

Let's get secure!

Een kijkje in de wereld van HTTPS

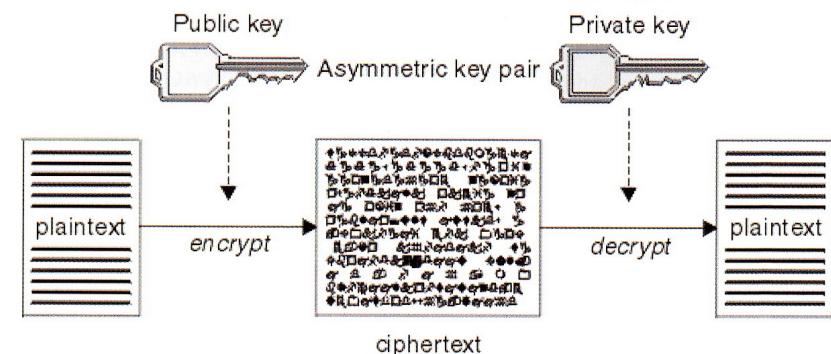
Heb je een website die HTTPS connecties aanbiedt aan je bezoekers? Maak je gebruik van HTTPS als je inlogt of als je gegevens stuurt? Heb je HTTPS eigenlijk nodig? Ook voor je eigen blogje op je eigen server? Voor de meesten van ons is het antwoord op deze vragen duidelijk 'Ja!'. Sommigen zullen ook al ervaren hebben dat dit nog niet zo gemakkelijk is te realiseren...

Misschien denk je "Ach, die site van mij, wie heeft hier nou interesse in om die te haken?" Misschien heb je daar in gelijk, maar ondertussen zijn het zowel de overheden als criminelen die interesse hebben in de communicatie die over het internet gaat. De discussie met betrekking tot de overheden staan we in dit artikel over. Criminelen daarentegen hebben veel baat bij jouw site. Op het moment dat ze de controle krijgen over je site, zullen ze niet alleen een deface doen, maar ook direct malware installeren voor jouw bezoekers. En waarschijnlijk zijn ze ook direct actief op je eigen (privé) netwerk.

In dit artikel nemen we je mee in de wereld van HTTPS. We beginnen bij de certificaten en van daaruit nemen we je mee door het protocol om uiteindelijk bij een praktijkvoorbeeld te komen. Daarnaast beschrijven we het gebruik van de gratis Certificate Authority (CA) LetsEncrypt. In de voorbeelden bespreken we de configuratie voor Apache HTTP Webserver. LetsEncrypt heeft een paar beperkingen en we geven een voorbeeld dat je zal helpen in het beheer van de certificaten die je met LetsEncrypt hebt gemaakt.

Certificaten?

Om een veilige verbinding te starten, moet je in het bezit zijn van een key value pair in combinatie met een certificaat. Met dit certificaat kun je twee dingen bewerkstelligen, namelijk encryptie en authenticatie. Maar wat is een certificaat? Bij certificaten draait het om private en public key pairs. In **Afbeelding 1** wordt weergegeven hoe de berichtuitwisseling eruit ziet. Het te versturen bericht wordt door middel van de public



Afbeelding 1

key van de ontvanger versleuteld. Als je het bericht wilt lezen, moet je in het bezit zijn van de bijpassende private key.

Hiermee hebben we de veiligheid van de inhoud van het bericht gewaarborgd, maar nog geen garantie dat de informatie verstuurd is door iemand die we vertrouwen. Om met certificaten te werken, hebben we een vertrouwenspersoon nodig. De vertrouwenspersoon noemen we een Root CA (Certificate Authority). Alle moderne webbrowsers hebben een standaardset van Root CA's. Een certificaat bevat jouw public key en een handtekening van de verstrekende CA. De handtekening is te verifiëren met de public key van de betrokken Certificate Authority.

Waarom encryptie

Door recente discussies in binnen- en buitenland bestaat steeds meer aandacht voor encryptie. Denk aan Edward Snowden en de laatste strijd tussen Apple en de FBI. Het doel is om ervoor te zorgen dat alleen de betrokken partijen in staat zijn om de informatie die uitgewisseld wordt daadwerkelijk te



Ivo Woltring is een enthousiaste Java ontwikkelaar, blogger en Coach voor Young professionals bij Ordina JTech.



Pieter van der Meer: Scala, Java Big en Fast Data hacker bij Codestar.

kunnen lezen. Tijdens het handshake proces van de communicatie wordt informatie uitgewisseld, zodat de verbinding beveiligd kan worden. Hierover later meer.

Authenticatie

Het tweede dat je met een certificaat kan doen, is identificatie (wie ben ik) en authenticatie (wat mag ik). De zwakke schakel is de zogenaamde Certificate Authority. Deze autoriteit is een organisatie die certificaten ondertekend. Met andere woorden: zij bevestigen dat jij (houder van het certificaat) ook daadwerkelijk degene bent die je zegt dat je bent. Het is dus belangrijk dat deze bedrijven te vertrouwen zijn.

HTTPS en TLS/SSL

Deze termen worden door elkaar gebruikt, maar zijn wezenlijk verschillend. TLS en SSL worden over het algemeen in één adem genoemd en zijn in werkelijkheid een implementatie van hetzelfde protocol. De naam TLS (Transport Layer Security) is ontstaan, omdat SSL (Secure Sockets Layer) bedacht is door de uitvinders van het protocol, namelijk Netscape. TLS is een open protocol dat is vastgelegd in RFC2226. De RFC beschrijft een beveiligde bi-directionele tunnel voor het versturen van willekeurige data tussen twee hosts. Door deze naamswijziging is SSL een deprecated protocol. Verder in het artikel spreken we daarom alleen nog over TLS.

Het HTTP protocol beschrijft het versturen van requests en responses van specifieke headers en mogelijk data. HTTP verkeer wordt ook uitgewisseld via een tunnel voor willekeurige data. Door een toepassing van het TLS protocol op deze tunnel krijgen we een HTTPS verbinding.

Samenvattend, HTTPS is een combinatie van deze twee protocollen (HTTP en TLS). Hetzelfde geldt bijvoorbeeld ook voor SMTP, IMAPS, SSH en FTPS.

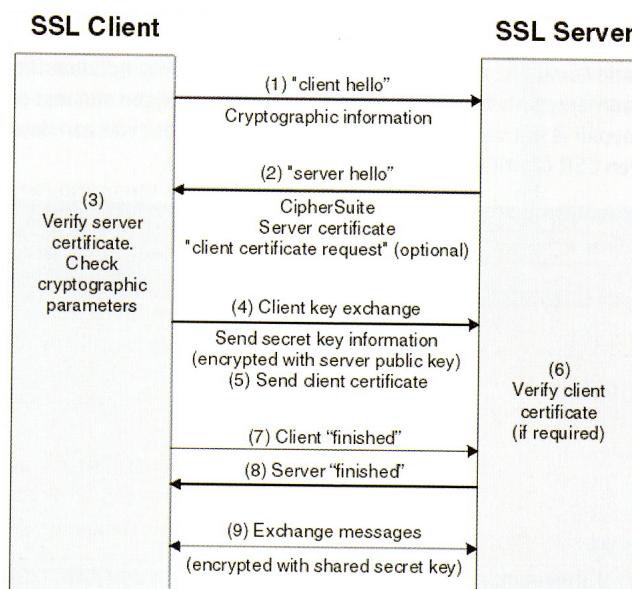
Handshake protocol

Het is handig om een goed beeld te krijgen van hetgeen wat moet gebeuren, voordat we een TLS (HTTPS) verbinding hebben. We willen namelijk op een veilige manier informatie uitwisselen. Om dit te kunnen doen, moeten beiden partijen het eens worden over de key en het algoritme. De stappen die hiervoor doorlopen worden, noemen we het "Handshake protocol" en dit is de basis voor alle TLS gebaseerde communicatie. Het volgende voorbeeld geeft dit proces weer.

Stel, je wilt inloggen op je webmail account. Aangezien je niet wilt dat je username en wachtwoord over een onbeveiligde verbinding verzonden worden, verzoek je de server om een beveilige verbinding op te zetten. Dit wordt ook wel de *Handshake (ClientHello)* fase genoemd. In deze handshake zit alle informatie die de server nodig heeft om te verbinden met de server via SSL, inclusief de ondersteunende cypher algoritmen en de hoogste, ondersteunde versie van TLS. De Server zal dan antwoorden met de *ServerHello*. In dit antwoord zit vergelijkbare informatie die de cliënt nodig heeft, inclusief het besluit welk cypher algoritme gekozen is en de versie van TLS die gebruikt gaat worden.

Nu de *Handshake* heeft plaatsgevonden, gaat de server zich identificeren aan de cliënt. Dit gebeurt door het certificaat op te sturen. Dit certificaat functioneert als een soort paspoort. Het bevat een aantal gegevens, waaronder de naam van de eigenaar, het domein waar het toe behoort, de publieke sleutel, de digitale handtekening en informatie over de geldigheidsduur. De cliënt kiest ervoor om expliciet het certificaat te vertrouwen of controleert dat het een certificaat betreft dat vertrouwd wordt door een vertrouwde CA (*Certificate Authority*). Zodra het vertrouwen is vastgesteld bij de *Certificate Exchange* fase zal op het daarop volgende verkeer encryptie toegepast worden door gebruik te maken van het encryptie algoritme dat tijdens de Handshake is afgesproken.

EEN CERTIFICAAT VOOR EEN (SUB) DOMAIN IS GRATIS EN MAKKELIJK TE VERKRIJGEN



Afbeelding 2

LetsEncrypt

LetsEncrypt is een initiatief van Internet Security Research Group (ISRG). Met LetsEncrypt hebben zij een dienst gemaakt die gratis en automatisch SSL certificaten aanbiedt. Het aangeboden certificaat wordt door alle moderne browsers aangemerkt 'als' getekend door een trusted CA'.

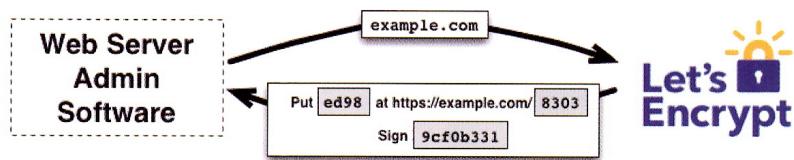
De implementatie van het mechanisme van LetsEncrypt is gebaseerd op het ACME (Automated Certificate Management Environment) protocol van de IETF (The Internet Engineering Task Force). Deze specificatie beschrijft hoe je op automatische wijze een certificaat kunt aanmaken en beheren. LetsEncrypt volgt nog niet helemaal de standaard, omdat de specificatie nog niet definitief is. Wat dat voor de toekomst gaat betekenen, is niet bekend.

ACME

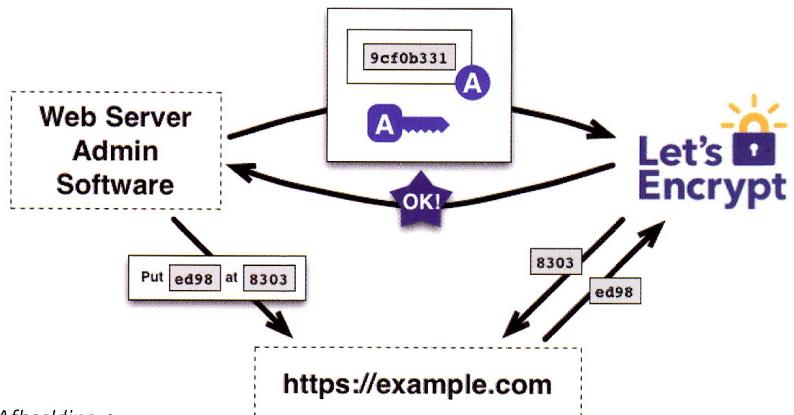
Voor het aanvragen van een certificaat moet de CA er zeker van zijn dat de aanvrager ook in het bezit is van het domein. In deze context wordt dat gedaan door een resource (een bestand met een specifieke URL) te plaatsen op een specifieke locatie binnen het domein (zie **Afbeelding 3**).

In dit geval wordt aan de server gevraagd om de content "*edg8*" te plaatsen in de file "*8303*" op het domein. Ten tweede wordt hij gevraagd om de nonce (een arbitrair 'nummer' dat maar één keer gebruikt kan worden) "*gcf0b331*" te tekenen met zijn eigen private key en deze terug te sturen naar de LetsEncrypt server (zie **Afbeelding 4**).

Als de LetsEncrypt server al zijn controles kan uitvoeren en de informatie correct is, zal hij het Keypair van de server aanmerken als Authorized keypair. Met dit keypair is het vanaf dat moment mogelijk om een CSR (Certificate



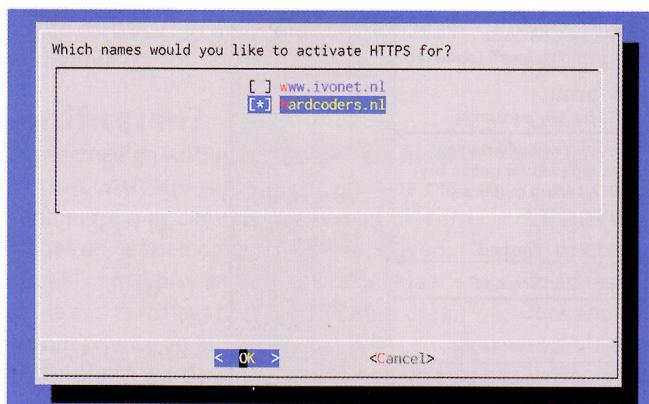
Afbeelding 3



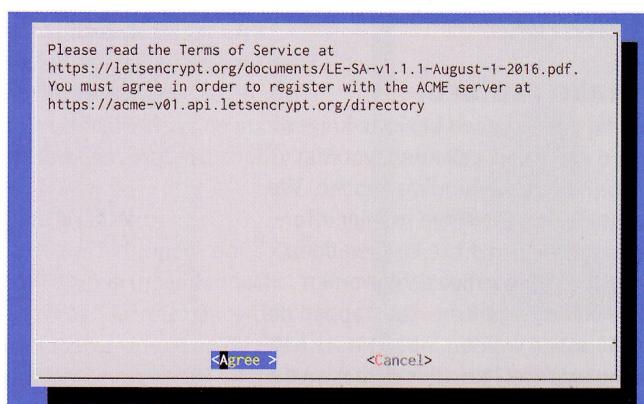
Afbeelding 4

Signing Request) te versturen. Zonder al teveel in detail te treden, is een CSR een versleuteld bericht met daarin informatie over de aanvrager en de publieke sleutel van de server. Met dit CSR kan de CA een certificaat genereren voor het aangevraagde domein.

LetsEncrypt heeft ook een aantal beperkingen. Op dit moment ondersteunt LetsEncrypt geen zogenaamde ster certificaten (*.example.com). Voor elk subdomein zal dus een eigen certificaat moeten worden aangevraagd. Ook moet LetsEncrypt in staat zijn om het domein te verifiëren en het domein zal dus vanaf internet toegankelijk moeten zijn. Verder zijn er verschillende rate limits, onder andere een beperking van het aantal certificaten per week, die je zeker als je met LetsEncrypt gaat beginnen in de gaten moet houden. Dit gezegd hebende, is het verkrijgen van een certificaat voor een (sub)domein dat wel aan de voorwaarden voldoet, makkelijk.



Afbeelding 5



Afbeelding 6

Getting started: mijn eerste certificaat

In dit praktijk voorbeeld wordt ervan uitgegaan dat je beschikking hebt over een eigen server. Ook heb je een domeinnaam dat je naar het IP van de server kan laten verwijzen door middel van een A record of een CNAME. Op je Linux machine heb je Apache HTTP Webserver draaien waar je jouw website aan gekoppeld hebt. Deze website wil je ook via HTTPS aanbieden en daarvoor genereren we een sleutelpaar waar LetsEncrypt een certificaat voor zal leveren. Dit voorbeeld lijkt misschien te simpel voor de meeste situaties, omdat de meeste configuraties te maken krijgen met reverse proxies en dergelijke, maar dat valt wel mee. De configuratie is wellicht wat moeilijker, maar het is nog steeds de buitenste laag (Apache / Nginx) die het certificaat levert en dat is, in het hier beschreven voorbeeld, eigenlijk hetzelfde. Voordat je met LetsEncrypt kunt beginnen, moet je ervoor zorgen dat poort 80 (HTTP Poort) en 443 (HTTPS Poort) benaderbaar zijn voor LetsEncrypt.

Nu ben je klaar om te beginnen. Eerst dien je Certbot te installeren. Certbot is door LetsEncrypt aanbevolen, maar er zijn vele implementaties beschikbaar. Enig onderzoek naar de juiste cliënt kan dus lonen.

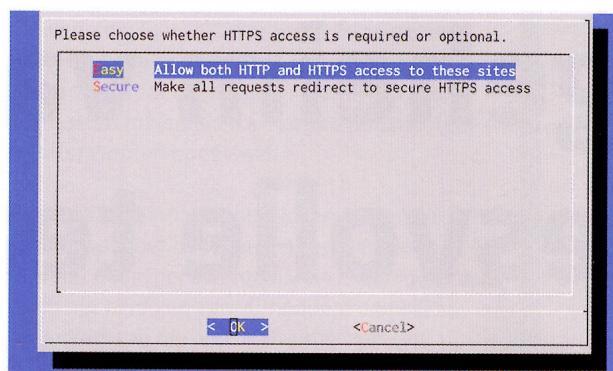
```
$ wget https://dl.eff.org/certbot-auto
$ chmod a+x certbot-auto
$ ./certbot-auto
```

Certbot is geïnstalleerd en gestart. Certbot is een tool waarmee je geautomatiseerd certificaten van LetsEncrypt kunt aanvragen en installeren. Certbot zal op zoek gaan naar je Apache configuratie en je de sites tonen die het zelf kan vinden. In dit geval gaan we als voorbeeld hardcoders.nl van een certificaat voorzien. Je krijgt een scherm te zien waarin om een e-mail adres gevraagd wordt (zie **Afbeelding 5**).

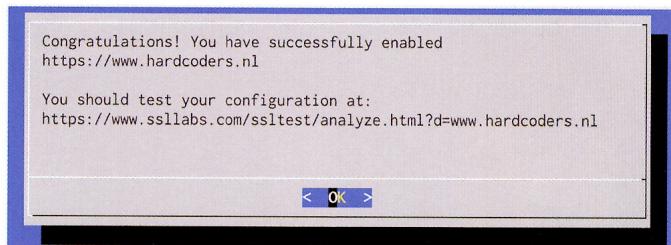
Nu krijg je het verzoek om akkoord te gaan met de Terms of Service (zie **Afbeelding 6**). In dit geval wordt gekozen dat zowel poort 80 als poort 443 gaan werken voor deze site (zie **Afbeelding 7**).

Op het moment dat je felicitaties krijgt heb je een certificaat en is Apache ook geconfigureerd met dit certificaat (zie **Afbeelding 8**).

Gefeliciteerd! Je hebt nu een met HTTPS beveiligde website! Certbot heeft de configuratie van Apache aangepast, zodat de LetsEncrypt gegenereerde certificaten zijn toegevoegd en



Afbeelding 7



Afbeelding 8

```
SSLCertificateFile      /etc/letsencrypt/live/www.hardcoders.nl/cert.pem
SSLCertificateKeyFile  /etc/letsencrypt/live/www.hardcoders.nl/privkey.pem
SSLCertificateChainFile /etc/letsencrypt/live/www.hardcoders.nl/chain.pem
```

Listing 2

```
$ crontab -e
13 7 * * * ./path/to/certbot-auto renew --quiet --no-self-upgrade
3 13 * * * ./path/to/certbot-auto renew --quiet --no-self-upgrade
```

Listing 3

geactiveerd. De regels hieronder geven aan wat er door Certbot aangepast is in de Apache configuratie (zie **Listing 2**). Alle certificaten van LetsEncrypt zijn drie maanden geldig en moeten daarna opnieuw worden uitgegeven. Dit kan gelukkig helemaal geautomatiseerd worden door een cron / systemd job hiervoor aan te maken.

Het commando in **Listing 3** zal een verversing van het certificaat uitvoeren als het certificaat verlopen is. Dit wordt elke dag om 7:13 uur en 13:03 uur gedaan in dit voorbeeld. Deze tijden zijn willekeurig gekozen en het is handig om dit voor jezelf ook te doen, zodat de LetsEncrypt servers niet overbelast raken op een specifiek tijdstip.

Nu heb je je site beveiligd en ervoor gezorgd dat dat zo blijft. Dezelfde oefening kan je ook uitvoeren voor andere (sub)domeinen en natuurlijk heeft de Certbot nog veel meer mogelijkheden, die goed gedocumenteerd staan op de website van LetsEncrypt. ■

REFERENTIES

- <https://letsencrypt.org/>
- https://en.wikipedia.org/wiki/Transport_Layer_Security
- <http://www.tech-faq.com/os-model.html>
- <http://tools.ietf.org/html/rfc2246>
- <https://certbot.eff.org>