# Eigene CA bauen und Zertifikate ausrollen

Grundegedanke war hier die ganze Hürde mit gekauften Zertifikaten zu umgehen, und das ganze noch einfacher zu gestalten. Für Organisationen intern, aber auch extern. Mit Installation von deinem DEB Paket in Ubuntu, und in Windows mit einem MSI. Wir erstellen eine eigene CA mit einem Wildcardzertifikat. Somit benötigen wir pro Domäne nur ein Zertifikat und können dies auf allen Geräten und Rechnern installieren. Daraus kann man natürlich auch Anmeldezertifikate (p12) erstellen, um die Sicherheit in relevanten Seiten hinauf zu schrauben.

Spenden

Hauseigenes Apt-Repo: https://apt.iteas.at

## Erstellen der eigenen CA

**Für's erste hier mal einige nützliche OpenSSL-Befehle:**

Zertifikat anzeigen lassen:

```
openssl x509 -in /etc/ldap/ssl/01cacert.pem -noout -text
```

Auflisten aller Zertifikate im globalen CA-Speicher von Java:

```
keytool -list -keystore /etc/ssl/certs/java/cacerts
```

Firefoxcertifikate anzeigen lassen:

```
NSS_DEFAULT_DB_TYPE="sql" certutil -d ~/.mozilla/firefox/*.default -L
```

Firefoxcertifikat hinzufügen:

```
certutil -A -d sql:$HOME/.mozilla/firefox/2gjkcvvk.default  -i cacert.pem -n
"tux.at Wildcard Selfsigned from ITEAS IT Services" -t TCP,TCP,TCP
```

Firefoxcertifikat löschen:

```
certutil -D -d sql:$HOME/.mozilla/firefox/2gjkcvvk.default  -i cacert.pem -n
"tux.at Wildcard Selfsigned from ITEAS IT Services" -t TCP,TCP,TCP
```

CA Zertifikat in Chrome/Chromium importieren

```
ertutil -d sql:$HOME/.pki/nssdb -A -n '' -i cacert.pem -t TCP,TCP,TCP
```

Alle Zertifiakte des Systems anzeigen:

```
locate .pem | grep "\.pem$" | xargs -I{} openssl x509 -issuer -enddate -
```

```
noout -in {}
```

PFX Inhalt anzeigen:

```
openssl pkcs12 -info -in testcert.pfx
```

## Vorbereiten des Servers

Wir verwenden hier ein Ubuntu 18.04 als CA Server. Sehr gut eigent sich auch ein LXC-Container oder Docker. Auch kann man die natürlich überall wo es passt dazu installieren. Folgende Pakete sollten installiert sein.

```
apt-get install ca-certificates ca-certificates-java ca-certificates-mono
openssl gnutls-bin libnss3-tools
```

Nun passen wir noch unsere `/etc/ssl/openssl.cnf` an. Mit diesem File kann man das meiste komplett automatisieren und vorallem für Google Chrome anpassen. Google Chrome ist der einzige Browser der spezielle Richtlinien für das Zertifikat verwendet. Das File nach eigenem Ermessen abändern.

Thread about

openssl.cnf

```
...
[ CA_default ]
...
default_days    = 3650                  # how long to certify for
default_crl_days= 30                    # how long before next CRL
default_md      = default               # use public key default MD
preserve        = no                    # keep passed DN ordering
...
# Extension copying option: use with caution.
# copy_extensions = copy
...
[ v3_req ]
...
subjectAltName = @alt_names
...
[ alt_names ]
DNS.1 = *.osit.cc
DNS.2 = localhost
DNS.3 = ip6-localhost
IP.1 = 127.0.0.1
IP.2 = ::1
...
[ v3_ca ]
...
subjectAltName = @alt_names
```

```
...
```

Alle anderen Anpassungen im File sind zwecks Automatisierung empfohlen. Hier das ganze File zum Download: openssl.cnf

## Erstellen der CA und bauen des Zertifikates

Man erstellt sich ein Verzeichnis wo man seine CA(s) baut, und erstellt diese. Den Wizard dabei einfach folgen.

```
mkdir Zertifizierungsstelle
cd Zertifizierungsstelle
/usr/lib/ssl/misc/CA.pl -newca
```

Im aktuellen Verzeichnis sollte jetzt ein neues Verzeichnis **demoCA** mit einigen Dateien und weiteren Unterverzeichnissen entstanden sein. Folgende davon sind wichtig:

- **cacert.pem** - Das öffentliche Zertifikat der CA. Diese Datei sollte auf allen Clients importiert werden. Danach vertrauen diese Clients automatisch allen von der CA ausgestellten Zertifikaten.
- **certs/** - Das Verzeichnis in dem die signierten Zertifikate untergebracht werden.
- **private/** - Das Verzeichnis für die privaten Schlüssel.
- **private/cakey.pem** - Der private Schlüssel der CA. Dieser sollte die CA nie verlassen, außer zu Backup-Zwecken. (Und auch dann gesondert gesichert werden.)
- **index.txt** - Sogenannte Datenbank der CA mit einer Liste aller ausgestellten Zertifikate und deren Status.

### Zertifikate signieren

Mit dieser CA fängt man nun an die eigentlichen Zertifikate zu signieren, die man dann bspw. auf seinen Servern einsetzt. Auch hierfür verwendet man das CA.pl-Skript. Das Signieren läuft in zwei Stufen ab. Als erstes erstellt man einen neuen privaten Schlüssel und einen „Zertifikatsantrag“ (CA.pl -newreq). Anschließend signiert man den Zertifikatsantrag (CA.pl -sign). Damit das Skript funktioniert, muss man sich im selben Verzeichnis befinden in dem man auch die CA erstellt hat, also eines oberhalb von demoCA.

Bei „Common Name“ trägt man dann sein Wildcard, z.B. `*.osit.cc` ein.

```
/usr/lib/ssl/misc/CA.pl -newreq
```

Beim Signieren des Zertifikatsantrags wird nicht nur die Gültigkeitsdauer (s. Erstellen der neuen CA) sondern auch der Verwendungszweck festgelegt. Per Default werden Zertifikate ohne besonderen Verwendungszweck ausgestellt. Für Webserver empfiehlt es sich allerdings den Zertifikatstyp und Verwendungszweck als zusätzliche X.509 v3-Erweiterungen hinzuzufügen. Dazu muss wieder die OpenSSL-Konfiguration `/etc/ssl/openssl.cnf` angepasst werden.

```
...
```

```
[ usr_cert ]
...
basicConstraints=critical,CA:FALSE
...
nsCertType = server
...
keyUsage = critical,keyEncipherment
extendedKeyUsage = serverAuth
...
```

Jetzt wird der Zertifikatsantrag signiert und das neue Zertifkat mit den oben konfigurierten Attributen (X.509 v3-Erweiterungen) ausgestellt.

```
/usr/lib/ssl/misc/CA.pl -sign
```

Hat man die „copy_extensions = copy“ von oben aktiviert (nicht empfohlen), bekommt man folgende Fehlermeldung:

```
failed to update database
TXT_DB error number 2
```

Das Problem lies sich umgehen in dem ich die Datei `demoCA/index.txt.attr` bearbeitete und folgendes änderte:

```
-- unique_subject = yes
++ unique_subject = no
```

Die Passphrase im Signierschritt ist natürlich die des privaten CA-Schlüssels, und nicht die des Server-Schlüssels der gerade erstellt worden ist. Außerdem muss man darauf achten, am Ende wirklich mit „y“ zu bestätigen, sonst bricht das Skript mit einem Fehler ab.

Im aktuellen Verzeichnis - immer noch das von **demoCA** übergeordnete - befinden sich jetzt die Dateien:

- **newreq.pem** - Zertifikatsantrag
- **newcert.pem** - Signiertes öffentliches Zertifikat
- **newkey.pem** - Privater Schlüssel

**newreq.pem** ist inzwischen nutzlos geworden und kann gelöscht werden. Wichtig sind nur die beiden Dateien **newcert.pem** und **newkey.pem**. Diesen sollte man aussagekräftigere Namen geben und sie zur Archivierung in das jeweilige **demoCA**-Unterverzeichnis verschieben. Wer das Schlüsselpaar auf einem Server benutzen möchte und nicht bei jedem Server-Start die Passphrase des privaten Schlüssels eingeben will, kann dieses noch vorher aus dem privaten Schlüssel löschen. In diesem Fall sollte man aber auf jeden Fall darauf achten, dass der private Schlüssel ohne Passphrase durch restriktive Dateirechte[4] auf dem Server vor neugierigen Blicken geschützt wird:

```
penssl rsa -in newkey.pem -out newkey-without-password.pem
mv newkey-without-password.pem demoCA/private/worpswede_key.pem
mv newcert.pem demoCA/certs/worpswede_cert.pem
```

Diese zwei Dateien sind es auch, die der betreffende Server zum SSL-Betrieb benötigt. Die Datei **newkey.pem** kann jetzt ebenfalls gelöscht werden (nachdem man sich davon überzeugt hat, dass mit der Umwandlung alles geklappt hat, und **newkey-without-password.pem** nicht etwa leer ist).

Wo diese Dateien auf dem Server letztendlich hinkopiert werden müssen, hängt vom benutzten Server ab. Üblich sind aber die Verzeichnisse **/etc/ssl/certs/** und **/etc/ssl/private/**.

# CA global in Ubuntu einspielen

Das CA ganz einfach unter `/usr/local/share/ca-certificates/` ablegen. Mit dem nachfolgenden Befehl wird das Zertifikat in den globalen Zertifikatsspeicher des Systems importiert.

```
update-ca-certificates
```

Alternativ interaktiv: `dpkg-reconfigure ca-certificates`

# Automatische Installation in Firefox (Linux und Windows)

Folgendes Tool muss in Ubuntu installiert werden.

```
apt install libnss3-tools
```

## Policies in Firefox (Autoimport global)

Mit einem File ist es möglich sämtlich Dinge wie Berechtigungen, fixe Bookmarks und vieles vieles mehr dem System für alle Benutzer vor zu geben. Man kann Dinge Sperren, und auch Zertifikate ausrollen. Zertifikate funktionieren mit fully qualified path erst ab Firefox Version 65. Das gilt auch für Windows.

### Beispielfile, Syntax getestet auf KDE Neon 18.04 LTS

```
cat /usr/lib/firefox/distribution/policies.json
```

policies.json

```
{

 "policies": {
   "BlockAboutAddons": true,
   "BlockAboutConfig": true
 }
}
```

Beispiel zwei, mit Zertifikat:

policies.json

```
{

 "policies": {
   "BlockAboutAddons": true,
   "BlockAboutConfig": true,
      "Certificates": {
      "ImportEnterpriseRoots": true,
      "Install": ["/usr/local/share/ca-certificates/osit.cc-wildcard-
selfsigned-cacert.crt"]
    }
 }
}
```

Und noch ein Beispiel:

policies.json

```
{
   "policies":{
      "DisableBuiltinPDFViewer":true,
      "Extensions":{
         "Install":[
"https://addons.mozilla.org/firefox/downloads/file/1672871/ublock_origi
n-*.xpi",
"https://addons.mozilla.org/firefox/downloads/file/879506/ip_address_an
d_domain_information-*.xpi",
"https://addons.mozilla.org/firefox/downloads/file/1205950/keepassxc_br
owser-*.xpi"
         ]
      },
      "Certificates":{
         "ImportEnterpriseRoots":true,
         "Install":[
            "/usr/local/share/ca-certificates/osit.cc-wildcard-
selfsigned-cacert.crt",
            "/usr/local/share/ca-certificates/osit.cc-
Fortinet_CA_SSL_deepinspection.crt"
         ]
      }
   }
}
```

Man kann auch Zertifikate in dem Verzeichnis des Benutzers ablegen `~/.mozilla/certificates`. Diese werden dann ohne den gesamten Pfad unter „Install“ vermerkt und auch automatisch

installiert. Der globale Zertifikatsordner von Firefox in Ubuntu `/usr/share/ca-certificates/mozilla` funktioniert aus unverfindlichen Gründen nicht. Daher die Empfehlung immer den fully qualified path verwenden.

#### Beispielfile, Syntax getestet auf Windows 10

```
cat C:\Program Files\Mozilla Firefox\distribution\policies.json
```

policies.json

```
{

 "policies": {
   "BlockAboutAddons": true,
   "BlockAboutConfig": true,
      "Certificates": {
      "ImportEnterpriseRoots": true,
      "Install": ["C:\\Company\\bla.crt", "bla.crt"]
    }
 }
}
```

Auch hier gibt es zwei Möglichkeiten. Einmal ein fully qualified path, oder im Ordner `C:\Program Files\Mozilla Firefox\distribution\certificates`. Auch hier funktioniert das erst ab Firefox Version 65.

## Automatische Installation in Google Chrome, Chromium (Linux und Windows)

#### Policies in Chrome (Autoimport global)

## Fertige Pakete für die Installation der CA am Client (Linux und Windows)

## Import in Android

Fix Me!

https://forum.xda-developers.com/google-nexus-5/help/howto-install-custom-cert-network-t2533550 (Methode 1)

# Links

- Komplette Doku gibt es hier
- Mozillabeitrag Windows10
- Mozillabeitrag Ubuntu Linux
- https://thomas-leister.de/ca-zertifikat-importieren-linux-windows/

In Windows muss der Ordner Stammzertifizierungsstellen für das CA manuel ausgewählt werden um in Edge oder IE zu trusten. Das trusted wie es aussieht auch Chrome und Firefox, das wird aber nochmal getestet.

From:
https://deepdoc.at/dokuwiki/ - **DEEPDOC.AT - enjoy your brain**

Permanent link:
**https://deepdoc.at/dokuwiki/doku.php?id=server_und_serverdienste:eigene_ca_bauen_und_zertifikate_ausrollen&rev=1614865909**

Last update: **2021/03/04 14:51**