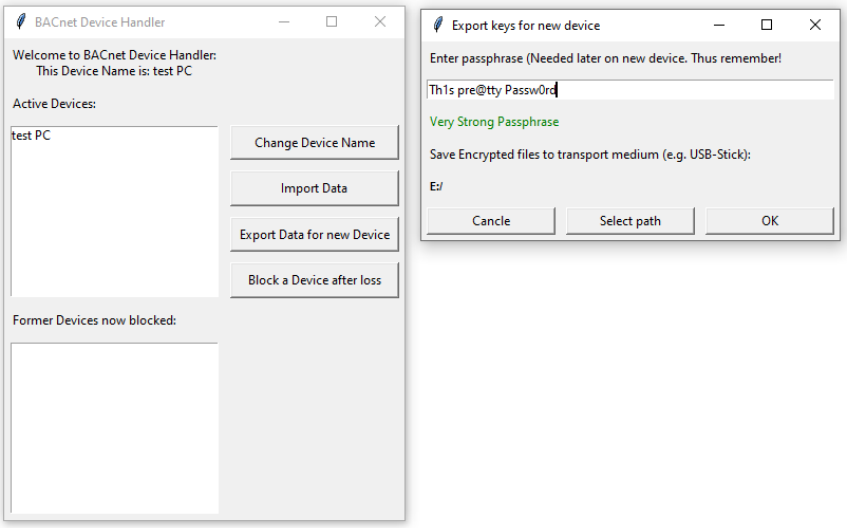
Kurs: Introduction to Internet and Security (IaS), Frühjahrsemester 2021
Dozenten: Prof. Dr. Christian Tschudin
Datum: 18. Juli 2021

Projekt Report

BACnet:
Multi Device Feed



Gruppe 10
Matthias Müller matthias01.mueller@stud.unibas.ch
Patrick Steiner pa.steiner@stud.unibas.ch
Reto Krummenacher reto.krummenacher@unibas.ch

1 Einleitung

Das BACnet ist ein dezentralisiertes Netzwerk, welches ohne das Internet und dessen gängige Protokolle (IP, TCP) die Kommunikation zwischen Teilnehmern ermöglicht. Die einfachste Art der Informationsübertagung ist die Übergabe von USB-Sticks mit den gewünschten Daten der Applikationen. Jede Applikation erstellt ihre eigenen Feeds mit einem privaten und öffentlichen Schlüssel. Ein Feed kann als erweiterbare aber nicht veränderbare verkettete Liste betrachtet werden. Anhand des öffentlichen Schlüssels sind die einzelnen Feeds unterscheidbar. Ein Nutzer im BACnet wird anhand eines sogenannten Master-Feeds[1] eindeutig identifiziert.

Bisher war es einem Teilnehmer im BACnet nicht möglich, die gleiche Applikation von verschiedenen Geräten zu bedienen. Dazu müssen die Privaten Schlüssel der Feeds geteilt werden, damit der gleiche Benutzer auf einem anderen Gerät die selben Feeds erweitern kann. Damit besteht die Gefahr von Kollisionen. Ein Nutzer schreibt von Gerät A und B gleichzeitig in einen Chat. Das dezentrale BACnet hat keine automatische Synchronisation, sprich die Daten von Gerät A werden nicht sofort auf das Gerät B übertragen. Für den Chatpartner müssen die unterschiedlichen Einträge von Gerät A und B zusammengefügt werden. Vor dem Hintergrund der BACnet Prämisse von nicht veränderbaren Feeds ist dies nicht möglich. Eine Lösung sind virtuelle Feeds.

Ziel dieses Begleitprojekts im Rahmen der Vorlesung ‘Internet and Security’ ist das Entwickeln eines Systems virtueller Feeds für das BACnet sowie einer Möglichkeit der Geräteverwaltung für den Benutzer. Dies aufbauend auf den Arbeiten des Frühjahrsemester 2020. Der vorliegende Bericht erläutert die erarbeiteten Lösungen und beschreibt die erstellten Python-Module [2] konzeptionell. In einem ersten Teil werden die virtuellen Feeds behandelt, während die Geräteverwaltung in einem zweiten Teil besprochen wird. Abschluss bildet die kritische Würdigung der Arbeit. Im Anhang finden sich sämtliche Abbildungen sowie die Liste verwendeter Python-libraries.

2 Virtuelle Feeds

Der virtuelle Feed ist der Grundbaustein des Projekts. Durch ihn eröffnet sich die Möglichkeit des "Multidevice", indem ein Feed virtuel in mehreren anderen Feeds "platziert" wird. Doch die Idee selbst wurde erst nach ein paar verworfenen Iterationen und Rücksprache mit Professor Tschudin festgelegt.

2.1 Idee und Probleme

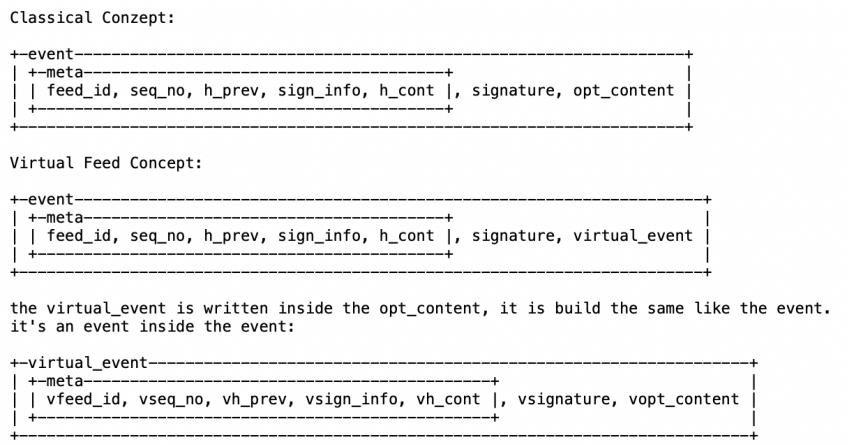
Einen Nutzer für mehrere Geräte zu haben ist rein vom Grundbau des BACnet Protokolls nur sehr schlecht möglich. Ein Ansatz war der erste Feed als Benutzerfeed an zu erkennen und dann die restlichen Feeds mittels dem Benuzerfeed zu authentifizieren. Ein Anderer war, einfach auf allen Geräten den gleichen privaten Schlüssel zu haben, was aber zu Fork Konflikten führen kann (‚dass aufteilen der Feed Blockchain in zwei unterschiedliche Stränge).

Um das Problem von Parallelismus zu umgehen, haben wir die Rahmenbedingungen angepasst und uns darauf geeinigt dass wir paralleles posten per seh ausschliessen, sprich der Benutzer verwendet immer nur ein Gerät aufs mal. Um das Hierarchieproblem zu lösen (welches ist der Benutzerfeed und welches untergeordnete Feeds) haben wir das Konzept des virtuellen Feeds ausgearbeitet. Dabei wird auf eine Idee zurückgegriffen die bereits Shakespeare bei seinen Theaterstücken hatte: anstatt ein Stück in einem

[1] Dies basiert auf Arbeiten aus dem Frühjahrsemester 2020:
https://github.com/cn-uofbasel/BACnet/tree/master/20-fs-ias-lec/groups/14-feedCtrl
(letzter Aufruf 11.07.2021)

[2] Sämtlicher Code findet sich unter:
https://github.com/cn-uofbasel/BACnet

Stück zu inszenieren, haben wir einen Feed in einem Feed (oder besser gesagt in mehreren) erschaffen. Es gibt zum einen die Gerätefeeds (auch Hostfeed genannt). Jedes Gerät besitzt einen eigenen Gerätefeed, mit einem eigenen privaten Schlüssel, der lokal gespeichert bleibt. Dann hat jedes gerät einen privaten Schlüssel des Virtuellen Feeds. Dieser Feed hat jedoch kein eigenes Pcap file, sondern die Events des Feeds werden in den Gerätefeeds als Content gespeichert.



Struktur des Virtuellen Events inerhalb des Gerätefeed Events

2.2 Umsetzung

Bei der Umsetzung haben wir uns vorallem an dem alten BACnet orientiert, nicht an der version vom BACnet Core Team, da wir mit den Files besser umgehen können, als wenn alles in einer Datenbank ist, ausserdem war es nicht möglich den Virtuellen Feed Schlüssel passwortgeschützt an die Datenbank weiter zu geben, da wir ein völlig neues format verwenden für die Schlüsselpaare. Und zwar ist der öffentliche Schlüssel nun im Filenamen gespeichert und der private Schlüssel als Bytestring im file. So ist beides einfacher auszulesen und braucht weniger Speicherplatz.

Wir haben bereits vom kleinen auch ins grosse gedacht und dabei ist uns aufgefallen dass bei grossen Feeds mit vielen Geräten, der Aufwand die ganzen virtuellen Events durchzurechnen sehr hoch ist. Deshalb haben wir uns dazu entschieden eine kleine "Abkürzung" zu nehmen um performance beim Nachrichten schreiben raus zu holen. Nämlich nutzen wir eine zusätzliche Stats Datei, um die Sequenznummer und den letzten Hash zu speichern, so können diese direkt zur Berechnung der neuen Nachricht eingesetzt werden. Was wenn jetzt irgendwer kommt und die sequenznummer verändert? Die Datenkorrektheit im Stats file haben wir gewährleistet indem wir eine Signatur aus dem privaten Schlüssel, der Sequenznummer und dem Contetnhash erzeugen und ebenfalls im Stats file speichern.

Nun zu den eigentlichen virtual Feeds. Wie bereits im Abschnitt 2.1 angetönt wird der virtuelle Event direkt in den Contentbereich des Hostfeeds geschrieben. Um diesen danach wieder einzulesen haben wir uns an der Methode `to_wire` und `from_wire` aus der `event.py` klasse orientiert und letztendlich `from_wire` so modifiziert, dass man damit ganze events aus einem Binarystream einlesen kann. Aus dem event können wir wiederum den Content, die Hashes und die Sequenznummern extrahieren. Dadurch können wir die Nachrichten chronologisch nach sequenznummer ordnen. Auch kann man dadurch die Stats vom Feed einlesen, falls dieses File mal ausversehen gelöscht wird.

3 CLI

Da `virtualFeed.py` keine wirkliche Klasse ist, sondern mehr daraus ausgelegt ist als aktives Programm zu funktionieren, wurde dafür ein Command Line Interface geschaffen, in dem der Benutzer die Funktionen live testen kann. Der funktionsumfang ist jedoch aufs nötige beschränkt, man kann den virtuellen Feed lesen und man kann in den Feed hineinschreiben. Um alles andere muss der Benutzer sich nicht kümmern, denn die Arbeit übernimmt das Programm im hintergrund. Auch Probleme mit den Files sollte das Programm beheben, sofern dies möglich ist.

4 Geräteverwaltung

Zu den Aufgaben der Geräteverwaltung zählt einerseits das Verteilen von privaten Schlüssel auf weitere Geräte eines Nutzers sowie das Gerätemanagement generell. Die Klassen und Methoden stellen die Funktionalität sowie eine GUI bereit.

4.1 Funktionalität

Die gesamte Funktionalität ist in `uiFunctions.py` implementiert.

Schlüsselverteilung

Wie im vorangehenden Kapitel erläutert, benötigen virtuelle Feeds einen gemeinsamen geheimen Schlüssel. Dieser muss auf allen Geräten identisch sein und daher geteilt werden können, ohne für Dritte lesbar zu sein . Daraus folgt, dass Geräte A und B einen gemeinsamen privaten Schlüssel benötigen, um den virtuellen Feed Schlüssel chiffrieren und dechiffrieren zu können. Das teilen von privaten Schlüssel ist unter dem Begriff 'Key exchange problem' bekannt. Dies kann mittels Verwendung eines asymmetrischen Schlüsselpaars umgangen werden. Zu sendende Informationen werden mit dem öffentlichen Schlüssel des Empfängers codiert. Nur mit dem privaten Schlüssel des Empfängers ist eine Decodieren möglich. Im vorliegenden Fall ist dies nicht praktikabel, da das Erstellen eines asymmetrischen Schlüsselpaars den Austausch der dazu notwendigen Informationen benötigt. Im dezentralen BACnet erfordert dies ein mehrmaliges hin und her reichen eines USB-Sticks. Aus diesem Grund haben wir uns für eine symmetrischen Verschlüsselungsverfahren mit einem gemeinsamen privaten Schlüssel entschieden.

Unser Lösungsansatz ist, dass der gemeinsame private Schlüssel gar nicht geteilt, sonder auf jedem Gerät individuell erstellt wird. Dazu verwenden wird den gleichen Algorithmus auf beiden Geräten zusammen mit einem vom Benutzer bereitgestellten Passwort. Dieses Verfahren ist als 'password based key derivation' bekannten und von der IETF[3] empfohlen. Die Daten, hier die geheimen Feed Schlüssel eines Benutzers, werden mit dem passwortbasierten Schlüssel auf Gerät A chiffriert und auf einem Transportmedium gespeichert. Nur mit dem Passwort ist ein Dechiffrieren auf Gerät B möglich.

Um die Stärke des eingegebenen Passworts zu bewerten, wird dessen Entropie berechnet. Die Entropie wird durch die Länge und die Menge der möglichen Zeichen bestimmt.[4] Je höher der Wert, desto mehr Kombinationen ($2^{Entropie}$) gibt es. Entsprechenden länger dauert ein Brute-Force erraten des Passworts.

[3] Internet Engineering Task Force, RFC 8018, Password-Based Cryptography Specification Version 2.1
https://datatracker.ietf.org/doc/html/rfc8018#appendix-A.2
(letzter Aufruf 11.07.2021)

[4] Berechnung der Entropie im Detail:
https://www.omnicalculator.com/other/password-entropy
(letzer Aufruf 9.7.2021)

Geräte hinzufügen, benennen und blockieren

Neben der Schlüsselweitergabe ist das Benennen und Blockieren der registrierten Geräte von Bedeutung. Jedes Gerät verfügt über eine Identifikationsnummer und einen privaten Schlüssel. Beides wird beim erstmaligen Ausführen erstellt und lokal gespeichert. Gleichzeitig wird ein provisorischer Benutzername zugeordnet. Anhand der eindeutigen, unveränderlichen Nummer kann ein Gerät identifiziert werden, während der Name selbst vom Benutzer nach Wunsch geändert werden kann. Ein Gerät kennt zwei Zustände, aktiv und blockiert. Jedes neu registrierte Gerät wird zunächst als aktiv gekennzeichnet. Die gesamten Informationen werden lokal gespeichert und simultan zur Schlüsselverteilung ebenfalls auf das Transportmedium übertragen. Über die Zeit entsteht ein Verzeichnis aller Geräte eines Benutzers.

Ein wesentliches Problem im Zusammenhang mit verschiedenen Geräten ist ein möglicher Verlust eines solchen. Darauf enthaltene Informationen sind in jedem Fall verloren. Noch vorhandenen historischen Meldungen des abhanden gekommen Geräts können weiterhin verwendet werden, falls dessen Identifikationsnummer bekannt bleibt. Auf Basis dieser Tatsache haben wir uns dazu entschlossen, keine Geräte zu löschen, sondern diese nur zu blockieren. Die Nummer verbleibt im Geräteverzeichnis.

4.2 GUI

Die GUI wurde mit `tkinter` erstellt und ist in `ui.py` implementiert. Die graphische Benutzeroberfläche bildet dehn Einstiegspunkt der Geräteverwaltung. Beim erstmaligen Ausführen auf einem Gerät wird automatisch eine Geräteschlüssel und ein Gerätename kreiert. Das Hauptfenster ermöglicht dem Benutzer die Funktionalitäten der Geräteverwaltung per Klick auszuführen. Darunter das Exportieren und Importieren der Feed-Schlüssel. Der jeweiligen Button öffnet ein Dialog-Box, worin der Nutzer die benötigten Informationen wie Pfad oder Passwort eingeben kann. Abbildung 2.1 zeigt das Hauptfenster zusammen mit dem geöffneten Exportdialog. Letzterer enthält einen Hinweis zur Stärke des eingegebenen Passworts. Das angezeigte Geräteverzeichnis bietet die Optionen, Geräte zu blockieren. Ein Umbenennen ist nur für das aktuel in Gebrauch befindliche Gerät erlaubt.

5 Fazit

Die Geräteverwaltung bietet die geforderte Funktionalität sowie eine benutzerfreundliche GUI zur chiffrierten Übertragung privater Schlüssel zwischen Geräten. Ebenfalls gelang die Umsetzung des virtuellen Feeds gut mitsamt implementierung eines Command-Line Interfaces zur einfachen Verwendung. Die Gruppe Masterfeed bedankt sich bei Professor Tschudin für den Vorschlag des virtuellen Feeds und dessen Erläutreung.

Zusammenfassend lässt sich sagen, dass die im Verlauf dieses Projekts festgelegten Anforderungen mehrheitlich erfüllt wurden. Nicht erreicht wurde der Aufbau unserer Lösung für virtuelle Feeds auf das im Frühjahrsemester 2021 von der BACnet Core Gruppe entwickelte System. Dies ist der Tatsache geschuldet, dass den dortigen Neuerungen von uns zu spät Beachtung geschenkt wurde. In einer Gesamtbetrachtung überwiegen jedoch die positiven Punkte, weshalb wir dieses Begleitprojekt als Erfolg bewerten.

Anhang

Anhang 1 Python libraries

- `cryptography`
  https://cryptography.io/en/latest/ (letzter Aufruf 11.07.2021)

  `.hazmat.primitives.kdf.pbkdf2.PBKDF2HMAC`
  Erstellen eines Key auf Basis eines Passworts

  `.hazmat.primitives.ciphers.aead.AESGCM`
  Verschlüsselung mittels AES und GCM mittels Key

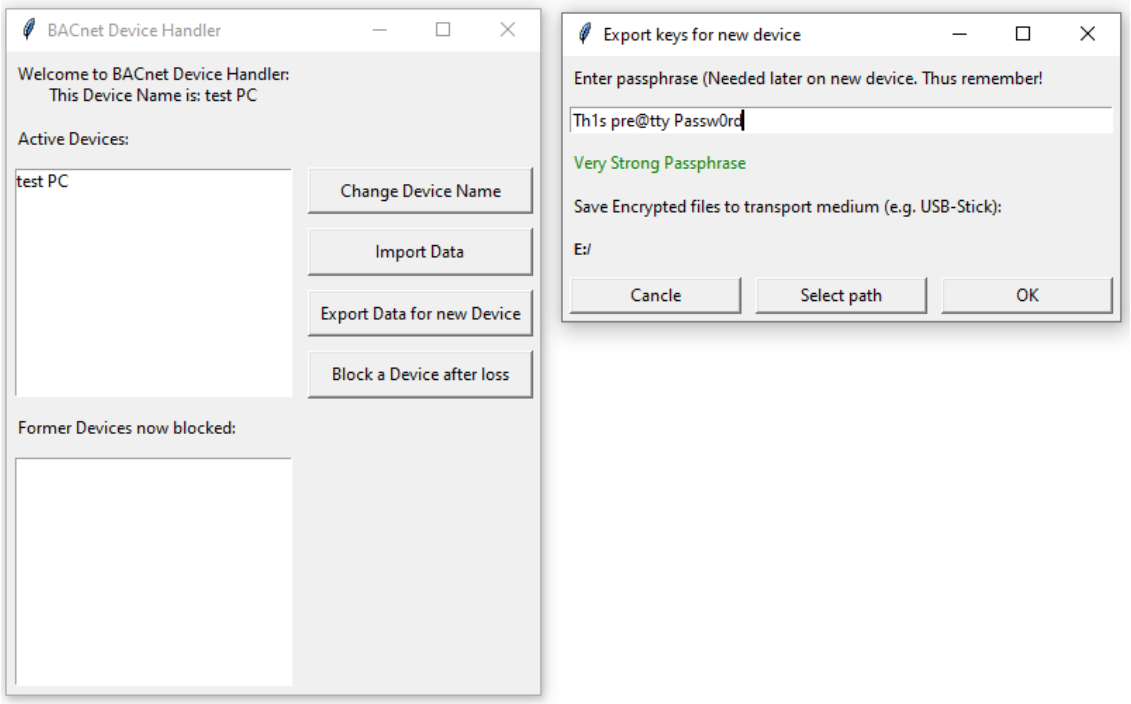
  `.hazmat.primitives.hashes`
  Enthält den Algorithmus SHA256

  `.exceptions.InvalidTag`
  Exception beim Entschlüssel mit falschem Passwort

- `getpass`
  Bestimmung des Benutzernamens auf dem Betriebssystem
- `json`
  Library für den Umgang mit JSON-Dateien
- `math`
  Mathematische Operationen
- `os`
  Pfad und Datei Methoden
- `re`
  Verwendung von regulären Ausdrücken
- `secrets`
  Erstellen von Test-Keys für das Testing
- `shutil`
  Kopieren von Dateien
- `tkinter`
  Erstellen einer GUI
- `unittest`
  Wird in der Klasse `TestMethods` verwendet. Diese enthält Funktionstest des Device Handlers.
- `lib/crypto`
  Erzeugen von HMAC Objekten und Asymetrischen Schlüsseln
- `lib/feed`
  Erstellen eines Feeds
- `lib/event`
  Erstellen eines Events
- `hashlib`
  Berechnen einer SHA256 Checksum

Anhang 2 Abbildungen

2.1 GUI



Das Hauptfenster und der geöffnete Exportdialog, welcher ein Hinweis zur Passwortstärke enthält.

2.2