Eine eigene OpenSSL CA erstellen und Zertifikate ausstellen

OpenSSL bringt umfassende Werkzeuge mit, um eine eigene, kleine Certificate Authority (CA) betreiben zu können. Die Nutzung einer eigenen CA ist besonders dann sinnvoll, wenn mehrere Dienste über SSL/TLS kostenlos abgesichert werden sollen. Neben dem Nachteil, dass die eigene CA vor Benutzung zuerst auf den Clientrechnern bekannt gemacht werden muss, gibt es aber auch einen Vorteil: Mit einer CA unter der eigenen Kontrolle ist man im Zweifel auf der sicheren Seite: In den letzten Jahren wurden immer wieder Fälle bekannt, in denen große Certificate Authorities falsche Zertifikate ausgestellt haben. Es gibt Grund genug, die Vertrauenswürdigkeit großer CAs anzuzweifeln.

Mit dieser Anleitung werdet ihr in der Lage sein, beliebig viele Zertifikate für eure Dienste ausstellen zu können, die in jedem Browser als gültig erkannt werden, sofern vorher das Root-Zertifikat eurer CA importiert wurde.

Certificate Authority (CA) erstellen

Zu Beginn wird die Certificate Authority generiert. Dazu wird ein geheimer Private Key erzeugt:

```
openssl genrsa -aes256 -out ca-key.pem 2048
```

Der Key trägt den Namen „ca-key.pem“ und hat eine Länge von 2048 Bit. Wer es besonders sicher haben will, kann auch eine Schlüssellänge von 4096 Bit angeben. Die Option „-aes256“ führt dazu, dass der Key mit einem Passwort geschützt wird. Die Key-Datei der CA muss besonders gut geschützt werden. Ein Angreifer, der den Key in die Hände bekommt, kann beliebig gefälsche Zertifikate ausstellen, denen die Clients trauen. Die Verschlüsselung dieses Keys mit einem Passwort gibt zusätzlichen Schutz. Das gewünschte Passwort wird bei der Generierung abgefragt.

Einen geheimen Key für die CA gibt es nun also schon – fehlt noch das Root-Zertifikat, das von den Clients später importiert werden muss, damit die von der CA ausgestellten Zertifikate im Browser als gültig erkannt werden. Das Root-Zertifikat „ca-root.pem“ wird mit folgendem Befehl erzeugt: (ggf. wird das Passwort für den vorher erstellen Key abgefragt!)

```
openssl req -x509 -new -nodes -extensions v3_ca -key ca-key.pem -days 1024 -out ca-root.pem -sha512
```

In diesem Fall wird die CA 1024 Tage lang gültig bleiben. Während der Generierung werden das Passwort für die CA und einige Attribute abgefragt (hier ein Beispiel):

```
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:BY
Locality Name (eg, city) []:Landshut
Organization Name (eg, company) [Internet Widgits Pty Ltd]:trashserver.net
Organizational Unit Name (eg, section) []:IT
Common Name (eg, YOUR name) []:trashserver.net
Email Address []:sslmaster@domain.com
```

Root-Zertifikat auf den Clients importieren

Damit ein Rechner die selbst ausgestellten Zertifikate akzeptiert, muss auf diesem Rechner das Root-Zertifikat (Public Key der CA) importiert worden sein. Die Root-CA Datei ist „ca-root.pem“.

Mozilla Firefox / Thunderbird

Mozilla Firefox verwaltet Zertifikate selbst. Ein neues Zertifikat wird importiert unter „Einstellungen => Erweitert => Zertifikate => Zertifikate anzeigen => Zertifizierungsstellen => Importieren“. Wählt die Datei „ca-root.pem“ aus. „Wählt die Option „Dieser CA vertrauen, um Websites zu identifizieren“.

Chromium / Google Chrome

„Einstellungen“ => „Erweiterte Einstellungen anzeigen“ (unten) => „HTTPS/SSL“ => „Zertifikate verwalten“ => „Zertifizierungsstellen“ => „Importieren“ => „ca-root-pem“ auswählen => „Diesem Zertifikat zur Identifizierung von Websites vertrauen“

Debian, Ubuntu

```
sudo cp ca-root.pem /usr/share/ca-certificates/myca-root.crt
sudo dpkg-reconfigure ca-certificates
```

=> Neue Zertifikate akzeptieren

Ein neues Zertifikat ausstellen

Die CA ist nun fertig und kann genutzt werden. Zeit, das erste Zertifikat auszustellen!

Grundlage ist immer ein privater Schlüssel. Wie auch bei der CA wird dieser Private Key erzeugt:

```
openssl genrsa -out zertifikat-key.pem 4096
```

An dieser Stelle ein Passwort zu setzen ist in den meisten Fällen nicht besonders sinnvoll. Ein Webserver, der des Zertifikat verarbeitet, müsste bei jedem Start das Passwort abfragen. Das ist in der Praxis mehr lästig und hinderlich als nützlich. (=> Passwortfelder einfach leer lassen). Die Schlüssellänge wurde hier auf paranoide 4096 Bit gesetzt. 2048 sind auch okay ;)

Nun wird eine Zertifikatsanfrage erstellt, bei der wieder einige Attribute abgefragt werden. Besonderheit ist hier: Das Feld „Common Name“ *muss* den Hostnamen des Servers tragen, für den es gültig sein soll. Soll z.B. die Verbindung zum Rechner mit der IP-Adresse „192.168.2.2“ mit dem Zertifikat abgesichert werden, muss die IP-Adresse hier angegeben werden. Soll das Zertifikat dagegen für die Domain thomas-leister.de gelten, muss das ebenso eingetragen werden. Es ist auch möglich, sog. Wildcard-Zertifikate zu erstellen. Wird z.B. „*****.thomas-leister.de“ als Common Name angegeben, gilt das Zertifikat für alle Domains von thomas-leister.de, also login.thomas-leister.de, start.thomas-leister.de usw. – nicht aber für thomas-leister.de selbst. Das Challenge Passwort wird nicht gesetzt (leer lassen).

```
openssl req -new -key zertifikat-key.pem -out zertifikat.csr -sha512
```

Sobald die Zertifikatsanfrage „zertifikat.csr“ fertiggestellt ist, kann sie von der CA verarbeitet werden. Dabei entsteht der öffentliche Schlüssel (Public Key) zum angefragten Zertifikat. Dieser wird zusammen mit dem Private Key des Zertifikats für die Verschlüsselung benötigt.

Mit folgendem Befehl wird ein Public Key „zertifikat-pub.pem“ausgestellt, der 365 Tage lang gültig ist:

```
openssl x509 -req -in zertifikat.csr -CA ca-root.pem -CAkey ca-key.pem -CAcreateserial -out zertifikat-pub.pem -days 365 -sha512
```

(Das Passwort für die CA wird erneut abgefragt.) Die Zertifizierungsanfrage zertifikat.csr kann gelöscht werden – sie wird nicht mehr benötigt. Übrig bleiben Private Key und Public Key des neuen Zertifikats (zertifikat-key.pem und zertifikat-pub.pem) sowie Private- und Public Key der CA (ca-key.pem und ca-root.pem)

Die Zertifikate nutzen

In der Webserver-Konfiguration müssen üblicherweise drei Zertifikatsdateien angegeben werden:

- Private Key des Zertifikats (zertifikat-key.pem)
- Public Key des Zertifikats (zertifikat-pub.pem)
- Public Key der CA (ca-root.pem)

Der Public Key der CA kann auch an die Public Key Datei des Zertifikats angehängt werden:

```
cat ca-root.pem >> zertifikat-pub.pem
```

Diese Integration ist immer dann nötig, wenn es keinen Parameter in der Konfiguration gibt, bei dem man das Rootzertifikat einer CA angeben kann – beim XMPP Server Prosody und beim Webserver Nginx ist das z.B. der Fall: Hier können nur Public- und Private Key des Zertifikats angegeben werden.

Wie ihr SSL/TLS für euren Webserver nutzt könnt ihr in diesen beiden Beiträgen nachlesen:

- Apache Webserver mit SSL
- Nginx Webserver mit SSL

Quellen: http://datacenteroverlords.com/2012/03/01/creating-your-own-ssl-certificate-authority/

Post published on 1. September 2014 | Last updated on 17. Dezember 2014
Tags: #CA #Open Source #Server #SSL #TLS #Ubuntu #Verschlüsselung

Diesen Blog unterstützen

Wenn Dir der Beitrag gefallen hat, freue ich mich über einen kleinen Obolus :-)

PayPal-Seite: https://www.paypal.me/ThomasLeister
Meine **Bitcoin**-Adresse: **15z8 QkNi dHsx q9WW d8nx W9XU hsdf Qe5B 4s**

Siehe auch: Unterstützung

Diesen Beitrag teilen

spenden spenden teilen e-mail twittern teilen teilen teilen mitteilen

Informationen zum Autor

Thomas Leister

Geb. 1995, Kurzhaar-Metaller, Geek und Blogger. Nutzt seit Anfang 2013 ausschließlich Linux auf Desktop und Servern. Student der Automobilinformatik an der Hochschule für angewandte Wissenschaften in Landshut.

Weitere Beiträge zum Thema

- SSH Anmeldung über privaten Schlüssel und Passwortauthentifizierung abschalten
- Android startet nicht mehr nach Verschlüsselung und neuer ROM – Lösung
- Nginx: PHP-FPM unter Ubuntu Server 14.04 installieren und einrichten
- Einzeiler: Dateien mal eben über's Netzwerk schubsen