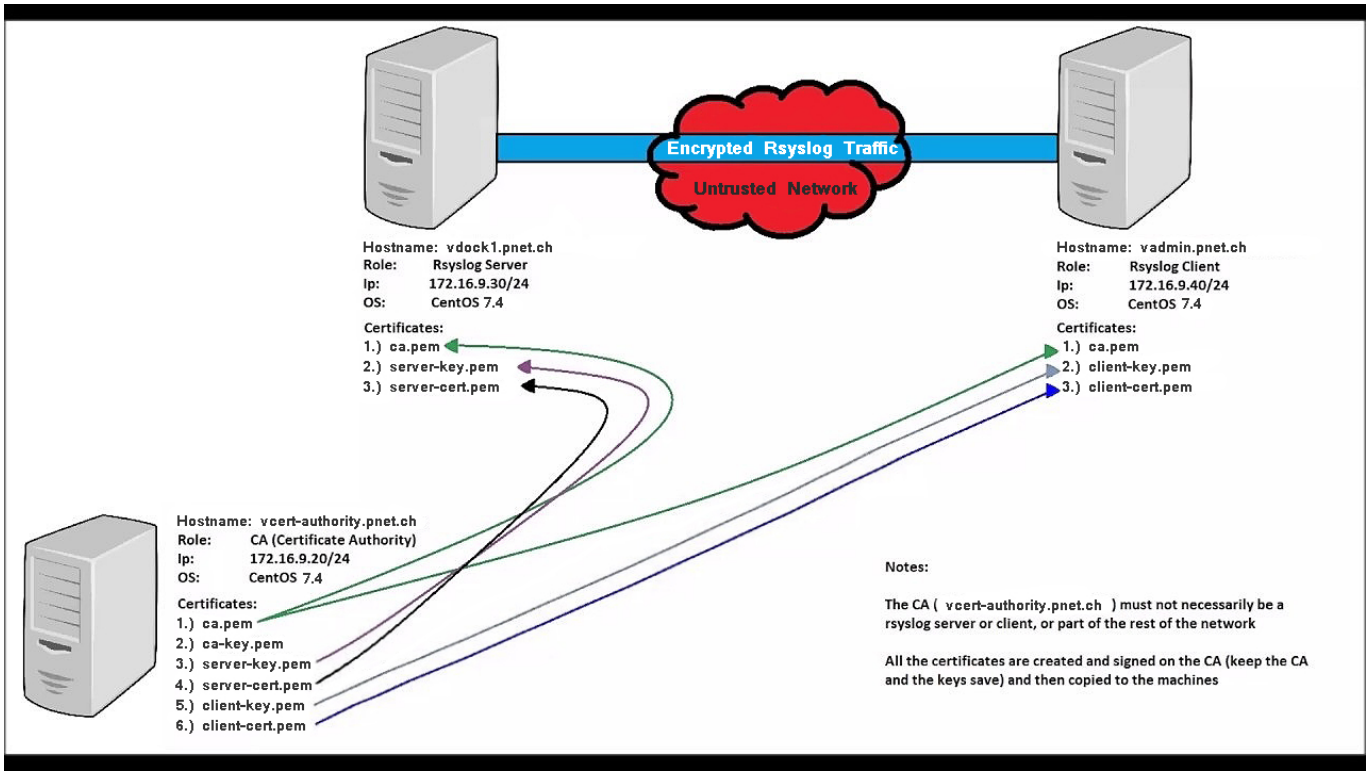
Create Client/Server TLS Certificate

Funktionsprinzip einer TLS Verschlüsselung

TLS (`Transaction Layer Security`) ist die standardisierten Weiterentwicklungen von SSL (TLS 1.0 = SSL 3.1). SSL wird also nun unter dem Namen TLS weiterentwickelt. TLS und SSL werden daher oft auch synonym verwendet. `TLS ist laut RFC 2246 eine Technologie zur Sicherung des Netzwerk-Datenaustausches zwischen Anwendungen, das Datenschutz und Datenintegrität (“privacy and integrity”) gewährleistet (bzw. gewährleisten soll).` Der generelle Ablauf bei TLS beginnt mit dem Aufbau einer Verbindung vom Client zum Server. Dabei schickt er auch gleich eine Liste an unterstützten Cipher Suites mit. Anschliessend authentisiert sich der Server gegenüber dem Client mit seinem Zertifikat und der ausgewählten Cipher Suite. Der Client überprüft das Zertifikat und authentisiert sich ggf. selbst auch noch gegenüber dem Server mit einem eigenen Zertifikat. Nun schickt entweder der Client dem Server eine verschlüsselte Zufallszahl, die mit dem öffentlichen Schlüssel des Servers verschlüsselt ist, oder beide Parteien berechnen ein gemeinsames Geheimnis mit dem Diffie-Hellman-Schlüsselaustauschverfahren. Mit dem daraus abgeleiteten kryptographischen Schlüssel, werden nun alle Nachrichten der Verbindung mit einem ausgewählten symmetrischen Verschlüsselungsverfahren verschlüsselt.

Beispiel Umgebung:



Erstellungsvorgang von unabhängigen Client / Server Zertifikaten

Im folgenden Beispiel, erstelle ich **ein neues Server Zertifikat** als sowohl **ein neues Client Zertifikat**, welche **beide mit einer lokal erstellten CA signiert** werden. Der Aufbau entspricht der obigen Übersicht.

Vorbereiten der Linux Umgebung

Um überhaupt anschliessend eine eigene CA zu erstellen, benötige ich zuerst das passende OpenSSL Package für mein System sowie eine passende Ordnungsstruktur. Ich gehe daher wie folgt vor:

```
# yum install openssl.x86_64
# mkdir -p /opt/TLS_Certs/{Server_Cert-Key,Client_Cert-Key}
```

Eigene CA einrichten

Um unsere späteren Client/Server Zertifikate zu erstellen, wird zuerst die neue CA erstellt. Dazu wird fogendermassen vorgegangen:

1. Wechseln in das CA Hauptverzeichnis:

```
# cd /opt/TLS_Certs/
```

2. Erstellen des Initialen CA-Private-Keys **(Wird mit starkem Passwort geschützt!)**:

```
# openssl genrsa -aes256 -out ca-key.pem 4096
```

3. Generieren des CA Zertifikats **(mit 10 jähriger Gültigkeit)**:

```
# openssl req -new -x509 -days 3650 -key ca-key.pem -sha256 -out ca.pem
```

4. Zugriffe auf CA Key/Cert absichern:

```
# chmod 0400 /opt/TLS_Certs/*.pem
```

Server Zertifikat erstellen

Nun werden im ersten Teil einmal einen neuen Server Key, sowie ein neues Server Cert für den Server “vdock1” erstellt. Wir gehen wie folgt vor:

1. Wechseln in das Server_Cert-Key Hauptverzeichnis:

```
# cd /opt/TLS_Certs/Server_Cert-Key/
```

2. Erstellen der Zertifikat Konfigurationsdatei:

```
# vim /opt/TLS_Certs/Server_Cert-Key/extfile.cnf
```

```
subjectAltName = DNS:vdock1.pnet.ch
extendedKeyUsage = serverAuth
```

3. Erstellen des Server-Private-Keys:

```
# openssl genrsa -out server-key.pem 4096
```

4. Erstellen des CSR (Signing Request) für den Server:

```
# openssl req -subj "/CN=vdock1.pnet.ch" -sha256 -new -key server-
key.pem -out server.csr
```

5. Generieren des Server Zertifikats **(mit 5 jähriger Gültigkeit)**:

```
# openssl x509 -req -days 1825 -sha256 -in server.csr -CA ../ca.pem -
CAkey ../ca-key.pem -CAcreateserial -out server-cert.pem -extfile
extfile.cnf
```

Client Zertifikat erstellen

Jetzt müssen wir im zweiten Teil, noch ein passendes Client Zertifikat für unsere Clients erstellen. **(Dieses Zertifikat, kann anschliessend auf alle**

Clients verteilt werden!)

1. Wechseln in das Client_Cert-Key Hauptverzeichnis:

```
# cd /opt/TLS_Certs/Client_Cert-Key/
```

2. Erstellen der Zertifikat Konfigurationsdatei:

```
# vim /opt/TLS_Certs/Client_Cert-Key/extfile.cnf
```

```
extendedKeyUsage = clientAuth
```

3. Erstellen des neuen Client-Private-Keys:

```
# openssl genrsa -out client-key.pem 4096
```

4. Erstellen des CSR (Signing Request) für den Client:

```
# openssl req -subj '/CN=vadmin.pnet.ch' -new -key client-key.pem -out client.csr
```

5. Generieren des Client Zertifikats **(mit 5 jähriger Gültigkeit)**:

```
# openssl x509 -req -days 1825 -sha256 -in client.csr -CA ../ca.pem -CAkey ../ca-key.pem -CAcreateserial -out client-cert.pem -extfile extfile.cnf
```

Verteilen der neu erstellten Keys / Certs

- Der Inhalt von: **/opt/TLS_Certs/Server_Cert-Key/** wird in unserem Beispiel, **auf dem Applikationsserver "vdock1.pnet.ch" kopiert** und dort in der entsprechenden Applikation eingerichtet.
- Der Inhalt von: **/opt/TLS_Certs/Client_Cert-Key/** wird **auf unseren Admin-Client(Server) "vadmin.pnet.ch" kopiert** und dort eingerichtet, damit der Zugriff via TLS auf die Applikation möglich wird.

Weiteres

Ansible Skript zur vollautomatischen Erstellung und Verteilung der Key / Certs:

- Praktische Experimente mit OpenSSL, sowie Certificate-Validation und Analyse