SLAC 2024

Zertifikate

Es geht nicht ohne, aber meist auch nicht mit!

Daniel van Soest

VORSTELLUNG

Daniel van Soest

- Jahrgang 1981
- Über 20 Jahre IT
- Publikationen

INHALT

01

GESCHICHTE

Wie alles begann …

VERÖFFENTLICHUNG

*"X.509 wurde erstmals **1988** veröffentlicht. Die Entwicklung von X.509 begann in Verbindung mit dem X.500-Standard, der **nie vollständig** implementiert wurde.“*

[Quelle: Wikipedia]

ENTWICKLUNG

1988 Veröffentlichung

1996 Erweiterung v3

2008 Letzte Überarbeitung

02 THEORIE

Von A wie Attribute, bis Z wie Zertifikate.

ZUTATENLISTE

Benutzer

Schlüssel

→ *private key*

Zertifikatsignierungsanforderung

→ *certificate singing request = CSR*

Registrierungsstelle

→ *registration authority = RA*

Zertifizierungsstelle

→ *certificate authority = CA*

Zertifikat

→ *public key*

Validierungsdienst

→ *validation authority = VA*

ZERTIFIKATSARTEN

SSL/TLS

Server Authentifizierung

CODE-SIGNING

Signierung von Quellcode

Benutzer / Clients

Authentifizierung von Personen und Geräten

Mischung

Unendlich viele Kombinationen der zuvor genannten Arten

Aufbau Zertifikatsdatei [1/2]

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            08:01:4d:70:3a:93:64:0d:87:6b:d9:a6:7a:69:2e:0f
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com, CN=Thawte TLS RSA CA G1
        Validity
            Not Before: Mar 18 00:00:00 2024 GMT
            Not After : Apr 10 23:59:59 2025 GMT
        Subject: CN=*.heinlein-support.de
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                […]
```

Aufbau Zertifikatsdatei [2/2]

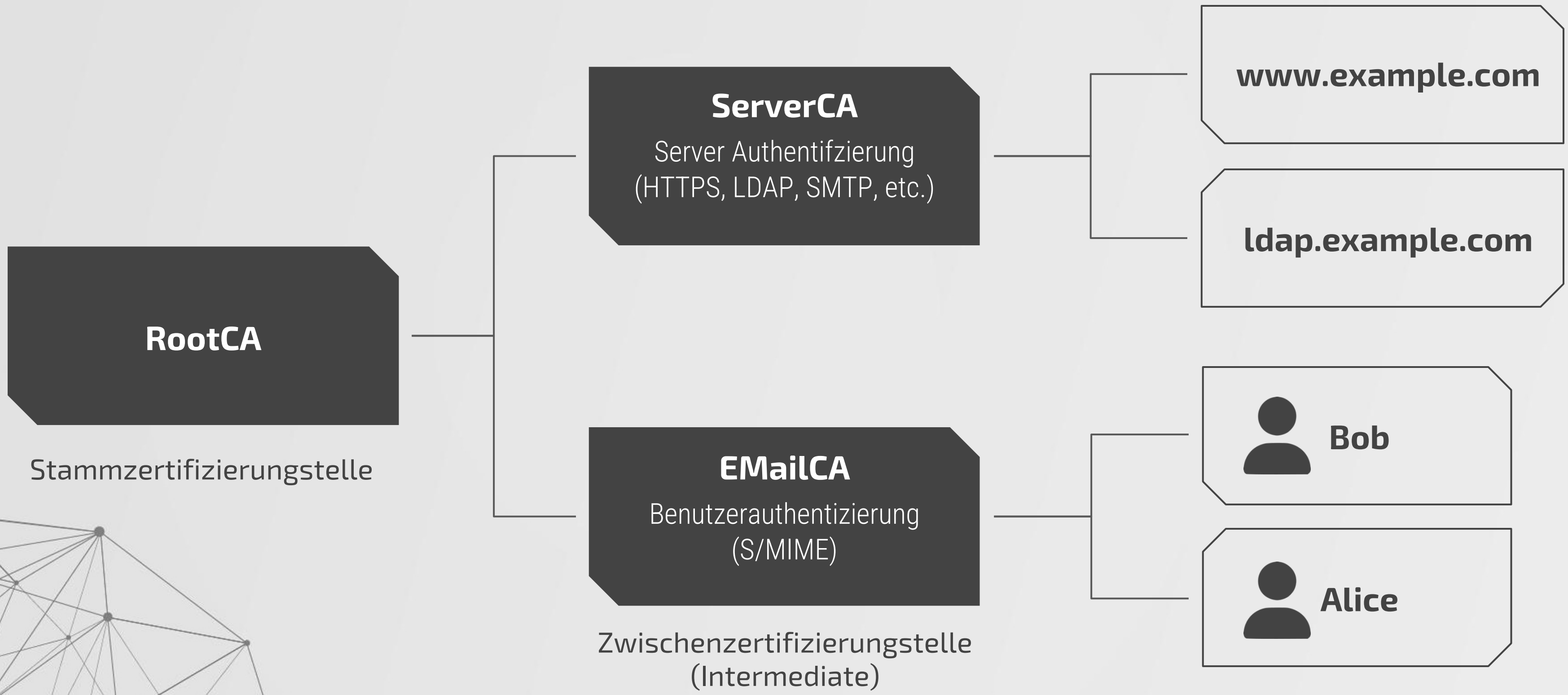
```
X509v3 extensions:
    X509v3 Authority Key Identifier:
        keyid:A5:8C:FE:32:CC:EB:0F:2C:D4:19:C6:08:B8:00:24:88:5D:C3:C5:B7

    X509v3 Subject Key Identifier:
        65:D1:A1:19:DC:34:89:2D:DF:68:12:DF:87:09:5C:84:DA:7A:10:A8
    X509v3 Subject Alternative Name:
        DNS:*.heinlein-support.de, DNS:heinlein-support.de
    X509v3 Certificate Policies:
        Policy: 2.23.140.1.2.1
          CPS: http://www.digicert.com/CPS

    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 CRL Distribution Points:
        […]
    X509v3 Basic Constraints: critical
        CA:FALSE
        […]
```

03

CERTIFICATE AUTHORITY

Mach’s am besten selbst!

Struktur


RootCA
Stammzertifizierungsstelle
ServerCA
Server Authentifzierung
(HTTPS, LDAP, SMTP, etc.)
EMailCA
Benutzerauthentizierung
(S/MIME)
Zwischenzertifizierungsstelle
(Intermediate)
www.example.com
ldap.example.com
Bob
Alice

openssl.cnf: RootCA

```
[ RootCA ]
dir             = /root/CA/RootCA          # Where everything is kept
certs           = $dir/certs               # Where the issued certs are kept
crl_dir         = $dir/crl                 # Where the issued crl are kept
database        = $dir/index.txt           # database index file.
new_certs_dir   = $dir/newcerts            # default place for new certs.
certificate     = $dir/RootCA.cacert.pem   # The CA certificate
serial          = $dir/serial              # The current serial number
crl             = $dir/crl.pem             # The current CRL
private_key     = $dir/private/RootCA.cakey.pem   # The private key
RANDFILE        = $dir/private/.rand       # private random number file
x509_extensions = ca_cert                  # The extentions to add to the cert
name_opt        = ca_default               # Subject Name options
cert_opt        = ca_default               # Certificate field options
crl_extensions  = crl_ext
default_days    = 365                      # how long to certify for
default_crl_days= 30                       # how long before next CRL
default_md      = sha256                   # which md to use.
preserve        = no                       # keep passed DN ordering
policy          = policy_match
```

openssl.cnf: Sektionen [1/2]

```
[ ca_cert ]
basicConstraints         = critical,CA:TRUE
nsComment                = "OpenSSL Generated CA Certificate"
subjectKeyIdentifier     = hash
authorityKeyIdentifier   = keyid:always,issuer:always
keyUsage                 = cRLSign,keyCertSig
```

openssl.cnf: Sektionen [2/2]

```
[ server_cert ]
basicConstraints        = CA:FALSE
nsComment               = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer
keyUsage                = digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth

[ email_cert ]
basicConstraints        = CA:FALSE
nsComment               = "OpenSSL Generated E-Mail Certificate"
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid,issuer
keyUsage                = digitalSignature,keyEncipherment
extendedKeyUsage        = emailProtection
subjectAltName          = email:copy
```

[Details: https://www.openssl.org/docs/manmaster/man5/x509v3_config.html]

Exkurs: keyUsage

```
Value                  Meaning
-----                  -------
digitalSignature       Signatur (Authentifizierung)
nonRepudiation         Signatur (Inhaltsbestätigung)
keyEncipherment        Schlüsselverschlüsselung
dataEncipherment       Datenverschlüsselung
keyAgreement           Schlüsselaustausch
keyCertSign            Signatur von öffentlichen Schlüsseln
cRLSign                Signieren von Sperrlisten
encipherOnly           Nur verschlüsseln (keyAgreement)
decipherOnly           Nur entschlüsseln (keyAgreement)
```

Exkurs: extendedKeyUsage

Value	Meaning according to RFC 5280 etc.
serverAuth	SSL/TLS WWW Server Authentication
clientAuth	SSL/TLS WWW Client Authentication
codeSigning	Code Signing
emailProtection	E-mail Protection (S/MIME)
timeStamping	Trusted Timestamping
OCSPSigning	OCSP Signing
ipsecIKE	ipsec Internet Key Exchange
msCodeInd	Microsoft Individual Code Signing (authenticode)
msCodeCom	Microsoft Commercial Code Signing (authenticode)
msCTLSign	Microsoft Trust List Signing
msEFS	Microsoft Encrypted File System

openssl.cnf: Policy

```
[ policy_match ]
countryName             = match
stateOrProvinceName     = match
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied
emailAddress            = optional
```

[Details: https://www.openssl.org/docs/manmaster/man1/openssl-ca.html#POLICY-FORMAT]

openssl.cnf: Standardwerte [1/2]

```
[ req ]
default_bits         = 2048
default_keyfile      = privkey.pem
distinguished_name   = req_distinguished_name
Attributes           = req_attributes
x509_extensions      = v3_ca   # The extensions to add to the self signed cert
string_mask          = utf8only
```

openssl.cnf: Standardwerte [2/2]

```
[ req_distinguished_name ]
countryName                    = Country Name (2 letter code)
countryName_default            = AU
countryName_min                = 2
countryName_max                = 2
stateOrProvinceName            = State or Province Name (full name)
stateOrProvinceName_default = Some-State
localityName                   = Locality Name (eg, city)

0.organizationName             = Organization Name (eg, company)
0.organizationName_default  = Internet Widgits Pty Ltd

organizationalUnitName         = Organizational Unit Name (eg, section)
commonName                     = Common Name (e.g. server FQDN or YOUR name)
commonName_max                 = 64
emailAddress                   = Email Address
emailAddress_max               = 64
```

[Details: https://www.openssl.org/docs/manmaster/man1/ca.html#POLICY-FORMAT]

Erstellung einer PKI mit openssl [1/3]

```
### Verzeichnisse und Dateien erstellen
$ mkdir -p CA/{RootCA,ServerCA,EMailCA}/{certs,crl,newcerts,private}

$ touch CA/{RootCA,ServerCA,EMailCA}/index.txt
$ echo "00" | tee CA/{RootCA,ServerCA,EMailCA}/serial

$ openssl rand -out CA/RootCA/private/.rand 1024
$ openssl rand -out CA/ServerCA/private/.rand 1024
$ openssl rand -out CA/EMailCA/private/.rand 1024


### RootCA: Schlüssel, Ausstellungsanfrage und Zertifikat erstellen
$ openssl genpkey -algorithm RSA -out CA/RootCA/private/RootCA.cakey.pem
$ openssl req -new -x509 -days 1820 -set_serial 00 \
  -key CA/RootCA/RootCA.cakey.pem -out CA/RootCA/RootCA.cacert.pem
  […]
```

Erstellung einer PKI mit openssl [2/3]

```
### Intermediate-CAs erstellen
$ openssl genpkey -algorithm RSA -out CA/RootCA/private/ServerCA.cakey.pem
$ openssl req -new -key CA/RootCA/private/ServerCA.cakey.pem \
  -out ServerCA.cacert.req.pem

$ openssl genpkey -algorithm RSA -out CA/RootCA/private/EMailCA.cakey.pem
$ openssl req -new -key CA/RootCA/private/EMailCA.cakey.pem \
  -out EMailCA.cacert.req.pem



### Intermediate-CAs Zertifikate erstelle
$ openssl ca -name RootCA -in ServerCA.cacert.req.pem \
  -out CA/RootCA/certs/ServerCA.cacert.pem

$ openssl ca -name RootCA -in EMailCA.cacert.req.pem \
  -out CA/RootCA/certs/EMailCA.cacert.pem
```

Erstellung einer PKI mit openssl [3/3]

```
### Verteilen & Aufräumen
$ cp CA/RootCA/private/ServerCA.cakey.pem CA/ServerCA/
$ cp CA/RootCA/certs/ServerCA.cacert.pem CA/ServerCA/

$ cp CA/RootCA/private/EMailCA.cakey.pem CA/EMailCA/
$ cp CA/RootCA/certs/EMailCA.cacert.pem CA/EMailCA/

$ rm *.req.pem
```

Arbeiten mit der PKI [1/2]

```
### Zertifikate ausstellen: ServerCA
$ cd CA/ServerCA/
$ penssl genpkey -algorithm RSA –out mail.example.com.key.pem
$ openssl req -new -key private/mail.example.com.key.pem \
  -out mail.example.com.req.pem

$ openssl ca -name ServerCA -in mail.example.com.req.pem \
  -out mail.example.com.cert.pem

$ mv newcerts/00.pem certs/00.pem
$ cd certs
$ ln -s 00.pem $(openssl x509 -hash -noout -in 00.pem).0
```

Arbeiten mit der PKI [2/2]

```
### Zertifikate widerrufen: ServerCA
$ openssl ca -name ServerCA -revoke CA/ServerCA/certs/01.pem

$ openssl ca -name ServerCA -gencrl -out crl/crl.pem

$ openssl crl -in crl/crl.pem -noout -text
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = DE, ST = NRW, O = ACME Ltd, OU = Dev, CN = ServerCA,
emailAddress = ca@example.com
        Last Update: May  2 11:28:47 2024 GMT
        Next Update: Jun  1 11:28:47 2024 GMT
        CRL extensions:
            X509v3 Authority Key Identifier:
                C3:55:4E:29:EB:FB:69:39:67:0E:60:12:4B:AD:35:6A:E9:46:4E:1B
Revoked Certificates:
    Serial Number: 01
        Revocation Date: May  2 11:28:11 2024 GMT
    [...]
```

MYTHEN

04

urban legends, epic fails and many more

Was? Was. Was !

- Highlander: Es kann (mehr als) nur einen geben!

  → index.txt.attr: `unique_subject = no`
- Shit in = shit out! Der CSR muss stimmen!

  → SAN
- Für Zertifikate mit SAN wird immer eine eigene openssl.cnf benötigt?

  → Falsch! (siehe *Tipps & Tricks 1*)
- Browser lehnen das Zertifikat ab?

  → Altes openssl? `default_md = md5` (nicht mehr zulässig!)

  → Laufzeit zu lang? Nur noch 1 Jahr!

  → Webserver liefert evtl. die Chain nicht mit aus (siehe *Tipps & Tricks 2*)
- Unmöglich zu prüfen (siehe *Tipps & Tricks 2*)
- “Ich brauche genau das Zertifikat nochmal”: `x509toreq` (siehe *Tipps & Tricks 1*)
- Eure CA wird nicht akzeptiert? → `critical` (see RFC 5280)
- Wie war nochmal der Befehl/Parameter/Option? → https://certificatetools.com/

Tipps & Tricks [1/2]

```
### Zertifikat anzeigen
openssl x509 -in server.crt -noout -text

### SAN-one-liner
openssl req -section ServerCA -nodes -x509 -sha256 -newkey rsa:4096 \
  -keyout example.org.key \
  -out example.org.crt \
  -days 356 \
  -subj "/C=DE/ST=NRW/L=Moers/O=ACME Inc/OU=Dev/CN=example.org"  \
  -addext "subjectAltName = DNS:example.org,DNS:www.example.org"

### Passen CRT & KEY zusammen? (RSA)
CRT="example.org.crt"
KEY="example.org.key"
CRT_MOD=$(openssl x509 -noout -modulus -in $CRT | openssl md5 | awk '{print $2}')
KEY_MOD=$(openssl rsa -noout -modulus -in $KEY | openssl md5 | awk '{print $2}')
[[ "$CRT_MOD" -eq "$KEY_MOD" ]] && echo "OK" || echo "MISSMATCH"

### CSR aus bestehendem CRT
$ openssl x509 -x509toreq -in server.crt -out server.csr -signkey server.key
```

Tipps & Tricks [2/2]

```
### Verify
# Einfache CA (CA > Server)
$ openssl verify -CAfile ca.crt test.crt

# Intermediate CA (CA > Interm. > Server)
$ openssl verify \
  -CAfile CA/RootCA/RootCA.cacert.pem \
  -untrusted CA/ServerCA/ServerCA.cacert.pem \
  CA/ServerCA/certs/mail.example.com.cert.pem



### Check Chain
openssl s_client -connect www.example.com -servername www.example.com
CONNECTED(00000003)
depth=2 C = DE, ST = NRW, L = Moers, O = ACME Inc, CN = RootCA
verify return:1
depth=1 C = DE, ST = NRW, L = Moers, O = ACME Inc, CN = ServerCA
verify return:1
depth=0 CN = www.example.com
verify return:1
[…]
```

05
GAME CHANGER

Ein Coyote räumt auf …

Let's Encrypt

11/2014	01/2015	12/2015
Ankündigung EFF, University of Michigan und Mozilla → *ISRG*	**ACME** Vorlage bei der IETF zur Standardisierung	**BETA** Beginn der offenen Betatest-Phase

03/2016	05/2016	03/2018	02/2020
1 Mio. Bereits nach 3 Monaten wurden 1 Millionen Zertifikate ausgestellt	**2 Mio.** 1,5 Monate später wurde bereits die 2 Millionen-Marke gerissen	**Wildcard** Ja LE unterstützt auch Wildcard-Zertifikate!	**1 Mrd.** Größenverhältnis in Sek.: 1 Mio. = 11 Tage 2 Mio. = 22 Tage 1 Mrd. = 31,7 Jahre

Challenge Typen

HTTP-01

Häufigste Methode

DNS-01

Gute Alternative für "interne Webserver"

TLS-ALPN-01

Beste Lösung für TLS-terminirende Reverse-Proxys (100% TLS)

75%*

20%

5%

**   Prozentangabe sind reine Willkür*

*** z.Z. DRAFT Token Authentifizierung: tkauth-01*

self-hosted ACME

06

Lokale Zertifizierungsstelle wie Let‘s Encrypt

acme2certifier

python

In python geschrieben
→ *läuft als CGI auf Webservern*

Unterstützt viele ACME-Clients: acme.sh, certbot, traefik, win-acme, dehydrated
→ *Mit Beispielen*

github

Frei verfügbar unter GPLv2:

https://github.com/grindsa/acme2certifier

ACME

Entwicklungsimplementierung als ACME Protokoll Proxy zur Anbindung von CAs die das Protokoll (noch) nicht unterstützen

Unterstützte CAs

- NetGuard Certificate Manager/Insta Certifier
- NetGuard Certificate Lifecycle Manager
- Insta ActiveCMS
- EJBCA
- OpenXPKI
- Microsoft Certificate Enrollment Web Services
- Microsoft Windows Client Certificate Enrollment Protocol (MS-WCCE) via RPC/DCOM
- Generic ACME protocol handler supporting Letsencrypt, BuyPass.com and ZeroSSL
- Generic EST protocol handler
- Generic CMPv2 protocol handler
- Openssl
- XCA
- acme2dfn

[Details: https://github.com/grindsa/acme2certifier/blob/master/docs/ca_handler.md]

Vorbereitungen

```
### CA Erstellen
# Dateien & Verzeichnisse
$ mkdir -p acmeCA/{certs,crl,newcerts,private}
$ touch acmeCA/index.txt
$ echo "00" > acmeCA/serial
$ openssl rand -out acmeCA/private/.rand 1024

# KEY & CSR
$ openssl genpkey -algorithm RSA -out CA/RootCA/private/acmeCA.cakey.pem
$ openssl req -new -key CA/RootCA/private/acmeCA.cakey.pem \
  -out acmeCA.cacert.req.pem

# Signieren
$ openssl ca -name RootCA -in acmeCA.cacert.req.pem \
  -out CA/RootCA/certs/acmeCA.cacert.pem


### Abhängigkeiten
$ apt install apache2 libapache2-mod-wsgi-py3 python3-pip apache2-data git
```

Installation & Konfiguration [1/3]

```
### Installation
$ cd /opt
$ git clone https://github.com/grindsa/acme2certifier
  […]

$ cd acme2certifier
$ pip3 install -r requirements.txt
  […]


### Konfiguration
$ cp examples/apache2/apache_wsgi.conf \
  /etc/apache2/sites-available/acme2certifier.conf
```

Installation & Konfiguration [2/3]

```
### Serverprogramm kopieren
$ mkdir /var/www/acme2certifier
$ cp examples/acme2certifier_wsgi.py /var/www/acme2certifier

### Beispiele kopieren
$ mkdir /var/www/acme2certifier/examples
$ cp -R examples/ca_handler/ /var/www/acme2certifier/examples/ca_handler
$ cp -R examples/db_handler/ /var/www/acme2certifier/examples/db_handler
$ cp -R examples/eab_handler/ /var/www/acme2certifier/examples/eab_handler
$ cp -R examples/hooks/ /var/www/acme2certifier/examples/hooks
$ cp -R examples/acme_srv.cfg /var/www/acme2certifier/examples/

### Hilfswerkzeuge kopieren
$ cp -R tools/ /var/www/acme2certifier/tools

### Funktionen kopieren
$ cp -R acme_srv/ /var/www/acme2certifier/acme_srv
```

Installation & Konfiguration [3/3]

```
### OpenSSL- und DB-Handler
$ cp examples/ca_handler/openssl_ca_handler.py acme_srv/ca_handler.py
$ cp examples/db_handler/wsgi_handler.py acme_srv/db_handler.py

### Konfigurationsdatei
$ cp examples/acme_srv.cfg acme_srv/

### Zertifikate
$ mkdir -p acme_srv/ca/certs
$ cp /root/CA/acmeCA/private/acmeCA.cakey.pem acme_srv/ca/
$ cp /root/CA/acmeCA/acmeCA.cacert.pem acme_srv/ca/
$ cp /root/CA/RootCA/RootCA.cacert.pem acme_srv/ca/
$ touch acme_srv/ca/crl.pem
```

Anpassungen »acme_srv.cfg«

```
[CAhandler]
issuing_ca_key: /var/www/acme2certifier/acme_srv/ca/acmeCA.cakey.pem
issuing_ca_key_passphrase: <PASSHRASE>
issuing_ca_cert: /var/www/acme2certifier/acme_srv/ca/acmeCA.cacert.pem
issuing_ca_crl: /var/www/acme2certifier/acme_srv/ca/crl.pem
cert_validity_days: 30
cert_save_path: /var/www/acme2certifier/acme_srv/ca/certs
ca_cert_chain_list: ["RootCA.cacert.pem"]
openssl_conf: /var/www/acme2certifier/acme_srv/ca/openssl.conf
allowed_domainlist: ["foo.bar\\$", "foo1.bar.local"]
blocked_domainlist: ["google.com.foo.bar\\$", "host.foo.bar$", "\\*.foo.bar"]
save_cert_as_hex: True
cn_enforce: True
```

Anpassungen »acme_srv/ca/openssl.conf«

```
[extensions]
subjectKeyIdentifier   = hash, issuer:always
keyUsage               = digitalSignature, keyEncipherment
basicConstraints       = critical, CA:FALSE
authorityKeyIdentifier = keyid:always, issuer:always
extendedKeyUsage       = critical, clientAuth, serverAuth
```

Finalisieren

```
### Rechte setzen und Apache-Module laden
$ chown -R www-data.www-data /var/www/acme2certifier/
$ a2enmod cgi rewrite
$ systemctl restart apache2.service

### Prüfen
$ curl -s http://127.0.0.1/directory | jq
  {
    "newAuthz": "http://127.0.0.1/acme/new-authz",
    "newNonce": "http://127.0.0.1/acme/newnonce",
    "newAccount": "http://127.0.0.1/acme/newaccount",
    "newOrder": "http://127.0.0.1/acme/neworders",
    "revokeCert": "http://127.0.0.1/acme/revokecert",
    "keyChange": "http://127.0.0.1/acme/key-change",
    "meta": {
        "home": "https://github.com/grindsa/acme2certifier",
        "author": "grindsa <grindelsack@gmail.com>",
        "name": "acme2certifier",
        "version": "0.34"
    },
    "c701[…]69f62": "https://community.letsencrypt.org/t/[…]/33417"
  }
```

War‘ das schon?

Bei weitem nicht…

VIELEN DANK

Irgendwelche Frage?

daniel@vansoest.org

https://www.rheinwerk-verlag.de/linux-server-das-umfassende-handbuch/