Geheimnisse für Container sicher und effizient verwalten

Docker Container, essendi xc Zertifikatemanager und HSM.

Docker-Plattformen bilden die ideale Infrastruktur für DevOps. Docker packt eine Anwendung sowie alle für deren Ausführung benötigten Systembestandteile in einen „Container".

Mit der Docker-Technologie lassen sich ganze Serverlandschaften abbilden und Anwendungen können auf Knopfdruck zur Verfügung gestellt werden.

Docker-Images benötigen für ihre Betriebsfähigkeit Geheimnisse wie Kennwörter und Schlüsselmaterial. Dieses Whitepaper zeigt Möglichkeiten auf für sicheren und effizienten Umgang mit Schlüsselmaterial in Containerplattformen.

essendi it GmbH
Dolanallee 19

74523 Schwäbisch Hall

Sicherer Umgang mit Zertifikaten in virtualisierten Docker-Umgebungen

Einleitung DevOps und Docker

In der heutigen Zeit sind kurze Zyklen zur Bereitstellung von IT-Applikationen der Standard. Die schnelllebigen Prozesse werden durch agile Softwaremethoden flankiert. Immer häufiger treten cross-funktionale Teams in Aktion, welche ein Verschmelzen der Aufgaben aus den Bereichen Entwicklung, Administration und Betrieb der Anwendungskomponenten vorantreiben. Sogenannte DevOps-Teams vereinen alle notwendigen Skills in einem Lösungsteam.

Docker-Plattformen bilden die ideale Infrastruktur für DevOps. Docker packt eine Anwendung sowie alle für deren Ausführung benötigten Systembestandteile in einen „Container". Docker-Plattformen sorgen dafür, dass die Anwendung beim Transfer über verschiedene Laufzeitumgebungen oder Stages hinweg verlässlich und ohne Betriebseingriffe läuft.

Docker-Infrastrukturen sind in diesem Kontext die erste Wahl zur Virtualisierung der Laufzeitumgebungen und Infrastrukturkomponenten. Daraus ergibt sich die Notwendigkeit die administrativen Prozesse für Build, Deployment, Administration, Konfiguration und Betrieb gleichermaßen zu organisieren, wie die Design- und Implementierungsaufgaben in Softwareprojekten.

Mit der Docker-Technologie lassen sich ganze Serverlandschaften abbilden und Anwendungen können auf Knopfdruck zur Verfügung gestellt werden. Speziell für komplexe Architekturen und Infrastrukturen spielt das einfache Management der Umgebungen eine wichtige Rolle.

Um mögliche Fehlerquellen auszuschließen, müssen die Umgebungen, idealerweise von der Entwicklung bis hin zur Produktion, identisch sein. Alle umgebungsspezifischen Aspekte sind über Konfigurationen in Umgebungsvariablen abzubilden. An den Images sollen über die gesamte Deploymentpipeline hinweg von der Entwicklung über Test bis zur Produktion keine Änderungen mehr notwendig sein.

Vorteile durch die Verwendung von essendi xc in Containerplattformen

- ✓ hoher Automationsgrad
- ✓ Effiziente und sichere Zertifikatsprozesse
- ✓ Einhalten von Regeln und Vorgaben an Zertifikate
- ✓ Weniger Fehlerquellen im Umgang mit Zertifikaten
- ✓ Keine Betriebsstörungen durch abgelaufene Zertifikate

Vorteile durch die Verwendung von essendi xc in Verbindung mit HSM

- ✓ Mächtige Krypto-Funktionen
- ✓ Umsetzung Ihrer Compliance Vorgaben
- ✓ Hardwaregesicherter Ablageplatz im HSM
- ✓ Sensible Dateien werden außerhalb potenziell unsicherer Container-Images gespeichert
- ✓ Schlüsselpaare direkt auf HSM erzeugen und nutzen
- ✓ Private und andere Schlüssel werden direkt im HSM erzeugt und müssen das HSM nicht verlassen

Sicherheitsrelevante Artefakte in Docker-Images

Die Anwendungskomponenten in Docker-Images treten in jeder Art auf, z.B. in Webseiten, Online-Portalen, Shopsystemen, SOAP-Webservices, Microservices in REST, Rechenkernen, etc. Die Softwarekomponenten müssen ihre notwendigen Vorkehrungen für die IT-Sicherheit enthalten, z.B. SSL-Zertifikate, Signaturzertifikate für Webservices oder Dokumente, Private-Keys, Datenbank-Kennwörter und andere Geheimnisse.

Der Umgang mit sicherheitsrelevanten Artefakten wie Schlüsselpaaren stellt dabei eine spezielle Herausforderung dar. Daher werden die Entwicklungsstufen voneinander getrennt und für jede Umgebung ein eigenes Docker-Image für eine jeweils eigene Test-/Umgebung angelegt. Insofern sind in jedem Stage wieder andere, passende Zertifikate, Schlüssel oder Kennwörter notwendig, die verwaltet werden müssen. Zudem ist die Speicherung von Sicherheitsartefakten direkt in den Docker-Images kein sicherer Ablageplatz und logistisch kompliziert.

Hier entsteht ein nicht unerheblicher administrativer und logistischer Aufwand. Für agile Softwareprozesse sind daher intelligente Lösungen mit hohem Automationsgrad bei gleichzeitig hohen Sicherheitsanforderungen erforderlich.

Abhilfe schafft hier ein Bundle bestehend aus einer Docker-Plattform, dem „essendi xc“ Zertifikatemanager in Kombination mit Hardware-Security-Modulen (HSM).

Vault-Verwaltung der Docker-Plattformen

Ein Vault-Speicher ist eine Komponente zur sicheren Speicherung von Geheimnissen wie Schlüssel und Zertifikate. Ein Vault ist in der Regel Bestandteil einer Docker–Plattform. Es gibt verschiedene Distributionen von Docker-Plattformen, z.B.

- Kubernetes von Google
- Red Hat - OpenShift, als eine kommerzielle Distribution von Kubernetes
- Docker Swarm

Die Plattformen machen die Docker-Technologie in Hochleistungssegmenten und Betriebssituationen mit hohen Anforderungen nutzbar. Hier ein Auszug aus dem Funktionsumfang:

- Orchestrierung von Containern, Skalierung, Clustering, Lastverteilung,
- Deploymentpipeline für einfachen Transfer der Anwendungskomponenten und aller Artefakte über die verschiedenen Stages hinweg
- Schnittstelle zur Quellcodeverwaltung, Git/Nexus, Build-Management,
- Service-Katalog-Management,
- Sicherheitsfunktionen, z.B. logische und physische Trennung von Laufzeitumgebungen innerhalb einer Containerplattform, z.B. durch Overlaynetzwerke.
- Erzeugen, speichern und Verwalten von Geheimnissen in Vaults

Vault-Stores gibt es von diversen Anbietern in unterschiedlichen Ausprägungen.

Vault:

- Free und Premium Variante – nur Premium enthält ein Rechte und Rollenkonzept und eine Segmentierungsmöglichkeit auf Mandantenebene
- kann z.B. mit Red Hat – OpenShift verwendet werden, bzw. ist in der offenen Version Bestandteil von Open Shift
- Sichert, speichert und kontrolliert Sicherheitsartefakte wie Tokens, Passwörter, Zertifikate, API-Keys… über die Vault CLI oder über REST APIs
- Verfügt über ein Feature zur Erzeugung von Schlüsselmaterial
- erlaubt Zugriff auf Key/Value Store, bietet Netzwerk Entschlüsselung als Service als auch die Generierung von Credentials wie X.509 Zertifikaten, SSH Credentials usw.

Docker Secrets – Raft Store:

- Docker native Lösung zum Management von Secrets
- Orchestrierung der Container über Docker Swarm. Secrets (Passwörter, SSH Schlüssel, TSL Zertifikate,…) werden als Sicherheitsobjekte über SSL zu den jeweiligen Containern geliefert
- Secrets werden vom Service Container über Manager (Docker Komponente) angefordert und anschließend im verschlüsselten internen Raft Store gesichert
- Konfiguration über Docker CLI / Commands

Kubernetes – Secrets:

- Erstellung von Secrets über Kubernetes Commands (kubelet)
- Beherrscht Ver- und Entschlüsselung von Passwörtern, Tokens oder Keys
- Einbindung des Secret über Pods (Umgebungskonfigurationsdatei), welche entweder als Volume in den Container eingebunden wird oder über den Kommandobefehl kubelet beim Beziehen von Images
- etcd dient als Speicher (hierarchischer Key-Value-Store) zur Verwaltung von diesen Artefakten in verteilten Systemen

Amazon AWS – Key Management Service:

- Zentrale Schlüsselverwaltung – Schlüssel erstellen, importieren (auch eigener bestehender Schlüssel in AWS Umgebung) und rotieren
- Hosting bei Amazon – Skalierbarkeit, Beständigkeit und hohe Verfügbarkeit, als auch Einhaltung von Compliance Regeln
- Integration und Verwendung des Dienstes in eigene Anwendungen über Verwendung der AWS SDK, AWS-CLI oder über eine RESTful-API

In der Regel haben alle verfügbaren Lösungen die Dezentralität gemeinsam. In orchestrierten Umgebungen wie Docker Swarm übernimmt die Managerkomponente das Verteilen der Secrets innerhalb von Docker Container, auf Applikationsebene stehen Kommandozeilenprogramme oder APIs zur Verfügung, die von der jeweiligen Lösung angeboten werden und der Endwender diese letztlich konsumieren kann.

Die Stores bieten in der Regel über mehrere Faktoren abgesicherte Zugänge zur Speicherung und zum Lesen der Geheimnisse. Diese werden als Blobs auf dem Dateisystem oder in Datenbanken verschlüsselt hinterlegt. Über sichere Interfaces und Umgebungsvariablen werden die Schlüssel, Zertifikate, etc. den Docker-Containern zur Laufzeit zur Verfügung gestellt.

Die Authentifikation und Autorisierung findet hauptsächlich über die Verwendung der REST-APIs statt.
Exemplarisch sei die Vault Lösung von HashiCorp herangezogen, über welche eine Authentifizierung stattfindet und anschließend Secrets generiert und wieder ausliest:

Annahme: Vault ist bereits eingerichtet, Authentifizierung über AppRole mit eingerichteten ACLs

1. Login über API /auth/approle/login mit Parametern “role_id” und “secret_id”
2. Verwendung des „client_token“ aus Login Antwort (für weitere Anfragen gegen das Vault benötigt)
3. Anlage von Secret über POST Aufruf der API /secret/secretname mit Übersendung des zuvor erstellten „client_tokens“
4. Abfrage eines Secrets über GET an API /secret/secretname, wobei die entschlüsselten Daten im data-Part der Serviceantwort übermittelt wird

Management sensibler Daten in HSMs

Die sicherste Variante zur Verwaltung von Geheimnissen, wie Schlüsselmaterial, Zertifikate und Kennwörter ist der Einsatz von Hardware-Sicherheits-Modulen (HSM).

Mit einem HSM werden die Schlüssel direkt darin erzeugt und gespeichert und sie müssen das HSM nicht mehr verlassen. HSMs liefern eine umfangreiche und breite Palette an leistungsfähigen Verschlüsselungsalgorithmen. Hardwaregesicherte Infrastruktur Sicherungsmechanismen sind sicherer als eine bloße Absicherung über Software, da sie schwerer angreifbar sind.

Ein HSM verfügt über umfangreiche Funktionalitäten:

- Key-Management mit vielen verschiedenen Verschlüsselungsalgorithmen
- HSMs sind über mächtige JCA-Interfaces oder UI-Konsole ansprechbar
- Client-/Server-Authentifikation und Autorisierung
- SSL-Prüfung, Ver- und Entschlüsselung von Webseiten und Webservices
- Internes Load-Balancing bei SSL-Offloading
- Granulare Authorisierungsmöglichkeiten - Segmentierung, d.h. Trennung der Bereiche im HSM nach Anwendungsgruppen oder Mandanten.
- Damit kann das Schlüsselmaterial für Docker-Plattformen nach Anwendungsgruppen, Mandanten, etc. isoliert werden
- Erfüllt Governance Compliance Anforderungen wie „FIPS 140“
- Erzeugen der Signaturen von E-Mails oder Webservices (Soap / REST) wird von der Anwendung / Programmcode direkt in das HSM delegiert. Das Zertifikat und der Private-Key verlässt hierbei das HSM nicht.

➢ Hier liegt der entscheidende Vorteil gegenüber den Vault-Stores

Management von Zertifikaten mit essendi xc

Mit essendi xc wird das Management von Zertifikaten automatisiert. Die Key-Generierung, Anforderung und Beglaubigung bis hin zum Enrolment von Zertifikaten durch interne oder öffentliche CAs wird damit weitgehend automatisiert und standardisiert. Über Interfaces können die Vault-Stores der Containerplattformen an den essendi xc Zertifikatemanager angebunden werden. Damit wird bereits ein hoher Grad an Standardisierung und Vereinfachung der Prozesse des Zertifikatsmanagements im Zusammenspiel mit Docker-Plattformen erreicht. essendi xc kann zudem über das JCA-Interface ein HSM direkt ansprechen und Schlüsselmaterial direkt darin erzeugen bzw. ablegen.

Weitere Informationen zu essendi xc siehe gesondertes Whitepaper essendi xc.

Anforderungen an den Umgang mit Zertifikaten in Docker-Umgebungen

Hier in der Zusammenfassung noch einmal die Anforderungen an das Zertifikatsmanagement in Docker-Umgebungen:

- Sichere Verwahrung der (privaten) Schlüssel, Zertifikate, Passwörter
- Sicherer Austausch der Schlüssel insbesondere private Keys
- Transparente und sichere Prozesse von der Anforderung bis zur Verteilung und Verlängerung von Zertifikaten (kein manueller Zertifikatsaustausch, z.B. per e-Mail)
- Konventionen für die Belegung der Zertifikate-Attribute, wie CN, DN, O, Schlüssellänge, Algorithmus etc. da diese auch für Authentifikation herangezogen werden
- Der Zertifikateprozess soll die Einhaltung der Konventionen sicherstellen
- Einhaltung von Zertifikate-Konventionen wie Keylängen, Algorithmen, CN Namen und andere Attribute
- Geregelte Rollen und Rechte für die Beteiligten der DevOps-Teams
- Funktionstrennung und eine definierte Stelle für die Genehmigung von Zertifikate-Requests, auch bei privaten CAs

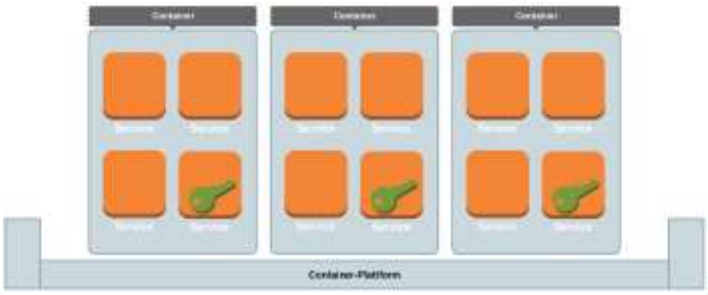
- Nutzung einer oder mehr CAs der Partner
- Je Entwicklungs- und Teststufe gesonderte Zertifikate
- Einfaches Handling in den Containern und der Plattform
- Warnung bei Ablauf von Zertifikaten
- Dokumentation der Zertifikate in zentralem Repository
- Einfaches Austauschen und Erneuern von Zertifikaten
- Einfaches Enrolment, d.h. Ausbringung der Zertifikate in ihre Ziel-Lokation
- Sperren von Zertifikaten z.B. bei kompromittiertem Schlüsselmaterial

Haben wir Ihr Interesse geweckt?

Gerne präsentieren wir Ihnen die Anwendung in einer Online-Demo per Webkonferenz oder präsentieren essendi xc live bei Ihnen im Haus. Kontaktdaten für eine Terminabstimmung:

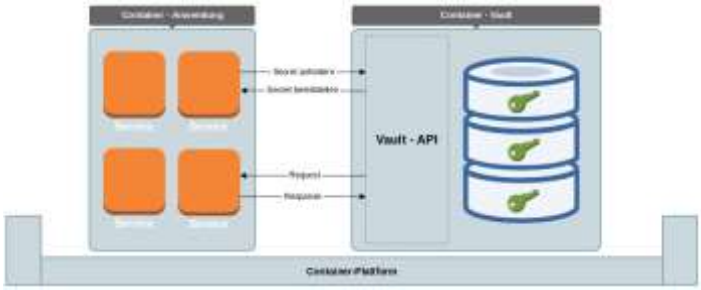
sarah.zuegel@essendi.de | Tel: +49 (0)89 944 697-71

Geheimnisse und Schlüsselmaterial für Container im Überblick



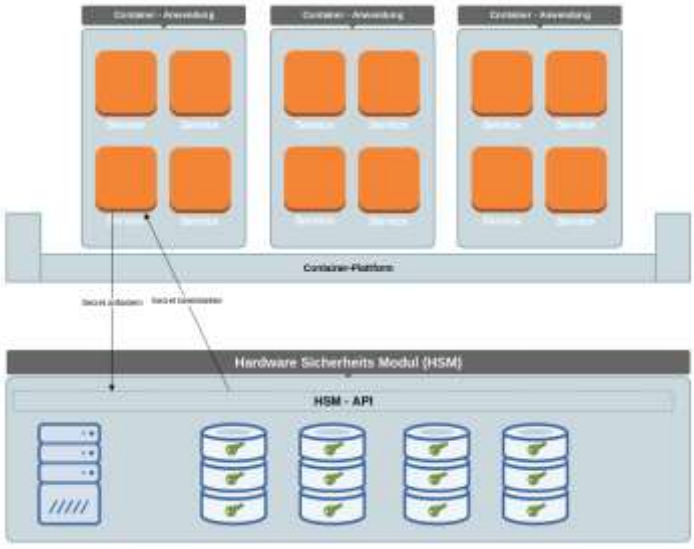
Stufe 0 – Schlüsselmaterial im Container

- Geheimnisse sind Bestandteil der Docker-Images
- Bestandteile des Quellcodes oder in Java-Keystores
- Schlüsselmaterial incl. Private Keys einfach zugänglich
- Transport / Deployment der Schlüssel über unsichere Kanäle
- Images sind über die Entwicklungsstufen hinweg nicht gleich



Stufe 1 – Schlüsselmaterial im Vault der Plattform

- Geheimnisse liegen in sicherem Vault-Speicher der Containerplattform
- Geregelter, sicherer Zugang zum Vault über Multifaktor-Authentifikation
- Stufenweises Autorisierungskonzept
- Transport der Schlüssel über unsichere Kanäle
- Images sind über die Entwicklungsstufen hinweg identisch

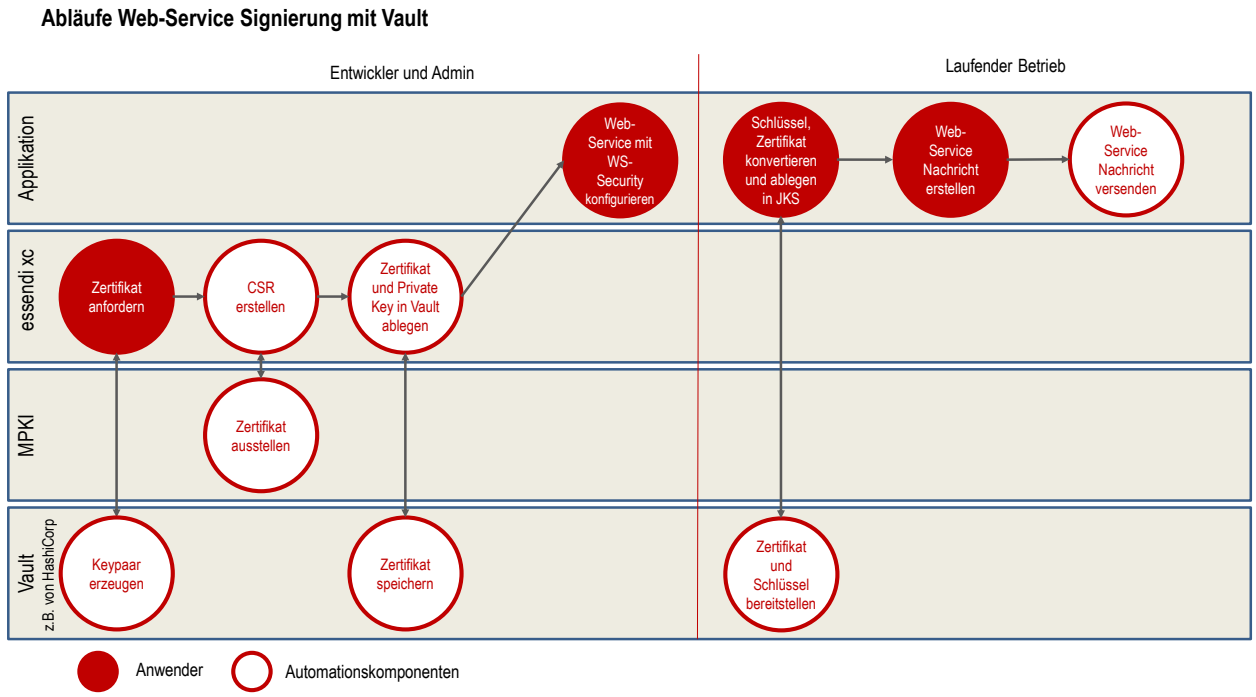


Stufe 2 – Schlüsselmaterial im HSM

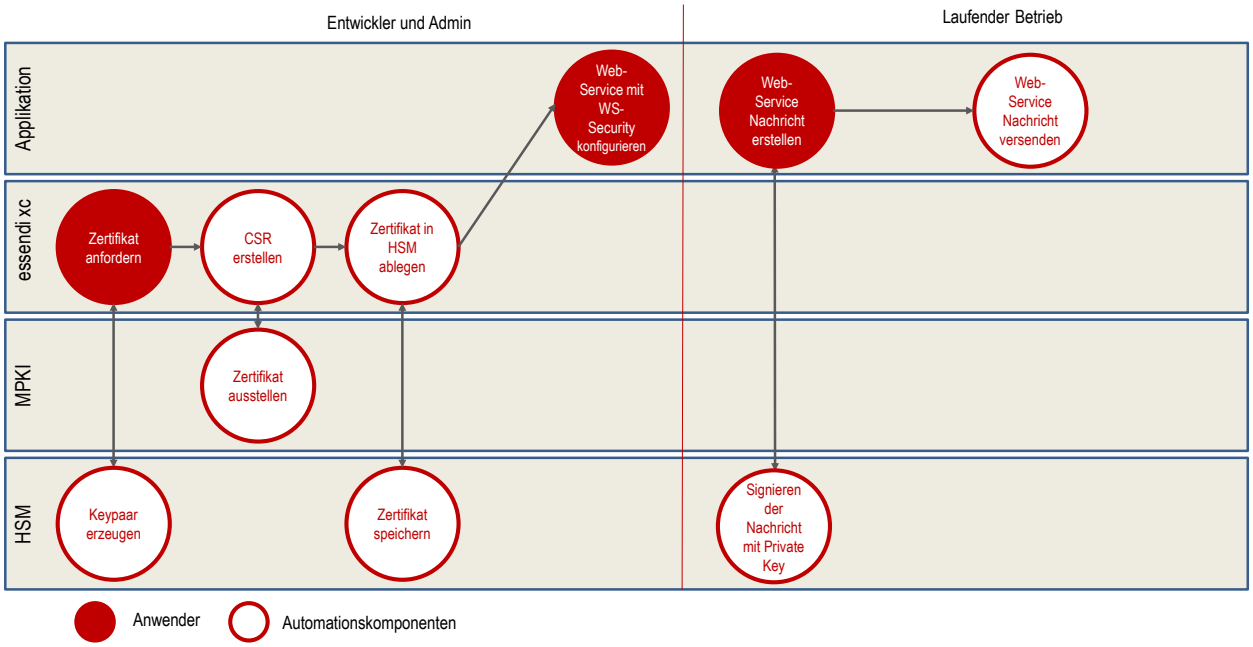
- Geheimnisse liegen ausgelagert in sicherem Hardware-Speicher
- Performante und umfangreiche Krypto-Algorithmen
- Geregelter, sicherer Zugang zum HSM über Multifaktor-Authentifikation
- Mandanten- oder Anwendungsbezogene Segmentierung der Speicherorte im HSM
- Autorisierungskonzept passend zur Segmentierung des HSM
- Kein Transport der Schlüssel über unsichere Kanäle
- Schlüssel werden im HSM erzeugt, private Schlüssel und andere Geheimnisse müssen das HSM nicht verlassen, da z.B. die Signatur von Webservice-Nachrichten direkt im HSM erfolgt
- Images sind über die Entwicklungsstufen hinweg identisch

Prozessabläufe schematisch

Das nachfolgende Schema zeigt den logischen Ablauf der Erzeugung von Schlüsseln und Geheimnissen, deren Speicherung / Bereitstellung über die verschiedenen Möglichkeiten mit Vault bzw. HSM.



Abläufe Web-Service Signierung mit HSM



Abläufe Webserver-Admin zur ssl-Absicherung Webseite (https)