je dient deze opdracht met sudo uit te voeren omdat step de http challenge (http-01) van het ACME-protocol gebruikt. Het programma krijgt immers van de ACME-server de opdracht om een willekeurig getal op een willekeurige url van het domein te plaatsen. De ACME-server verifieert zo of je de controle over het domein hebt. Maar dat gebeurt op poort 80, en een normale gebruiker kan geen server op poort 80 draaien. Daarvoor dienen de rootrechten. Met de opties –ca-url en –root dien je de url van de ca en het rootcertificaat aan te duiden. Overigens dient de firewall op je client ook poort 80 toe te staan.

Met de pijltjestoetsen kies je nu je ACME-provisioner, en daarna voert step de aanvraag uit. Na enkele seconden krijg je het certificaat en de sleutel in de bestanden x1.crt respectievelijk x1.key. Deze kun je nu als tls-certificaat gebruiken in je favoriete software. Een certificaat vernieuwen is even eenvoudig:

```
client:~$ sudo step ca renew x1.crt x1.key --ca-url https://pow.home:8443
```

AUTOMATISCHE VERNIEUWING

Het leuke van het ACME-protocol is uiteraard dat het aanvragen, toekennen en vernieuwen van certificaten volledig op de achtergrond kan gebeuren. Op het blog van Smallstep vind je informatie over hoe je dit configureert in talloze programma's, zoals Caddy, Nginx, Apache, Traefik en Kubernetes.

Op **https://gist.github.com/mmalone/12f5422b2ec68e64e9d11eae0c6ca47d** vind je een voorbeeld-webserver in Python (helaas nog in Python 2…) die automatisch een certificaat verkrijgt en vernieuwt voordat het niet meer geldig is. Download het en installeer dan eerst enkele dependency's:

```
client:~$ sudo pip install acme pem
```

Open dan het Python-script https.py en pas de volgende variabelen aan (de CA is hier pow.home op poort 8443 en de webserver x1.home op poort 10443):

```
# This is the ACME Directory URL for `step-ca`
DIRECTORY_URL = 'https://pow.home:8443/acme/acme/directory'

# This is the root certificate for `step-ca`
ROOT_CERTIFICATE = "/root/.step/certs/root_ca.crt"

# Email address used for creating ACME acount
ACC_EMAIL = "you@example.com"

# Domain name for the certificate.
DOMAIN = 'x1.home'

# Port to listen on for `http-01` challenge
ACME_PORT = 80

# Port our HTTPS server will listen on
HTTPS_PORT = 10443
```

Start daarna de webserver met:

```
client:~$ sudo python https.py
```

Als alles goed gaat, krijg je nu in de uitvoer het certificaat te zien dat je webserver van je CA verkrijgt. Je ziet ook dat de webserver elke 15 seconden controleert of het certificaat al vernieuwd dient te worden. De echte test is natuurlijk de website bezoeken. Bezoek je in je webbrowser de webserver met als url **https://x1.home:10443**, dan zou je de boodschap "Hello, TLS!" moeten zien. Klik je op het slotje links in de adresbalk, dan krijg je te zien dat de verbinding beveiligd is. Je kunt het certificaat onderzoeken om te verifiëren of het door je CA ondertekend is. En als je deze testwebserver lang genoeg laat draaien, zul je zien dat het certificaat automatisch vernieuwd wordt.