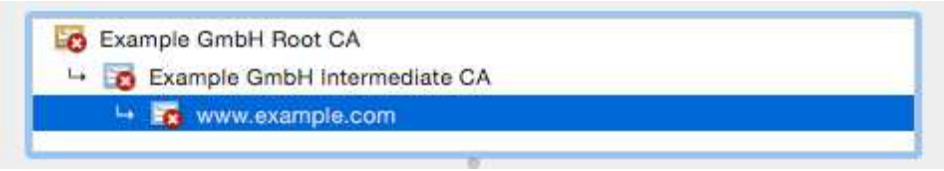
Mit der eigenen Root CA SSL Zertifikate ausstellen

Trotz kostenloser SSL Zertifikate mittels Let's Encrypt, ist es manchmal hilfreich für interne Projekte SSL Zertifikate selbst auszustellen.

Das folgende Tutorial beschreibt die Erstellung aller nötigen Schlüssel und Zertifikate um selbst Zertifikate zu erstellen bzw. zu signieren. Diese werden von gängigen Browsern zwar nicht als vertrauenswürdig eingestuft, jedoch reicht es das Root Zertifikat einmal zu den vertrauenswürdigen Zertifikaten des Systems hinzuzufügen, damit der Browser allen damit signierten Zertifikaten zu vertraut.

Es wird zunächst eine Root CA (Certificate Authority) erstellt. Diese beglaubigt das Zertifikat der Intermediate CA. Die Intermediate CA unterschreibt alle weiteren Serverzertifikate. Damit entsteht eine Chain of Trust (Root CA > Intermediate CA > Serverzertifikat), wie sie auch bei den großen CA's Anwendung findet.



Mein Dank geht an Jamie Nguyen dessen Anweisungen als Grundlage für dieses Tutorial dienten.

Root Certificate Authority (CA)

Schritt: 1: Ordnerstruktur und Dateien für die Root CA erstellen

```
mkdir /root/ca && mkdir /root/ca/certs && mkdir /root/ca/crl && mkdir /root/ca/newcerts && mkdir /root/ca
```

```
chmod 700 /root/ca/private
```

```
touch /root/ca/index.txt
```

```
echo 1000 > /root/ca/serial
```

Schritt 2: OpenSSL Konfiguration der Root CA erstellen

```
nano /root/ca/openssl.cnf
```

Folgende Anweisungen sollte die Konfigurationsdatei enthalten.

```
# OpenSSL root CA configuration file.
# Copy to `/root/ca/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default

[ CA_default ]
# Directory and file locations.
dir = /root/ca
certs = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/private/.rand

# The root key and root certificate.
private_key = $dir/private/ca.key.pem
certificate = $dir/certs/ca.cert.pem

# For certificate revocation lists.
crlnumber = $dir/crlnumber
crl = $dir/crl/ca.crl.pem
crl_extensions = crl_ext
default_crl_days = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256

name_opt = ca_default
cert_opt = ca_default
default_days = 375
preserve = no
policy = policy_strict

[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_loose ]
```

```
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits = 2048
distinguished_name = req_distinguished_name
string_mask = utf8only

# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256

# Extension to add when the -x509 option is used.
x509_extensions = v3_ca

[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName = Locality Name
0.organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName = Common Name
emailAddress = Email Address

# Optionally, specify some defaults.
countryName_default =
stateOrProvinceName_default =
localityName_default =
0.organizationName_default =
organizationalUnitName_default =
emailAddress_default =

[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
```

```
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

Schritt 3: Key für die Root CA erstellen

```
openssl genrsa -aes256 -out /root/ca/private/ca.key.pem 4096
```

Hinweis: Für den Key der CA sollte ein starkes Passwort vergeben werden!

Im Anschluss werden noch die Rechte angepasst, sodass kein anderer User außer Root den Key einsehen kann.

```
chmod 400 /root/ca/private/ca.key.pem
```

Schritt 4: Zertifikat der Root CA erstellen

Nun wird das Zertifikat der CA erstellt.

```
openssl req -config /root/ca/openssl.cnf -key /root/ca/private/ca.key.pem -new -x509 -days 7300 -sha256 -
```

Hier werden die Daten der CA abgefragt. z.B.:

```
Country Name (2 letter code) []:DE
State or Province Name (full name) []:Bavaria
Locality Name (eg, city) []:Munich
Organization Name (eg, company) []:Example GmbH
Organizational Unit Name (eg, section) []:Example GmbH Certificate Authority
Common Name (e.g. server FQDN or YOUR name) []:Example GmbH Root CA
Email Address []:admin@example.com
```

Im Anschluss werden noch die Rechte angepasst.

```
chmod 444 /root/ca/certs/ca.cert.pem
```

Alle Dateien für die Root CA sind nun erstellt. Es folgt die Intermediate CA.

Intermediate CA

Schritt 1: Ordnerstruktur und Dateien für die Intermediate CA erstellen

```
mkdir /root/ca/intermediate && mkdir /root/ca/intermediate/certs && mkdir /root/ca/intermediate/crl && mk
```

```
chmod 700 /root/ca/intermediate/private
```

```
touch /root/ca/intermediate/index.txt
```

```
echo 1000 > /root/ca/intermediate/serial
```

```
echo 1000 > /root/ca/intermediate/crlnumber
```

Schritt 2: OpenSSL Konfiguration der Intermediate CA erstellen

```
nano /root/ca/intermediate/openssl.cnf
```

Folgende Anweisungen sollte die Konfigurationsdatei enthalten.

```
# OpenSSL intermediate CA configuration file.
# Copy to `/root/ca/intermediate/openssl.cnf`.

[ ca ]
# `man ca`
default_ca = CA_default
```

```
[ CA_default ]
# Directory and file locations.
dir = /root/ca/intermediate
certs = $dir/certs
crl_dir = $dir/crl
new_certs_dir = $dir/newcerts
database = $dir/index.txt
serial = $dir/serial
RANDFILE = $dir/private/.rand

# The root key and root certificate.
private_key = $dir/private/intermediate.key.pem
certificate = $dir/certs/intermediate.cert.pem

# For certificate revocation lists.
crlnumber = $dir/crlnumber
crl = $dir/crl/intermediate.crl.pem
crl_extensions = crl_ext
default_crl_days = 30

# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256

name_opt = ca_default
cert_opt = ca_default
default_days = 375
preserve = no
policy = policy_loose

[ policy_strict ]
# The root CA should only sign intermediate certificates that match.
# See the POLICY FORMAT section of `man ca`.
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ policy_loose ]
# Allow the intermediate CA to sign a more diverse range of certificates.
# See the POLICY FORMAT section of the `ca` man page.
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
# Options for the `req` tool (`man req`).
default_bits = 2048
distinguished_name = req_distinguished_name
string_mask = utf8only
```

```
# SHA-1 is deprecated, so use SHA-2 instead.
default_md = sha256

# Extension to add when the -x509 option is used.
x509_extensions = v3_ca

[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName = Country Name (2 letter code)
stateOrProvinceName = State or Province Name
localityName = Locality Name
0.organizationName = Organization Name
organizationalUnitName = Organizational Unit Name
commonName = Common Name
emailAddress = Email Address

# Optionally, specify some defaults.
countryName_default =
stateOrProvinceName_default =
localityName_default =
0.organizationName_default =
organizationalUnitName_default =
emailAddress_default =

[ v3_ca ]
# Extensions for a typical CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ v3_intermediate_ca ]
# Extensions for a typical intermediate CA (`man x509v3_config`).
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = critical, CA:true, pathlen:0
keyUsage = critical, digitalSignature, cRLSign, keyCertSign

[ usr_cert ]
# Extensions for client certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = client, email
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, nonRepudiation, digitalSignature, keyEncipherment
extendedKeyUsage = clientAuth, emailProtection

[ server_cert ]
# Extensions for server certificates (`man x509v3_config`).
basicConstraints = CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer:always
```

```
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth

[ crl_ext ]
# Extension for CRLs (`man x509v3_config`).
authorityKeyIdentifier=keyid:always

[ ocsp ]
# Extension for OCSP signing certificates (`man ocsp`).
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid,issuer
keyUsage = critical, digitalSignature
extendedKeyUsage = critical, OCSPSigning
```

Schritt 3: Key für die Intermediate CA erstellen

```
openssl genrsa -aes256 -out /root/ca/intermediate/private/intermediate.key.pem 4096
```

Hinweis: Auch für der Key der Intermediate CA sollte durch ein starkes Passwort geschützt werden!

Im Anschluss werden noch die Rechte angepasst, sodass kein anderer User außer root den Key einsehen kann.

```
chmod 400 /root/ca/intermediate/private/intermediate.key.pem
```

Schritt 4: Certificate Signing Request (CSR) der Intermediate CA erstellen

Nun muss das Zertifikat der Intermediate CA von der Root CA unterschrieben werden. Hierzu erstellt man zuerst einen CSR (Certificate Signing Request)

```
openssl req -config /root/ca/intermediate/openssl.cnf -new -sha256 -key /root/ca/intermediate/private/int
```

Auch hier werden wieder die Daten der Certificate Authority angegeben. Der Common Name muss allerdings zwingend anders lauten (vgl. Root CA, Schritt 4).

```
Country Name (2 letter code) []:DE
State or Province Name (full name) []:Bavaria
Locality Name (eg, city) []:Munich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Example GmbH
Organizational Unit Name (eg, section) []:Example GmbH Certificate Authority
Common Name (e.g. server FQDN or YOUR name) []:Example GmbH Intermediate CA
Email Address []:admin@example.com
```

Schritt 5: CSR der Intermediate Certificate Authority unterschreiben

Ist der CSR erstellt, kann nun das Zertifikat von der Root CA unterschrieben werden.

```
openssl ca -config /root/ca/openssl.cnf -extensions v3_intermediate_ca -days 3650 -notext -md sha256 -in
```

Im Anschluss werden auch hier wieder die Rechte angepasst.

```
chmod 444 /root/ca/intermediate/certs/intermediate.cert.pem
```

Die eigene Certificate Authority ist nun einsatzbereit.

Schritt 6: Chainfile erstellen

Um die Vertrauenswürdigkeit eines Zertifikats zu prüfen braucht der Browser imm die komplette Zertifikatskette (Chain). Das Chainfile enthält die Zertifikate von Intermediate CA und Root CA.

```
cat /root/ca/intermediate/certs/intermediate.cert.pem /root/ca/certs/ca.cert.pem > /root/ca/intermediate/
```

Serverzertifikat erstellen

Sind alle Files erstellt können nun Serverzertifikate unterschrieben werden. Meist erfolgt die Erstellung eines privaten Schlüssels sowie eines CSR auf dem jeweiligen Rechner auf dem das Zertifikat eingesetzt werden soll. Am einfachsten geht das mit folgendem Befehl, welcher Schlüssel und CSR in einem Zug erstellt.

Schritt 1: Schlüssel und CSR erstellen

```
openssl req -nodes -newkey rsa:2048 -sha256 -keyout ~/www.meine-domain.com.key.pem -out ~/www.meine-domai
```

Der Befehl erstellt einen „servertauglichen“ Schlüssel ohne Passwort, damit bei einem Neustart des Webservers (z.B. Apache) kein Passwort abgefragt wird, und ein CSR.

Nun müssen die Daten für das Zertifikat eingegeben werden. Wichtig ist, dass hier unter Common Name die Domain angegeben wird, für welche das Zertifikat gelten soll.

```
Country Name (2 letter code) []:DE
State or Province Name (full name) []:Bavaria
Locality Name (eg, city) []:Munich
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Hallo Welt GmbH
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:www.meine-domain.com
Email Address []:
```

Schritt 2: Serverzertifikat unterschreiben

Im zweiten Schritt muss der CSR auf den Rechner der CA übertragen werden (z.B. in /root), wo er mit folgendem Befehl unterschrieben wird.

```
openssl ca -config /root/ca/intermediate/openssl.cnf -extensions server_cert -days 375 -notext -md sha256
```

Schritt 3: Serverzertifikat installieren

Ist das Zertifikat erstellt, kann es auf dem Server installiert werden. Hierfür braucht der Server das in Schritt 2 erstellte Zertifikat, den Key, der den Server nie verlassen hat, und das Chainfile. Je nach Webserver werden diese Files in der Konfiguration entsprechend angegeben (hier Apache2)

```
SSLCertificateFile /path/to/www.meine-domain.com.cert.pem
SSLCertificateKeyFile /path/to/www.meine-domain.com.key.pem
SSLCertificateChainFile /path/to/ca-chain.cert.pem
```

Quelle: https://jamielinux.com/docs/openssl-certificate-authority/index.html

Dieser Beitrag wurde am 27. Dezember 2015 [http://blog.mansshardt.net/mit-der-eigenen-root-ca-ssl-zertifikate-ausstellen/] von Markus Mansshardt in IT-Sicherheit veröffentlicht. Schlagworte: ca, ssl, zertifikate.

Über Markus Mansshardt

Markus Manßhardt befasst sich seit einigen Jahren im privaten und beruflichen Umfeld mit System- und Netzwerkadministration sowie den Themen Virtualisierung, Cloud und IT-Sicherheit. Sein Schwerpunkt liegt bei der Administration von Linux Systemen und deren Intergration in Mac und Windows Umgebungen.

Zeige alle Beiträge von Markus Mansshardt →

---

3 Gedanken zu „Mit der eigenen Root CA SSL Zertifikate ausstellen“

Casper
10. Januar 2017 um 17:17

Üblicherweise schreibe ich keine Kommentare, aber hier mache ich eine Ausnahme :)
Tolles Tutorial! Daumen hoch!

Markus Mansshardt Beitragsautor
10. Januar 2017 um 19:34

Na das liest man doch gern! Danke für dein Feedback Casper!

Guest
9. September 2017 um 7:21

Wirklich sehr nice. :)

Ein kleiner Bugfix:

touch /root/ca/index.txt.attr
