Konzeption einer hochsicheren TLS-Auth Infrastruktur zur Authentifizierung von Industrie-Routern an einer Management-Software

Christopher Neldner
INSYS MICROELECTRONICS GmbH
Hermann-Köhl-Str. 22, 93049 Regensburg, Germany
E-Mail: cneldner@insys-icom.de

Professor Dr. Frank Herrmann
Ostbayerische Technische Hochschule Regensburg
Innovationszentrum für Produktionslogistik
und Fabrikplanung
Galgenbergstraße 32, 93053 Regensburg, Germany

Djordje Gladovic
INSYS MICROELECTRONICS GmbH
Hermann-Köhl-Str. 22, 93049 Regensburg, Germany
E-Mail: dgladovic@insys-icom.de

ABSTRAKT

IT-Sicherheit spielt besonders im IoT-Bereich eine kritische Rolle. In dieser Arbeit soll eine bestehende Public-Key-Infrastruktur, die im *Router Management* der INSYS icom für die Authentifizierung von Routern eingesetzt wird, auf optimierbare Stellen untersucht werden, und diese durch ein neues, verbessertes Konzept optimiert werden. Ein *Proof of Concept* zeigt die Realisierbarkeit dessen.

1 EINFÜHRUNG

Das *Internet of Things* (IoT) und darauf aufbauende Konzepte wie Industrie 4.0 und das *Smart Grid* erlangen täglich an größerer Bedeutung [1]. Um diese Konzepte praktikabel in einem realen Anwendungsszenario umzusetzen, bedarf es passender Hardware- und Softwarelösungen.

Das mittelständische Unternehmen INSYS MICROELECTRONICS GmbH [2] beschäftigt sich mittlerweile seit über 30 Jahren mit der Entwicklung spezieller Hardware für Kommunikation in verschiedenen Branchen. Aber auch Softwarelösungen, welche sich auf den Lebenszyklus von Routern oder auf das Aufbauen sicherer Kommunikationskanäle beziehen, befinden sich im Portfolio des Unternehmens. Dabei ist zu beachten, dass Kunden, die sich mit der Bereitstellung von kritischer Infrastruktur befassen, wie beispielsweise Energie- oder Wasserversorger, den strengen KRITIS-Vorlagen unterliegen [3]. Um diesen Markt zu erschließen, müssen Softwarelösungen diese Vorlagen ebenfalls erfüllen. Das Softwareprodukt, welches in dieser Arbeit im Fokus steht, ist das INSYS Router Management (iRM). Dieses ermöglicht dem Nutzer, eine große Anzahl von Routern komfortabel und effizient mit Firmware- und Softwareupdates zu versorgen. Um unbefugten Zugriff auf die Anwendung zu verhindern, nutzt das Produkt Transport Layer Security (TLS)-Clientauthentifizierung. Dabei muss ein Router bei Verbindungsaufbau ein gültiges Zertifikat vorweisen um überhaupt erst mit dem iRM kommunizieren zu können. Diese Funktionalität ist bereits funktionsfähig implementiert, weist jedoch Optimierungspotential in Handhabung und Implementierung auf.

In Abschnitt 2 wird dafür zunächst die Problemstellung erläutert und das Optimierungspotential der Implementierung aufgezeigt. Darauf folgend wird in Abschnitt 3 ein Konzept vorgestellt, welches eben genannte Stellen verbessern soll. In Abschnitt 4 werden im *Proof of Concept* verwendete Technologien vorgestellt sowie deren Zusammenspiel erläutert, um eine vollständige Public-Key-Infrastruktur zu erhalten. Abschnitt 5 greift die vorher aufgezählten optimierbaren Stellen wieder auf und analysiert, ob das neue Konzept diese tatsächlich beseitigt und alle weiteren Anforderungen erfüllt.

2 PROBLEMSTELLUNG

Das iRM nutzt eine eigene *Public-Key- Infrastruktur* (PKI) um Routern zu ermöglichen, sich am Backend des iRM zu authentifizieren. Diese Public-Key-Infrastruktur ist nicht für die Verschlüsselung der Daten und den sicheren Datentransport zuständig, sondern lediglich für die Authentifizierung, sodass nicht autorisierte Geräte erst keine Verbindung zum iRM aufbauen können. Der Aufbau eines TLS-Tunnels zur gesicherten Kommunikation erfolgt vor diesem Schritt an anderer Stelle. Die Prozesse und Gegebenheiten der bestehenden Public-Key-Infrastruktur sind dabei an einigen Stellen noch zu optimieren. Ziel der Arbeit ist es, durch die Konzeption einer neuen Public-Key-Infrastruktur die in Abschnitt 2.2 genannten Schwachstellen auszumerzen, sodass eine Public-Key-Infrastruktur entsteht, welche aktuellen Empfehlungen entspricht [4].

2.1 Analyse der bestehenden Public-Key-Infrastruktur

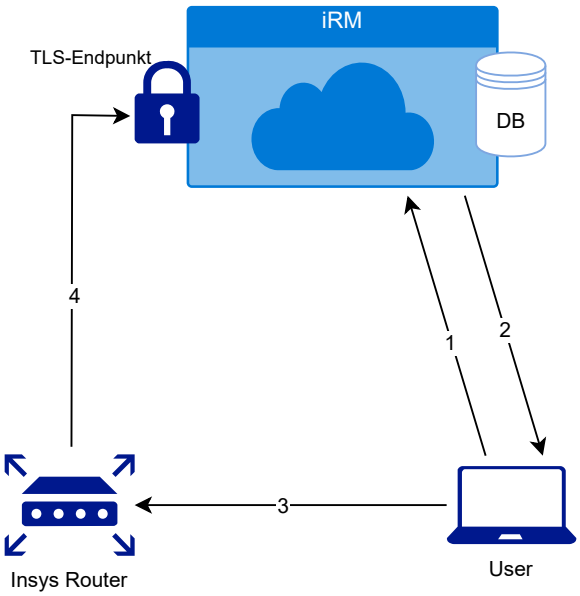
Die zurzeit eingesetzte Public-Key-Infrastruktur ist eine Eigenimplementierung. Diese benutzt gängige

kryptographische Bibliotheken und Pakete um alle nötigen Funktionen zu implementieren.

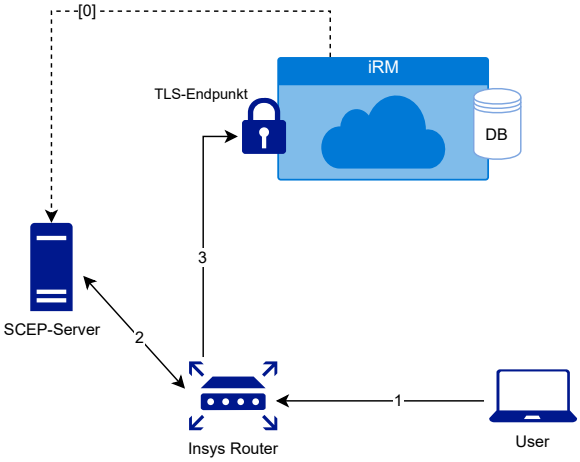
Funktionsweise Es wird für jedes neu angelegte Benutzerkonto im iRM eine eigene, selbst signierte *Root-CA* erstellt. Diese Zertifikatsautorität ist die *Wurzel* der Zertifikatsstruktur und dadurch der *Vertrauensanker*. Vertraut man dieser Autorität, kann man gleichzeitig allen Zertifikaten vertrauen, die diese ausstellt und signiert. Die *Root-CA* ist hier also dafür zuständig, die entsprechenden Client-Zertifikate für die Router auszustellen.

Die Auslieferung der Zertifikate erfolgt über eine sogenannte *Startkonfiguration*, welche beim einrichten eines Routers im iRM erstellt wird und jederzeit abrufbereit ist. Dies ist in Abbildung 1a zu sehen. Diese enthält verschiedenste Einstellungen als auch das Clientzertifikat samt privatem Schlüssel, um den Router mit dem iRM zu verbinden. Der Nutzer lädt diese über eine verschlüsselte Verbindung herunter und spielt die *Startkonfiguration* selber auf den Router ein. Der Router kann sich nun mit dem iRM verbinden und authentifizieren.

Ein weiterer Weg, Zertifikate auszuliefern, besteht darin, die bestehende Implementierung des Simple Certificate Enrollment Protocol (SCEP)-Protokolls im Betriebssystem des Routers zu nutzen, was in Abbildung 1b abgebildet ist. SCEP ist ein standardisiertes Protokoll zur Auslieferung und Erneuerung von Zertifikaten. Mehr dazu kann in Abschnitt 3.2 nachgelesen werden. Die Router können als SCEP-Klienten fungieren, wobei es jedoch zurzeit nötig ist, den SCEP-Server selber bereitzustellen. Dieser benötigt eine Zertifikatsautorität samt privatem Schlüssel um Zertifikate auf Anfrage zu signieren. Wird eine erfolgreiche Zertifikatsanfrage gestellt, wird dem Router ein gültiges Zertifikat zurückgesandt, mit welchem er sich am TLS-Endpunkt authentifizieren kann.



(a) Zertifikatsausbringung über die Startkonfiguration



(b) Zertifikatsausbringung über SCEP

Abbildung 1: Zertifikatsausbringung der PKI des iRM

Zertifikatsstruktur Das System benutzt Standard *X.509-Zertifikate* der Version drei. Diese sind komplexe, definierte Datenstrukturen, welche über bestimmte Notationen und Kodierungen verfügen. Zudem ermöglicht dieser Standard in Version drei den Zertifikaten *Erweiterungen* hinzuzufügen, welche diese mit Informationen anreichern. Wird im Laufe der Arbeit von Zertifikaten gesprochen, ist damit immer ein *X.509v3-Zertifikat* gemeint.

In Abbildung 2 ist die aktuelle Zertifikatsstruktur zu sehen. Bei Erstellung eines Nutzerkontos wird eine eigene selbst signierte *Root-CA* ausgestellt, welche wiederum die Client-Zertifikate ausstellt. Dies bedeutet, dass mehrere voneinander komplett unabhängige Public-Key-Infrastrukturen nebeneinander existieren. Zudem ist die *Root-CA* die signierende Instanz, wenn ein Zertifikat ausgestellt werden soll. Dies sollte grundsätzlich vermieden werden, da bei der Signierung von Dokumenten der private Schlüssel in Benutzung ist und sich dieser daher nicht mehr an einem sicheren Ablageort befindet. Sollte der private Schlüssel der *Root-CA* kompromittiert werden, ist die komplette Public-Key-Infrastruktur nicht mehr sicher.

Ebenfalls wird das Überprüfen eines Zertifikates auf Gültigkeit am TLS-Endpunkt erschwert, da dieser die *Root-CA* jedes Nutzerkontos besitzen muss um das Zertifikat der entsprechenden *Root-CA* zuordnen zu können.

2.2 Schwachstellen der bestehenden Public-Key-Infrastruktur

Im Folgenden werden die Schwachstellen der Public-Key-Infrastruktur beschrieben, welche im Laufe der

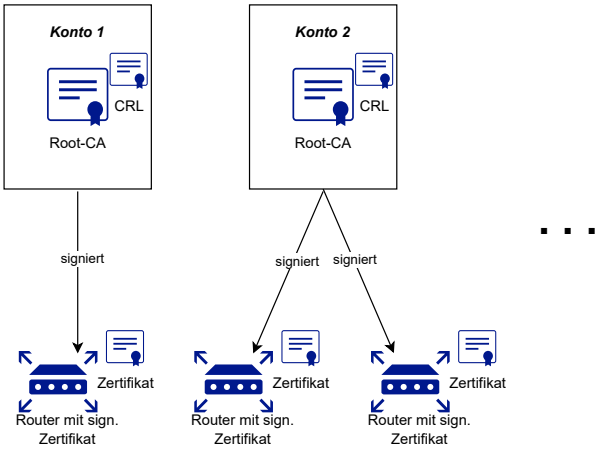


Abbildung 2: Bestehende Zertifikatsstruktur im System

Arbeit durch das angefertigte Konzept eliminiert werden sollen:

Rückdatierung und Laufzeit Router bekommen beim Aufspielen der ersten *Startkonfiguration* ein Zertifikat mitgesendet, welches von 1970 an rückdatiert ist und 10 Jahre ab Datum der Erstellung gültig ist. Somit hat das Zertifikat eine Laufzeit von über 50 Jahren, was für Zertifikate dieser Art sehr ungünstig ist. Diese Eigenheit ist durch die Weiterentwicklung der Produkte nicht mehr von Nöten. Es gilt dennoch ein System zu entwerfen, welches mit solchen möglichen Eigenheiten zurecht kommen würde. Ebenfalls ist Rückdatierung ein seltener Randfall, welcher im besten Fall vermieden werden sollte.

Ausbringung der Zertifikate Wenn die *Startkonfiguration* auf den Router übertragen wird, wird ebenfalls das Zertifikat samt Schlüsselmaterial mitgesendet. Dies führt dazu, dass der private Schlüssel des Zertifikates im Klartext in der *Startkonfiguration* vorhanden ist. Dies gefährdet die Integrität des Schlüsselmaterials, da nun die sichere Handhabung dessen beim Benutzer liegt und nicht mehr bei der eigenen Anwendung.

Rückziehung von Zertifikaten Client-Zertifikate sind in diesem Anwendungsfall relativ flüchtige Komponenten. Jedes mal, wenn ein Router im iRM angelegt wird, wird ebenfalls ein Zertifikat erzeugt. Wird ein Router aus der Managementsoftware entfernt, wird dieses Zertifikat deaktiviert. Dies wird in der Datenbank hinterlegt. Jedoch sollte dies eigentlich in einer Certificate Revocation List (CRL) hinterlegt werden und vielleicht sogar mittels dem Online Certificate Status Protocol (OCSP) abgefragt werden können. Mit OCSP kann die Gültigkeit eines bestimmten Zertifikats direkt abgefragt werden ohne erst die CRL herunterzuladen. In einer CRL werden die Seriennummern von zurückgezogenen Zertifikaten gespeichert, sodass zu einem späteren Zeitpunkt die Gültigkeit von Zertifikaten überprüft werden kann. Der Grundstein für diese Funktionalität wurde gelegt, jedoch ist diese nicht funktionsfähig. Bei diesem Aspekt spielt ebenfalls die lange Laufzeit von Zertifikaten hinein. Werden viele Router angelegt und wieder entfernt, würde die CRL schnell anwachsen, da diese alle Zertifikate enthalten muss, die nach Rückziehung von sich aus noch gültig wären, und die Zertifikate nicht in naher Zeit ablaufen.

Terminierung der Verbindung Ist ein Router gelöscht worden, existiert das Zertifikat immer noch und ist zu jetzigem Stand in keiner abrufbaren CRL hinterlegt. Es gibt also keine Möglichkeit für solch einen Endpunkt zu überprüfen, ob ein Zertifikat gültig ist oder nicht. Erst am Backend des iRM wird die Verbindung abgelehnt, da dieses über die Deaktivierung des Zertifikats Bescheid weiß.

3 KONZEPTION EINER VERBESSERTEN PUBLIC-KEY-INFRASTRUKTUR

Um die Public-Key-Infrastruktur des Router Managements zu verbessern, wird eine neue Zertifikatsstruktur entworfen sowie die Verwaltung dieser weitestgehend auf bestehende und erprobte Protokolle und Softwarelösungen verlagert.

Eine Verbesserung der Public-Key-Infrastruktur anzustreben hat vielfältige Gründe. Zum einen wird der Wartungsaufwand der Anwendung bei Auslagerung von komplexen Public-Key-Infrastruktur-Funktionalitäten gesenkt. Das Signieren und Ausstellen von Zertifikaten sowie das Verwalten von Schlüsselmaterial dieser sind aufwendige und sicherheitskritische Aktionen. Wird dies auf erprobte Softwarelösungen und Protokolle ausgelagert, sinkt die Fehleranfälligkeit und Komplexität des Systems, sodass die Wartung dessen erleichtert wird. Außerdem wird durch ein modernes und schlüssiges Konzept die Sicherheit des iRM und der zugehörigen Komponenten verbessert. Dies erhöht die Attraktivität des Produkts, vor allem für Kunden, bei denen Sicherheit einen hohen Stellenwert hat.

Das entworfene Konzept wird in den nächsten Kapiteln erläutert.

3.1 Zertifikatsstruktur

Die Zertifikatsstruktur ist ein zentraler Teil einer Public-Key-Infrastruktur und bestimmt das Verhalten der einzelnen Komponenten sowie Vorgänge in dieser.

Alle in Abschnitt 2.2 genannten Punkte zeigen auf, dass an der Zertifikatsstruktur Verbesserungsbedarf besteht. Ein Weg dies zu erreichen besteht darin, das

Prinzip des *Hierarchischen Vertrauens* [5] zu nutzen. Dafür wird eine übergeordnete *Root-CA* erstellt, welche Zwischenzertifikate ausstellt; diese ersetzen bei den Benutzerkonten die eigenen *Root-CAs*. Die übergeordnete *Root-CA* übernimmt hier die Rolle des *Vertrauensankers*. Diese Zwischenzertifikate sind von der übergeordneten *Root-CA* signiert und können selber Client-Zertifikate ausstellen. Somit wird die Zertifikatshierarchie um eine weitere Ebene erweitert, wie in Abbildung 3 abgebildet.

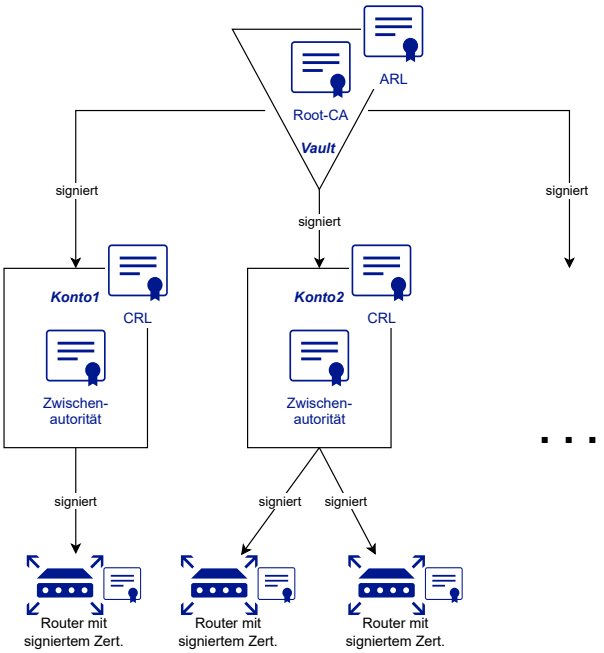


Abbildung 3: Neu angedachte Zertifikatsstruktur

Möchte man nun eine CRL pflegen, kann dies über die jeweiligen Zwischenautoritäten geschehen. Diese stellen und signieren CRLs für jegliche von ihnen ausgestellte Zertifikate bereit. Es wäre zwar möglich, dies über eine indirekte CRL, welche von der obliegenden Zertifikatsautorität ausgestellt wird, zu verwalten, jedoch wird davon abgeraten, da die Sicherheit des Systems darunter leidet. Bei der Signierung einer CRL wird der private Schlüssel der Zertifikatsautorität benötigt. Dadurch ist dieser in Benutzung und somit im Speicher des Systems geladen, wodurch das Risiko der Kompromittierung des Schlüsselmaterials steigt. Zwar bleibt der Verwaltungsaufwand der CRLs im Vergleich zum Bestandssystem ähnlich, jedoch verbessert sich die Sicherheit, da das Schlüsselmaterial der *Root-CA* sicher ruhen kann.

Auch besteht nun die Möglichkeit, eine *Authority Revocation List* (ARL) anzufertigen, sollte eine Zwischenautorität zurückgezogen werden, zum Beispiel bei Löschung oder Kompromittierung eines Nutzeraccounts. Dies ist möglich durch die Delegation der *Root-CA*-Aufgaben an eine ausgewiesene selbst signierte Zertifikatsautorität. Dies erleichtert diesen Randfall zu behandeln.

Soll die Gültigkeit eines Client-Zertifikates ermittelt werden, muss ein valider Zertifizierungspfad ermittelt werden. Solch ein Pfad ist eine endliche Sequenz aus Zertifikaten mit der Eigenschaft, dass das Subjekt der Aussteller des nächsten Zertifikates ist, außer dem Letzten [5]. Der Pfad ist gültig, wenn das Subjekt im letzten Zertifikat das zu prüfende Zertifikat ist und der Aussteller des ersten Zertifikates der Vertrauensanker ist.

3.2 Zertifikatsaussbringung

Damit die ausgestellten Zertifikate auch genutzt werden können, müssen diese über einen sicheren und effizienten Mechanismus beantragt, ausgestellt und zugesandt werden. Solch einen Mechanismus kann SCEP bieten [6]. SCEP ist ein Protokoll zur Ausstellung, Erneuerung und Anfrage von Zertifikaten sowie zur Verteilung von öffentlichen Schlüsseln der Zertifikatsautoritäten. Ebenfalls können zumindest CRL-Anfragen über das Protokoll bedient werden.

SCEP wird bereits von den Routern der INSYS unterstützt, weswegen primär dieses Protokoll betrachtet wird. Zu Abwandlungen des Konzepts wird sich in späteren Kapiteln der Arbeit geäußert.

Das Protokoll sendet und empfängt Nachrichten unverschlüsselt über HTTP, weswegen die Nachrichten in gesicherten, verschlüsselten PKCS-Containern versendet werden. So kann verhindert werden, dass Dritte in-Transit Zugriff auf die Nachrichten haben könnten. Um zu verhindern, dass Zertifikate an nicht autorisierte Geräte ausgestellt werden, arbeitet SCEP mit *Challenge*-Passwörtern. Da diese Schnittstelle über das Internet erreichbar ist, muss diese entsprechend abgesichert werden. Das *Challenge*-Passwort ist ein geteiltes Geheimnis und muss dem Router vor der Zertifikatsanfrage zukommen. Hier eignet sich der Weg über die *Startkonfiguration*, welche zurzeit das gesamte Zertifikat samt privatem Schlüssel enthält. Das Passwort sollte kein konventionelles Passwort sondern ein schwer zu erratendes sein [6]. Im besten Fall ist dieses ein Einmal-Passwort, das nur für eine Zertifikatsanfrage gültig ist und eine begrenzte Lebenszeit besitzt. Die Passwörter an sich müssen natürlich auch sicher abgelegt werden.

Für das Router-Management bietet es sich an, diese Passwörter bei Anlegen des Routers im iRM zu erstellen und für jeden Router an sicherer Stelle abzulegen. Möchte sich nun ein Router ein Zertifikat über SCEP ausstellen lassen, muss dieser das *Challenge*-Passwort mitliefern. Der SCEP-Server kann nun prüfen, ob das Subjekt für die Zertifikatsausstellung autorisiert ist. Wird ein Zertifikat nach Anfrage ausgestellt, wird das zugehörige Passwort ungültig. Sollte ein Router länger mit der Zertifikatsanfrage warten als die gesetzte Lebenszeit des Passworts, wird die Anfrage fehlschlagen.

Zertifikate können aber auch über SCEP erneuert

werden. Dabei sendet der Router eine Erneuerungs-Anfrage, welche mit dem alten Zertifikat signiert wurde, an den SCEP-Server. An dieser Stelle kann ein Zertifikat mit dem bestehenden Schlüsselmaterial oder mit neuem Schlüsselmaterial angefordert werden.

3.3 Eigenschaften der Zertifikate

An den Zertifikaten selbst werden keine großen Änderungen vorgenommen. Diese enthalten weiterhin nötige Felder wie Subjektname, Aussteller, Gültigkeit, öffentlicher Schlüssel und Informationen zu benutzten Verschlüsselungsalgorithmen. Die Zertifikate enthalten ebenfalls Erweiterungen, welche in Version 3 von X.509-Zertifikaten vorhanden sind.

Für Zertifikatsautoritäten sind die Erweiterungen *Basiseinschränkungen*, welche angeben ob das vorliegende Zertifikat eine Zertifikatsautorität sein darf, und *Schlüsselnutzung*, welche die möglichen Anwendungsfälle der Schlüssel des Zertifikats definiert, wichtig. Diese müssen so gesetzt werden, dass vorliegende Zertifikate als Zertifikatsautoritäten fungieren und Zertifikate sowie CRLs signieren können. Die Zertifikatsautoritäten besitzen ebenfalls *Erweiterungen* für den Zugriff auf die CRLs, welche auf den Ablageort dieser verweisen.

Da Zertifikate nun über einen definierten Mechanismus nach Bedarf angefordert sowie auch erneuert werden können, kann die Laufzeit der Zertifikate niedrig gehalten werden. Beruhend auf der Annahme, dass die Router eine längerfristige Verbindung mit dem Router-Management aufbauen und sich trotz zeitweiliger Abschaltung wieder mit dem Router Management verbinden sollen, wäre eine Laufzeit von drei Monaten akzeptabel.

4 REALISIERUNG

Das folgende Kapitel beschäftigt sich mit der Implementation des im vorherigen Kapitel erarbeiteten Konzepts. Die Implementation und Integration des Konzepts zielt in dieser Arbeit auf einen *Proof of Concept* ab, welcher die Möglichkeiten als auch Grenzen des Konzepts im realen Produktivumfeld aufzeigen soll. Dafür werden bestehende Technologien untersucht und ihre Funktionen genutzt, um als Endergebnis eine performante, sichere und flexible Public-Key-Infrastruktur zu erhalten.

4.1 Technologien

Im Folgenden werden die ausgewählten Technologien beschrieben und Gründe für die Auswahl dieser dargelegt.

HashiCorp Vault *HashiCorp Vault* [7] ist eine Anwendung zur sicheren Verwaltung und Ablage von Geheimnissen, welche sehr vielfältig sein können. Diese sind zum Beispiel Passwörter für Datenbanken, Tokens für Authentifizierung oder auch Zertifikate. Vault verwaltet solche Geheimnisse über *Secret Engines*. Diese sind nach Bedarf hinzufügbare Module, die verschiedene Funktionalitäten für Verwaltung und Ausstellung der Geheimnisse bieten. Solch eine Engine ist zum Beispiel die *Public-Key-Infrastruktur-Certificates Engine*, welche ermöglicht, Zertifikatsautoritäten zu erstellen, das Schlüsselmaterial dieser zu verwalten und mithilfe dieser Zertifikate auszustellen. Diese können ebenfalls bei Bedarf zurückgezogen werden. Auch werden Schnittstellen angeboten, welche den Zugriff auf etwaige CRLs und CA-Zertifikate bieten.

Bei der Verwaltung von Geheimnissen setzt HashiCorp auf das Prinzip von kurzlebigen Geheimnissen. Dies bedeutet, dass ausgestelltes Schlüsselmaterial so kurz wie möglich seine Gültigkeit behält. Im Fall dieser Arbeit bedeutet dies, dass Zertifikate nur so lange wie nötig für einen reibungslosen Lebenszyklus dieser gültig sind, also etwa drei Monate. Dabei können die Funktionen der Engine über eine HTTP- oder HTTPS-Schnittstelle aufgerufen werden.

Vault ist aber nicht nur eine gute Softwarelösung zur Bereitstellung einer Public-Key-Infrastruktur, sondern ebenfalls ein Technologiewunsch von Seiten der Produktentwicklung. Dies ist der Fall, da das Router-Management die Fähigkeit bietet, Konfigurationsdaten für Router in Vorlagen und eigenen Feldern zu speichern. Diese können vertrauliche Daten beinhalten und Aufschluss über etwaige Netzwerke geben, was als vertraulich anzusehen ist. Dabei kann *Vault* helfen, diese Daten über eine andere *Secret-Engine* sicher abzulegen. Dieser Mechanismus kann auch genutzt werden, um die Zertifikatsanforderung über SCEP mithilfe von *Challenge*-Passwörtern abzusichern. Mehr dazu in Abschnitt 4.2.

Alle genannten Anwendungsfälle kombiniert, sowie verwendete Algorithmen und Technologien in der Implementierung des *Vault*, ergeben ein großes Potenzial für den Einsatz dessen im bestehenden System der INSYS .

SCEP Wie bereits angemerkt, können Router Zertifikate für das iRM über eine SCEP-Implementation verwalten. Um diese Funktionalität zu nutzen, wird ein SCEP-Server benötigt, welcher kommende Aufrufe nach dem SCEP-Protokoll annehmen und verarbeiten kann. An dieser Stelle wurde sich für den micromdm/scep [8] Server entschieden. Dies hat vielfältige Gründe:

Zum einen ist das vorliegende Paket in der Programmiersprache *Go* implementiert, welche die verwendete Sprache des iRM-Backends ist. Dadurch passt sich dieses Paket gut in das Ökosystem des iRM ein.

Außerdem ist das Paket sehr flexibel was Erweiterbarkeit angeht. Im vorangehenden Punkt wur-

de geschildert, das die Public-Key-Infrastruktur über den *Vault* verwaltet werden soll. SCEP sowie auch die enthaltene Serverimplementation im *micromdm/scep*-Paket beschreiben jedoch das Ausstellen und Erneuern von Zertifikaten als Aufgabe des Servers selbst. In Fall des erarbeiteten Konzepts soll der SCEP-Server als Proxy für etwaige Anfragen an den *Vault* dienen. So müssen Router nicht mit dem *Vault* direkt kommunizieren können, sondern können dafür das standardisierte SCEP-Protokoll nutzen. In der Beschreibung des betrachteten Pakets wird die Möglichkeit, dieses als Proxy für Anfragen zu nutzen, explizit genannt, was auch durch die generisch gehaltene Implementierung bestätigt werden kann.

So kann eine Eigenimplementierung eines SCEP-Servers mit getesteten Funktionen angefertigt werden, welche sogar in das bestehende Backend des iRM integriert werden kann.

Dabei ist jedoch zu beachten, dass SCEP nicht über verschlüsselte Kanäle kommuniziert. Dies bedeutet, dass SCEP sich selbst um Verschlüsselung der Daten in-Transit kümmert. Dafür wird der zu signierende Certificate Signing Request (CSR) mit dem öffentlichen Schlüssel der im Server hinterlegten Zertifikatsautorität verschlüsselt, welche diesen daraufhin bei Erhalt mit Hilfe des privaten Schlüssels entschlüsseln kann. Das signierte Zertifikat kann nun mit dem eigenen öffentlichen Schlüssel wieder verschlüsselt und zurückgeschickt werden. Der SCEP-Server benötigt also stets alle Zwischenautoritäten samt Schlüsselmaterial, unabhängig davon, an welcher Stelle Zertifikate signiert werden.

EST Enrollment over Secure Transport (EST) ist ein Protokoll, welches, ähnlich wie SCEP, das Ausstellen von Zertifikaten auf Anfrage sowie das Bereitstellen von Zertifikatsautoritäten ermöglicht [9].

Der größte Unterschied im Vergleich zu SCEP besteht im Mechanismus des sicheren Transports der Daten. Während SCEP sich selbst um die Verschlüsselung kümmert, nutzt EST den bereits vorhandenen TLS-Mechanismus, um die Daten verschlüsselt zu transportieren. Ebenfalls ist die Generierung von Schlüsselmaterial auf Klientenseite als auch auf Serverseite im Standard spezifiziert. Zudem ist im Standard von EST die Möglichkeit von Erweiterung des Protokolls nicht nur möglich, sondern auch vorhergesehen. Dies bietet mehr Freiheit bei der Gestaltung des Ausstellungsprozesses.

Außerdem bietet EST die Wahlmöglichkeit von Chiffretypen, wie zum Beispiel *Eliptic Curve Cryptography*.

EST ist jedoch noch nicht in der Firmware der Router implementiert.

4.2 Funktionsweise

Da nun etwaige Technologien und Gründe für deren Benutzung bekannt sind, können diese im Folgenden Kapitel miteinander kombiniert und deren Zusammenspiel erläutert werden, um im Anschluss eine vollwertige Public-Key-Infrastruktur zu erhalten. Dieses ist in folgender Grafik veranschaulicht, welche im Folgenden in Reihenfolge der angemerkten Schritte erläutert wird.

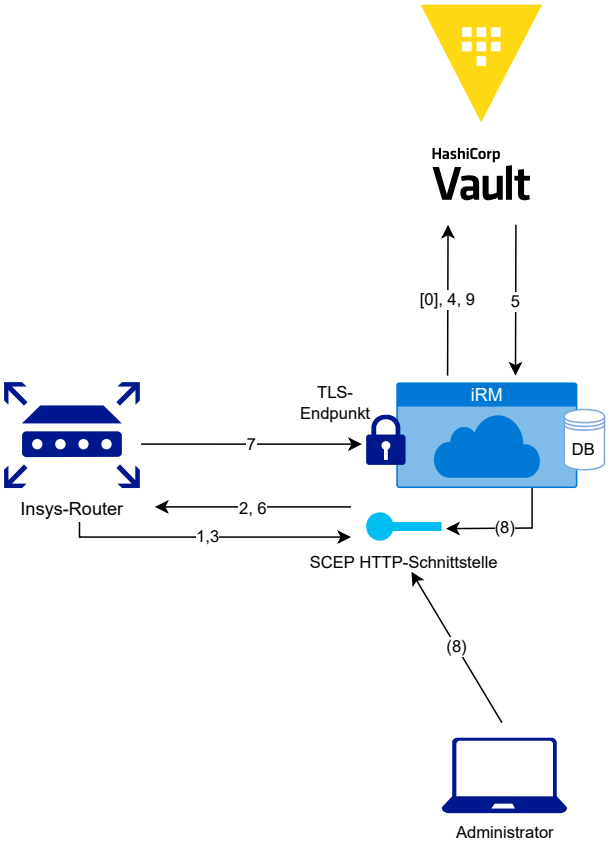


Abbildung 4: Abläufe in der PKI

In Abbildung 4 ist das Aufsetzen der Public-Key-Infrastruktur und der Ablauf bei Zertifikatsanforderung sowie der optionalen Zertifikatsrückziehung enthalten.

Begonnen wird mit dem Aufsetzen und Vorbereiten des *Vaults* für das Ausstellen und Verwalten von Zertifikaten, angemerkt durch Schritt null in eckigen Klammern, da dies nicht direkt zum Lebenszyklus der Zertifikate gehört. Das Vorbereiten des *Vaults* kann in zwei Arten aufgeteilt werden:

Beim ersten Start des Komplettsystems wird dieser hochgefahren und eine selbst signierte Root-CA ausgestellt. Diese bleibt im besten Fall bis Laufzeitende bestehen. Ebenfalls werden zugehörige Rollen und Konfigurationen erstellt.

Läuft das System, ist es bereit Nutzereingaben zu verwalten. Erstellt ein Nutzer nun einen Account, erzeugt das Backend eine Zwischenautorität, welche später genutzt wird, die Client-Zertifikate zu signieren. Diese Art ist in Abbildung 4 in Schritt null dargestellt.

Ist das System aufgesetzt, können Zertifikatsanfragen gestellt werden. In Schritt eins fragt der Router das Zertifikat der Zwischenautorität an, welches für das Generieren und Verschlüsseln des CSRs benötigt wird. Schritt zwei ist die zugehörige Antwort, also das Zertifikat der Autorität. Soll tatsächlich das richtige Zertifikat geliefert werden, ist es nötig, sich einen Mechanismus zu überlegen, welcher die Auswahl der richtigen, zum Nutzerkonto gehörigen, Zwischenautorität ermöglicht, da SCEP standardmäßig lediglich ein bestimmtes hinterlegtes Zertifikat zurückgibt.

Schritt drei beschreibt die eigentliche Zertifikatsanfrage. Dabei generiert der Router das Schlüsselmaterial für das Zertifikat und packt dieses in einen CSR, welcher verschlüsselt an den SCEP-Server gesendet wird. In Schritt vier reicht die Schnittstelle die Anfrage an den *Vault* weiter, welcher letztendlich die Anfrage erhält, das gewünschte Zertifikat über einen bestimmten Endpunkt signiert und dieses in Schritt fünf zurück an den SCEP-Server sendet. Dieser sendet wiederum in Schritt sechs das aufbereitete Ergebnis der Anfrage, verschlüsselt mit dem öffentlichen Schlüssel, an den Router weiter, welcher daraufhin alle nötigen Daten besitzt, um eine Verbindung mit dem Router-Management herzustellen.

Möchte man das Schlüsselmaterial serverseitig generieren lassen, wenn zum Beispiel das Generieren von Schlüsseln zu aufwendig für den Klienten ist, bietet die Spezifikation von SCEP jedoch keine Mechanismen. Zwar gibt es SCEP-Server-Implementationen, die diese Funktionalität bieten, jedoch ist diese nicht im Standard spezifiziert. An dieser Stelle ist der Einsatz des EST-Protokolls empfohlen, welches die Verschlüsselung der Daten an den TLS-Mechanismus abgibt, sofern alle Parteien diesen unterstützen.

Das Aufbauen der Verbindung passiert in Schritt sieben, wobei der Router sich an der TLS-Terminierung mit dem erhaltenen Zertifikat authentifiziert. Die TLS-Terminierung besitzt alle nötigen übergeordneten Zertifikatsautoritäten sowie Zugriff auf alle CRLs, wodurch diese in der Lage ist, die Gültigkeit des Zertifikats zu prüfen. Erst bei erfolgreicher Prüfung des Zertifikats kann der Router mit dem iRM kommunizieren.

Ist es zu einem späteren Zeitpunkt erforderlich, ausgestellte Client-Zertifikate zurückzuziehen, sollte dies über einen definierten Weg machbar sein. In Schritt acht wenden sich die Anwendung oder ein Administrator mit der Seriennummer des zu annullierenden Zertifikats an eine passende Schnittstelle. Soll diese die SCEP-Schnittstelle sein muss, dieser Endpunkt selber implementiert werden, da das SCEP-Protokoll keinen Mechanismus definiert, um Zertifikate zurückzuziehen, der *Vault* jedoch schon. Da Rückziehung nur von der Anwendung selber, zum Beispiel bei Löschung eines Routers, oder von Administratoren durchgeführt wird, kann der *Vault* auch direkt mit einem passenden Token angesprochen werden. Schritt neun ist hier entweder die direkte Anfrage an den *Vault* für Rückziehung oder eine weitergeleitete Anfrage. Zurückgezogene Router sind nun in der vom *Vault* ausgestellten und bereitgestellten CRL aufzufinden.

4.3 Umsetzung

Damit gezeigt werden kann, dass das Konzept auch in der Realität umsetzbar ist, wurde ein *Proof of Concept* angefertigt. Dieser soll keine komplette Implementation darbieten, sondern zeigen, dass die einzelnen Schritte der automatisierten Zertifikatsausbringung tatsächlich implementierbar sind. Auch sollte sich nicht auf die Wahl eines bestimmten Protokolls zur Ausbringung der Zertifikate versteift werden.

Der *Proof-of-Concept* basiert auf *Bash*-Skripten. Hier wird größtenteils mit *Curl*-Befehlen gearbeitet, welche API-Aufrufe absetzen. Folgende Aufgaben sollen automatisiert werden:

Aufsetzen des Vault Der *Vault* muss zu Beginn konfiguriert und eingerichtet werden. In diesem Fall besteht dies daraus, die selbstsignierte *Root-CA* zu erstellen und zwei Public-Key-Infrastruktur-*Mounts* zu erzeugen. Einer fungiert als Ablageort für Zertifikatsautoritäten, der andere für Zertifikate. Weiterhin müssen noch einige URLs gesetzt werden und Rollen erstellt werden.

Erzeugen von Zertifikaten Einige Zertifikate müssen zur Laufzeit erzeugt werden, darunter fallen die Zwischenautoritäten und Client-Zertifikate. Hier wurden Skripte angelegt, die die minimal nötigen Daten sammeln und damit die geforderten Zertifikate anfordern.

Ausbringen von Zertifikaten Erstellte Zertifikate müssen ebenfalls zur anfragenden Instanz gesendet werden. Dabei ist ein Server, der ein passendes Protokoll wie SCEP implementiert und ebenfalls mit dem *Vault* kommunizieren kann, gefordert. Der Server ist somit das Bindeglied zwischen Public-Key-Infrastruktur und Klient.

Konfiguration der TLS-Terminierung Damit die TLS-Terminierung prüfen kann, ob ein Zertifikat gültig ist, muss diese alle nötigen Zertifikate, wie zum Beispiel die *Root-CA* und CRLs, erhalten können. Diese sollen ebenfalls an richtiger Stelle platziert werden.

5 ANALYSE DER NEUEN PUBLIC-KEY-INFRASTRUKTUR

Im Folgenden wird untersucht, ob die in Abschnitt 2 aufgezeigten Ungereimtheiten der bestehenden Public-Key-Infrastruktur durch das neue Vorgehen beglichen werden.

Rückdatierung und Laufzeit In der bestehenden Public-Key-Infrastruktur wurden Zertifikate rückdatiert und mit einer sehr hohen Laufzeit versehen. Im neuen Konzept können Zertifikate dynamisch nach Bedarf generiert und erneuert werden. So kann die Laufzeit beliebig klein gehalten werden. Diese kann zum Beispiel drei Monate betragen, welche jedoch in einem produktiven Test nachzuprüfen ist und gegebenenfalls optimiert werden kann.

Ausbringung der Zertifikate An dieser Stelle ist wieder die dynamische Generierung von Zertifikaten wichtig. In der bestehenden Implementierung werden generierte Zertifikate im Klartext mit der *Startkonfiguration* übertragen und dauerhaft abgelegt. Nun muss lediglich ein *Challenge*-Passwort mit beschränkter Laufzeit übertragen werden, damit ein Router autorisiert wird, sich ein Zertifikat signieren zu lassen.

Rückziehung von Zertifikaten Im bestehenden System ist das Zurückziehen von Zertifikaten nicht möglich. Durch Kombination von niedriger Laufzeit und erprobten Softwarelösungen können über den *Vault* die möglichst minimalen CRLs der jeweiligen Autoritäten bereitgestellt werden.

Terminierung der Verbindung Da nun, anders als im bestehenden System, CRLs an zentraler Stelle abrufbar sind, können diese auch an einem TLS-Endpunkt angefordert werden. Somit können ungültige Verbindungen terminiert werden, bevor sie am System eingehen.

Weitere Anforderungen Zusätzlich zur Optimierung der grundsätzlichen Abläufe sollte die Public-Key-Infrastruktur auch weitere Eigenschaften wie Flexibilität bei Chiffren und Schlüssellänge, aber auch bestimmte implementationstechnische Eigenschaften, vorweisen.

Eine Anforderung ist die freie Wahl des Ortes der Schlüsselmaterialgenerierung. Da das System Zertifikate mit CSRs anfragt, kann entweder Schlüsselmaterial mitgesendet werden oder diese Aufgabe an den *Vault* delegiert werden, sofern EST zum Einsatz kommt.

Zudem sollte die Ablage von Zertifikaten und Schlüsselmaterial sicher gestaltet werden. Um diesen Punkt kümmert sich der *Vault* selber, da dieser Paradigmen implementiert, welche wenig bis kaum Möglichkeiten geben, an dieses Heranzukommen. Zusätzlich sollte die Public-Key-Infrastruktur so konzipiert sein, dass kein Zertifikat oder Schlüsselmaterial tatsächlich den *Vault* verlassen muss, um ein Zertifikat auszustellen. Hier sollte wieder EST in Betracht gezogen werden, um diese Anforderung vollends zu erfüllen.

Außerdem sollte die Public-Key-Infrastruktur im Kontext einer Cloudnative-Anwendung skalierbar sein, also zum Beispiel in Kubernetes ausgebracht werden können. *HashiCorp Vault* bietet diese Funktionalität. Da der Server für Zertifikatsanfragen in die Anwendung integriert werden soll, benötigt dieser kein weiteres Vorgehen für Skalierbarkeit.

6 ZUSAMMENFASSUNG

In diesem Paper wurde die bestehende Public-Key-Infrastruktur für TLS-Authentifizierung von Routern am *INSYS icom* Router Management beschrieben und einige optimierbare Stellen aufgezeigt (Abschnitt 2). Daraufhin wurde ein Konzept entworfen, welches diese ausmerzen soll sowie flexibel und einfach in der Handhabung ist (Abschnitt 3). Im Darauffolgenden Abschnitt wurden Technologien und deren Zusammenspiel erläutert, welche solch ein System in der Realität implementieren können (Abschnitt 4). Schlussendlich werden die vorher genannten optimierbaren Stellen erneut aufgenommen und gegen das Lösungskonzept geprüft, um zu erschließen, ob diese verbessert und weitere Anforderungen erfüllt wurden (Abschnitt 5).

In diesem Paper zeigt sich, wie durch das geschickte Zusammenfügen von Technologien eine flexible und sichere Public-Key-Infrastruktur in ein Bestandssystem implementiert werden kann.

LITERATUR

[1] A. Koohang, C. S. Sargent, J. H. Nord und J. Paliszkiewicz, „Internet of things (IoT): From awareness to continued use," *International Journal of Information Management*, Jg. 62, S. 102 442, 1. Feb. 2022, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2021.102442. Adresse: https://www.sciencedirect.com/science/article/pii/S0268401221001353 (besucht am 28. 01. 2023).

[2] „Insys-icom Homepage," INSYS icom. (), Adresse: https://www.insys-icom.com/ (besucht am 03. 01. 2023).

[3] „Was sind Kritische Infrastrukturen?" Bundesamt für Sicherheit in der Informationstechnik. (), Adresse: https://www.bsi.bund.de/DE/Themen/KRITIS-und-regulierte-Unternehmen/Kritische-Infrastrukturen/Allgemeine-Infos-zu-KRITIS/allgemeine-

infos-zu-kritis.html?nn=126640 (besucht am 03. 01. 2023).

[4] „BSI TR-02102 Kryptographische Verfahren: Empfehlungen und Schlüssellängen," Bundesamt für Sicherheit in der Informationstechnik. (), Adresse: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr-02102.html?nn=433400 (besucht am 28. 01. 2023).

[5] J. A. Buchmann, E. Karatsiolis und A. Wiesmaier, *Introduction to Public Key Infrastructures*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN: 978-3-642-40656-0 978-3-642-40657-7. DOI: 10.1007/978-3-642-40657-7. Adresse: http://link.springer.com/10.1007/978-3-642-40657-7 (besucht am 04. 10. 2022).

[6] P. Gutmann, „Simple Certificate Enrolment Protocol," Internet Engineering Task Force, Request for Comments RFC 8894, Sep. 2020, Num Pages: 42. DOI: 10.17487/RFC8894. Adresse: https://datatracker.ietf.org/doc/rfc8894 (besucht am 11. 10. 2022).

[7] „Vault by HashiCorp," Vault by HashiCorp. (), Adresse: https://www.vaultproject.io/ (besucht am 10. 01. 2023).

[8] *scep*, original-date: 2016-05-29T20:53:09Z, 8. Jan. 2023. Adresse: https://github.com/micromdm/scep (besucht am 10. 01. 2023).

[9] M. Pritikin, P. E. Yee und D. Harkins, „Enrollment over Secure Transport," Internet Engineering Task Force, Request for Comments RFC 7030, Okt. 2013, Num Pages: 53. DOI: 10.17487/RFC7030. Adresse: https://datatracker.ietf.org/doc/rfc7030 (besucht am 06. 12. 2022).

AUTOREN

Christopher Neldner ist Master of Science des Studiengangs Software Engineering.

Prof. Dr. Frank Herrmann wurde in Münster geboren und studierte Informatik an der Rheinisch-Westfälischen Technischen Hochschule in Aachen. Nach seinem Diplom 1989 arbeitete er bei dem Fraunhofer Institut IITB in Karlsruhe. Während dieser Zeit promovierte er 1996 über Ressourcenbelegungsplanungsprobleme. Von 1996 bis 2003 arbeitete er für die SAP AG in verschiedenen Funktionen, zuletzt als Direktor. Im Jahr 2003 wurde er Professor für Produktionslogistik an der Ostbayerischen Technischen Hochschule in Regensburg. Seine Forschungsthemen sind Planungsalgorithmen, Optimierung und Simulation für die operative Produktionsplanung und -steuerung. Er ist Leiter des Innovations- und Kompetenzzentrums für Produktionslogistik und Fabrikplanung (IPF).