keyon

keyon / Validation Server

Produktbeschreibung

V2.0 - 03/2014

Keyon AG
Schlüsselstrasse 6
8645 Jona
Switzerland

Tel +41 55 220 64 00
Fax +41 55 220 64 01

www.keyon.ch
info@keyon.ch

Inhaltsverzeichnis

1 Einführung

Der *keyon / Validation Server* ist ein Modular aufgebauter, OCSP V1 (RFC 2560) konformer Responder für die Online-Prüfung von Zertifikaten. OCSP V1 fähige Clients können OCSP V1 Anfragen an den *keyon / Validation Server* senden, um den Status eines Zertifikats abzufragen, ohne dabei clientseitige Sperrlisten (CRL) zu verwenden. Der *keyon / Validation Server* unterstützt zusätzlich zu RFC 2560 das Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments nach RFC 5019.

Der *keyon / Validation Server* erfüllt alle im RFC 2560 beschriebenen MUSS (MUST) Kriterien.

2 Validation Server

2.1 High-Level Architektur

Die High-Level Architektur des Validation Server ist wie folgt[1]:

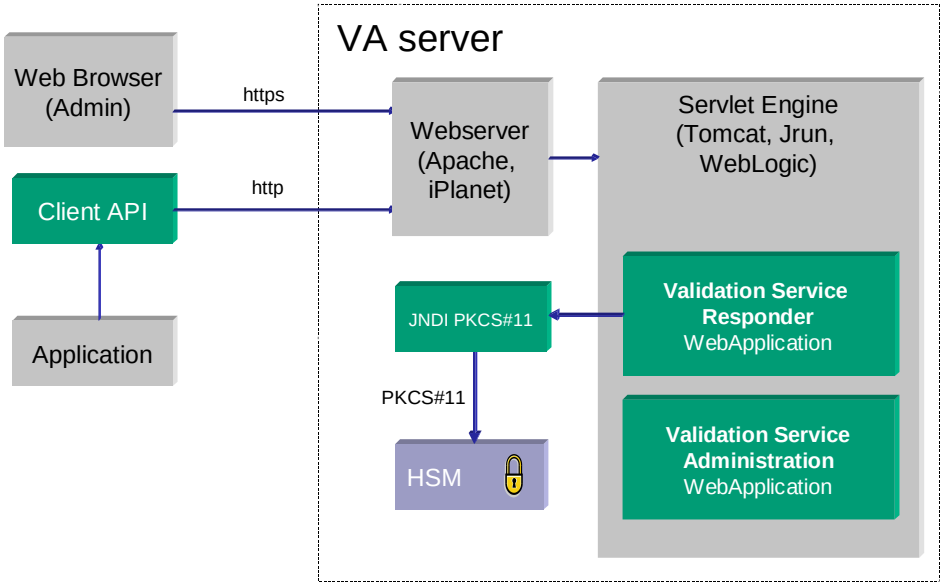

Bild 1: Architektur Validation Server

[1] Der Validation Service Responder kann auch als Standalone Server betrieben werden. Siehe Kapitel 9 für die verschiedenen Deployment Möglichkeiten.

2.1.1 Admin

Die Kontrolle (start/stop) sowie die Konfiguration des Validation Servers erfolgt über ein Webinterface. Der Administrator benötigt nur einen Standard Webbrowser sowie ein Client-Zertifikat.

Benutzer können Gruppen zugeordnet werden welchen über ein Access-Control Management entsprechende Administrationsrechte zugeteilt werden.

Das Datenbank Passwort kann über den Administration Service lokal verschlüsselt abgelegt werden, damit der Validation Server auch automatisch (unattended) gestartet werden kann.

Trusted Root Zertifikate können über das Webinterface installiert werden so dass nach dem Initialen Setup die gesamte Konfiguration über das Webinterface realisiert wird. Konfigurationen können exportiert und importiert werden.

2.1.2 Web-Server/Servlet Engine

Alle gebräuchlichen Protokolle für die Zertifikatsvalidierung (OCSP, SCVP, OCSPv2) verwenden das HTTP Protokoll zum Transport der Meldungen. Als Basis für den Validation Server kann daher ein Webserver mit Servlet Engine eingesetzt werden welcher das Connection Handling übernimmt. Ob eine Kombination Web-Server/Servlet Engine oder nur eine Servlet Engine eingesetzt wird spielt aus der Sicht des Validation Server keine Rolle, kann aber einen Einfluss auf die Performance des System haben.

Alternativ kann der Validation Server auch als Standaloneserver betrieben werden. Ein eingebauter, minimaler http Server kommt in diesem Fall zum Einsatz.

2.1.3 Validation Server Responder

Der Responder ist als Webapplikation und als Standaloneapplikation ausgeführt. Die detaillierte Architektur des Responders ist in Kapitel 4 beschrieben.

2.1.4 Validation Server Administration

Die Administration des Validation Servers ist als Webapplikation ausgeführt. Die Web-Applikation ist kompatibel zu den gängigen J2EE Applikationsservern.

Ein eigner Applikationsserver (Tomcat) wird mitgeliefert falls kein Applikationsserver zur Verfügung steht.

2.1.5 JNDI PKCS#11

Um von Java aus das optionale Hardware Sicherheitsmodul (HSM) anzusteuern wird eine Java<->PKCS#11 Bridge mitgeliefert.

2.1.6 Hardware Sicherheitsmodul (HSM)

Das HSM dient zur Beschleunigung von kryptographischen Operationen sowie zur sicheren Ablage von Schlüsseln und wird über die PKCS#11 V2.01 angesteuert. Der Schlüssel zum Signieren der OCSP Antworten wird im HSM generiert und das signieren mit dem Schlüssel erfolgt auf dem HSM selbst.

Falls kein HSM verwendet wird, wird der Schlüssel in einer mit einem zufällig generierten, starken Passwort geschützten PKCS#12 Datei abgelegt.

2.1.7 Weshalb diese Architektur?

Untenstehend sind die wesentlichen Gründe für die Wahl der Architektur aufgelistet:

Webserver/Servlet Architektur

- Bewährte Technologie, wird auch für grosse Installationen eingesetzt.
- Es existiert eine Vielzahl von Technologien für die Skalierung und Last-Verteilung von Web-Applikationen.
- Die Administration und Konfiguration wird über ein Web-Interface durchgeführt und benötigt keinen direkten Zugriff auf den Server.
- Das Deployment von Web Applikationen ist einfach.
- Die Implementation ist nicht von einem bestimmten Betriebsystem abhängig. (Ausnahme Native-Code für die HSM Anbindung).
- Kann mit anderen Webapplikationen ko-existieren und benötigt keinen eigenen dedizierten Webserver.

Java

- Stabiles Framework, keine Speicherlecks bei Serverapplikationen.
- Exceptionhandling ermöglicht eine saubere Behandlung von Fehlern was für einen Non-Stop Betrieb wichtig ist.
- Keine Gefahr von Buffer-Overflows oder anderen Attacken.
- Durch konsequente Objekt-Orientierte Modellierung kann der Server jederzeit einfach erweitert werden.
- Hohe Rechenleistung wird primär für die kryptographischen Operationen benötigt, diese werden jedoch durch das HSM oder den WebServer (SSL) durchgeführt

Die Vorteile dieser Architektur gegenüber einer Realisierung als eigene Applikation komplett in Native-Code überwiegen nach unseren Erfahrungen bei weitem.

3 Administration

3.1 Weboberfläche

Die Administration erfolgt vollständig über eine Weboberfläche welche über SSL mit Client Authentisierung gesichert ist.

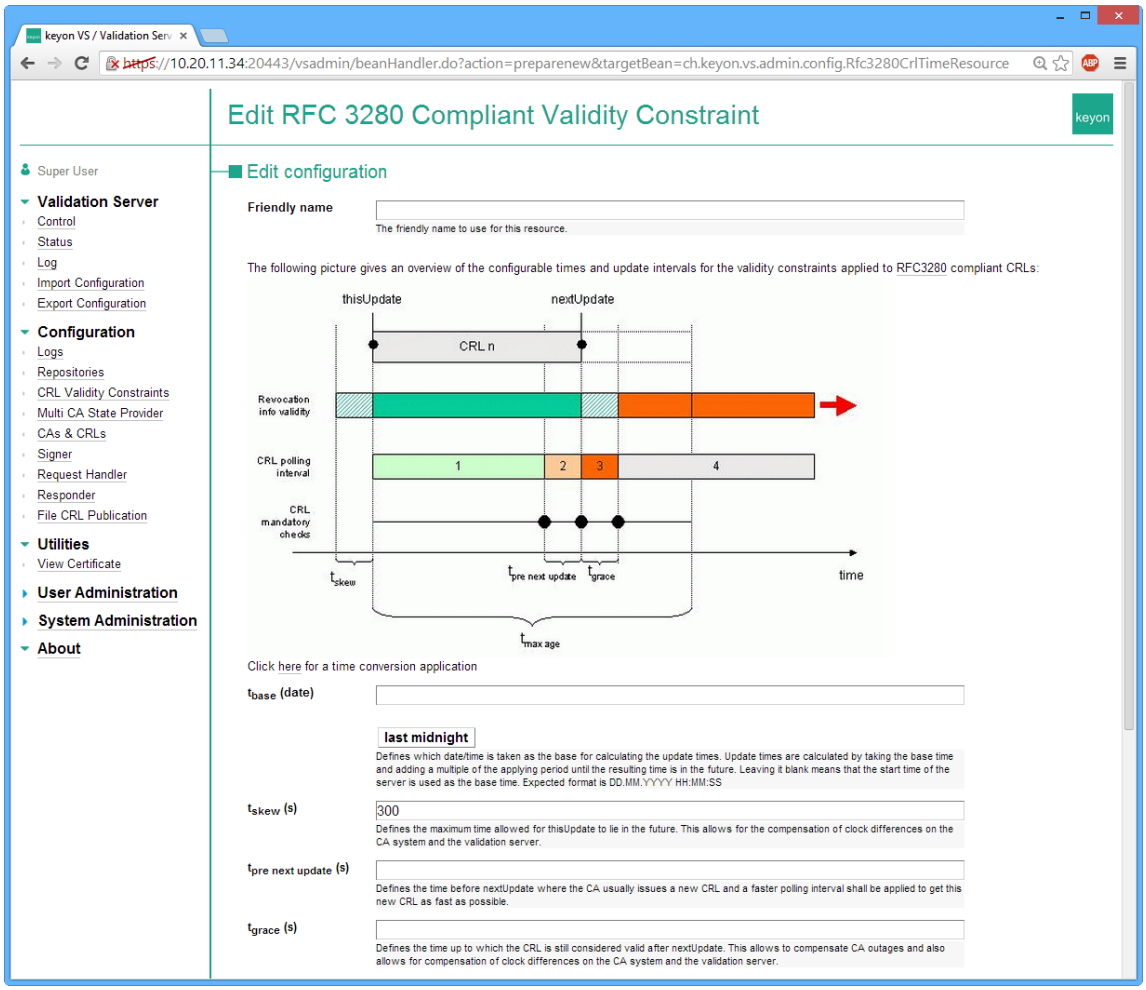

Bild 2: Screenshot Administrationsoberfläche

3.2 Rechteverwaltung

Administratoren werden über den DN ihres Zertifikats identifiziert. Die Benutzer werden Gruppen zugeordnet welche die Rechte der Benutzer definieren. Die Rechte lassen sich pro Gruppe detailliert vergeben:

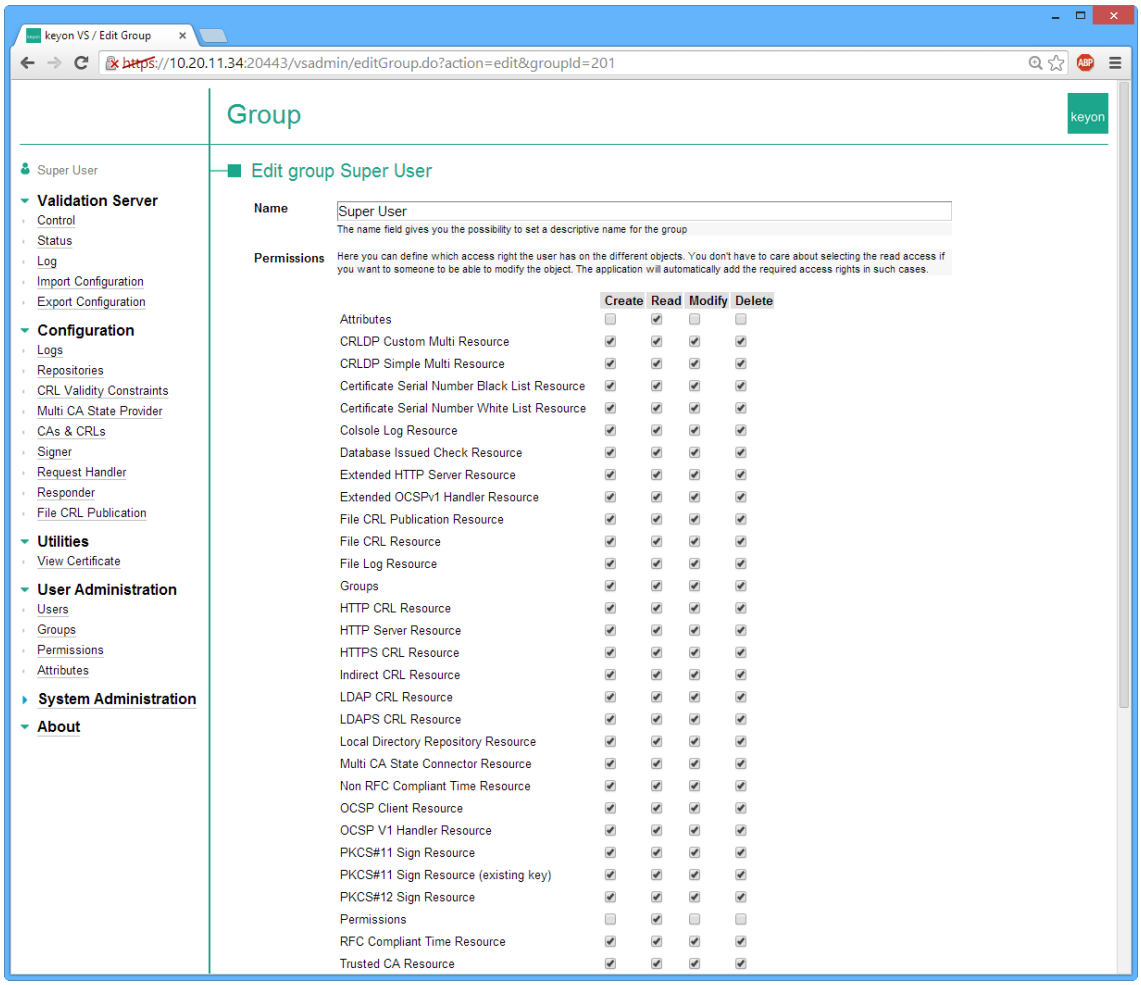

Bild 3: Screenshot Administration Gruppenrechte (Auszug)

Benutzer einer Gruppe mit nur Leserechten können beispielsweise die Konfiguration und die Log-Dateien über die Weboberfläche anschauen aber keine Änderungen an der Konfiguration selbst vornehmen.

Die Rechteverwaltung erlaubt eine klare Trennung zwischen Sicherheitsadministration, Betrieb und Audit.

4 Validation Responder

Der Validation Server besitzt eine hochmodulare und erweiterbare interne Struktur:

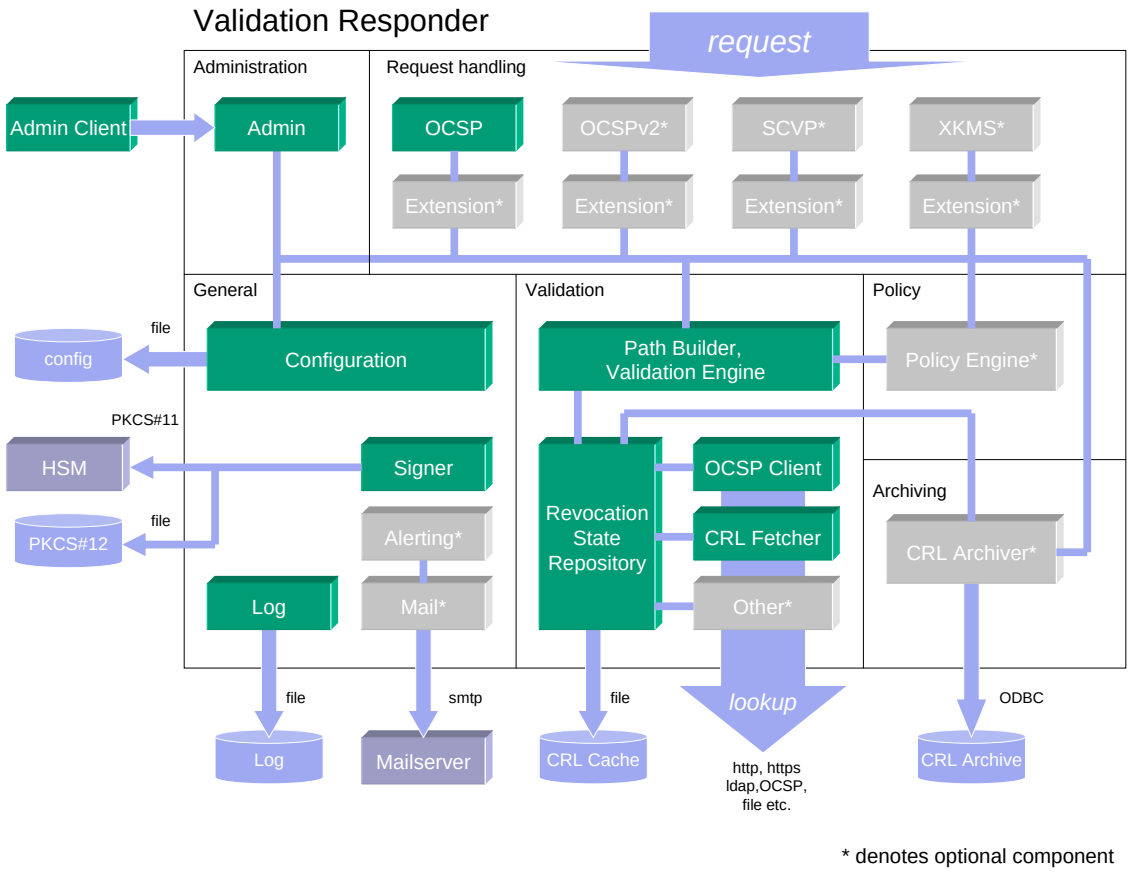

Bild 4: Interne Struktur der Validation Server

Validation Requests werden wie folgt verarbeitet:

1. Der Request wird durch den entsprechenden Handler entgegengenommen
2. Der Request wird auf Gültigkeit und Format überprüft
3. Die Informationen über das zu überprüfende Zertifikat werden aus dem Request extrahiert
4. Optional können Request Erweiterungen (Extensions) behandelt werden
5. Die Revokationsinformation für das gewünschte Zertifikat wird ermittelt. Dabei werden auch die Revokationsinformation der Aussteller ermittelt.
6. Die Policies für die Zertifikate im Pfad werden zugeordnet und überprüft[2]

[2] Nur mit Option Policy Engine und falls das verwendete Validierungsprotokoll das komplette zu prüfende Zertifikat übermittelt. Nicht für OCSPv1.

7. Optional werden Response Erweiterungen (Extensions) hinzugefügt
8. Der Response wird erstellt
9. Der Response wird signiert
10. Der Response wird an den Client zurückgeliefert

In den folgenden Kapiteln werden die technischen Spezifikationen der Basiskonfiguration des Responders erläutert.

4.1 OCSP Responder

Die Basiskonfiguration beinhaltet einen OCSP Responder nach RFC 2560. Nicht unterstützt werden signierte Requests da diese nur für die Erbringung von bezahlten Validierungsdienstleistungen von Interesse sind um den Benutzerkreis einzuschränken.

Requests gemäss RFC 5019 - The Lightweight Online Certificate Status Protocol (OCSP) Profile for High-Volume Environments werden unterstützt.

4.2 Unlimitierte Anzahl von Trusted CAs

Die Anzahl der unterstützten CAs ist nicht beschränkt. Die Quelle der Revozierungsinformationen wird pro CA unabhängig definiert. Jeder CA kann eine beliebige Anzahl von Quellen für die Revozierungsinformationen zugeordnet werden.

CA Hierarchien werden unterstützt.

Die maximal möglich Anzahl von CAs und CRL Quellen wird durch den verfügbaren Arbeitsspeicher des Servers definiert.

4.3 CRL Unterstützung

CRLs gemäss RFC 3280 werden unterstützt. Mit Ausnahme von Delta-CRLs und unbekannten kritisch markierten Extensions werden alle nach X.509 definierten CRL Formate unterstützt.

CRLs können im Binärformat (DER) mit den folgenden Protokollen bezogen werden:

- LDAPv2 und LDAPv3 (ohne Authentisierung)
- HTTP (Auch über Proxy, Basic Authentication wird für Server und Proxy unterstützt)
- HTTPS (Auch über Proxy, Basic Authentication wird für Server und Proxy unterstützt)
- Direkt aus lokalen Dateien

Beim Einsatz von CRL Distribution Points (wie z.B. von Entrust verwendet), können die CRLs dynamisch per Wildcard definiert werden, d.h. es müssen nicht alle CRLs manuell konfiguriert werden. (In einem OCSP Request sind leider nicht die notwendigen Informationen zur Ermittlung der CRL, d.h. die CRLDP Extension, enthalten.)

Die Intervalle für den CRL Update sind frei konfigurierbar. Vor Ablauf der CRLs werden diese unabhängig vom konfigurierten Intervall automatisch aktualisiert. Die Aktualisierung erfolgt im Hintergrund und blockiert den Responder nicht.

Die folgenden Grafiken geben einen Überblick über die Konfigurationsmöglichkeiten der Updateintervalle und Zeiten:

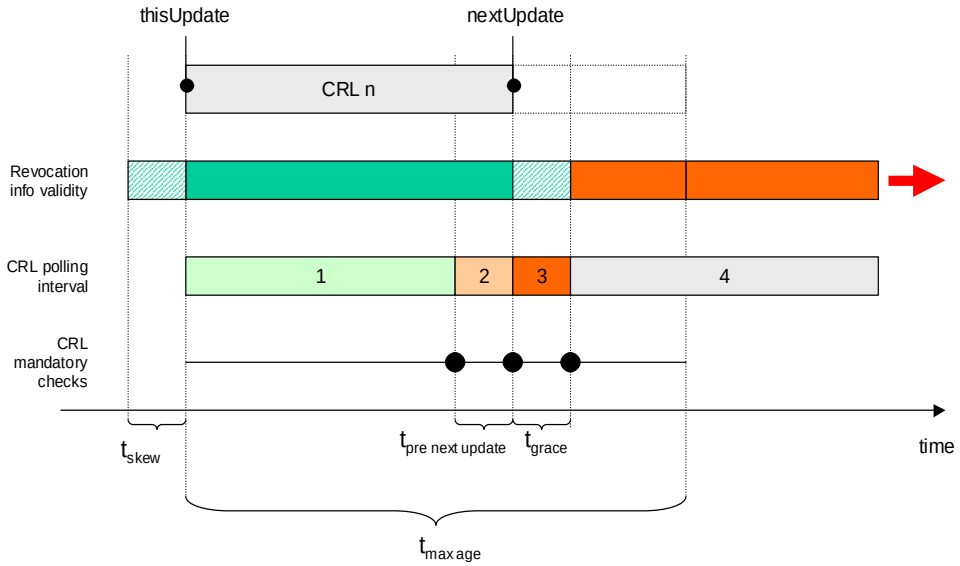

Bild 5: Beispiel der konfigurierbaren Zeiten und Intervalle falls nextUpdate im CRL vorliegt

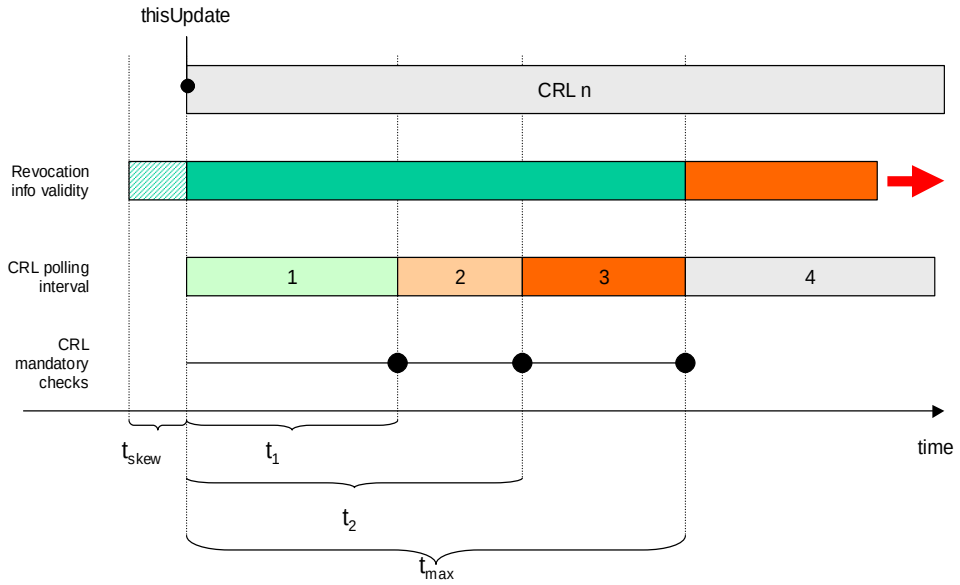

Bild 6: Beispiel der konfigurierbaren Zeiten und Intervalle falls nextUpdate im CRL nicht vorhanden ist

Die Informationen welche im OCSP Response geliefert werden basieren auf den zugrundeliegenden Revozierungsinformationen (thisUpdate wird vom CRL übernommen) sowie das nächste Update dieser Informationen auf Grund der Konfiguration (nextUpdate).

Die konfigurierbaren Zeiten und Intervalle werden als Templates angelegt welche den CRL Quellen zugeordnet werden. Damit können einheitliche Standards beispielsweise für interne CAs und externe CAs durchgesetzt werden.

4.4 Lokaler CRL Cache

Alle aktuellen CRLs werden lokal in einem dateibasierten Cache abgelegt um im Falle eines Neustarts des Systems als Basis für die Validierungsinformationen sofort und unabhängig von anderen Komponenten zur Verfügung zu stehen. (Nur gültige CRLs werden verwendet.)

Bei einem Neustart werden die aktuellsten CRLs jeweils unmittelbar geladen, der Cache dient nur dazu, die Zeit zu überbrücken bist die aktuellsten CRLs zur Verfügung stehen.

4.5 Revozierungsinformationen über OCSP ermitteln

Anstelle von CRLs als Quelle von Revozierungsinformationen können für entsprechend konfigurierte CAs die aktuellen Revozierungsinformationen über OCSP abgefragt werden.

4.6 Abfrage CA Datenbank ob Zertifikat ausgestellt wurde

Optional kann für jede CA eine Datenbank konfiguriert und abgefragt werden, ob die geprüfte Serienummer in der Datenbank existiert.

4.7 Signaturschlüssel in Hardware

Der Schlüssel mit welchem die Antworten signiert werden wird beim Einsatz eines HSMs im HSM erzeugt und verwendet. Bei einem Betrieb ohne HSM, z.B. in einer Testumgebung, kann der Schlüssel auch in Software generiert und verwendet werden.

Es gilt zu beachten, dass das für den Responder ausgestellt Zertifikat die für einen OCSP Responder notwendigen Extensions und Key Usages hat.

Die Administrationsoberfläche erlaubt die Erstellung von Certificate Signing Requests nach PKCS#10 sowie das Ausstellen von Self-Signed Zertifikaten.

4.8 Konfiguration

Die Konfiguration ist kryptographisch geschützt (MAC) und wird über die Administrations-Webapplikation erstellt und modifiziert. Die DNs der autorisierten Administratoren werden Initial lokal festgelegt und können später über das Administrationsinterface modifiziert werden.

Die Kommunikation des Admin Clients mit dem Responder ist über Standard SSL verschlüsselt und authentisiert.

4.9 Log

Es wird ein Log mit allen relevanten Events erzeugt. Die Events werden klassifiziert (Info, Fehler etc.) so dass eine automatische Auswertung für die Überwachung möglich ist.

Es kann eine beliebige Anzahl von Log Files konfiguriert und einzelnen Elementen (CAs, CRL Quelle etc.) zugeordnet werden. Die Log-Files werden optional täglich automatisch umbenannt was eine Archivierung erlaubt.

Die Admin Oberfläche erlaubt die Betrachtung der Logs sowie die Suche nach spezifischen Einträgen (Text, Loglevel).

5 Notfallmassnahmen

Der Validation Server kann in Notfällen die Revozierungsinformationen übergehen bzw. trotz Ablauf als gültig erachten um einen Weiterbetrieb der Clients zu ermöglichen.

5.1 Akzeptanz von Revozierungsinformationen

Es kann konfiguriert werden, dass CRLs trotz Ablauf weiterhin als gültig erachtet werden. Damit kann ein Betrieb trotz Ausfalls einer CRL Quelle (Directory, CA welche die CRLs erstellt) gewährleistet werden. Dies muss vom Administrator pro CRL Quelle konfiguriert werden.

Die Verwendung von Templates für die Konfiguration der Zeiten und Intervalle erlaubt es, ein Template für den Notfall zu erstellen und den Quellen bei Bedarf zuzuordnen.

5.2 Übersteuerung von Revozierungsinformationen

Die folgenden Massnahmen erlauben eine manuelle Übersteuerung von Revozierungsinformationen:

- Pro CA wird eine Blacklist mit Zertifikats-Seriennummern geführt welche in jedem Fall als Revoziert anzusehen sind. Der Administrator kann neue Seriennummern zu dieser Liste hinzufügen und vorhandene Seriennummern entfernen. Der Status der aufgelisteten Zertifikate wird mit Revoziert, Ursache *certificateHold* und Hold Instruktion *id-holdinstruction-reject* zurückgeliefert.
- Pro CA wird eine Whitelist mit Zertifikats-Seriennummern geführt welche in jedem Fall als gültig anzusehen sind. Der Administrator kann neue Seriennummern zu dieser Liste hinzufügen oder Seriennummern daraus entfernen. Damit kann bei unbeabsichtigter Revokation bis zum Ausstellen eines neuen Zertifikats der Betrieb einer kritischen Komponente aufrechterhalten werden. (Dieses Feature kann entfernt werden falls Sicherheitsbedenken bestehen) Der Status der aufgelisteten Zertifikate wird mit Revoziert, Ursache *certificateHold* und Hold Instruktion *id-holdinstruction-reject* zurückgeliefert.
- Trusted Root Zertifikate können vom Administrator als Revoziert markiert werden was den Status für alle von dieser CA ausgegebenen Zertifikate automatisch auf Revoziert mit dem Datum der Markierung und Ursache *cACompromise* setzt. Die Revozierung der CA im Validation Server kann nicht rückgängig gemacht werden.

6 Performance

Die Performance des Validation Servers ist von einer Vielzahl von Faktoren abhängig:

- Server Hardware und Software
- Webserver
- Servlet Engine
- OCSP Features (Ohne NONCE kann z.B. der Responder die Antworten auf Kosten der Sicherheit wiederverwenden.)
- Einsatz eines Hardware Sicherheitsmoduls
- Grösse des Signaturschlüssels
- Eingestellte Log Levels
- Verfügbarkeit von CRLs
- Konfiguration von externen OCSP Servern für die Statusabfrage

Die effektive Performance kann daher nur seriös in der direkten Zielinfrastruktur mit den gewünschten Einstellung ermittelt werden.

Tests mit einer Produktionsinfrastruktur und einer LunaSA 4 als HSM haben ergeben, dass mit einer Instanz des keyon / Validation Servers konstant mehr als 1‘000 OCSP Requests pro Sekunde beantwortet werden konnten wobei jeder Request den Status für zwei Zertifikate abgefragt hat. Dies ergibt für den getesteten Fall eine Kapazität von 3‘600‘000 OCSP Abfragen pro Stunde resp. 86,4 Millionen Abfragen pro Tag.[3]

Durch den Einsatz mehrerer Maschinen kann die Performance prinzipiell beliebig gesteigert werden zumal ein einfaches Load-Balancing wie bei normalen Webservern möglich ist.

[3] Unverbindliche Angabe, HSM Limite LunaSA 4 ist 1200 Signaturen pro Sekunde

7 Betriebssysteme

Der Validation Server ist, mit Ausnahme der PKCS#11 Anbindung, komplett in Java erstellt und daher Betriebssystem unabhängig. Die Java Runtime Umgebung 1.6.x wird benötigt, die Servlet Engine muss die Servlet Spezifikation 2.5 unterstützen.

Für die folgenden Plattformen wird eine PKCS#11 Anbindung geliefert:

- Solaris 9, 10 (Sparc, x86)
- Windows 2003, 2008, 2008R2, 2012, 2012R2 (x86, x64)
- Linux Systeme mit Kernel 2.4 oder höher (x86, amd64, x64)

Die Portierung auf andere Betriebssysteme ist möglich.

8 Optionen

Der modulare Aufbau lässt eine Vielzahl von Optionen zu. Um die minimalen Anforderung für OCSPv1 zu erfüllen werden die nachfolgenden Optionen nicht benötigt. Für die langfristige Sicherung der Investition ist es jedoch wichtig, dass der Validation Server jederzeit erweitert werden kann.

8.1 Unterstützung von Policies

Nachdem der Zertifizierungspfad erstellt wurde wird der Pfad gemäss der PKIX Policy überprüft. (z.B. path length restrictions, key usages etc.). Eigene Policies können optional implementiert werden (z.B. das Auswerten von Certificate Policies Extensions).

8.2 OCSPv2, SCVP, XKMS Unterstützung

OCSPv2 und SCVP Responder (beide Protokolle sind noch nicht definitive verabschiedet) oder einen XKMS Responder für die Validierung erlauben direkt die Verwendung von Clients welche diese Protokolle unterstützen.

8.3 CRL Archivierung

Die CRL Archivierung erlaubt die Ablage der CRLs in einer strukturierten Form. Diese CRLs können dann auch verwendet werden um den Zertifikatsstatus für Zeitpunkte in der Vergangenheit festzustellen, was insbesondere für das nachvollziehen einer Zertifikats-Suspendierung von Interesse sein kann.

8.4 Alerting

Beliebige Alarmierungsmodule können implementiert werden, welche z.B. im Fehlerfall (CRL kann nicht abgeholt werden etc.) eine Email versenden oder externe Applikationen anstossen.

9 Deployment

9.1 In Form einer Web Applikation

Der Validation Responder kann zusammen mit der Administrationsapplikation direkt in einer Servlet-Engine ausgeführt werden:

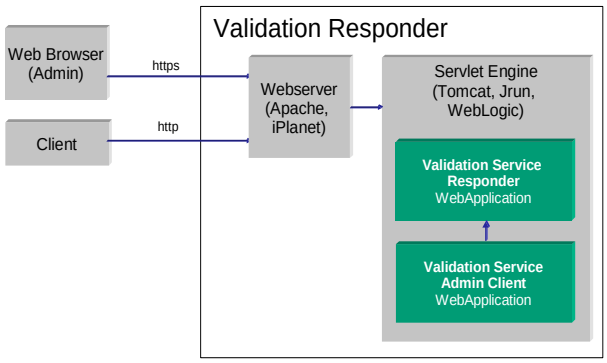

Bild 7: Deployment als Web Applikation

9.2 Als Standalone Server

Der Validation Responder kann mit Hilfe des eingebauten minimal http Servers auch im Standalone Betrieb eingesetzt werden. Für die Administrationsapplikation ist allerdings weiterhin eine Servlet-Engine notwendig.

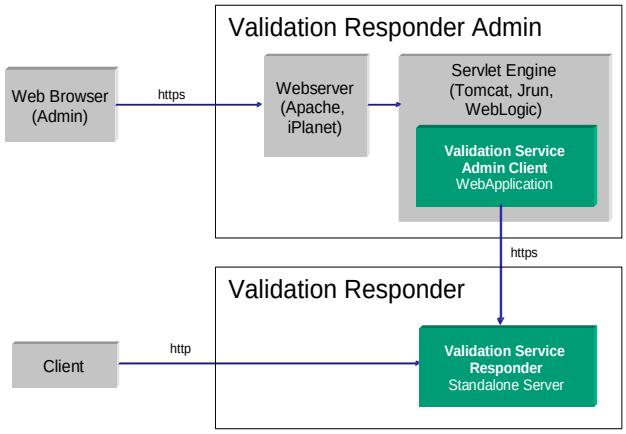

Bild 8: Deployment Standalone Responder