OpenSSL

Kleine OpenSSL FAQ Ecke

- OpenSSL FAQ
- So unterscheiden sich Clientzertifikate von Serverzertifikaten

Creating a Self-Signed Certificate With OpenSSL

Creating a Private Key

First, we'll create a private key. A private key helps to enable encryption, and is the most important component of our certificate.

Let's create a password-protected, 2048-bit RSA private key (domain.key) with the openssl command:

```
openssl genrsa -des3 -out domain.key 2048
```

We'll enter a password when prompted. The output will look like:

```
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................+++++
.........+++++
e is 65537 (0x010001)
Enter pass phrase for domain.key:
Verifying - Enter pass phrase for domain.key:
```

If we want our private key unencrypted, we can simply remove the -des3 option from the command.

Nun wird die Passphrase aus dem Schlüssel entfernt.

```
root@linux# openssl rsa -in domain.key -out domain.key
Enter pass phrase for serverkey.pem: jaja
writing RSA key
```

Creating a Certificate Signing Request

If we want our certificate signed, we need a certificate signing request (CSR). The CSR includes the public key and some additional information (such as organization and country).

Let's create a CSR (domain.csr) from our existing private key:

```
openssl req -key domain.key -new -out domain.csr
```

We'll enter our private key password and some CSR information to complete the process. The output will look like:

```
Enter pass phrase for domain.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:AU
State or Province Name (full name) [Some-State]:stateA
Locality Name (eg, city) []:cityA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:companyA
Organizational Unit Name (eg, section) []:sectionA
Common Name (e.g. server FQDN or YOUR name) []:domain
Email Address []:email@email.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

An important field is "Common Name," which should be the exact Fully Qualified Domain Name (FQDN) of our domain.

"A challenge password" and "An optional company name" can be left empty.

Creating a Self-Signed Certificate

A self-signed certificate is [b]a certificate that's signed with its own private key[/b]. It can be used to encrypt data just as well as CA-signed certificates, but our users will be shown a warning that says the certificate isn't trusted.

Let's create a self-signed certificate (domain.crt) with our existing private key and CSR:

```
openssl x509 -signkey domain.key -in domain.csr -req -days 365 -out
domain.crt
```

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.

Erstellen der CA

Legen Sie zunächst ein Verzeichnis an, in dem Sie das Zertifikat ablegen wollen. Wir nehmen in unserem Beispiel /root/ca:

```
root@linux# mkdir /root/ca
root@linux# cd /root/ca
```

Die Gültigkeit setzen wir mit 10 Jahren bewusst sehr hoch an. Läuft die CA aus, so werden nämlich auch alle damit signierten Serverzertifikate ungültig. Die CA enthält einen geheimen Schlüssel, welcher automatisch erzeugt und in der Datei cakey.pem abgelegt wird. Das CA-Zertifikat wird nach cacert.pem geschrieben. Der folgende Befehl erzeugt das einen Schlüssel für das Zertifikat mit einer Länge von 2048 Bit:

```
root@linux# openssl req -new -x509 -newkey rsa:2048 -keyout cakey.pem -out
cacert.pem -days 3650
Generating a 2048 bit RSA private key
.....................................................................
.....................................................................
.........................................+++
.....................................+++
writing new private key to 'cakey.pem'
```

Wer den geheimen Schlüssel der CA kennt, kann damit beliebige Serverzertifikate signieren. Deshalb wird diese Schlüsseldatei nicht im Klartext auf der Festplatte abgelegt, sondern mit einer Passphrase verschlüsselt. Diese Passphrase benötigen Sie immer dann, wenn Sie mit der CA neue Zertifikate ausstellen wollen:

```
Enter PEM pass phrase: wrzlprmpft
Verifying - Enter PEM pass phrase: wrzlprmpft
```

Nun werden Sie gebeten, Daten einzugeben, welche die CA identifizieren. Diese werden dem Client angezeigt, wenn er aufgefordert wird, das Zertifikat zu akzeptieren oder abzulehnen. Der Code für Deutschland ist DE. Wenn Sie ein Feld leerlassen möchten, so geben Sie einen Punkt ein. Ansonsten wird der in eckigen Klammern stehende Defaultwert eingetragen:

```
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: DE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Muenchen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Hinz und Kunz AG
Organizational Unit Name (eg, section) []:.
```

Das Feld Common Name (CN) ist hier der offizielle Name der Zertifizierungsstelle. Für Ihre eigene CA können Sie einfach Ihren eigenen Namen eintragen:

```
Common Name (eg, YOUR name) []: Adam Hinz
Email Address []: adam@hinzag.eu
```

Fertig. Folgende zwei Dateien sind entstanden:

```
root@linux# ll
insgesamt 9
drwxr-xr-x   2 root root  112 2006-04-30 12:08 .
drwx------  12 root root  600 2006-04-30 11:54 ..
-rw-r--r--   1 root root 1212 2006-04-30 12:08 cacert.pem
-rw-r--r--   1 root root  963 2006-04-30 12:08 cakey.pem
```

Vorsichtshalber sollten Sie die Rechte so setzen, dass die Schlüsseldatei nur für root lesbar ist:

```
root@linux# chmod 600 cakey.pem
```

Sie können nun ausprobieren, ob sie den Schlüssel mit der Passphrase wieder öffnen können:

```
root@linux# openssl rsa -in cakey.pem -noout -text
Enter pass phrase for cakey.pem: wrzlprmpft
Private-Key: (1024 bit)
modulus:
    00:d5:a5:37:51:e9:d9:fa:e3:97:e7:46:b2:88:1a:
    b5:46:80:47:76:14:ae:2b:8b:3e:35:5c:ab:15:84:
    53:d9:63:2e:7f:08:4b:ec:77:db:02:45:f8:c7:46:
    58:cd:2d:f9:29:4d:96:3d:d8:6c:5d:9f:79:8a:04:
    cf:b7:3a:89:da:a9:63:9f:44:b3:83:cf:0d:70:7d:
```

usw...

Schlüssel für das Serverzertifikat erzeugen

Nachdem wir nun eine eigene CA haben, kann diese nun endlich für unseren Server ein Zertifikat herausgeben. Dazu erzeugen wir zunächst einen 2048 Bit langen RSA Schlüssel, der mit AES 128 verschlüsselt auf der Platte abgelegt wird (ja wirklich, auch hier wieder ein verschlüsselter Schlüssel). Die Passphrase muss diesmal nicht sonderlich geheim sein, da wir sie ohnehin im Anschluss wieder entfernen werden. OpenSSL lässt allerdings keine leere Phrase zu:

```
root@linux# openssl genrsa -out serverkey.pem -aes128 2048 -days 730
Generating RSA private key, 2048 bit long modulus
....+++
.......................................+++
e is 65537 (0x10001)
Enter pass phrase for serverkey.pem: jaja
Verifying - Enter pass phrase for serverkey.pem: jaja
```

So. Nun entfernen wir die Passphrase wieder. Warum? Der Serverdienst (Apache, Cyrus, etc.) muss schließlich in der Lage sein, den Schlüssel ohne Ihr Zutun zu lesen. Oder wollen Sie bei jedem Booten des Servers ein Passwort eingeben müssen?

```
root@linux# openssl rsa -in serverkey.pem -out serverkey.pem
Enter pass phrase for serverkey.pem: jaja
writing RSA key
```

Certificate Signing Request erzeugen

Der nächste Schritt zum eigenen Zertifikat ist ein CSR. Dies muss dann nur noch von der CA signiert werden. Hier sind wieder Angaben analog zum Erstellen der CA nötig, was oft Verwirrung stiftet. Die allgemeinen Daten kann man ggfl. gleich wie oben eingeben:

```
root@linux#  openssl req -new -key serverkey.pem -out req.pem -nodes -config
openssl.cnf
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]: DE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:Muenchen
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Hinz und Kunz AG
Organizational Unit Name (eg, section) []:.
```

ACHTUNG, jetzt kommt das Wichtige: Beim Serverzertifikat ist der Common Name von entscheidender Bedeutung. Hier muss der DNS-Name stehen, unter dem der Client den Server anspricht! Wird das Zertifikat für eine HTTPS-Verbindung zu www.hinzag.eu verwendet, so muss der Common Name eben genau www.hinzag.eu heißen. Anderfalls wird der Browser das Zertifikat nicht akzeptieren, da er

davon ausgehen muss, auf dem falschen Server gelandet zu sein.

```
Common Name (eg, YOUR name) []: www.hinzag.eu
Email Address []: adam@hinzag.eu
```

Weitere Optionen kann man einfach leer lassen:

```
A challenge password []:
An optional company name []:
```

Mittlerweile tummeln sich schon vier Dateien in unserem Verzeichnis:

```
root@linux# ll
insgesamt 17
drwxr-xr-x   2 root root  168 2006-04-30 12:29 .
drwx------  12 root root  600 2006-04-30 11:54 ..
-rw-r--r--   1 root root 1212 2006-04-30 12:08 cacert.pem
-rw-------   1 root root  963 2006-04-30 12:08 cakey.pem
-rw-r--r--   1 root root 1017 2006-04-30 12:29 req.pem
-rw-r--r--   1 root root 1679 2006-04-30 12:21 serverkey.pem
```

OpenSSL-Konfiguration anpassen

Leider kann man bei OpenSSL nicht alle Daten als Kommandozeilenargumente übergeben. Einige Einstellungen muss man lästigerweise in der Datei /etc/ssl/openssl.cnf ändern, bevor man signieren kann. Öffnen Sie diese Datei und passen Sie folgende Zeilen in der Sektion [CA_default] an:

```
/etc/ssl/openssl.cnf:
dir             = .             # Where everything is kept
new_certs_dir   = $dir          # default place for new certs
private_key     = $dir/cakey.pem # The private key
RANDFILE        = $dir/.rand    # private random number file
default_days    = 3650          # how long to certify for
```

Das Feld default_days ist auf 365 Tage voreingestellt und gibt die Gültigkeit des Zertifikates an. Abgelaufene Zertifikate sind im Übrigen ein sehr häufiges Problem. Wenn es soweit ist, kennt sich damit nämlich schon lange keiner mehr aus. Deswegen können Sie wie im Beispiel angegeben die Lebensdauer z.B. auf 10 Jahre heraufsetzen.

Wenn Sie beim Serverzertifikat keinen Bundesstaat angegeben haben, benötigen Sie noch folgende Änderung unter [policy_match]:

```
stateOrProvinceName     = optional
```

Nun muss man noch einige Dateien anlegen:

```
root@linux# echo 01 > serial
root@linux# touch index.txt
```

Serverzertifikat signieren

Kommen wir zum feierlichen Abschluss. Damit später Chrome und Chromium Ihr Serverzertifikat akzeptiert, muß ein sogenannter „Subject Alternative Names (SAN)“ ins Zertifikat eingetragen werden. Dafür wird zusätzlich eine Datei angelegt:

```
root@linux# vim oats.extensions.cnf

basicConstraints=CA:FALSE
subjectAltName=@my_subject_alt_names
subjectKeyIdentifier = hash

[ my_subject_alt_names ]
DNS.1 = fhem01.tuxnet.local
DNS.2 = fhem02.tuxnet.local
DNS.3 = fhem01-vpn.tuxnet.local
```

Unsere CA signiert nun das Zertifikat:

```
root@linux# openssl ca -in req.pem -notext -extfile oats.extensions.cnf -out
servercert.pem
Enter pass phrase for ./cakey.pem: wrzlprmpft

...

Certificate is to be certified until Apr 27 10:45:36 2016 GMT (3650 days)
Sign the certificate? [y/n]: y


1 out of 1 certificate requests certified, commit? [y/n] y
Write out database with 1 new entries
Data Base Updated
```

Zertifikate installieren

Wohin sie die Zertifikate installieren, hängt natürlich vom jeweiligen Serverdienst ab. Was allen gemeinsam ist: Sie benötigen nur die Dateien cacert.pem, servercert.pem und serverkey.pem. Die Datei cakey.pem wird nicht benötigt. Sie sollte am besten auch nicht auf dem Server liegen sondern an einer sicheren Stelle auf einem anderen Rechner.