EBC | Institute for Energy Efficient Buildings and Indoor Climate
E.ON Energy Research Center
RWTH AACHEN UNIVERSITY

RWTH-EBC 2021-003

Best Practices zur Gateway-Entwicklung für Internet of Things - Anwendungen in der Gebäudeautomation

[a]Sebastian Blechmann, [b]Kai Jansen, [c]Bernd Scheuffele, [a]Markus Schraven, [a]Alexander Kümpel, [b]Mario Pietersz, [a]Dirk Müller

[a] Lehrstuhl für Gebäude- und Raumklimatechnik, E.ON Energieforschungszentrum, RWTH Aachen
[b] PHYSEC GmbH
[c] Automation-One GmbH

Please cite this article as:

S. Blechmann, et. al., *Best Practices zur Gateway-Entwicklung für Internet of Things - Anwendungen in der Gebäudeautomation*, White Paper
RWTH-EBC 2021-003, Aachen, 2021, DOI: 10.18154/RWTH-2021-04853

RWTH Aachen University
E.ON Energy Research Center
Institute for Energy Efficient Buildings and Indoor Climate (EBC)
Mathieustr. 10, 52074 Aachen

T +49 241 80-49760, F +49 241 80-49769
ebc-office@eonerc.rwth-aachen.de, www.eonerc.rwth-aachen.de/ebc

Best Practices zur Gateway-Entwicklung für Internet of Things – Anwendungen in der Gebäudeautomation

Sebastian Blechmann*[a], Kai Jansen[b], Bernd Scheuffele[c], Markus Schraven[a], Alexander Kümpel[a], Mario Pietersz[b], Dirk Müller[a]

* sebastian.blechmann@eonerc.rwth-aachen.de

[a] Lehrstuhl für Gebäude- und Raumklimatechnik, E.ON Energieforschungszentrum, RWTH Aachen

[b] PHYSEC GmbH

[c] Automation-One GmbH

Aachen, 01.05.2021

Kurzzusammenfassung

Dieses Whitepaper ergänzt die Ausführungen des ersten Whitepapers, welches ebenfalls im Rahmen des Forschungsprojektes NextGenBAT entstanden ist, und widmet sich dediziert den sicherheitstechnischen Aspekten in der Gateway-Entwicklung für IoT-Anwendungen. Es wurden Best Practices formuliert, um einen - nach aktuellem Stand - sicheren Betrieb des Gateways zu gewährleisten und somit einen einfacheren Einstieg in die nächste Generation der Gebäudeautomation zu ermöglichen.

I. Einleitung

In der Gebäudeautomation müssen für die Kommunikation über das Internet of Things (IoT) verschiedene Aspekte und Akteure in die Planung einbezogen werden. In diesem Whitepaper werden sicherheitstechnische Aspekte zur *Gateway-Entwicklung* beleuchtet und Best Practices formuliert, die Hilfestellung bieten sollen, damit ein sicherer Betrieb gewährleistet werden kann. Die Ausführungen bauen auf dem ersten Whitepaper [1] auf, welches im Rahmen des Projektes „Next Generation Building Automation Technology (NextGenBAT)“ [2] veröffentlicht wurde.

Der Begriff des *Gateways* wird wie folgt benutzt: Das Gateway bildet die Schnittstelle zwischen den daran angeschlossenen Feldgeräten (Sensoren und Aktoren der Gebäudetechnik) und der Cloud bzw. den auf Cloud-Servern laufenden Diensten. Das Gateway sowie mögliche IP-fähige Sensoren und Aktoren sind netzwerktopologisch hinter einer eigenen, korrekt konfigurierten Firewall platziert.

Die vorgestellten Best Practices basieren auf den Ergebnissen der Sicherheitsanalyse eines exemplarischen Gateways. Das betrachtete Gateway ist zuvor im Rahmen des Projektes NextGenBAT entwickelt und in einem Versuchsstand in Betrieb genommen worden. Für die Software des Gateways wurde auf Open-Source-Komponenten zurückgegriffen. Die identifizierten Risiken und Schwachstellen sollen durch die Anwendung von Best Practices vermieden oder minimiert werden.

Zunächst wird in Kapitel II die notwendige abgesicherte Kommunikation zwischen Gateway und Cloud-Server(n) erläutert, gefolgt von der Authentisierung des Gateways und der Autorisierung am Beispiel der Kommunikation über das Message Queuing Telemetry Transport (MQTT) - Protokoll gegenüber den Cloud-Diensten in Kapiteln III und IV. In Kapitel V wird beschrieben, wie Nutzer sich am Gateway einloggen können und was bei der Rechtevergabe zu beachten ist. Danach wird in Kapitel VI auf die physische Härtung, z. B. die räumliche Absicherung, des Gateways eingegangen, bevor in Kapitel VII die Absicherung auf Netzwerkebene sowie Firewalls beleuchtet werden. Im Anschluss werden in Kapitel VIII die Verwendung eines aktuellen Virenschutzes und in Kapitel IX Eigenschaften und Verwendung eines Secure Elements beschrieben. Abschließend werden die Notwendigkeit und die sichere Durchführung von (Sicherheits-) Updates in IoT-Anwendungen in Kapitel X vorgestellt.

II. Sichere Kommunikation durch TLS

In diesem Kapitel wird die Absicherung der Kommunikation zwischen einem Server und einem Client, also z. B. zwischen Cloud-Server und Gateway, beschrieben.

Die Kommunikation zwischen Gateway und Cloud (s. Abbildung 1) muss mit kryptographischen Protokollen abgesichert werden, um die Übertragung von Daten zwischen Server und Client zu schützen und Man-In-The-Middle-Angriffe zu verhindern, damit Vertraulichkeit, Integrität und Authentizität gewährleistet werden. Laut der Technischen Richtlinie TR-02102-2 [3] ist die Absicherung des Transportweges durch das Transport Layer Security (TLS)-Protokoll die beste Option, um Daten aus der Anwendungsschicht verschlüsselt über ein Netzwerk, z. B. über das Internet, zu übertragen. Die Richtlinie empfiehlt außerdem die Nutzung der TLS-Versionen 1.2 und 1.3, allerdings ist TLS 1.3 zu bevorzugen, weil es nicht nur erhöhte Sicherheit, sondern auch schnellere Verbindungen ermöglicht und weniger Ressourcen benötigt. Außerdem gelten nicht mehr alle verfügbaren Cipher Suites als sicher. Wir empfehlen daher die in der folgenden Zusammenfassung aufgeführten Suites.

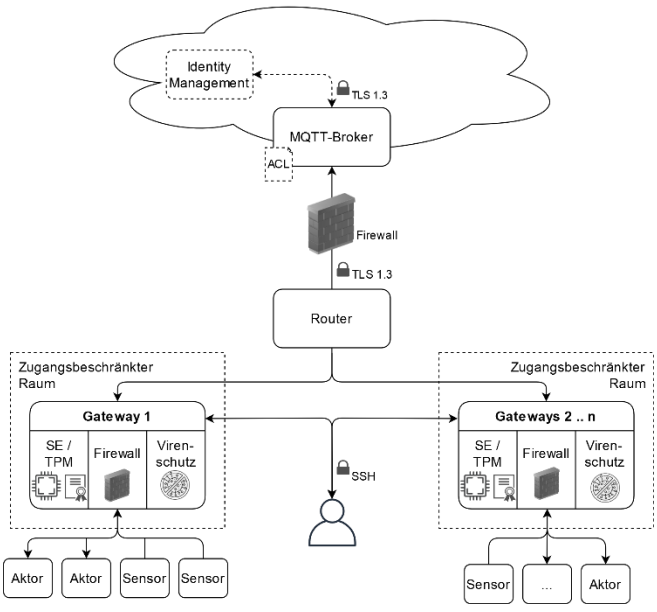

Abbildung 1: Exemplarischer Überblick der Kommunikation in der betrachteten Systemstruktur

Best Practices:

- Server und Client sollten TLS 1.3 unterstützen.
- Falls Server und Client TLS 1.3 unterstützen, sollte ausschließlich TLS 1.3 verwendet und alle anderen Versionen verboten werden.

- Falls eine Partei TLS 1.3 nicht unterstützt, muss auf TLS 1.2 zurückgegriffen werden.
- TLS 1.1 und ältere Versionen dürfen nicht verwendet werden und müssen explizit verboten sein.
- Fallback-Angriffe auf niedrigere TLS-Versionen müssen verhindert werden.
- Für TLS 1.2 sollten nur folgende Cipher Suites erlaubt sein:
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Für TLS 1.3 sollten nur folgende Cipher Suites erlaubt sein:
    - TLS_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
- Folgende elliptische Kurven sollten unterstützt werden:
    - Curve25519
    - P-256 (alias secp256r1)
    - P-384 (alias secp384r1)

III. Authentisierung des Gateways Richtung Cloud

In diesem Kapitel wird der Aspekt der Authentisierung des Gateways in Richtung Cloud betrachtet.

Während des Authentisierungsprozesses muss sich das Gateway gegenüber dem jeweiligen Cloud-Dienst authentisieren. Das kann beispielsweise durch die simple Verwendung einer Kombination aus Username und Passwort, durch die Verwendung von Client-Zertifikaten oder eine Kombination aus beidem erfolgen.

Auf der Transportebene ist die Kommunikation durch Verwendung des TLS-Protokolls abgesichert. Auf der Applikationsebene verwendet das betrachtete Gateway das vielfach in IoT-Anwendungen genutzte, schlanke MQTT-Protokoll zum Datenaustausch mit der Cloud [4]–[7].

Bei der Verwendung von Passwörtern müssen immer einige Aspekte beachtet werden, unabhängig davon, ob reine MQTT- oder Single-Sign-On-Zugangsdaten verwendet werden. Passwörter sollten stets mindestens 16 Bytes, besser 32 Bytes lang sein und eine zufällige Kombination aus großen und kleinen Buchstaben, Zahlen und Sonderzeichen aufweisen. Nutzer und deren zugewiesenen Rechte (vgl. Kapitel IV), also verschiedene Gateways, können sich entweder über Access Control Lists (ACLs) oder ein externes Identity Management bzw. einen Identity Provider am MQTT-Broker authentisieren. Einen stärkeren Schutz bietet die Authentisierung durch Client-Zertifikate während des Aufbaus einer TLS-Verbindung auf Transportebene. Dies erfordert ein eigenes Client-Zertifikat des Gateways, zusätzlich zu den für eine TLS-Verbindung immer nötigen Server-Zertifikaten. Um diese Authentisierung umzusetzen, wird eine dedizierte Public-Key-Infrastructure (PKI) benötigt. Das Aufsetzen und Betreiben einer PKI ist jedoch aufwendig und erfordert Fachkenntnisse, da Fehlkonfigurationen schnell zu schwerwiegenden Sicherheitslücken führen können.

Beide Ansätze zur Authentisierung, also Benutzername und Passwort über MQTT und Client-Zertifikate über TLS, lassen sich auch zu einer Zweifaktor-Authentisierung kombinieren. In diesem Fall muss sich der Client sowohl durch Wissen (Passwort) als auch Besitz (Zertifikat) authentisieren. Dies steigert die Sicherheit des Authentisierungsprozesses. Die Anzahl an erlaubten fehlgeschlagenen Login-Versuchen sollte durch geeignete „rate limiting“ Verfahren eingeschränkt werden, um Brute-Force-Angriffe auf den Authentifikationsprozess abzuschwächen.

Passwörter und andere sensible Informationen sollten niemals statisch im Quellcode hinterlegt werden. Besser ist das Ablegen dieser Informationen in separaten Dateien, wobei auch diese Lösung keinen vollständigen Schutz bietet. Am besten werden Passwörter, sensible Informationen sowie mögliche Client-Zertifikate an besonders geschützten Orten, beispielsweise einem Trusted Platform Module (TPM) (s. Kapitel IX) gespeichert.

Best Practices:

- Die Verwendung von Client-Zertifikaten zur Client-Authentisierung wird empfohlen. Die Nutzung ist jedoch aufwendig und erfordert eine dedizierte PKI. Diese ist wiederum mit einem erhöhten Aufwand und nötigen Fachkenntnissen verbunden, da sonst schwerwiegende Sicherheitslücken entstehen können.
- Falls Passwörter statt Zertifikaten zur Authentisierung verwendet werden, muss jedes Gateway einen eigenen Benutzernamen und ein einzigartiges Passwort besitzen. Zugangsdaten dürfen niemals mit mehreren Entitäten geteilt oder bei verschiedenen Anwendungen hinterlegt werden.

- Ein vom Gateway verwendetes Passwort muss zufällig aus einer sicheren Zufallsquelle generiert werden, zum Beispiel in linux-basierten Betriebssystemen /dev/urandom bzw. /dev/random.
- Ein vom Gateway verwendetes Passwort sollte idealerweise 32 Bytes lang sein.
- Zugangsdaten dürfen niemals statisch im Quellcode hinterlegt werden.
- Idealerweise sollten Passwörter an besonders geschützten Orten, beispielsweise einem TPM, abgespeichert werden.

IV. Autorisierung des Gateways an der Cloud

In diesem Kapitel werden die Möglichkeiten und Schwierigkeiten der Autorisierung im MQTT-Protokoll erläutert.

Wenn sich der Client beim Server erfolgreich authentisiert und der Server die Identität des Clients verifiziert hat, findet die Autorisierung des Clients statt. Dieser Prozess prüft die Rechte des jeweiligen Clients und sichert diese zu.

Die Autorisierung innerhalb von MQTT stellt einige Probleme dar, wie Jia et. al. [7] bereits ausgearbeitet haben. Ein Angreifer kann z. B. einen Hijacking-Angriff auf die MQTT Client ID durchführen, indem sich der Angreifer mit legitimen Zugangsdaten authentifiziert, jedoch die Client ID eines anderen Clients verwendet. Dies muss vom MQTT-Broker durch die Regel unterbunden werden, dass die Client ID dem Benutzernamen entsprechen muss.

Best Practices:

- Ankommende Nachrichten müssen auf Korrektheit und Sinnhaftigkeit überprüfbar sein.
- Sollte die Verifikation fehlschlagen, müssen Nachrichten verworfen werden.
- Die MQTT Client ID muss gleich dem Benutzernamen sein. Dies muss durch den Broker überprüft werden.

V. Authentisierung des Nutzers am Gateway

Zur Wartung des Gateways muss eine sichere Verbindung zum Gateway möglich sein. Die Verfahren zur Anmeldung am Gateway werden in diesem Kapitel näher betrachtet.

Eine Authentisierung mittels Passwort birgt gewisse Risiken, insbesondere wenn Passwörter von Menschen anstatt von Maschinen generiert werden. Auf der einen Seite müssen Menschen in der Lage sein, sich Passwörter zu merken, was einen starken Einfluss auf deren Länge und Zusammensetzung hat, und auf der anderen Seite müssen Passwörter kryptographisch stark genug sein um ausreichenden Schutz zu bieten. Deshalb muss bei der Wahl eines Passwortes durch Menschen besonders darauf geachtet werden, ein in Relation zum Use Case sicheres Passwort zu wählen.

Ein sicheres Passwort besteht aus einer zufälligen Kombination aus großen und kleinen Buchstaben, Zahlen und Sonderzeichen. Dabei sollte das Passwort mindestens 16 Bytes, besser 32 Bytes lang sein. Bei der Festlegung der Passwörter sollten diese Kriterien geprüft werden. Als Hilfestellung können Passwort-Manager verwendet werden, damit Nutzer sich die Passwörter nicht selber merken müssen. In diesem Fall muss ein besonders starkes Master-Passwort gewählt werden und der Speicherort der Passwortdateien muss gesichert sein.

Außerdem können je nach Use Case bzw. je nach Zugriffsrechten einzelner Nutzer weitere Maßnahmen zur Wahl sicherer Passwörter eingesetzt werden, sofern die Passwörter vom Nutzer selbst gesetzt werden: Es sollten Hinweise auf Mindestlänge und Zusammensetzung der Passwörter angegeben werden. Die kryptographische Stärke des Passworts kann während des Eintippens durch einen Passwort-Stärke-Messer visualisiert und nur Passwörter im bspw. grünen Bereich akzeptiert werden. Vor der Festlegung des gewünschten Passworts sollte überprüft werden, ob sich das gewünschte Passwort bereits in einer öffentlichen Passwort-Datenbank befindet.

Jeder menschliche Nutzer einer Anwendung muss einen eigenen Benutzernamen sowie ein einzigartiges Passwort verwenden. Dieses Passwort muss geheim bleiben und darf mit niemandem geteilt werden. Darüber hinaus darf jedes Benutzerkonto nur die Rechte besitzen, die für die Ausübung der Tätigkeiten der jeweiligen Person notwendig sind. Der Login als Root-Benutzer muss verboten werden bzw. darf maximal nur über eine lokale Schnittstelle erfolgen. Nach erfolgreichem Login, auch remote, darf dann auf den Root-Benutzer gewechselt oder Anwendungen über diesen gesteuert werden. Für lokale Logins oder über SSH sollten zusätzlich RSA-Token bzw. SSH-Schlüssel mit einer Mindestschlüssellänge von 3072 Bits für eine Zweifaktor-Authentisierung verwendet werden.

Die Anzahl erlaubter fehlgeschlagener Login-Versuche sollte ebenfalls durch geeignete „rate limiting“

Verfahren eingeschränkt werden, um Brute-Force-Angriffe auf den Authentifikationsprozess am Gateway abzuschwächen.

Best Practices:

- Jeder Nutzer muss einen eigenen Benutzernamen und ein einzigartiges Passwort haben. Zugangsdaten dürfen niemals von mehreren Entitäten geteilt oder bei verschiedenen Anwendungen hinterlegt werden.
- Im besten Fall sollten auch für Personen zufällige Passwörter verwendet werden. Diese können vom Nutzer durch Passwort-Manager verwaltet werden.
- Falls der Nutzer ein eigenes Passwort wählen darf, sollten zusätzliche Maßnahmen ergriffen werden, um den Nutzer dazu zu bringen ein möglichst starkes Passwort zu wählen:
    - Anzeige der Passwortstärke durch einen Passwort-Stärke-Messer.
    - Erzwingen einer Mindestlänge.
    - Erlauben von möglichst vielen Zeichen (Klein-, Großbuchstaben, Zahlen, Sonderzeichen).
    - Überprüfung des gewählten Passwortes auf Einträge in öffentlichen Passwort-Datenbanken.
- Bei lokalem Login sollte ein RSA-Token als Zweifaktor-Authentisierung gewählt werden.
- Bei SSH-Login sollte ein SSH-Schlüssel als Zweifaktor-Authentisierung gewählt werden.
- Der Login über SSH als Root-Benutzer muss verboten werden bzw. darf nur über die lokale Schnittstelle erfolgen. Nach Login über SSH als regulärer Nutzer kann bei Bedarf durch sudo/su zum Root-Benutzer gewechselt werden.
- SSH-Passwörter müssen immer besonders lang sein (mindestens 16 bis 32 Bytes), da diese Schnittstelle online durch Brute-Force angegriffen werden kann.
- Jedes Benutzerkonto darf nur die Rechte besitzen, die von der entsprechenden Person benötigt werden.

VI. Physischer Schutz des Gateways

Ein physischer Schutz des Gateways ist notwendig, da es sich auf Feldebene und nicht in einem geschützten Serverraum befindet, wie es bei Cloud-Komponenten der Fall ist. Die physische Härtung wird in diesem Kapitel erläutert.

Sollte ein Angreifer physischen Zugang zum Gateway erhalten, kann dies enorme sicherheitskritische Folgen haben. Das Verschließen des Gateways ergibt nur dann Sinn, wenn wenige, autorisierte Personen einen Schlüssel und damit physischen Zugang zum Gateway erhalten. Welche Personen hierzu autorisiert sind, sollte in regelmäßigen Abständen erneut geprüft werden. Ebenso muss sichergestellt werden, dass nicht mehr autorisierte Personen ihre Schlüssel zurückgeben.

Am Gateway selbst müssen nicht benötigte physische Schnittstellen deaktiviert werden. Außerdem sollte eine Whitelist für autorisierte Hardware geführt und regelmäßig überprüft werden. Fest verbaute Hardware, wie z. B. Speicherkarten, müssen so gesichert werden, dass diese nicht einfach austauschbar sind. Jede neu angeschlossene Hardware sollte mit aktueller Software auf Malware überprüft werden.

Weiterhin wird softwareseitig eine Fallback-Strategie empfohlen, die im Falle eines Wegfalls oder einer Störung der Verbindung zum MQTT-Broker in Kraft tritt.

Best Practices:

- Verteilen der Schlüssel für betroffene Räumlichkeiten oder Schaltschränke nur in begründeten Fällen.
- Stetige Kontrolle der Notwendigkeit aller verteilten Schlüssel.
- Einfordern verteilter Schlüssel, wenn diese nicht mehr benötigt werden.
- Deaktivieren nicht benötigter physischer Schnittstellen, bspw. nicht benutzte USB-Ports.
- Fester Einbau der genutzten SD-Karte, sodass sie nicht leicht entnommen werden kann.
- Führen einer Whitelist über autorisierte, angeschlossene Hardware.
- Generelle Überprüfung neu angeschlossener Hardware auf Malware.

VII. Absicherung des Netzwerks

In diesem Kapitel wird darauf eingegangen, dass das Netzwerk, die verbundenen Geräte sowie ein- und ausgehende Verbindungen überwacht werden müssen.

Ein- und ausgehende Verbindungen müssen durch Firewalls überwacht und gefiltert sowie alle

Schnittstellen ins Internet segmentiert werden, um sich vor Angriffen zu schützen und möglichst wenig Angriffsfläche zu bieten. Es wird empfohlen, das Netzwerk zwischen Router und Gateway in mehrere Subnetze aufzuteilen, um kritische Systeme von anderen Systemen zu entkoppeln. Firewalls und Netzwerke sollten von Fachkundigen konfiguriert und erstere stets auf dem neuesten Stand gehalten werden. Die Default-Regel der Firewall muss so konfiguriert werden, dass alle Anfragen, die nicht explizit erlaubt sind, verworfen werden. Regeln müssen differenzierbar für ein- und ausgehende Verbindungen gestaltet werden. Außerdem dürfen nur notwendige Ports geöffnet sein, alle anderen müssen dementsprechend geschlossen sein. Dies kann durch Tools wie nmap [8] überprüft werden. Durch die Nutzung einer Whitelist für zugelassene IP-Adressen und IP-Adressbereiche können Angriffsversuche von außen direkt unterbunden werden.

Best Practices:

- Es muss mindestens eine Firewall existieren und von einer fachkundigen Person konfiguriert sein.
- Wenn nicht bekannt ist, ob zusätzlich zu einer lokalen Software-Firewall eine weitere Firewall existiert, muss eine lokale Software-Firewall entsprechend sicher konfiguriert werden.
- Die Default-Regel einer Firewall muss alle Anfragen verwerfen, sofern keine anderen Regeln diese Anfragen explizit erlauben.
- Diese Regeln müssen zwischen ein- und ausgehenden Daten differenzieren.
- Alle nicht notwendigen Ports müssen geschlossen werden. Dies kann durch Tools wie nmap überprüft werden.
- Durch das Erstellen einer Whitelist für IP-Adressen und IP-Adressbereiche können Angriffsversuche direkt unterbunden werden.

VIII. Virenschutz

Im Falle eines dennoch erfolgreichen Angriffs und das Eindringen eines Virus, hilft ein Virenscanner bei der Detektion und beim Entfernen der Bedrohung bevor Schaden angerichtet werden kann. Unter Windows ist standardmäßig der Windows Defender aktiv. Unter Linux ist ClamAV [9] eine mögliche Open-Source Antiviren-Lösung. Während andere Virenscanner Hybrid-Lösungen sind und beispielsweise noch eine Firewall mit sich bringen, wurde ClamAV speziell dafür entwickelt, Viren, Trojaner, Malware und ähnliche Gefahren zu detektieren und abzuwehren.

Best Practices:

- Es sollte stets ein Virenschutz verwendet werden.
- Die Datenbank des Virenschutzes muss regelmäßig mit Updates versorgt werden.

IX. Hardware-Sicherheits-Module

In diesem Kapitel werden zwei Hardware-Sicherheits-Module beleuchtet: Das Secure Element (SE) und das Trusted Platform Module (TPM).

Der Einbau eines SE oder eines TPM in ein IoT-Gerät wie das betrachtete Gateway inkludiert eindeutige Vorteile bezüglich Gerätesicherheit und Effizienz. Für ein SE existieren mehrere Anwendungsfälle, von denen das Gateway in unserem Fall Gebrauch macht: Schlüssel-Safe, Krypto-Prozessor und Secure Boot. Ein TPM bietet im Regelfall zusätzliche Funktionen.

Mit der Verwendung des SE als Schlüssel-Safe können kryptographische Schlüssel sicher in diesem Safe gespeichert werden. Selbst wenn ein Angreifer physischen Zugriff auf das Gateway erlangt, ist das gespeicherte Schlüsselmaterial nicht auslesbar.

Als Krypto-Prozessor kann das SE asynchron kryptographische Berechnungen durchführen. Dieser ist für diese Art der Berechnungen ausgelegt und entlastet dadurch nicht nur die Host-CPU, sondern ist auch wesentlich effizienter als diese. Für die Berechnungen verwendet das SE das im Schlüssel-Safe gespeicherte kryptographische Material.

Durch Secure Boot kann kryptographisch verifiziert werden, dass während des Boot-Vorgangs nur Software verwendet wird, die im Vorfeld als vertrauenswürdig definiert wurde. Dies verhindert das Einschleusen und Ausführen bösartiger Software während des Startvorgangs. Für Secure Boot muss ein Unified Extensible Firmware Interface (UEFI) verwendet werden, das eine erweiterte Firmware-Schnittstelle bereitstellt.

Best Practices:

- Die Nutzung eines Secure Elements wird ausdrücklich empfohlen.
- Hauptanwendungsfälle des Secure Elements: Schlüssel-Speicher, Krypto-Prozessor und Secure-Boot.

X. Firmware Updates

Firmware Updates sind essenziell um die Software auf dem Endgerät stets aktuell zu halten. Die Absicherung

des Firmware Update-Prozesses stellt jedoch aus sicherheitstechnischer Perspektive eine nicht geringe Herausforderung dar, da kleine Fehler bereits große Auswirkungen haben können. Ein erfolgreicher Angriff über den Update-Mechanismus skaliert besonders stark, da die Updates und somit auch bösartige Softwareteile direkt an alle Geräte verteilt werden können.

Best Practices:

- Bei Firmware Updates sollte auf die vom Hersteller vorgesehenen Lösungen zurückgegriffen werden.
- Zugangsdaten für die clientseitige Update-Software sollten in separaten Dateien und in einem Secure Element abgespeichert werden.
- Es sollte auf Client-Authentisierung durch Client-Zertifikate zurückgegriffen werden, sofern der Update-Server dieses Verfahren anbietet.

XI. Zusammenfassung

Dieses Whitepaper ergänzt die Ausführungen des ersten Whitepapers, welches ebenfalls im Rahmen des Forschungsprojektes NextGenBAT entstanden ist, und widmet sich dediziert den sicherheitstechnischen Aspekten in der Gateway-Entwicklung für IoT-Anwendungen. Es wurden Best Practices formuliert, um einen - nach aktuellem Stand - sicheren Betrieb des Gateways zu gewährleisten und somit einen einfacheren Einstieg in die nächste Generation der Gebäudeautomation zu ermöglichen. Dazu zählen u. a. die Absicherung des Datenverkehrs auf Transportebene, starke Authentisierungsverfahren, sichere Autorisierung, die Verwendung aktueller Firmware, Firewalls und eines aktuellen Virenschutzes. Auch physikalisch sollte das Gateway abgesichert sein und nur Personen zugänglich sein, die diesen Zugang wirklich benötigen. Jede Sicherheitslösung muss abschließend berücksichtigen, dass auch menschliche Fehler zu Sicherheitslücken und -problemen führen können.

Neben den Gateways müssen auch die Cloud-Applikationen nach aktuellen Standards abgesichert sein und die cloudseitigen Anforderungen des Gateways erfüllen. Denn generell gilt der Grundsatz, dass ein System immer nur so sicher ist wie die unsicherste Stelle. Im Rahmen des Projektes sind die Analyse einer Cloudplattform sowie Veröffentlichung der Ergebnisse als nächsten Schritt angedacht.

Danksagung

Wir danken für die finanzielle Unterstützung durch das BMWi (Bundesministerium für Wirtschaft und Energie), Förderkennzeichen 03ET1657.

Referenzen

[1] S. Blechmann *et al.*, “Security-by-Design in der Gebäudeautomation,” RWTH Aachen University, Aachen, Whitepaper, 2020. doi: 10.18154/RWTH-2020-07469.

[2] “NextGenBAT | Next Generation of Building Automation Technology.” https://nextgenbat.de/ (accessed May 01, 2021).

[3] Bundesamt für Sicherheit in der Informationstechnik, “BSI-Standard 200-2.” Nov. 15, 2017, Accessed: May 01, 2021. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/BSI_Standards/standard_200_2.html;jsessionid=66621FA19BF21701D5134574A09E8D36.internet081?nn=128640.

[4] “MQTT - The Standard for IoT Messaging.” https://mqtt.org/ (accessed May 01, 2021).

[5] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[6] N. Naik, “Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP,” in *2017 IEEE International Systems Engineering Symposium (ISSE)*, Oct. 2017, pp. 1–7, doi: 10.1109/SysEng.2017.8088251.

[7] Y. Jia *et al.*, “Burglars’ IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds,” in *2020 IEEE Symposium on Security and Privacy (SP)*, May 2020, pp. 465–481, doi: 10.1109/SP40000.2020.00051.

[8] “Nmap: the Network Mapper - Free Security Scanner.” https://nmap.org/ (accessed May 01, 2021).

[9] “ClamavNet.” http://www.clamav.net/ (accessed May 01, 2021).