TLS für Mailserver richtig aufsetzen

Andreas Schulze

DATEV eG

6. Mailserverkonferenz

Berlin, 2014

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikaten
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

Browser vs. MTA

- erst Zertifikat prüfen, dann verschlüsseln
- Wenn die Zuordnung zum Namen nicht passt, ist alles böse
- HSTS / RFC 6797: verhindert Rückfall auf Klartext
- HTTPS Everywhere
- Mehr als 100 Root-Zertifikate
  - Juur-SK/AS Sertifitseerimiskeskus/EE
  - E-Tuğra EBG Bilişim Teknolojileri ve Hizmetleri A.Ş.
  - IZENPE S.A.
- Vertrauen?

Browser vs. **MTA**

- Hauptsache nicht Klartext
- Sicherheit vorm ***passiven*** Mitlesen durch externe Dritte
  - „Transport Layer Security“
- Optional: Validierung von X509 Zertifikaten
- → **aktive** Sicherheit

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikaten
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

Krypto

- Austauschen von geheimen Informationen in der Öffentlichkeit
- Üblich: asymmetrisch zum Tausch eines temporären Schlüssels
- Danach symmetrisch

- Schlüsseltausch
- Hash
- Symmetrische Verschlüsselung der Nutzdaten

Schlüsseltausch

- RSA
  - Übertragung einer Zufallszahl mittels asymmetrischer Verschlüsselung
  - Problem:
    geheimer Schlüssel muss geheim bleiben
- Diffie Hellmann
  - Zusenden von Nachrichten, aus denen sich ein gemeinsames Geheimnis ableiten lässt

Hash

- Belegen die Unverändertheit von Daten
- MD4
- MD5
- SHA1
- SHA2 / SHA3

Symmetrische Verschlüsselung

- Blockverschlüsselung
  - DES, 3DES, RC2, AES + AESGCM
  - BLOWFISH, IDEA
  - brauchen einen „Betriebsmodus“
    - ECB, CBC, CFB (ISO10116)
- Stromverschlüsselung
  - RC4

graue Theorie?

- openssl ciphers -v
- man ciphers
- gnutls-cli –list
- **http://en.wikipedia.org/wiki/Comparison_of_TLS_implementations**
- Cipher-Suiten+Namen von der IETF standardisiert (openssl nutzt eigene Namen)

- Einige Algorithmen möchte man heute nicht mehr verwenden
- Andere muss man leider noch anbieten :-/

RC4-MD5 ist keine Verschlüsselung

- RC4 wird vermutlich in Echtzeit dechiffriert
  http://www.theregister.co.uk/2013/09/06/nsa_cryptobreaking_bullrun_analysis/
  → Klartext
- MD5
  1996 … 2006 Uhhhh
  2012 MS-Zertifikat nachgemacht
- SHA1
  2005: Bruce Schneier: SHA-1 has been broken
- DSS
  Der Digital Signature Algorithm (DSA) ist ein Standard der US-Regierung für Digitale Signaturen
- Fefe: GCM ist nur in Hardware sauber implementierbar

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikate
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

Organisation von Zertifikaten

- Wichtige Parameter eines Zertifikates:
  - Für welchen Namen?
  - Wie lange gültig ?
  - Welche Root CA ?
  - Zwischen-CAs ?
  - Welches Dateiformat ?
    - PEM
- Weniger wichtig: die benutze Anwendung

Root CA

- X509 Subject — Worum geht’s ?
- X509 Issuer — Wer bestätigt das ?
- Subject == Issuer
  - Root Zertifikat / Zertifizierungsinstanz / Root CA
  - Ich bestätige hiermit, dass ich wirklich ich bin
- ~150 sind durchaus üblich

Zwischen CA

- Subject != Issuer
  - Zwischenzertifikat / Sub CA / Intermediate
- ist nicht zwingend auf jedem System vorhanden
- muss also dem Gegenüber bekannt gemacht werden, **wenn** dieser die Daten prüfen soll.

Organisation von Zertifikaten

- Dateiablage laut Dokumentation, HowTos und diversen Mailinglisten:
  - `/etc/postfix/server.crt`
  - `/etc/postfix/neu2.cert`
  - `/etc/ssl/dovecot.pem`
  - `/etc/apache/conf/ssl.crl/ca-bundle-client`

Organisation von Zertifikaten

- eigenes Schema zur Dateiablage
- Unterstützung durch Makefiles für Standardaktionen

  - /etc/ssl/
  - Zertifikatsgegenstand → /etc/ssl/${FQDN}/

Organisation von Zertifikaten

- Typische Dateinamen
  - cert.pem
  - root.pem
  - key.pem
  - intermediate.pem
  - ...
- sind SYMLINKS auf die aktuellen Dateien nach folgendem Schema:

/etc/ssl/${FQDN}/${FQDN}-${TYP}-${DATE}.pem

Organisation von Zertifikaten

- cert.pem → mx.example.org-cert-20140909.pem
- key.pem → ../private/mx.example.org-key-20100815.pem
- root.pem → ../certs/cacert.org-root.pem
- intermediate.pem → ../certs/cacert.org-class3.pem
- Makefile → ../path/to/Makefile_etc-ssl-fqdn

  (Quelle: postmaster.datev.de/mk6)

/etc/ssl/${FQDN}/

- vertrauenswürdige CAs als einzelne Dateien im Verzeichnis /etc/ssl/${FQDN}/trusted_cas/
- Makefile erzeugt daraus
  /etc/ssl/${FQDN}/trusted_cas.pem
- und bei Bedarf für den SMTP-Server
  die Hash-Links in der chroot

```
echo /var/spool/postfix/etc/ssl/${FQDN}/trusted_cas/ \
 > /etc/ssl/${FQDN}/trusted_cas/CHROOT
```

- **smtp_tls_cafile = /etc/ssl/${myhostname}/trusted_cas.pem**
- **smtpd_tls_capath = /etc/ssl/${myhostname}/trusted_cas/**

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikaten
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

Demo: Zertifikat erstellen

- `export FQDN=alice.example.org`
- `export STARTDATE=20140512`
- `install -d /etc/ssl/${FQDN}/ && cd /etc/ssl/${FQDN}/`
-
```
cat <<EOF > ${FQDN}-openssl.cnf
[ req ]
distinguished_name  = req_distinguished_name
[ req_distinguished_name ]
commonName          = CommonName
commonName_default = ${FQDN}
EOF
```
-
```
openssl req -nodes -new -sha256 -newkey rsa:4096 \
 -keyout ../private/${FQDN}-key-${STARTDATE}.pem \
 -config ${FQDN}-openssl.cnf \
 -out ${FQDN}-request-${STARTDATE}.pem
```
- `chmod 400 ../private/${FQDN}-key-${STARTDATE}.pem`

Demo: Zertifikat erstellen

- **ln -s ../private/${FQDN}-key-${STARTDATE}.pem key.pem**
- **cat ${FQDN}-request-${STARTDATE}.pem**
  -----BEGIN CERTIFICATE REQUEST-----
  MIIEYTCCAkkCAQAwHDEaMBgGA1UEAxMRYWxpY2UuZXhhbXBsZ
  6xXfHEB8lw+acV53pAQp33p2CAME
  -----END CERTIFICATE REQUEST-----
- **damit zur CA …**
- **cat $was_die_ca_liefert.pem | openssl x509 -noout -enddate**
  notAfter=May 12 04:44:48 2016 GMT
- **export ENDDATE=20160512**
- **mv $was_die_ca_liefert.pem ${FQDN}-cert-${ENDDATE}.pem**
- **ln -s ${FQDN}-cert-${ENDDATE}.pem cert.pem**
- **ln /path/to/ca-root root.pem**
- **ln /path/to/intermediate.pem intermediate.pem**

Demo: Zertifikat erstellen

- **ln -s /path/to/Makefile[1] Makefile**
- **make**
- **echo 443 > DANE; # echo 25 >> DANE; # echo 636 > DANE**
- **make dane**

- **ln -s /path/to/client_cacert.pem trusted_cas/**
- **make → trusted_cas.pem**
- **echo /var/spool/postfix/etc/ssl/${FQDN}/trusted_cas/ \ > trusted_cas/CHROOT**
- **make → aktualisiert Links in der postfix-chroot-umgebung**

[1]) Quelle: postmaster.datev.de/mk6

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikaten
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

Postfix als SMTP-Client

- Quasi „Browser-Mode“
- Zertifikat nicht zwingend nötig
- Verschlüsselung einfach einschalten:
  - **`smtp_tls_security_level = may`**
- Postfix tritt gegenüber dem Server „anonym“ auf

Postfix als SMTP-Client

- oder mit Clientzertifikat …

```
smtp_tls_cert_file = /etc/ssl/${myhostname}/cert+intermediate.pem
smtp_tls_key_file  = /etc/ssl/${myhostname}/key.pem
```

- Dann weiß der Server, wer der Client ist
- Ob man dann 'andere Berechtigungen' hat,
  ist Sache des Servers

Postfix als SMTP-Server

- Zertifikat zwingend[1] nötig

```
smtpd_tls_cert_file = /etc/ssl/${myhostname}/cert+intermediate.pem
smtpd_tls_key_file  = /etc/ssl/${myhostname}/key.pem

smtpd_tls_security_level = may
smtpd_tls_loglevel = 1
```

- eingehende TLS-Verbindungen sind entweder „anonym“ oder „untrusted“ jenachdem ob
  - der Server überhaupt nach einem Clientzertifikat fragt (`smtpd_tls_ask_ccert`)
  - ein Clientzertifikat liefert

[1]) postfix Server ohne Zertifikat ist wohl auch möglich ...

TLS bei SMTP != HTTPS

- Postfix unterscheidet optionale und verpflichtende Verschlüsselung
- Dementsprechend unterschiedliche starke Kryptographie

- smtp(d)_tls_ciphers = export
- smtp(d)_tls_mandantory_ciphers = medium
- smtp(d)_tls_protocols = !SSLv2
- smtp(d)_tls_mandantody_protocols = TLSv1.2

- smtpd_tls_security_level / smtp(d)_policy_maps

TLS bei SMTP != HTTPS

- smtp(d)_tls_exclude_ciphers
  = aNULL, RC4, MD5
- smtp(d)_tls_exclude_mandantory_ciphers
  = 3DES, DSS, (AESGSM)
- smtpd_tls_dh1024_param = ${data_directory}/4096.pem
  (nice -n 19) openssl gendh -out … -2 4096
  - Submission Clients: ev. Default

Trust ?

- X509 Zertifikat binden einen öffentlichen Schlüssel an eine Identität
- Bestätigt durch eine externe Instanz
- → Identität wird überprüfbar
  - smtp_tls_CAfile / smtp_tls_CApath
  - smtpd_tls_CAfile / smtpd_tls_CAPath
- Und dann ?
  - Trusted TLS connection im LOG
  - (verified OK) im Received Header

Trust ?

- smtp: wer ist der Server an den ich sende ?
- smtpd: wer liefert gerade ein ?

- smtp: bekommt Zertifikat „frei Haus“
  - kann quasi immer prüfen
- smtpd: muss danach fragen (**`smtpd_tls_ask_ccert`**)
  - kann also auch mal leer ausgehen
    ->anonymer Client
- In beiden Fällen: Liste von vertrauenswürdigen CAs

Vertrauenswürdige CAs

- OpenSSL kennt verschiedene 2 Arten
  - alle CA-Dateien in einer großen Datei
    - `$ cat /path/to/trusted_cas/* > /path/to/trusted_cas.pem`
  - alle Dateien separat in einem Verzeichnis mit zusätzlichen Symbolischen Links
    - `$ ls /path/to/trusted_cas/`
      `datev-esecure9.pem`
    - `$ c_rehash /path/to/trusted_cas/`
      `Doing /path/to/trusted_cas/`
      `datev-esecure9.pem => d5a0e395.0`
    - `$ ls -l /path/to/trusted_cas/`
      `d5a0e395.0 → datev-esecure9.pem`
      `datev-esecure9.pem`

CAfile oder CApath ?

- CAfile wird geladen, bevor smtp / smtpd chroot aufruft
- CApath muss in der chroot liegen
- **http://www.postfix.org/TLS_README.html#server_cert_key**

  When you configure the Postfix SMTP server to request client certificates, the DNs of certificate authorities in $smtpd_tls_CAfile are sent to the client, in order to allow it to choose an identity signed by a CA you trust.
- CAfile macht also den TLS-Handshake groß und geschwätzig
- somit: **smtp_tls_CAfile** und **smtpd_tls_CApath**

Weiter TLS-Einstellungen

- tls_ssl_options = no_compression
- tls_preempt_cipherlist = yes
- smtp(d)_tls_fingerprint_digest = sha1
- smtp_tls_note_starttls_offer = yes

Agenda

- Browser vs. MTA
- Krypto
- Organisation von Zertifikaten
- Demo: Zertifikat erstellen
- TLS mit postfix
- Blick über'n Tellerrand

nicht nur für postfix

- Dovecot
  - ```
    ssl = yes
    ssl_cert = </etc/ssl/imap.example.org/cert-intermediate.pem
    ssl_key  = </etc/ssl/imap.example.org/key.pem
    ```
  - ```
    protocol pop3 {
     ssl_cert = </etc/ssl/pop3.example.org/cert+intermediate.pem
     ssl_key  = </etc/ssl/pop3.example.org/key.pem
    }
    ```

nicht nur für postfix

- Nginx

```
server {
 listen              443 ssl spdy;
 server_name         www.example.org;
 ssl_certificate     /etc/ssl/www.example.org/cert+intermediate.pem;
 ssl_certificate_key /etc/ssl/www.example.org/key.pem;
}
```

- Apache (weicht vom Schema ab)

```
SSLCertificateChainFile /etc/ssl/www.example.org/intermediate.pem
SSLCertificateFile      /etc/ssl/www.example.org/cert.pem
SSLCertificateKeyFile   /etc/ssl/www.example.org/key.pem
```

- Lighttpd (ebenfalls anders)

```
ssl.pemfile = „/etc/ssl/www.example.org/cert+key.pem“
ssl.cafile  = „/etc/ssl/www.example.org/intermediate.pem“
```

nicht nur für postfix

- OpenLDAP

```
TLSCertificateFile    =
     /etc/ssl/ldap.example.org/cert-intermediate.pem

TLSCertificateKeyFile =
     /etc/ssl/ldap.example.org/key.pem

TLSCertificateFile    =
     /etc/ssl/ldap.example.org/trusted_cas/
```

nicht nur für postfix

- Systematische Dateinamen für Zertifikatsdateien
  - machen TLS Setup transparenter
  - scriptgesteuerte Überwachung der Ablaufzeiten: check-sslcert-expiredate [1]

[1]) Quelle: postmaster.datev.de/mk6

Zum Abschluß

F R A G E N ?

A N M E R K U N G E N ?

Danke

- Andreas Schulze
- DATEV eG
- andreas . schulze @ datev . de

Zukunft gestalten. Gemeinsam.