Florian Effenberger
Privates Blog

X.509-Zertifikate mit acme.sh und Let's Encrypt

acme.sh ist ein alternativer Client für Let's Encrypt, der alle Funktionen zur Erzeugung von SSL-Zertifikaten in einem Shellskript zur Verfügung stellt

Let's Encrypt ist das Mittel der Wahl für jeden, der schnell und kostenfrei X.509-Zertifikate für seinen Server erstellen möchte. Neben dem offiziellen Client gibt es eine Vielzahl alternativer Tools, die über das ACME-Protokoll Kontakt zur Zertifizierungsstelle aufnehmen. acme.sh ist ein Ansatz, der komplett als Shellskript geschrieben ist. In diesem Beitrag zeige ich, wie man damit Zertifikate erstellt, verlängert und verwaltet.

Zum offiziellen Let's Encrypt-Client gibt's hier im Blog bereits einen Beitrag in englischer Sprache, ebenso zur Funktionsweise von Wildcard-Zertifikaten und zum Zusammenspiel mit RouterOS.

Auch mein Blog nutzt ein Zertifikat von Let's Encrypt, das mit acme.sh erstellt wurde

In diesem Beitrag stelle ich die Nutzung von acme.sh vor, das im Gegensatz zum offiziellen Tool komplett als Shellskript programmiert wurde und keinen Python-Interpreter voraussetzt.

Die Installation von acme.sh

Am einfachsten installiert man das Paket direkt aus den offiziellen GitHub-Quellen. Grundsätzlich lässt sich acme.sh ohne Root-Rechte nutzen – sollen allerdings Zertifikate eingespielt und Serverdienste neu gestartet werden, sind entsprechende Rechte vonnöten. Im folgenden Beispiel wird der Client durch ein einleitendes

```
sudo su
```

daher als root installiert. Zur Installation aus dem Git-Repository ist der entsprechende git-Client vonnöten, zudem wird die Installation von socat empfohlen, falls der Standalone-Modus für acme.sh genutzt wird – dazu später mehr. Unter Debian oder Ubuntu finden diese beiden Pakete mittels

```
apt-get install git socat
```

ihren Weg aufs System. Das eigentliche Repository wird dann ins Root-Verzeichnis ausgecheckt:

```
cd ~
git clone https://github.com/Neilpang/acme.sh.git
```

In dieser Form kann der Client allerdings noch nicht benutzt werden, da unter anderem die Integration per Cron fehlt. Diese erledigt das integrierte Installationsprogramm:

```
cd acme.sh
./acme.sh --install --home /opt/acme.sh --config-home
/etc/acme.sh --accountemail "mail@adresse"
```

Damit wird acme.sh nach /opt/acme.sh installiert, die Konfiguration unter /etc/acme.sh angelegt und die E-Mail-Adresse für Informationen zum Ablauf von Zertifikaten gesetzt. Anschließend lässt sich das Repository durch ein einfaches

```
cd ..
rm -rf acme.sh
```

vom System entfernen, es wird nicht mehr benötigt, da der Client nun installiert ist. Bevor's weitergeht sollte man sich einmal neu am System anmelden, um die aktualisierte Bash-Konfiguration zu aktivieren, die einen Alias für acme.sh setzt.

Als letzten Schritt der Installation deaktiviere ich das automatische Update mittels

```
acme.sh --upgrade --auto-upgrade 0
```

Das ist sicher Geschmackssache – mir gefällt der Gedanke nicht, ein mit root-Rechten laufendes Skript nicht über die offiziellen Paketquellen meiner Distribution zu aktualisieren. Von Zeit zu Zeit empfiehlt es sich daher, das Paket händisch über

```
acme.sh --upgrade
```

auf die aktuelle Version zu bringen.

Das Erstellen von Zertifikaten

Jetzt ist acme.sh bereit, das erste Zertifikat zu erstellen! Das Tool übernimmt dabei auf Wunsch alle notwendigen Schritte:

- das Erstellen des privaten Schlüssels samt Certificate Signing Request
- die Validierung der Domain
- die Beantragung des Zertifikats
- die Installation in den gewünschten Zielpfad
- das Neustarten der Serverdienste
- sowie die automatische Verlängerung

Um überhaupt ein Zertifikat zu erstellen, muss die Inhaberschaft an der Domain nachgewiesen werden, wofür diverse Betriebsmodi zur Verfügung stehen.

Der Webroot-Modus

Die vermutlich gängigste Variante führt über das so genannte .well-known-Verzeichnis: Dabei erwartet Let's Encrypt einen speziellen, zufällig gewählten Dateinamen in ebendiesem Verzeichnis. Existiert diese Datei mit dem korrekten Inhalt, kann das Zertifikat ausgestellt werden – diese Variante wird auch Webroot-Modus genannt und erfordert Schreibzugriff auf den Webserver. Ein Beispiel:

```
acme.sh --issue --test --keylength 4096 --domain
meine.domain --webroot /srv/www/meine.domain --cert-
file /etc/ssl/certs/meine.domain.cert.pem --key-file
/etc/ssl/private/meine.domain.key.pem --ca-file
/etc/ssl/certs/meine.domain.ca.pem --fullchain-file
/etc/ssl/certs/meine.domain.chain.pem --reloadcmd
"service nginx force-reload"
```

Dieser Befehl bedeutet Folgendes:

- Es soll ein Zertifikat ausgestellt werden (–issue).
- Die Anfrage erfolgt nur im Testbetrieb über den so genannten Staging-Server (–test). Somit kann gefahrlos getestet werden, ohne ein gültiges Zertifikat zu erzeugen oder die Rate Limits auszulösen.
- Es wird ein Schlüssel mit 4096 Bits erstellt (–keylength 4096). acme.sh erstellt auf Wunsch auch ECC-Schlüssel (–keylenght ec-384), die jedoch noch

nicht von jedem Server und Client unterstützt werden, weshalb ich sie an dieser Stelle nicht näher behandle.

- Die zu validierende Domain lautet meine.domain (–domain meine.domain). Ausschließlich dieser konkrete Name ist im Zertifikat abgedeckt.
- Das Root-Verzeichnis der Domain, in der acme.sh die temporäre .well-known-Datei erzeugt, liegt in /srv/www/meine.domain. Der Webserver muss Dateien aus diesem Verzeichnis ausliefern können. Probleme gab es in meinen Tests unter anderem bei Joomla, da das CMS die Adresse als interne Seite interpretiert. Abhilfe schaffen entweder spezielle Regeln im Webserver, oder die Nutzung im Standalone-Modus (siehe unten).
- Das erzeugte Zertifikat speichert acme.sh standardmäßig in /etc/acme.sh /domainname. Es wird aber ausdrücklich davon abgeraten, dieses Verzeichnis in den jeweiligen Diensten anzugeben. Stattdessen kopiert acme.sh auf Wunsch alle nötigen Dateien in ein definiertes Zielverzeichnis, so auch hier (–cert-file, –key-file, –ca-file, –fullchain-file). In der Regel genügt es, beispielsweise in nginx das Keyfile und das Fullchain-File zu nennen, um das Zertifikat samt Zertifizierungskette korrekt auszuliefern. Die anderen Dateien sind meist nicht nötig und können bei Bedarf auch weggelassen werden.
- acme.sh beherrscht zudem den Neustart von Serverdiensten nach erfolgreicher Zertifikatserneuerung. Im obigen Beispiel wird nginx neu gestartet, sobald das Zertifikat erneuert wurde.

War dieser Test erfolgreich, geht es an die eigentliche Erstellung des Zertifikats. Ein Beispiel, mit Änderungen zur vorherigen Befehlszeile in Fettdruck:

```
acme.sh --issue --force --keylength 4096 --domain
meine.domain --webroot /srv/www/meine.domain --cert-
file /etc/ssl/certs/meine.domain.cert.pem --key-file
/etc/ssl/private/meine.domain.key.pem --ca-file
/etc/ssl/certs/meine.domain.ca.pem --fullchain-file
/etc/ssl/certs/meine.domain.chain.pem --reloadcmd
"service nginx force-reload ; service postfix force-
reload ; service dovecot force-reload" --domain
www.meine.domain
```

Was ist anders?

- Der Befehl –test ist einem –force gewichen. Es wird nun kein Testzertifikat mehr angefordert, sondern ein gültiges von der "richtigen" Zertifizierungsstelle. Das –force ist nur nötig, weil auch das Testzertifikat im System hinterlegt ist und jetzt überschrieben werden muss. Wer direkt und ohne Testzertifikat ans Werk geht, der lässt –force sicherheitshalber weg, um nicht ungewollt etwas zu überschreiben.
- Ist ein Zertifikat in mehreren Diensten genutzt, so lassen sich diese der Reihe nach neustarten. Im Beispiel startet acme.sh neben nginx auch Dovecot und Postfix neu, wenn ein Zertifikat erfolgreich verlängert wurde.
- Let's Encrypt erlaubt auch mehrere Domains bzw. Hostnamen. Im Beispiel wird neben meine.domain auch die Adresse www.meine.domain abgedeckt, eine in der Praxis vermutlich sehr häufig auftretende Konstellation. acme.sh versucht die Validierung dieser Adressen jeweils im selben Verzeichnis. Theoretisch ist im so genannten Hybrid-Modus auch eine Kombination verschiedener Validierungen möglich.

Der Standalone-Modus

In manchen Fällen führt der Webroot-Modus nicht ans Ziel. Das ist unter anderem dann der Fall, wenn der genutzte Server die .well-known-Adresse nicht unterstützt. Joomla hat die URL in meinen Tests beispielsweise immer als Seitentitel interpretiert und seine eigene Fehlerseite zurückgeliefert, vermutlich dürften sich aber auch Redmine und Co. ähnlich verhalten. Eine Möglichkeit, dem Problem zu begegnen ist die Anpassung im Webserver, oder aber die Nutzung des Standalone-Modus. Der hilft auch dann, wenn auf der Maschine beispielsweise gar kein Webserver läuft.

Im Standalone-Modus bedient sich acme.sh des zuvor installierten Tools socat und lauscht beim Erstellen und Verlängern des Zertifikats kurzzeitig auf Port 80, um die .well-known-Anfrage zu beantworten. Ein Beispiel:

```
acme.sh --issue --force --keylength 4096 --domain
```

```
meine.domain --standalone --pre-hook "service nginx
stop" --post-hook "service nginx start" --cert-file
/etc/ssl/certs/meine.domain.cert.pem --key-file
/etc/ssl/private/meine.domain.key.pem --ca-file
/etc/ssl/certs/meine.domain.ca.pem --fullchain-file
/etc/ssl/certs/meine.domain.chain.pem --reloadcmd
"service nginx force-reload"
```

Im Vergleich zum vorherigen Befehl die Änderungen wieder fett hervorgehoben:

- Der Standalone-Modus ist aktiviert (–standalone).
- Sofern ein Webserver läuft, wird dieser vorher beendet (–pre-hook) und nach erfolgter Zertifikatserstellung wieder gestartet (–post-hook).
- Auch hier lassen sich wieder mehrere Domains und verschiedene Befehle zum Neustart der Dienste kombinieren, wie weiter oben gezeigt.

Der DNS-Modus

Eine weitere Möglichkeit, die für die Erstellung von Wildcard-Zertifikaten sogar zwingend ist, ist die Validierung der Domain via DNS-Eintrag. Dabei gibt acme.sh einen Text aus, der zur Zonendatei hinzugefügt werden muss. Großer Nachteil ist, dass Verlängerungen hier nur händisch möglich sind, denn der Text ändert sich regelmäßig. Das sieht dann beispielsweise so aus:

```
_acme-challenge IN TXT
123a3aBxz1tLPow0x7J3BL8Dj9atPkO3rjDo9J-PAtl
```

Die genannte Challenge erhält man durch folgende Befehlszeile:

```
acme.sh --issue --test --keylength 4096 --domain
meine.domain --dns --yes-I-know-dns-manual-mode-
enough-go-ahead-please --cert-file /etc/ssl/certs
/meine.domain.cert.pem --key-file /etc/ssl/private
/meine.domain.key.pem --ca-file /etc/ssl/certs
```

```
/meine.domain.ca.pem --fullchain-file /etc/ssl/certs
/meine.domain.chain.pem --reloadcmd "service nginx
force-reload"
```

Die wichtigsten Punkte sind wiederum fett hervorgehoben:

- Als Modus wird die DNS-Validierung aktiviert (–dns).
- Man bestätigt ausdrücklich, die Hinweise zu diesem Verfahren gelesen und verstanden zu haben (–yes-I-know-dns-manual-mode-enough-go-ahead-please).

Eine Besonderheit: Trotz mehrerer Versuche bei korrektem TXT-Rekord gelang es mir zunächst nicht, die Validierung mit –force durchzuführen. Der Client gab dann aber den entscheidenden Hinweis – nach dem ersten Versuch muss statt –issue der Befehl –renew genutzt werden, dann funktioniert es auch.

Weitere Modi

acme.sh stellt noch eine Vielzahl weiterer Modi zur Verfügung, die in der offiziellen Dokumentation vorgestellt werden. Besonders interessant, aber von mir leider noch nicht getestet, ist dabei der Stateless-Modus. Er scheint zu ermöglichen, die Validierung nochmals wesentlich einfacher über den Webserver zu erzeugen, selbst wenn kein Schreibzugriff auf das Webroot besteht. Dabei wird der Fingerprint des Let's Encrypt-Accounts als .well-known-Adresse veröffentlicht, was das ganze Verfahren nochmals vereinfachen dürfte.

Die Verlängerung

Die Verlängerung konnte ich noch nicht testen, da ich erst vor einigen Tagen auf acme.sh gestoßen bin. Der folgende Absatz ist also mit etwas Vorsicht zu genießen. ;-)

Bei der Installation richtet acme.sh einen Cronjob ein, der jeden Tag nachsieht, ob Domains zur Verlängerung anstehen (siehe als root: crontab -l). Die

Zertifikate von Let's Encrypt gelten 90 Tage, nach 60 Tagen führt der Client in der Regel eine Verlängerung durch, sodass bei Problemen genug Zeit bleibt um zu reagieren. Zumindest Fehler sollten vom Client per E-Mail an den Root-Nutzer gemeldet werden.

Die derzeit aktiven Zertifikate und die Ablaufzeiten listet folgender Befehl auf:

```
acme.sh --list
```

Es kann also nicht schaden, regelmäßig nachzusehen, ob auch wirklich alle Zertifikate rechtzeitig erneuert wurden oder ob es Probleme gab. Die bei der Beantragung des Zertifikats übermittelten Werte, insbesondere für den Neustart der Serverdienste, speichert acme.sh unter /etc/acme.sh/meine.domain /meine.domain.conf in den Variablen Le_Webroot, Le_PreHook, Le_PostHook, Le_RenewHook und Le_ReloadCmd, wo sich diese auch ändern oder ergänzen lassen.

Eines der Argumente für acme.sh ist, dass der private Schlüssel des Zertifikats im Gegensatz zum offiziellen Let's Encrypt-Client gerade nicht bei jeder Verlängerung neu erzeugt wird. Das ermöglicht, die Zertifikate auch für TLSA-Records (DANE) und HPKP (Key Pinning) einzusetzen und war mein primärer Grund, den Client zu wechseln.

Eine manuelle Verlängerung von Zertifikaten ist natürlich ebenfalls möglich:

```
acme.sh --renew --domain meine.domain
```

Unterstützt wird auch eine Verlängerung aller Zertifikate:

```
acme.sh --renew-all --stopRenewOnError
```

Das Entfernen von Zertifikaten geht mittels

```
acme.sh --remove --domain meine.domain
```

Die Dateien werden dabei jedoch nicht automatisch gelöscht – im Beispielsfall sollten das Verzeichnis /etc/acme.sh/meine.domain sowie die Dateien /etc/ssl/certs/meine.domain.cert.pem, /etc/ssl/certs/meine.domain.chain.pem, /etc/ssl/certs/meine.domain.ca.pem und /etc/ssl/private/meine.domain.key.pem händisch gelöscht werden.

---

18 Gedanken zu „X.509-Zertifikate mit acme.sh und Let's Encrypt“

Björn

17. August 2018 um 22:38 Uhr

Wirklich ein Super Beitrag, hat mir echt weitergeholfen endlich Let's Encrypt mit DNS Validation bei OVH zum laufen zu bringen

Florian Effenberger

18. August 2018 um 9:35 Uhr

Danke für die nette Rückmeldung, das freut mich! ;-)

Klaus

27. August 2018 um 17:02 Uhr

Das ist ein schöner und verständlicher Beitrag.

Nach certbot habe ich jetzt acme.sh für meine Vorstellungen getestet.

Ich möchte Beantragung/Erneuerung unbedingt vom Produktivserver mit mehreren virtuellen Webservern fernhalten und einen eigenen LE-Server nutzen. Dieser LE-Server soll beantragen, hat aber keinen Zugriff auf das .well-known-Verzeichnis, bzw. bei V2 auf den DNS-Server.
acme.sh wartet, bis man den token im webroot platziert hat (oder den TXT-Eintrag ins DNS gepackt hat) und geht dann weiter. Das ist für eine renewal-Automatisierung hinderlich.
Gibt es eine Möglichkeit den LE-Token für das renewal “wiederzuverwenden”?

 Florian Effenberger

27. August 2018 um 18:40 Uhr

> Ich befürchte (ohne es zu wissen), dass das nicht geht – die Domain muss meines Wissens immer neu validiert werden. Es gibt aber eine DNS-Alias-Möglichkeit, hilft dir die evtl. weiter?

 Florian Effenberger

3. September 2018 um 10:31 Uhr

Die automatische Verlängerung hat bei mir problemlos geklappt. Lediglich bei der Domain mit DNS-Validierung kam, wie zu erwarten, ein Hinweis per Cron-Mail, dass ich eine neue Challenge hinterlegen muss. Das Ganze lässt sich dann manuell mit `acme.sh --renew --domain meine.domain --yes-I-know-dns-manual-mode-enough-go-ahead-please` anstoßen.

Schön ist, dass in der Tat der bisherige Private Key beibehalten wurde (das Datum in /etc/acme.sh/meine.domain/meine.domain.key hat sich nicht geändert), es somit keine Probleme mit DANE/TLSA geben sollte. Auch der Neustart der Serverdienste scheint anstandslos funktioniert zu haben. Ein Backup der alten Zertifikate wurde nicht angelegt, war aber für mich auch nicht nötig.

Roland A.

18. April 2019 um 5:51 Uhr

Geiler Beitrag. Vielen Dank. War sehr hilfreich.

Florian Effenberger

18. April 2019 um 7:50 Uhr

Danke für die Blumen! ;)

Majo

21. April 2019 um 9:46 Uhr

Hallo,
vielen Dank für die ausführliche und beschreibende Erklärung.

Ich habe ein Problem, jeglicher acme.sh-Befehl wird mit einem „sh: 1: acme.sh: not found“ quittiert?
Neustart des Servers, Neu-Installation und alternative Installationswege von acme helfen nicht. Ausloggen und neue VNC-Verbindung nutzen, ebenfalls nicht. Ich habe keine Idee mehr was ich noch probieren könnte?

Florian Effenberger

21. April 2019 um 16:05 Uhr

Das klingt danach, als ob die Installation im Pfad nicht gefunden wird. Wie wirst du root? Bei mir funktioniert es mittels `sudo su`. Hast du die Zeile mit `acme.sh --install` ausgeführt?

Stefan

1. Juni 2019 um 14:23 Uhr

Hallo,
ein sehr guter Artikel. – Zur Optimierung der Verifikation bei Wildcardzertifikaten: Wenn man die Verwaltung der DNS-Zone auf Cloudflare übertragt (mindestens für eine Zone kostenlos), kann von acme.sh auch die Cloudflare-API (Schalter –dns dns_cf) nutzen lassen. Zuvor muss von Cloudflare ein API-Token erstellt werden (‘My Profile’ –> ganz unten ‘API Keys’).
Beispiel:
acme –issue –dns dns_cf –keylength 4096 –domain example.com –domain *.example.com
Damit wird der gesamte Zonen-Verifikationsprozess auch für Wildcars-Zertifikate automatisiert. Klappt bei mir wunderbar und ich kann in Verbindung mit OVH als Registrar auch günstig DNSSEC und DANE realisieren.

 Florian Effenberger

2. Juni 2019 um 11:49 Uhr

Klasse, vielen Dank für den Tipp!

---

tioan

22. Juli 2019 um 4:17 Uhr

OVH hat auch selbst eine DNS-API somit ist der Umweg über Cloudflare unnötig.

Tom

2. Juli 2019 um 10:04 Uhr

Ich habe mit Let´s Encrypt ganz andere Sorgen, ich kann nicht mehr verlängern ! Am Server habe ich nichts geändert und LE läuft seit fast 2 Jahren ohne Probleme. Jetzt kommt:
Let’s Encrypt-SSL/TLS-Zertifikat konnte nicht ausgestellt werden für meine_domaine.de. Die Autorisierung dieser Domain ist fehlgeschlagen.
Unter Details steht dann:
Invalid response from https://acme-v02.api.letsencrypt.org/acme/authz

/6JKF1pfr7bzjycza4tNRSZXsCtqyC2-wD2WiN11.
Details:
Type: urn:ietf:params:acme:error:connection
Status: 400

Selbst mit der Top Anleitung hier bekomme ich beim erstellen des Zertifikats diese Meldung:
Detail: Fetching https://www.meine_domaine.de/.well-known/acme-challenge/0Y1CjT01DDX3cFkcQil1uFM_ffCvkls0rKrNhU8: Timeout during connect (likely firewall problem)

ich stehe irgendie auf´m Schlauch ... weiß jemand was das soll oder besser wie fix ich das Problem ?

 Florian Effenberger

2. Juli 2019 um 19:32 Uhr

Mit welcher Befehlszeile erzeugst du denn das Zertifikat?

Tom

2. Juli 2019 um 23:50 Uhr

Ich habe vor 2 Jahren Let´s Encrypt via Plesk-Erweiterung installiert und dann meinen beiden Domains ein SSL Zertifikat zugewiesen. Seit 20 Tagen bekomme ich die Zertifikate nicht aktualisiert. Ich habe dann versucht acme mit der Anleitung von hier via Webroot-Modus zu installieren , leider ohne Erfolg , da auch hier die Fetching Meldung kommt. Ich finde den Fehler irgendwie nicht ... weiß jemand Rat ?

 Florian Effenberger

4. Juli 2019 um 16:57 Uhr

Mit Plesk hab ich leider keine Erfahrung. Es klingt danach, als ob entweder Port 80 bzw. 443 nicht offen sind, oder aber eine Konfiguration im Webserver die .well-known

URLs verhindert. Was sagt denn der Debug-Modus von acme.sh?

Tom

5. Juli 2019 um 14:18 Uhr

So , habe den Fehler gefunden , meine AAAA eingetragen ipv6 Adressen mochte LE nicht. Hatte in einem engl. Forum sowas gelesen , das LE damit ab und an Probleme hat. Nach dem löschen der Einträge nur noch die DNS wieder neu eingestellt und schon lief die Installation wie Sie hier beschrieben wurde. So wie hier beschrieben hat alles wirklich wunderbar geklappt ! Super Betrag und Danke !

 Florian Effenberger

5. Juli 2019 um 14:32 Uhr

Danke für die Rückmeldung! Hatte nie Probleme mit IPv6, aber gut zu wissen, dass es jetzt klappt. :-)