Inhaltsverzeichnis

Linux Server gestützte Public-Key-infrastruktur (PKI) im Unternehmen mit Microsoft Active Directory Umgebung

weiter Links

Präsentation des Projektes Bearbeiten der Präsentation

Impress - Präsentation via Impress

Projektbeschreibung

Projektumfeld

Das Projekt wird in einer virtualisierte Umgebung mit einer DMZ[(gl:dmz)]-Netzwerk-Architektur der Firma <FIRMA1> (<FIRMA1> hier Dienstleister / Auftragnehmer) als Tochter Unternehmen der <FIRMA2> (Kunde / Auftraggeber) entwickelt, getestet und für die gedachte Umsetzung im Hause der <FIRMA2> vorbereitet.

Projektziel

Hauptziel

Es soll ein Lösungskonzept zum Aufbau einer einfachen PK-Infrastruktur im Unternehmensumfeld der <FIRMA2> erstellt werden. Bislang gibt es Intern keine geeignete Möglichkeit, welche den aufkommenden Daten-Verkehr zwischen den Clients den unterschiedlichen Angeboten an Diensten (u.a. über dei Protokolle HTTP[(gl:prot:http)] und FTP[(gl:prot:ftp)] an Dokumenten-Server) zu Verschlüsselung. Mittelfristig wird der Einsatz einer neuen Software vorbereitet, welche diese einfache PK-Infrastruktur[(gl:pki)] ebenso als Schnittstelle zur Datenübertragung beinhalten könnte.

- Herstellen einer Kosten neutralen und überschaubaren Public-Key-Infrastruktur, Certificate Authority-Root (CA[(gl:ca)]-Root) & Certificate Authority Intermediate (CA-Sign) für den Einsatz in einer Microsoft Active-Directory Umgebung.
- Ausstellen von Root-CA und Sign-CA Zertifikat für die Gesamtstruktur (test.pp)
- Verteilung der CA-Zertifikate via Active Directory-DC Gruppen-Richtlinie und/oder per Software.
- Ausstellen und Unterschreiben (Sign-CA) von Server-, Client- und Code-Signing-Zertifikate welche valide zur Root-CA und Sign-CA sind.
- Kontinuierliche Erstellung und Bereitstellung einer Sperrliste

Teilziele

- Erstellen / Abrufen von Client Zertifikaten via Web-Oberfläche ohne Administrationsrechte.
- Aufheben (Revoke) von Zertifikaten via Web-Oberfläche mit u. ohne Administrationsrechten.
- Auffrischen von CA-Zertifikate vor Ablauf der zeitlichen Gültigkeit
- Auffrischen von Zertifikaten kurz vor Ende der zeitlichen Gültigkeit
- Überprüfung
- Client Authentifizierung via Client Certificate am Web-Server

Aufgabenstellung (IST Analyse - SOLL-Konzept)

- IST Zustand:
  - In einer betrieblichen Umgebung (Kunden IT-Infrastruktur) mit einer Microsoft Domäne im AD-DC Verbund von über 500 Clients und mehreren Server-Systemen, sowohl Hardware wie auch in virtualisierten Umgebungen z.B. über Terminal Server, werden Datenschutz relevanten Informationen über das bestehende Netzwerk zum großen Teil unverschlüsselt ausgetauscht.
- Soll-Zustand:
  - Es soll sichergestellt werden das innerhalb der Domäne keine dieser Informationen weiterhin unverschlüsselt über das Netzwerk übermittelt werden und somit vor unberechtigtem Zugriff bzw. Abgriff hinreichen Geschütz sind.
  - Den Clients soll ausschließlich nach Einbindung in die Domäne und entsprechender Gruppen-Zuweisung, die nötigen CA-Zertifikate übermittelt und anschließend durch ein Eigens für diesen Client erstelltes Zertifikat zur Authentifizierung, Autorisierung und Verschlüsselung auf den spezifischen Datenübertragungswege gegenüber den vorhandenen Server Systemen ermöglichen.

Projektbegründung

Spätestens seid den sogenannten Snoden-Enthüllungen und Ransomware (Krypto-/Erpressungstrojaner) Angriffen, ist offensichtlich geworden das nicht nur gezielte Angriffe erfolgen müssen um Schäden in Millionen Höhe zu verursachen.

Mit dem Internet of Things (IoT) wächst die Gefahr durch fehlerhafte Massenware auch annähernd Zufällig sensible Daten zu verlieren oder preis zugeben. Gerade auch in Bezug auf Industriespionage wäre ein solchen zustande kommen für ein Unternehmen fatal. Eine fehlerhafte Schnittstelle in einem Router oder Switch könnte bereits ausreichen, wie z.B. offene Lücken im `telnet`-Dienst.
Sicherheitsupdate in Sicht: Gravierende Telnet-Lücke bedroht zahlreiche Cisco-Switches | 20.03.2017

PDF Artikel - Heise Security - Gravierende "telnet"-Lücke

Abgrenzungen

- der Aufbau einer vollständigen Domänen-Struktur mit Domäne, Tree oder Forest und/oder AD-DC ist nicht die Aufgabe
- LDAP und andere Autentifizierungsverfahren sind nicht Bestandteil der Aufgabe
- Funktionalität bei 2 / 3 Browsern sind hinreichend
- User- oder E-Mail Zertifkate sind nicht Bestandteil der Aufgabe
- Bewertung, Analyse oder Erstellung von Krypto-Verfahren sowie Encryption und/oder

Decryption übertragender Daten sind nicht Bestandteil der Aufgabe
- Aufbau einer vollständige Sicherheits-Architektur rund um Webserver und andere Dienste ist nicht die Aufgabe

Projektplanung

Projektphasen

Das Projekt wurde für 35 Stunden gem. der IHK Vorgaben geplant. Die Meilensteine dienen nicht nur der Orientierung und Soll/Ist Vergleichspunkte sondern vor allem wie im iterativen Wasserfallmodell (erweit. Wasserfallmodell) vorgesehen, als Kontrollrückkopplung um eventuellen Schwierigkeiten bei der Umsetzung des Projektes mit geeigneten Maßnahmen frühzeitig entgegen wirken zu können. Und zwar rechtzeitig, bevor ein Projekt mangels Qualitätssicherungsmaßnahmen ganz oder teilweise scheitert.

Abweichungen vom Projektantrag

Ursprünglich geplantes Power-Shell Script für Firefox

- das signieren und ausführen des PS-Script zur installation des Firefox Zertifikat-Store konnte nicht hinreichend getestet werden. Die Ausführung war wie geplant durch das Herstellen einer mind. nötigen Vertraulichkeit in der Domäne möglich.
- der korrekte Speicherbereich konnte nicht Zuverlässig ermittelt werden, in der kürze der Zeit.
- gemäß einer Ankündigungen von Mozilla solle diese Maßnahmen in naher Zukunft nicht mehr notwendig sein.
- das Auffrischen der CA-Zertifikate ist aus zeitlichen Gründen nicht umgesetzt worden.

Ressourcen-Planung

Die zur Umsetzung des Projektes genutzte Hard- und Software sowie die mind. Systemvoraussetzungen werden im Anhang vollständig Aufgeführt und Beschrieben.

Zeitplanung / Meilenstein (Projektmanagement)

Projektphasen mit Zeitplanung (in Stunden)

Nr	Phase	Kurz Beschr.	Zeit
1	Projektstart	Vorstellung des Projektes	1 h
2	Analyse & Definition	Soll-Ist Analyse	2 h
3	Entwurf	Ausarbeitung Scripte	5 h
4	Implementation	Aufsetzen Server / Client Instanzen	10 h
5	Test / Validierung	Testlauf Zertifikat Verteilung	8 h
6	Einsatz	Roll-Out	2 h

Nr	Phase	Kurz Beschr.	Zeit
7	Abschluss	Übergabe	7 h

Detaillierte Zeitplanung und Meilensteine (mit Soll-Ist Datum)

Projektphase	Gesamtstunden	Meilenstein	Zwischensumme	Teilsumme	Datum
1. Projektstart	1 h				
1.1 Projekt Vorstellung			45 min		21.12.16
1.2 Projekt Genehmigung			15 min		21.12.16
2. Analyse & Definition	2 h				
2.1 Durchführung IST-Analyse			30 min		27.03.17
2.2 Durchführung SOLL-Analyse			1,25 h		27.03.17
2.3 Formulierung Definition			15 min		27.03.17
		1. Meilenstein	Abschluss A & D mit erfg. Formulierung		
3. Entwurf	5 h				
3.1 openSSL			2,75 h		
3.1.1 Template ca.conf				1 h	27.03.17
3.1.2 BASH Script CA-Cert.sh				15 min	27.03.17
3.1.3 Template cic.certs.conf				1 h	27.03.17
3.1.4 BASH Script CIC-Cert.sh				30 min	27.03.17
		2.1 Meilenstein	BASH-Scripte Lösung Formuliert		
3.2 Microsoft X.509-Storage			1,25 h		
3.2.1 C# Program Cert-Installer.exe				1h	27.03.17
3.2.1 PS Script Firefox Cert-Install				15 min	27.03.17
		2.2 Meilenstein	X509-Stores Lösung Formuliert		
3.3 Web-Interface			1 h		
3.3.1 PHP Scripte				30 min	27.03.17
3.3.1 Bootstrap Webpage				30 min	27.03.17
		2.3 Meilenstein	Web-Interface Konzipiert		
4. Implementation	10 h				
4.1 Linux Server			3 h		
4.1.1 Update & Installation				1 h	28.03.17
4.1.2 BASH-Scripte Lösung integrieren				15 min	28.03.17
4.1.3 Webserver Konfiguration				1 h	28.03.17
4.1.4 CA Zertifikate erzeugen & einbinden				30 min	28.03.17
4.1.5 Über Web-Server Prüfen				30 min	28.03.17
4.1.6 Fehler Dokumentation				30 min	28.03.17
		3.1 Meilenstein	BASH-Scripte Lösung integriert		
4.2 Windows Server & Client			5 h		
4.2.1 Windows Server Updates				1 h	28.03.17
4.2.2 Domäne-Controller konfigurieren				2 h	28.03.17
4.2.3 Active Directory konfigurieren				1 h	28.03.17
4.2.4 Windows Client Updates				1 h	28.03.17
4.2.5 Windows Client in Domäne Aufnehmen				15 min	29.03.17
4.2.6 Konnektivität zwischen allen Systemen prüfen				15 min	29.03.17
4.2.7 Fehler Dokumentation				30 min	29.03.17
		3.2. Meilenstein	Windows Server AD-DC funktionsfähig		

Projektphase	Gesamtstunden	Meilenstein	Zwischensumme	Teilsumme	Datum
14.3 Web-Interface			1 h		
4.3.1 Web-Interface einpflegen				15 min	29.03.17
4.3.2 Web Ausgabe überprüfen				15 min	29.03.17
4.3.3 HTTPs Verbindung protokollieren				15 min	29.03.17
4.3.4 Fehler Dokumentation				15 min	29.03.17
3.3. Meilenstein		Webserver & Interface erreichbar			
5. Test / Validierung	8 h				
5.1 Server Test			3 h		
5.1.1 Funktionsprüfung Bereitstellung Zertifikate				30 min	29.03.17
5.1.2 Funktionsprüfung Verteilung CA - Zertifikate				30 min	29.03.17
5.1.3 Auswertung und Dokumentation Logs bzw. Events				1 h	29.03.17
5.1.4 Fehlerkorrekturen				1 h	29.03.17
4.1 Meilenstein		Bereitstellung und Verteilung erfolgt			
5.2 Client Test			3 h		31.03.17
5.2.1 Abruf und Installation der Client Zertifikate				30 min	29.03.17
5.2.2 Überprüfung Windows.Store und Browser Ausgabe				30 min	29.03.17
5.2.3 Auswertung und Dokumentation Logs bzw. Events				1 h	29.03.17
5.2.4 Fehlerkorrekturen				1 h	30.03.17
4.2 Meilenstein		Client Authentifizierung erfolgt			
5.3 Web-Interface Test			2 h		
5.3.1 Web-Interface Funktionsprüfung				30 min	30.03.17
5.3.2 Automatischer Abruf Client Zertifikate prüfen				30 min	30.03.17
5.3.3 Fehlerkorrekturen				1 h	30.03.17
4.3. Meilenstein		Web Verwaltung und automatischer Abruf erfolgt			
6. Einsatz	2 h				
6.1 Zurücksetzen und 2. Durchlauf starten			1 h		30.03.17
5. Meilenstein		2. Durchlauf erfolgreich ohne Fehler			
6.2 Einweisung und Übergabe an Kunden			1 h		
6.2.1 Projekt Präsentation und Vorführung				30 min	29.03.17
6.2.2 Abnahme durch Kunden				30 min	29.03.17
7. Abschluss	7 h				
7.1 Nachbereitung			1 h		
7.1.1 Fehlerkorrekturen				30 min	30.03.17
7.1.2 Versionierung abschließen				30 min	30.03.17
7.2 Dokumentation			6 h		31.03.17
8. Projektende	0 h				

Wirtschaftlichkeit

Ein Schaden am Unternehmen, hervorgerufen durch ‚Men in the Middle' Angriffe in Verbindung mit der Entwendung von Daten oder ähnlichem, ist nicht kalkulierbar, auch ein wirtschaftlichen Totalschaden kann die Folge sein. Es gilt diese Schäden vom Unternehmen abzuwenden durch die Anwendung aller möglichen Maßnahmen, daher sind Sicherheitsmaßnahmen wie die Nutzung von kryptographischen Lösungen unabdingbar.

Projektkosten (Plan-/Ist-Kosten)

- Kurzfristige Kosten (fallen in diesem Projekt nicht an)
  - [empfohlen] Zusätzlicher Speicherplatz mit separater einfacher Sicherung (RAID 1) wenn nicht bereits Vorhanden
  - [empfohlen] Datenträger zur Aufbewahrung der CA-Zertifikate (3x CD’s und oder 2x USB-Sticks)
  - Angenommene Personalkosten

Vergabe	Stunden	Einzel	Summe
Extern	35 h	85 €	∑ 2975 €
Intern	35 h	35 €	∑ 1225 €

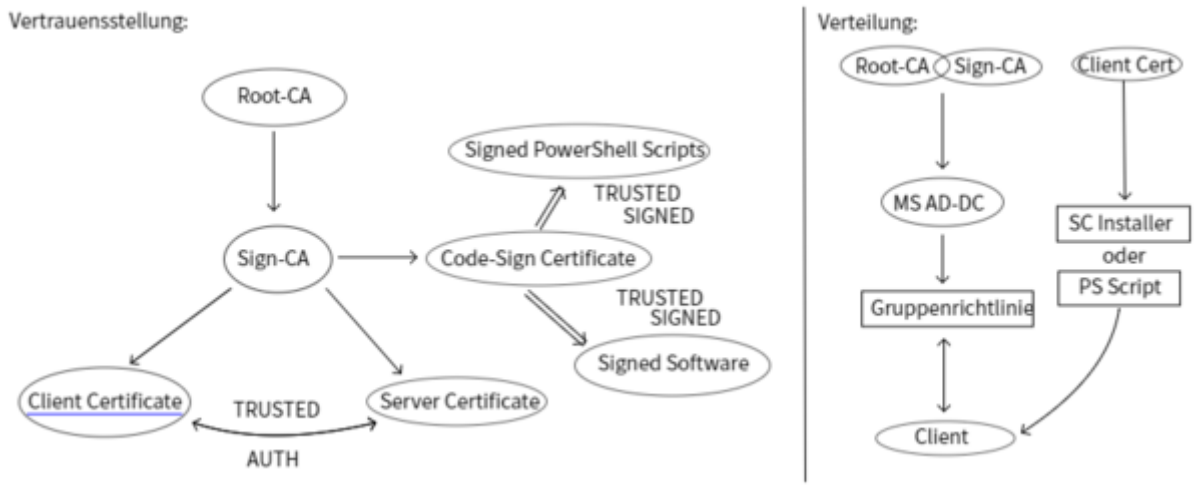
- Mittelfristige Kosten
  - zusätzlicher geringer Stromverbrauch
  - [empfohlen] 4 Jahre Turnus, Erneuerung der Festplatten im RAID 1 Verbund
- In diesem Projekt kam nur frei zugängliche Software bzw. Open Source Software zur Anwendung.

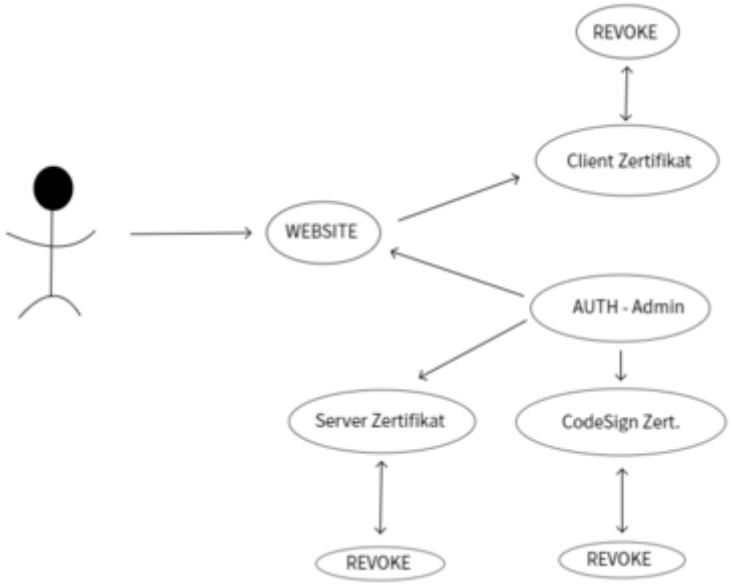
Projektverlauf

Dokumentation

Entwurf

Proof of Concept PKI-Struktur





Erleuterung PKI

- Die Nutzung eines öffentliches Schlüssels erfordert eine Vertrauenstellung und setzt voraus das der Schlüssel im Besitz des richtigem Subjektes ist (einer Person oder eines Systems) bei dem ein Verschlüsselungs- oder digitaler Signaturmechanismus zur Anwendung kommt.
- Diese Vertrauensstellung wird durch die Verwendung eines öffentlichen Schlüsselzertifikates erreicht, welcher der Datenstruktur des Subjektes im öffentlichen Schlüssel verbindlich entspricht.
- Diese Verbindlichkeit wird durch eine CA mit einer digitalen Unterschrift erklärt.
- Diese Erklärung basiert auf technischen Möglichkeiten der CA, z.B. über eine Challenge-Response-Authentifizierung , über das Vorzeigen des privaten Schlüssels oder über eine Erklärung des Subjektes.
- Solche Zertifikate haben nur einen definierte begrenzte Lebenszeit, welche ein Teil des unterschriebenen Inhaltes ist. Dies ermöglicht die unabhängige Überprüfung bei Client-Zertifikat Authentifizierung. Solche Zertifikate können über nicht Vertrauenswürdige Kommmunikatinos- und Server-System verteilt werden, und in ungesicherten Speicher des zu zertifizierenden Clients abgelegt werden.

Übersetzung aus [(rfc5280)]

[(rfc5280>https://tools.ietf.org/html/rfc5280 | RFC5280)]

RFC 5280 PDF RFC 5280 PKI Def. 3.1

Die PKI-Zertifikat Struktur für dieses Projekt

Certificate Authority		Subordinated CA		non-CA certificates
CA: <FIRMA2> Root CA X1	–>	Intermediate: <FIRMA2> Sign CA X2	–>	Server
			–>	Client
			–>	Codesign

Script oder Binary Parameter

Der Script Aufruf erfolgt mit den folgen Parametern :

Beispiel: `./use_cert.sh -srv create PC-Name_001 192.168.0.1`

	Host-Type:		Funktion:		notwendige Angaben	optional
		Parameter:		Parameter:		
1	Server	-srv	Ausstellen	create	<Hostname> <IP-Adresse>	<Tage>
2	Client	-cli	Sperren	revoke	<Hostname>	
3	Code	-code	Prüfen	check	<Hostname>	

Ordner-Struktur

Die angewandten Strukturen sollten bereits zu beginn des Projektes so konzipiert werden, das es möglich ist für andere Systeme einen Zugang zu schaffen. Beispielsweise müssen/sollten die Ordner für das Web-Interface nicht direkt im Hauptordner für den Webserver liegen.

Ausgearbeitet wurden die Funktionen: Erstellen, Sperren und Prüfen, welche auf die verschiedenen Zertifikats-Arten angewendet werden. Das sind Server, Client und Code Zertifikate die dazu notwendigen Angaben sind Hostname und IP-Adresse optional kann auch die Gültigkeitsdauer angegeben werden.

OpenSSL Ordner Strukturen:

- Hauptordner

Betriebssystem	Name	Location	Funktion	File-Konfiguration	User
Linux	PKI-CA Hauptordner	`/opt/ca/<CA-Class>`	Projekt Gesamt-Ordner	0755	root
Linux	CA-Root Zertifikate	`/opt/ca/<CA-Class>/certs`	Ablage CA-Root Zertifikate	0755	root
Linux	CA-Sign Hauptordner	`/opt/ca/<CA-Class>/intermediate`	Hauptordner Intermediate	0755	www-data
Linux	Sign Zertifikate	`/opt/ca/<CA-Class>/intermediate/certs/<HOSTNAME>`	Ablage CA-Sign-Zertifikate	0755	www-data
Linux	Web-Interface	`/opt/ca/<CA-Class>/intermediate/ppwi`	HTML und PHP Scripte	0755	www-data

- Unterordner

Betriebssystem	Name	Location	Funktion	File-Konfiguration	
Linux	Key Files	`../privates`	Private Keys	0755	www-data
Linux	Config Files	`../templates`	Konfigurationen	0755	www-data
Linux	Requests Files	`../requests`	Anfragen	0755	www-data
Linux	CRL Files	`../revokes`	Sperrungen	0755	www-data

Begriffe (CA-Root, Intermediate, Sign, Key , Config , Requests , CRL , File-Konfiguration)

Apache Webserver

Betriebssystem	**Name**	**Location**	**Funktion**	**File-Konfiguration**	
Linux	SSL-Config	`/etc/apache2/sites-enabled/default-ssl.conf`	Webserver SSL Konfigurations-File	0755	root
Linux	Web-Interface	`/var/www/html/ppwi`	`/opt/ca/<CA-Class>/intermediate/ppwi`	Symlink	www-data

Durch die gewählte Struktur ist der Zugriff auf die notwendigen Dateien für die Ausführung der PHP-Scripte vereinfacht worden, auch braucht so die Versionsverwaltung in nur einem Ordner angewandt werden.

Der Ordner mit den Web-Interface Dateien wurde als Submodule der Versionsverwaltung bekannt gemacht.

```
git submodule add [url] ppwi/
```

openSSL

Die Konfiguration der Profile für jeweilgen Zertifikat-Arten wurde mit der Version 3 des X.509 Standards ermöglicht.

Wichtige angaben wurden mit `default` vordefiniert. Diese sind ggf. bei Standort Änderungen und ähnlichen Anzupassen.

3.1.1 Template ca.conf

- Profil Datei für die CA:
  - <FIRMA2> CA Root X1
  - <FIRMA2> CA Sign X2 (ähnlich)

[SNIPPET]

```
countryName_default             =       DE
stateOrProvinceName_default     =       Sachsen-Anhalt
localityName_default            =       Magdeburg
0.organizationName_default      =       <FIRMA2> Test
organizationalUnitName_default  =       <FIRMA2> Root Certificate Authority
commonName_default              =       <FIRMA2> CA Root X1
emailAddress_default            =       <FIRMA2>@<FIRMA2>.local
```

[SNIPPET]

```
[ req_distinguished_name ]
# See <https://en.wikipedia.org/wiki/Certificate_signing_request>.
countryName                     = Country Name (2 letter code)
stateOrProvinceName             = State or Province Name
localityName                    = Locality Name
0.organizationName              = Organization Name
organizationalUnitName          = Organizational Unit Name
commonName                      = Common Name
emailAddress                    = Email Address
```

```
[ v3_ca ]
# Extensions for a typical CA (`man X509v3_config`).
subjectKeyIdentifier        = hash
authorityKeyIdentifier      = keyid:always,issuer:always
basicConstraints        = critical, CA:true
keyUsage            = critical, digitalSignature, cRLSign, keyCertSign,
dataEncipherment, keyEncipherment, nonRepudiation
extendedKeyUsage        = clientAuth, serverAuth, emailProtection,
msCTLSign, msEFS, codeSigning, caIssuers
issuerAltName               = issuer:copy
crlDistributionPoints       = URI:http://pp-linux1/revoke.crl
nsComment                   = <FIRMA2> Stammzertifikat
```

Nach dem das Grundprofil für die Certificate Authority angelegt wurde kann die Bash-Verarbeitung umgesetzt werden. Im weiteren Projekt Verlauf ist an dieser Stelle ein schwerwiegender Fehler deutlich geworden. Im Ursprünglichen Entwurf habe ich bei der `extendedKeyUsage` nur die Werte `caIssuers`, `msCTLSign` und `msEFS` definiert gehabt. Aus diesem Grund war es in der CA Hierarchie dem Intermediate-Zertifikat untersagt Client Zertifikate auszustellen, dies Führte bei der Client-Server Authentifizierung zur gegenseitigen Ablehnung Zertifikate. Es war notwendig auch `clientAuth` und `serverAuth` hinzuzufügen.

> Das unterzeichnende (Intermediate) Zertifikat muss die Funktionen welche sie signiert selbst beinhalten
>
> eine weitere Möglichkeit um auch die Verwendung des Client Zertifikates einzuschränken besteht darin das `extendedKeyUsage` als `critical` zu definieren. (ist für dieses Projekt jedoch nicht Vorgesehen)

3.1.2 BASH Script CA-Cert.sh

[SNIPPET]

- Root Certificate:

```
openssl genrsa -aes256 -out privates/ca.key.pem  \
      -passout pass:$capw 4096

openssl req -config $caconfig \
      -key privates/ca.key.pem \
      -new -x509 -days 730 -sha512 \
      -extensions v3_ca \
      -out certs/ca.cert.pem \
      -passin pass:$capw \
      -batch
```

- Intermediate Certificate:

```
openssl genrsa -aes256 \
      -out intermediate/priates/intermediate.key.pem \
      -passout pass:$caspw 4096
```

```
openssl req -config $caiconfig -new -sha512 \
      -key intermediate/priates/intermediate.key.pem \
      -out intermedicsrests/intermediate.csr.pem \
      -passin pass:$caspw \
      -batch
openssl ca -config $caiconfig -extensions v3_intermediate_ca \
      -days 3650 -notext -md sha512 \
      -in intermediate/csr/intermediate.csr.pem \
      -out intermediate/certs/intermediate.cert.pem \
      -passin pass:$capw \
      -batch
```

3.1.3 Template cic.certs.conf

[SNIPPET]

```
[ v3_usr_cert_ca ]
# Extensions for user certificates (`man X509v3_config`).
basicConstraints                                = CA:FALSE
nsCertType                                      = objsign, email
nsComment                                       = "<FIRMA2> CA :VAR6: Class
2 - :VAR5:"
subjectKeyIdentifier                            = hash
authorityKeyIdentifier                          = keyid,issuer
keyUsage                                        = critical,
digitalSignature, keyEncipherment, nonRepudiation, dataEncipherment,
keyAgreement
extendedKeyUsage                                = codeSigning,
emailProtection, msCodeInd, msCodeCom
crlDistributionPoints                   = URI:http://pp-
linux1/:VAR6:/<FIRMA2>_policy_:VAR6:.crl
authorityInfoAccess                             = @ocsp_info
subjectAltName                      = $ENV::UAN
issuerAltName                                   = $ENV::UAN

[ v3_srv_cert_ca ]
# Extensions for server certificates (`man X509v3_config`).
basicConstraints                                = CA:FALSE
nsCertType                                      = server
nsComment                                       = "<FIRMA2> CA :VAR6: Class
2 - Server :VAR2:"
subjectKeyIdentifier                            = hash
authorityKeyIdentifier                          = keyid,issuer:always
keyUsage                                        = critical,
digitalSignature, keyEncipherment, nonRepudiation, dataEncipherment,
keyAgreement
extendedKeyUsage                                = serverAuth, clientAuth
crlDistributionPoints                   = URI:http://pp-
```

```
linux1/:VAR6:/<FIRMA2>_policy_:VAR6:.crl
authorityInfoAccess                                = @ocsp_info
subjectAltName                          = $ENV::SAN


[ v3_cli_cert_ca ]
# Extensions for client certificates (`man X509v3_config`).
basicConstraints                                   = CA:FALSE
nsCertType                                         = client
nsComment                                          = "<FIRMA2> CA :VAR6: Class
2 - Client :VAR2:"
subjectKeyIdentifier                               = hash
authorityKeyIdentifier                             = keyid,issuer
keyUsage                                           = critical,
digitalSignature, keyEncipherment, nonRepudiation, dataEncipherment,
keyAgreement
extendedKeyUsage                                   = clientAuth
crlDistributionPoints                        = URI:http://pp-
linux1/:VAR6:/<FIRMA2>_policy_:VAR6:.crl
authorityInfoAccess                                = @ocsp_info
subjectAltName                          = $ENV::SAN
```

3.1.4 BASH Script CIC-Cert.sh

[SNIPPET]

- Client Certificate

```
openssl req -new -key $key -out $csr -config $cnf -batch;

openssl x509  -CA $int_crt_file -CAkey $int_key_file -CAcreateserial $passin
-req -in $csr -days $4 -outform PEM  -out $pem  -$sha_rsa -extensions
$v3_ext ;

openssl ca -config $cnf -in $csr -days $4 -out $crt -extensions $v3_ext
$passin -batch;

openssl pkcs12 -export -inkey $key -name "$2" -in $crt -out $pfx $passout ;
```

Microsoft X.509-Storage

C# Program Cert-Installer.exe

- Das zu installierende PFX-File enthält neben dem Client-Zertifikat auch den Private-Key. Dieser Private Schlüssel ist zur Client-Authentifizierung gegen über einem Server notwendig.

- Die Installation stellte eine besondere Herausforderung dar, der Download der <ClientName.pfx> und die Anschließende Installation in den Windows-Certificate Store ergab

doch einige Hürden. Die sinnvollste Lösung war den Bezug des PFX-Files für den Clienten über die `WebRequest` und `WebClient` Klassen per HTTPS zu realisieren.

- Im Anschluss kam das folgende Programm [SNIPPET] zur Anwendung.

Quelle: msdn.microsoft.com

[SNIPPET]

```
       private void InstallCertPfx(string certpw)
       {
            try
            {
               string certPath = $@"{resultTempPath}\<FIRMA2>-
{clientname}.pfx";
                var cert = new X509Certificate2(certPath, certpw);

                var store = new X509Store(StoreName.My,
StoreLocation.CurrentUser);
                store.Open(OpenFlags.ReadWrite);
                if (!store.Certificates.Contains(cert))
                {
                    store.Add(cert);
                }
                var storepers = new X509Store(StoreName.TrustedPeople,
StoreLocation.CurrentUser);
                storepers.Open(OpenFlags.ReadWrite);
                if (!storepers.Certificates.Contains(cert))
                {
                    storepers.Add(cert);
                }
            }
            ...
        }
```

Nach einigen Fehlversuchen und Fehlerkorrekturen konnte auch das hier vorliegende Mysterium zumindest umgangen werden. Sowohl der Edge-Browser von Microsoft wie auch der Chrome, welche beide auf den Windows.X509Store zugreifen konnten sich sporadisch nicht am Server mittels des im `Eigene Zertifikate` befindliche Zertifikat (inkl. Private Key) Authentifizieren. Im weiteren Verlauf erkante ich das bei dem Authentifizierung-Versuch auch auf den X509Store.Vertraute Personen zu gegriffen wurde. Daher entschied ich für auch diesen Zertifikat-Storage anzusteuern.

Weiter sollte sichergestellt sein das kein bereits Installiertes Zertifikat wieder Exportiert wird mit privatem Schlüssel, daher wird das PFX-File vom Server aus mit einer Passphrase versehen und ausgeliefert. Diese Passphrase ist im Programm `Cert-Installer.exe` codiert hinterlegt.

PS Script Firefox MFF-Cert-Install.ps1

Für Firefox wird nicht ausschließlich ein PFX-File benötigt sonder kan auch ein PEM-File sein. Quelle: developer.mozilla.org

```
$ComputerName = "<FIRMA2>-cli"+$env:COMPUTERNAME".pem"
```

```
$ProfilePath = "C:\Users\" + $env:username +
"\AppData\Roaming\Mozilla\Firefox\Profiles\"
$ProfilePath = $ProfilePath + (Get-ChildItem $ProfilePath | ForEach-Object {
$_.Name }).ToString()

C:\ctmp\certs\nss\bin\certutil -A -n "<FIRMA2>" -t "CT,C,C" -i
$certlocation\$ComputerName -d $ProfilePath
```

- Zur Ausführung des PS-Scriptes auf der Domäne muss dieses noch Signiert werden. Voraussetzung dazu sind eine Installierte CA-Hierarchie und eine CodeSign Zertifikat. Windows PowerShell signieren
- Wichtige Einstellungen und Einsicht in den X.509 Store

```
Set-ExecutionPolicy allsigned
$Zertifikat = @(gci cert:\currentuser\my -codesigning)[0]
Set-AuthenticodeSignature C:\powershell\MFF-Cert-Install.ps1 $Zertifikat
Sign MFF-Cert-Install.ps1 $Zertifikat
```

Web-Interface

- Übergabe der vom Web-Interface übergebenen Formular Informationen an das Script ./usr_cert.sh.

PHP-Code Snippet

```
    function cert($parm)
    {
        exec('cd /opt/ca/x2/intermediate/; ./use_cert.sh '.$parm.' 2>&1',
$out0);
                foreach ($out0 as $i)
                {
                    echo "<p> $i </p>";
                }
    }
```

- Bootstrap → Quelle des freien Bootstrap CSS-Framework

Implementation

4.1 Linux Server

- 4.1.1 Update & Installation

```
apt update && apt upgrade -y;

apt install openssl apache2 php7.0-cli php7.0-curl php7.0-gd \
            php7.0-geoip php7.0-intl php7.0-json \
            php7.0-mbstring php7.0-mcrypt php7.0-mysql \
```

```
            php7.0-opcache php7.0-readline php7.0-xml \
            php7.0-xsl php7.0-zip;
```

- 4.1.2 BASH-Scripte Lösung integrieren

  - über git-Repository einbringen

```
git clone [url] /opt/ca
```

oder

- Anlegen der Ordner Struktur (s.o) mittels `mkdir` - Die vorbereiteten Dateien in die Ordner Struktur übertragen via FTP oder SSH/SFTP
- 4.1.3 Webserver Konfiguration
  - Das Script welches später ausgeführt wird erzeugt für den Server die nötigen Zertifikate unter `/opt/ca/intermediate/certs/srv-PPLINUX001`
  - der Apache2 Server muss noch für SSL Konfiguriert werden

```
a2enmod ssl
```

- und die Konfiguration:

`nano /etc/apache2/sites-available/default-ssl.conf`

```
               SSLEngine on
               SSLUseStapling on
               SSLCertificateFile      /opt/ca/<CA
CLASS>/intermediate/certs/srv-PPLINUX.crt
               SSLCertificateKeyFile   /opt/ca/<CA
CLASS>/intermediate/privates/srv-PPLINUX001.pem
               SSLCertificateChainFile /opt/ca/<CA
CLASS>/intermediate/certs/intermediate.crt
               SSLCACertificatePath    /opt/ca/<CA
CLASS>/intermediate/certs/
               SSLCACertificateFile    /opt/ca/<CA
CLASS>/intermediate/certs/ca.bundle.crt
               SSLCARevocationPath     /opt/ca/<CA
CLASS>/intermediate/revokes/
               SSLCARevocationFile     /opt/ca/<CA
CLASS>/intermediate/revokes/<FIRMA2>_policy_X1.crl
               SSLVerifyClient         optional
               SSLVerifyDepth          10
```

- im Anschluss noch die Konfiguration aktivieren

```
   ln -s /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-
enabled/default-ssl.conf ;
   apachectl -t ;
```

Die Ausgabe sollte mit `Syntax OK` die korrekte Konfiguration bestätigen.

- 4.1.4 CA Zertifikate erzeugen & einbinden

```
    /opt/ca/ca_create.sh;
    /opt/ca/intermediate/use_cert.sh -srv create PP-LINUX1 192.168.6.201 365
;
    chown -R www-data:www-data /opt/ca/intermediate;
    sleep 10;
    apachectl restart ;
```

- 4.1.5 Über Web-Server Prüfen

```
curl -Iv http://pp-linux1
```

Die Ausgabe sollte `* Connected to ...` enthalten

oder den Web-Browser des Clients die Erreichbarkeit prüfen.

- 4.1.6 Fehler Dokumentation
    - `d.Z. Keine Daten vorhanden`

4.2 Windows Server & Client

- 4.2.1 Windows Server Updates

```
  Aufgrund der DMZ Restriktionen der virtuellen Umgebung können keine
Updates bezogen werden.
```

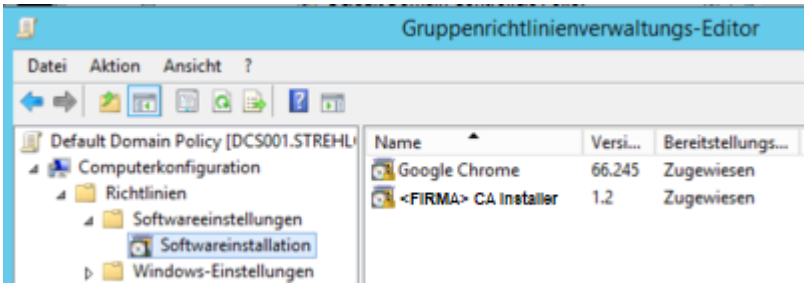
- 4.2.2 Domäne-Controller konfigurieren

- `d.Z. Keine weiteren Daten vorhanden`

> Auf den Inhalt des Projektes ist sind am Domäne-Controller keine Einstellungen notwendig
>
> Die Clients müssen sich in der Domäne befinden

- 4.2.3 Active Directory konfigurieren

- Gruppenrichtlinie



Das Programm Cert-Installer.exe konnte nicht als EXE-File über die GPO verteilt werden, dies erforderte noch die Integration in ein MSI-File, was über die Software „Advanced Installer 13“ realisiert wurde.

Im Anschluss kann über die GPO:

```
-> Gruppenrichtlinien-Editor -> Client OU -> Default Domain Policy ->
Computerkonfiguration -> Richtlinien -> Softwareeinstellungen ->
Softwareinstallation
```

die MSI Datei an die Clients Verteilt werden.

- 4.2.4 Windows Client Updates
  - `d.Z. Keine weiteren Daten vorhanden`
- 4.2.5 Windows Client in Domäne Aufnehmen
  - `d.Z. Keine weiteren Daten vorhanden`
- 4.2.6 Konnektivität zwischen allen Systemen prüfen
  - `d.Z. Keine weiteren Daten vorhanden`
- 4.2.7 Fehler Dokumentation
  - `d.Z. Keine Daten vorhanden`

4.3 Web-Interface

- 4.3.1 Web-Interface einpflegen

```
ln -s /opt/ca/x2/intermediate/ppwi/ /var/www/html/ppwi
chown www-data:www-data /var/www/html/ppwi
```

- 4.3.2 Web Ausgabe überprüfen

1. Prüfung

```
curl -L -k http://pp-linux1/ppwi
```

Die Ausgabe sollte HTML-Code enthalten

1. Prüfung

Über den Web-Browser die Webseite → http://pp-linux1/ppwi und https://pp-linux1/ppwi aufrufen

- 4.3.3 HTTPs Verbindung protokollieren

Beispiel einer Positiven Verbindung im Logfile von Apache:

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

Server certificate subject=/C=DE/ST=Sachsen-Anhalt/O=<FIRMA2>
Test/OU=<FIRMA2> Root Certificate Authority/CN=PP-Linux1/emailAddress=PP-
Linux1@192.168.6.201

issuer=/C=DE/ST=Sachsen-Anhalt/O=<FIRMA2> Test/OU=<FIRMA2> Root Certificate
Authority/CN=<FIRMA2> Sign CA X3/emailAddress=<FIRMA2>@192.168.6.201

Acceptable client certificate CA names /C=DE/ST=Sachsen-Anhalt/O=<FIRMA2>
Test/OU=<FIRMA2> Root Certificate Authority/CN=<FIRMA2> Sign CA
```

```
X3/emailAddress=<FIRMA2>@192.168.6.201 /C=DE/ST=Sachsen-
Anhalt/L=Magdeburg/O=<FIRMA2> Test/OU=<FIRMA2> Root Certificate
Authority/CN=<FIRMA2> CA Root X1/emailAddress=<FIRMA2>@192.168.6.201
Client Certificate Types: RSA sign, DSA sign, ECDSA sign Requested Signature
Algorithms: ...+SHA1:ECDSA+SHA1
Shared Requested Signature Algorithms: ...SHA1 Peer signing digest: SHA512

Server Temp Key: ECDH, P-256, 256 bits

SSL handshake has read 8085 bytes and written 5673 bytes
```

- 4.3.4 Fehler Dokumentation
    - `d.Z. Keine Daten vorhanden`

Test / Validierung

Die Zertifikate können auf dem Linux-Server mittels openSSL geprüft werden.

```
openssl verify -CAfile /opt/ca/intermediate/certs/sign-ca-chain.crt <client-
certificate>
```

Sollte die Ausgabe OK erzeugen.

Um einen ausschließlichen Client-Authentifizierungsvorgang zu provozieren muss der Apache2 Server noch einmal konfiguriert werden.

In der Datei `/etc/apache2/sites-available/default-ssl.conf` wird der Wert „optional“ durch „require“ ersetzt. Anschließend muss der Apache Dienst neugestartet werden.

```
     SSLVerifyClient require
```

Nach dem die Zertifikate an die Client(s) verteilt wurden, sollte mittels der Browser Edge und Chrome beim Aufrufen der Webseite https://pp-linux1/ eine Abfrage über das Zertifikat erfolgen und mit <OK> quittiert werden können.

Dann nach sollte die Apache Test Seite zu sehen sein.

> Fehlercode: SSL_ERROR_HANDSHAKE_FAILURE_ALERT oder ähnliches weißt auf das Fehlen oder ein Fehlerhaftes Zertifikat hin.

der Linux-Client PP-Linux2 ermöglicht es hier noch eine Prüfung zu vollziehen

```
openssl s_client -connect pp-linux1:443 -CAfile ca-chain.cert.pem -cert
publicCert.pem -key private.key -showcerts
```

Die Ausgabe sollte unter anderem `verify return:1` beinhalten.

Einsatz

> Bei der Wiederholung der oben beschriebenen Vorgänge kam es nur durch eine fehlerhafte Web-Server Konfiguration zu einer Störung. Dann jedoch gab es keinen weiteren Schwierigkeiten.
>
> Dem Windows Server 2008 konnte mittels des Cert_Installer.exe / bzw. der MSI kein Zertifikat übermittelt werden, da keine sichere Verbindung über HTTPS aufgebaut werden konnte bzw. von Windows ablehnt wurde. Als Ursache habe ich den älteren Internet Explorer und fehlenden Updates ermittelt.

Abschluss

Projekterfolg

Soll-Ist-Vergleich

Nr	Ziel	Kurz Beschr.	Status
1	CA-Root	Ausstellen eines Certificate Authority Root Zertifikat	
2	CA-Sign	Ausstellen eines Certificate Authority Sign Zertifikat	
3	Client Certs	Ausstellen Client Zertifikate per Script	
4	Deploy CAs	Verteilen CA-Root/CA-Sign Zertifikat	
5	Deploy Client Certs	Abrufen & installieren Client Zertifikate	
6	Renew on expired CAs	Erkennen von Ablauf CAs Zertifikate und Erneuerung	
7	Web-Gen. Client Cert	Erzeugen von Client-Zertifikat via Web-Interface	
8	Gen. Revoke List	Erzeugen einer abrufbaren Sperrliste	
9	Cli-Srv Auth	Authentifizierung des Clients am Server via Zertifikat	

Ausblick

- Weitere Einsatzmöglichkeiten
  - Software Signierung

    * LDAP Zertifikate und (Wikipedia LDAP)

      - RFC 2587
      - RFC 2559
      - RFC 2459
  - VPN Technologie

Folgeaktivitäten

- Erstellen einer Binary sowie eines Repository Package (deb bzw. rpm)
- OCSP Erweiterung
- Veröffentlichung GitHub

Fazit

Das Projekt umfasste in der Planung 35 Stunden, diese Zeit konnte aufgrund einer Störung im betrieblichen Ablauf im Vorfeld des Projektes und wegen meines gravierenden Fehlers bezüglich der PKI-Vererbungshierarchie[(#1)] nicht absolut Eingehalten werden.

Mangels der Internet Anbindung und dem dadurch ausgebliebenen Updates der Windows Systeme habe ich 2 Stunden gewonnen, welche ich wie oben beschrieb dann 4 Stunden Fehler und Ursachen Behebung gegenrechnen muss. Der zeitlich Vergleich zwischen Soll und Ist wurde dem Anhang hinzugefügt.

Auch musste ich feststellen das das strengende erw. Wasserfallmodell nicht zwangsläufig die beste Wahl sein muss für solche Projekte. Das ständige Hin und Her zwischen den Plattformen, die Abhängigkeiten bei Server Zertifikat und Client-Authentifizierung sowie das damit verbundene neustarten der Dienste konnte auch nicht den Fehler Frühzeit erkenntlich machen.

Glossar

[(gl:ca>CA - Certificate Authority)]

[(gl:dmz> DMZ - Demilitarisierte Zone)]

[(gl:pki> PKI - Demilitarisierte Zone)]

[(gl:prot:http> HTTP - Hyper Text Transfer Protokoll - Port 80)]

[(gl:prot:https> HTTPS - Hyper Text Transfer Protokoll Secure - Port 443)]

[(gl:prot:ftp> FTP - File Transfer Protokoll - Port 21)]

~~REFNOTES gl ~~

Protokolle

~~REFNOTES gl:prot ~~

Kürzel	Erklärung
DMZ	Demilitarisierte Zone
Isuser	

Abbildungsverzeichnis

Quellennachweis

~~REFNOTES~~

~~REFNOTES cite 10 ~~ ~~REFNOTES cite /2 ~~

Citations

<refnotes cite /2> refnote-id: i; note-preview: none </refnotes>

[(rfc5820>rfc5820)] - https://tools.ietf.org/html/rfc5820

[(RFC6818>RFC6818)] - https://tools.ietf.org/html/rfc6818

RFC3280

"anyPolicy" - Wildcard

PKI-Certificate-Chaining

Policy identifier - PEN

PEN Request - iana

Anlagen / Anhang:

Weitere Angaben

- Domäne: test.pp
- Signaturalgorithmus / Hash-Algorithmus : sha256rsa / sha256
- Root-CA: <FIRMA2> CA Root X1
- Sign-CA: <FIRMA2> CA Sign X2

<FIRMA2> CA Root X1

```
countryName_default             =       DE
stateOrProvinceName_default     =       Sachsen-Anhalt
localityName_default            =       Magdeburg
0.organizationName_default      =       <FIRMA2> Test
organizationalUnitName_default  =       <FIRMA2> Root Certificate Authority
commonName_default              =       <FIRMA2> CA Root X1
emailAddress_default            =       <FIRMA2>@<FIRMA2>.local
```

<FIRMA2> Sign CA X2

```
countryName_default             =       DE
stateOrProvinceName_default     =       Sachsen-Anhalt
localityName_default            =       Magdeburg
0.organizationName_default      =       <FIRMA2> Test
organizationalUnitName_default  =       <FIRMA2> Root Certificate Authority
commonName_default              =       <FIRMA2> Sign CA X2
emailAddress_default            =       <FIRMA2>@<FIRMA2>.local
```

Ressourcen

- Virtuelle Umgebung und Betriebssystem

Name	**IP**	**Betriebssystem**	**Funktion**	**Konfiguration**
PP-Linux1	192.168.6.201	#93-Ubuntu SMP	PKI-Server	Ground
PP-Linux2	192.168.6.202	#93-Ubuntu SMP	Example SRV/CLI	Ground
w2k8-template	192.168.6.203	Windows Server 2k8	AD-Domäne Controller	Template
TestCli	192.168.6.204	Windows 10	Example Client	Template

- Netzwerk Konfiguration

- Netzwerk: 192.168.6.0 / 24 GW .1

Anforderungen

- Version 3 X.509 Standard (X.509v3)
- mind. SHA256 in Zertifikaten
- Linux Server
  - Apache Web-Service / NginX Web-Service
  - MySql bzw. MariaDB Datenbank
  - SSL Client Certificate Authentication
  - BASH o.ä.
  - openSSL
- Windows Server 2012 RC2 bzw. Windows Server 2008 RC2
  - Domäne-Server (mind. Grundkonfiguration)
  - Active Directory - Domäne-Controller
  - [optioal] OCSP Rolle (Online Certificate Status Protocol)
- Windows Client(s)
  - MS Windows 7+
  - Clients in AD-DC migriert bzw. migrierbar
  - Browser: iE 8+, FireFox, Chrome
- PowerShell
- VisualStudio mit C#, C++
- Definierte Netzwerk-Umgebung

Systemvoraussetzungen

- mindest Hardwareanforderung

    - x64-Prozessor AMD-V mit NX-Bit oder Intel VT mit XD-Bit als DEP
    - 8 GB RAM
    - 2 GB freien Festplattenspeicher
    - integriertes 1-GBit/s Netzwerk
    - Internetverbindung
- optional bei realer/produktiver Umsetzung (Bsp. Debian Jessie)
    - empfiehlt es sich den „PKI“-Linux Server auf einer eigenständigen Hardware bzw. unabhängig vom Hyper-V mit Windows Servern und somit auf einem separaten Virtualisierungs-Server zu betreiben.
    - Nahezu alle x86-basierten (IA32) Prozessoren, 32-Bit AMD- und VIA- (früher Cyrix) Prozessoren sowie Prozessoren wie den Athlon XP und Intel P4 Xeon.
    - nicht auf 486- oder älteren Prozessoren oder Intel Quark
    - Intel Pentium und dessen Klone inklusive denen ohne eine FPU (Fließkommaeinheit oder mathematischem Coprozessor) werden unterstützt.
    - 80 MB / 128 MB / 512 MB RAM (Abs. Minimum / Minimum Betrieb / Empfohlen)
    - 680 MB / 2 GB / 10 GB freien Festplattenspeicher
    - integriertes 1-GBit/s Netzwerk
    - Internetverbindung
- mind. Softwareanforderung
    - Windows Server 2012 oder Windows Server 2008
    - Windows 10/8/7
    - Linux-Distro (Debian / Ubuntu / Fodera / openSuSe o.ä.) als Net-Install-ISO oder Standard-ISO’s

Entwicklungsumgebung

- VisualStudio 2015/2017 Community
- Eclipse (optional)
- VIM / Nano / gedit
- Brackets

Prüfprogramme

- WireShark [1)]
- tcpdump [2)]

Programmiersprachen

- C#
- PowerShell-Script
- Linux-Shell (sh/bash)
- PHP

Nützliche Tools

- git Versionsverwaltung

Soll-/Ist Vergleich Zeitplanung

Nr	Phase	Kurz Beschr.	Soll-Zeit	Ist-Zeit	Diff
1	Projektstart	Vorstellung des Projektes	1 h	1 h	
2	Analyse & Definition	Soll-Ist Analyse	2 h	2 h	
3	Entwurf	Ausarbeitung Scripte	5 h	5 h	
4	Implementation	Aufsetzen Server / Client Instanzen	10 h	8 h	- 2
5	Test / Validierung	Testlauf Zertifikat Verteilung	8 h	12 h	+4
6	Einsatz	Roll-Out	2 h	2 h	
7	Abschluss	Übergabe	7 h	7 h	
		Summe	**35 h**	**37 h**	

[1)]

WireShark | https://www.wireshark.org/

[2)]

tcpdump | http://www.tcpdump.org/tcpdump_man.html