DICOM TLS Encryption

Ticket Beschreibung

- TLS-Verschlüsselung in DICOM verfügbar machen

Begriffe

- Keystore: Schlüsseln und Zertifikaten enthalten sind
- Truststore: nur Zertifikate enthalten sind
- Zertifikate:
- Schlüsseln:
- Keytool: erstellt Keystores mit Java und manipuliert darin Schlüssel und Zertifikate. Es kann auch Schlüssel erstellen und Zertifikate signieren.
- Ein Weg Authentifizierung: der Client bestätigt die Identität des Servers, während die Identität des Clients anonym bleibt.
- Zwei Wege Authentifizierung: der Client bestätigt die Identität des Servers und der Server bestätigt die Identität des Clients. Grundsätzliche TLS Händedruck

```
|SSL Client      -----        SSL Server|
|-------------Client Hallo------------->|
|<------------Server Hallo-------------->|
|<------Zertifikate und Public Key------|
|<--------Zertifikate Anfrage-----------|
|<---------Server Hallo fertig----------|
|-------Zertifikate und Public Key----->|
|--------- Client Key Austausch-------->|
|-----------Zertifikate Verify--------->|
|-----------tausch Cipher Spec--------->|
|--------------Client fertig----------->|
|<----------tausch Cipher Spec----------|
|<-------------Server fertig------------|
```

- Verschlüsselung Algorithmus:
  - ECC
  - RSA
    - PKCS(Public Key Cryptography Standard): PKCS #12(Personal Information Exchange Syntax Standard)
  - DSA
- Zertifikate Format = X.509 Standard, was das ASN.1 (Abstract Syntax Notation One) Sprach bunutzt um Datensturctur der Zertifikate zu definieren

```
  X.509
    |
    Definieren
    |
    |<--ASN.1
    |
Datenstruktur
    |                 |---PEM---  .pem .crt .cer .key
    |                 |
    |--Base64 ASCII---|---PKCS#7--- .p7b .p7c
    |
    |                 |---DER--- .der .cer
    |                 |
    |--Binary---------|---PKCS#12--- .pfx .p12
```

- File Format:
    - PEM (Privacy-Enhanced Mail): Ein File Format für Schlüsseln, Zertifikate. Die Server Zertifikate und Intermediate Zertifikate und Private Key können in demselbe .pem File gelegen werden. Oder Server Zertifikate (nach der CA Verifikation) in .crt(.csr: CSR: Certificate Signing Request, meistens auf Windows Platform funktioniert wie .crt), Intermediate Zertifikate in .cer und Private Key in .key
        - Syntax Bsp: Zertifikat:

```
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

        - Syntax Bsp: Zertifikat Anfrage:

```
-----BEGIN CERTIFICATE REQUEST-----
-----END CERTIFICATE REQUEST-----
```

    - Zertifikatsanforderung in .csr, was mit Private Key beantragt
    - PKCS#12: Ein Binary File. Server Zertifikate in .p12. Intermediate Zertifikate und Private Key sind in .pfx und verschlüsselt. Diese Format ist hauptsächlicht für Windows Platform benutzt.

- TLS:
    - Komponente
        - aes-128-gcm
        - rsa-ecdhe
        - TLS-PRF-sha256
    - CipherSuite: TLS CipherSuites können in iana registiert werden.
        - authentication Algorithmus
        - encryption Algorithmus
        - message authentication code Algorithmus
        - key exchange Algorithmus
        - key derivation function Algorithmus

Zustand des Ticketsprozesses

- Java und OpenSSL sind zwei Tools zur Generierung von Kryptoschlüsseln.

Vergleich Java und OpenSSL

- Java arbeitet mit Schlüsseln und Zertifikaten, die in einem KeyStore gespeichert sind.
- OpenSSL funktioniert mit Standardformaten (PEM/CER/CRT/PKCS/etc), manipuliert jedoch keine KeyStore-Dateien.

Es ist möglich, einen Schlüssel und/oder ein Zertifikat mit OpenSSL zu generieren und diesen Schlüssel/dieses Zertifikat dann mit keytool in einen KeyStore zu importieren.

- Statt NIM3.0 wird NIMLU benutzt um Integrieren von TLS in DICOM zu testen, weil nim noch Bug im JAVA Loader hat. NIMLU hat völlstandig JDK mit.

Java

OpenJDK install

Anwendung von Keytool

- Erstellung Zertifikate

```
keytool -genkey -alias tomcat -keyalg RSA -keystore /etc/cas/keystore/tomcat.keystore
```

- Export Zertifikate zu /etc/cas/keystore/tomcat.cer

  ```
  keytool -export -alias tomcat -keystore /etc/cas/keystore/tomcat.keystore -storepass tomcat -rfc -file /etc/cas/keystore/tomcat.cer
  ```

- Export Zertifikate zu Java cacert lib

  ```
  keytool -import -alias tomcat -keystore cacerts -file /etc/cas/keystore/tomcat.cer -trustcacerts
  ```

- Delete Zertifikate

  ```
  keytool -delete -alias tomcat -keystore /etc/cas/keystore/tomcat.keystore -storepass tomcat
  ```

- Erstellung Keystore `keytool -genkeypair -alias certificatekey -keyalg RSA -validity 365 -keystore tomcat.keystore`
- Export Zertifikate Key `keytool -export -alias certificatekey -keystore tomcat.keystore -rfc -file tomcat.cer`
- Erstellung Truststore `keytool -import -alias certificatekey -file tomcat.cer -keystore tomcat.truststore`
- Umwandlung Keystore ins P12 `keytool -importkeystore -srckeystore tomcat.keystore -destkeystore tomcat.p12 -deststoretype PKCS12`

OpenSSL

Install

- Herunterladen: OpenSSL
- Unter Linux:

  ```
  tar -xzf openssl-1.1.1m.tar.gz

  cd openssl-1.1.1m

  mkdir /usr/local/openssl

  ./config --prefix=/usr/local/openssl

  make

  make install
  ```

  oder `yum install openssl`

- Unter Unbuntu:

  ```
  sudo apt-get install openssl
  ```

- Umgebung Variable in Linux hinzufügen: `./config -t`

  `make depend` gehen ins /usr/local `ln -s openssl ssl` oder `ln -s /usr/local/openssl/bin/openssl /usr/bin/openssl`

  In /etc/ld.so.conf fügen /usr/local/openssl/lib hinzu `ldconfig`

  In etc/profile fügen `export OPENSSL=/usr/local/openssl/bin` und `export PATH=$OPENSSL:$PATH:$HOME/bin` hinzu

Anwendungen von OpenSSL

Erzeugen eines selbst signiertem Zertifikat

- Erstellung des privaten Schlüssels im PEM-Format

```
openssl genrsa -out cakey.pem 2048
```

- Erstellung des privaten Schlüssels und einer Zertifikatsanforderung

```
openssl req -newkey rsa:4096 -keyout hostname_key.pem -out hostname_request.pem -nodes
```

- Erstellung eines selbst signiertem CER Zertifikat

```
openssl x509 -req -days 365 -sha1 -extensions v3_ca -signkey cakey.pem -in ca.csr -out
cacert.pem
```

- Erstellung CRT

```
openssl pkcs12 -in tomcat.p12 -nokeys -out tomcat.crt
```

- Erstellung Key

```
openssl pkcs12 -in tomcat.p12 -nocerts -nodes -out tomcat.key
```

Server Private Schlüsseln und Zertifikate

- Erstellung Server Private Schlüsseln (CA Private Key)

```
openssl genrsa -out key.pem 2048
```

- Erstellung Server Zertifikatsanforderung (CSR)

```
openssl req -new -key key.pem -out server.csr
```

- Mit root(selbst signierte) Server Zertifikate signierte Server Zertifikate (CA Root Zertifikate)

```
openssl x509 -req -days 365 -sha1 -extensions v3_req -CA ../CA/cacert.pem -
CAkey ../CA/cakey.pem -CAserial ca.srl -CAcreateserial -in server.csr -out cert.pem
```

- Mit CA Zertifikat verfizierte Server Zertifikate

```
openssl verify -CAfile ../CA/cacert.pem cert.pem
```

Client Private Schlüsseln und Zertifikate

- Erstellung Client Private Schlüsseln

```
openssl genrsa -out key.pem 2048
```

- Verschlüsseln des Private Schlüsselns

```
openssl rsa -in ssl.key -des3 -out encrypted.key
```

- Erstellung Client Zertifikatsanforderung

```
openssl req -new -key key.pem -out client.csr
```

- Mit selbst signierte Client Zertifikate signierte Client Zertifikate

```
openssl x509 -req -days 365 -sha1 -extensions v3_req -CA ../CA/cacert.pem -
CAkey ../CA/cakey.pem -CAserial ../server-cert/ca.srl -in client.csr -out cert.pem
```

- Mit CA Zertifikat verfizierte Client Zertifikate

```
openssl verify -CAfile ../CA/cacert.pem cert.pem
```

Umwandlung

- Umwandlung mittels Kommandozeilen Aufruf

```
sudo openssl genrsa -out private/ca.key
```

Server:

```
sudo openssl req -new -key private/ca.key -out private/ca.csr

sudo openssl x509 -req -days 365 -in private/ca.csr -signkey private/ca.key -out
private/ca.crt

sudo echo FACE > serial

sudo touch index.txt

sudo openssl ca -gencrl -out /root/ca/private/ca.crl -crldays 7 -config
"/root/ca/conf/openssl.conf"
```

Client:

```
sudo openssl genrsa -des3 -out /root/ca/users/client.key 1024

sudo openssl req -new -key /root/ca/users/client.key -out /root/ca/users/client.csr

sudo openssl ca -in /root/ca/users/client.csr -cert /root/ca/private/ca.crt -keyfile
/root/ca/private/ca.key -out /root/ca/users/client.crt -config
"/root/ca/conf/openssl.conf"

sudo openssl pkcs12 -export -clcerts -in /root/ca/users/client.crt -inkey
/root/ca/users/client.key -out /root/ca/users/client.p12
```

- Umwandlung mittels OpenSSL c-API
- Common PEM Conversions
    - View contents of PEM certificate file

```
openssl x509 -in CERTIFICATE.pem -text -noout
```

    - Convert PEM certificate to DER

```
openssl x509 -outform der -in CERTIFICATE.pem -out CERTIFICATE.der
```

    - Convert PEM certificate with chain of trust to PKCS#7 openssl crl2pkcs7 -nocrl -certfile CERTIFICATE.pem -certfile MORE.pem -out CERTIFICATE.p7b

```
openssl crl2pkcs7 -nocrl -certfile CERTIFICATE.pem -certfile MORE.pem -out
CERTIFICATE.p7b
```

    - Convert PEM certificate with chain of trust to PKCS#7

```
openssl pkcs12 -export -out CERTIFICATE.pfx -inkey PRIVATEKEY.key -in CERTIFICATE.crt -
certfile MORE.crt
```

- Common DER Conversions
    - View contents of DER-encoded certificate file

```
openssl x509 -inform der -in CERTIFICATE.der -text -noout
```

- Convert DER-encoded certificate to PEM

```
openssl x509 -inform der -in CERTIFICATE.der -out CERTIFICATE.pem
```