VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNOSTNÍ ANALÝZA DOMÁCÍ IOT SÍTĚ

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ ČIKEL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN PLUSKAL
SUPERVISOR

BRNO 2020

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

23135

Student: **Čikel Tomáš**
Program: Informační technologie
Název: **Bezpečnostní analýza domácí IoT sítě**
Security Analysis of Home IoT Network
Kategorie: Vestavěné systémy
Zadání:

1. Seznamte se s běžně používanými komponentami pro domácí IoT, např. Sonoff, Moeshouse a jiné dle doporučení vedoucího. Zjistěte jejich možnosti použití a integrace do oficiálních automatizačních systémů dodávaných jejich výrobci.
2. Proveďte analýzu síťové komunikace s cílem zjištění informací, které tyto komponenty prozrazují o uživateli. Zaměřte se také na možné zranitelnosti, které umožňují prozradit citlivé informace, nebo převzít kontrolu nad komponentou.
3. Po dohodě s vedoucím navrhněte, implementujte a otestujte opatření, jak znemožnit využití objevených zranitelností nebo zneužití dat, která zařízení o uživateli zasílá. Nahraďte, typicky cloudové, automatizační platformy výrobců jednotlivých komponent pomocí open source nástrojů třetích stran, např. automatizačního nástroje Home Assistant a alternativních open source firmware vyvinutých komunitou pro dané komponenty.
4. Diskutujte nalezené zranitelnosti/úniky dat a navrhněte soubor doporučení, jak vybudovat bezpečnou domácí IoT síť, která zamezí úniku informací o uživateli mimo jeho síť.

Literatura:

- Simpson, A. K., Roesner, F., & Kohno, T. (2017, March). Securing Vulnerable Home IoT Devices with an In-Hub Security Manager. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (pp. 551-556). IEEE.
- Ali, B., & Awad, A. (2018). Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. *Sensors*, *18*(3), 817.
- Oh, S. R., & Kim, Y. G. (2017, February). Security Requirements Analysis for the IoT. In *2017 International Conference on Platform Technology and Service (PlatCon)*(pp. 1-6). IEEE.
- Lin, H., & Bergmann, N. (2016). IoT Privacy and Security Challenges for Smart Home Environments. *Information*, *7*(3), 44.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Pluskal Jan, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Táto práca sa zaoberá problematikou bezpečnosti domácej IoT siete. Zameriava sa na bezpečnostné problémy komerčných IoT zariadení a ich automatizačných systémov. V samotnej práci boli najskôr analýzou zistené konkrétne bezpečnostné hrozby v sieti. Pre riešenie týchto hrozieb, boli nahradené pôvodné automatizačné systémy a firmware na zariadeniach open-source implementáciami, ktoré boli nakonfigurované tak aby riešili konkrétne objavené hrozby. Aj keď sa v sieti s open-source implementáciou nachádza niekoľko bezpečnostných hrozieb, podarilo sa eliminovať veľkú časť objavených bezpečnostných hrozieb. Výsledky tejto práce umožňujú nakonfigurovať domácu IoT sieť použitím open-source automatizačného systému a firmware a dokazujú, že takáto sieť je bezpečnejšia oproti sieti kde sa používaju pôvodné, výrobcom dodané automatizačné systémy a firmware na zariadeniach.

Abstract

This work focuses on the issue of the security in home IoT network. Specifically, it focuses on the security issues of commercial IoT devices and their automation systems. In the work itself, it was firstly analyzed and identified specific threats specific for our network. To address these threats, the original automation systems and firmware on the devices were replaced by open-source implementations that were configured to address discovered threats. Although, there are still several security issues in a network with an open-source implementation, a large part of the security issues which were discovered has been eliminated. The results of this work allows to configure a home IoT network using an opens-source automation system and firmware and proves that such a network is more secure than the network where the original factory-supplied automation systems and firmware on devices are used.

Kľúčové slová

Inernet vecí, bezpečnosť, Tasmota, ESPHome, Home Assistant, ESP8266, MQTT komunikácia, Wi-Fi, analýza, firmware, open-source, automatizácie

Keywords

Internet of Things, security, Tasmota, ESPHome, Home Assistant, ESP8266, MQTT communication, Wi-Fi, analysis, firmware, open-source, automatizations

Citácia

ČIKEL, Tomáš. *Bezpečnostní analýza domácí IoT sítě.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Bezpečnostní analýza domácí IoT sítě

Prehlásenie

Prehlasujem , že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jan Pluskal. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

. . . . . .
Tomáš Čikel
28. mája 2020

Poďakovanie

Rád by som sa poďakoval vedúcemu bakalárskej práce pánovi Ing. Janovi Pluskalovi, za odborné vedenie, konzultácie a trpezlivosť pri konzultáciach k tejto práci.

Obsah

1 Úvod **3**

2 IoT **4**
2.1 Zariadenia v IoT sieti 4
2.2 Automatizačné systémy 9
2.2.1 Automatizačné systémy dodávané výrobcami 9
2.3 Komunikácia IoT sieti 11
2.3.1 Protokoly na štandarte 802.15.4 a proprietarné protokoly 11
2.3.2 IoT protokoly v štandarte IEEE 802.11 12
2.3.3 Ostatné protokoly používane v IoT 13

3 Bezpečnosť **14**
3.1 Bezpečnosť v domácej IoT sieti 14
3.2 Bezpečnostné hrozby v IoT 15
3.2.1 Fyzické hrozby 15
3.2.2 Hrozby na nižších vrstvách OSI modelu 15
3.2.3 Hrozby na vyšších vrstviach OSI modelu 16
3.2.4 Analýza komunikácie 18
3.2.5 Simulácia MITM útoku 21
3.2.6 Simulácia nedostupného systému 22
3.2.7 OTA nahratie firmware 23
3.2.8 Známe a objavené hrozby v IoT sieti 24

4 Open-source systémy **25**
4.1 Firmware 25
4.2 Open-source systémy 29
4.3 Implementácia v našej IoT sieti 31
4.3.1 Konfigurácia Home Assistant 33
4.3.2 Firmware 35
4.3.3 Testovanie a porovnanie s oficialnými automatizačnými systémami . 38

5 Diskusia a doporučenia na vytvorenie bezpečnej IoT siete **41**
5.1 Diskusia 41
5.2 Doporučenia pre vytvorenie bezpečnej IoT siete 42

6 Záver **45**

Literatúra **47**

A Konfigurácia open-source IoT sieťe **50**
A.1 Home Assistant 50
A.1.1 Zabezpečenie a vzdialený prístup 51
A.1.2 MQTT a zálohovanie systému 53
A.2 Konfigurácia ESPHome 55
A.3 Tasmota 58

Kapitola 1

Úvod

Táto práca sa venuje bezpečnosti domácej IoT siete, viacej známej pod konceptom „Smart home“, konkrétne analýze komunikácie v nami vytvorenej sieti, potencionálnym bezpečnostným hrozbám a únikom citlivých informácií. Cieľom práce je zoznámenie s IoT zariadeniami, odhalenie potencionálnych zrantiteľnosti alebo možnosti zneužitia dát, implementovanie a otestovanie riešenia pre najdené zraniteľnosti.

IoT technológie zaznamevajú v posledných rokoch rýchly rast, z tohto dôvodu sa obava o bezpečnosť a súkromie stala jednou z hlavných oblastí, na ktoré sa práce ako je táto zameriavajú. S vysokým rastom používania IoT technológii rastie taktiež počet útokov na ne. Iba v prvej polovici roku 2019 bolo zdetekovaných vyše 100 miliónov útokov na IoT zariadenia, počet útokov za prvú polovicu roka 2019 je skoro o 9 krát väčší ako počet nameraných za prvú polovicu roka 2018 [26]. Ďalší posudok dokonca uvádza 2.9 miliardy detekovaných udalostí za prvú polovicu 2019, čo je 12 krát viac, než bolo detekovaných za ten istý čas v roku 2018 [22]. Aj keď sa dáta medzi týmito dvoma zdrojmi líšia, je jasne vidieť trend nárastu počtu útokov na IoT siete.

IoT siete často používajú na komunikáciu protokoly, ktoré sú nešifrované alebo sú postavené na úplne iný účel ako použitie v IoT sieťach napr. IPDC. Vďaka tomuto sú tieto sieťe vystavené bezpečnostným hrozbám, ktoré môže potencionálny útočník využiť na získanie dát,či prístupu do siete. Taktiež je tu problém s aktualizovaním firmware na zariadeniach, ktoré sa už používaju. Nie všetky zariadenia ponúkaju túto možnosť. Často potom čo sa dostane zariadenie k zákaznikovi, sa zariadenie buď nedá alebo samotný výrobca sa ani nesnaží aktualizovať zariadenie proti známym bezpečnostným hrozbám. Automatizačné systémy, ktoré sú typicky cloudové, ponúkaju možnost nielen ovládania IoT siete z hocijakého miesta, ale tiež možnosť odchytenia komunikácie mimo lokálnej siete potencionálnym útočníkom. Aj vďaka tomu sa open-source systémy zameriavajú na bezpečnosť a kontrolu zariadení z lokálnej siete.

V prvej časti práce sa čitatel zoznámi so zariadeniami v našej sieti a ich automatizačnými systémami. Zistí akú funkcionalitu majú zariadenia a aký je rozdiel medzi ich automatizačnými systémami. V druhej časti je prevedená analýza sieťovej komunikácie našej IoT siete. Čitatel v nej nájde ukážky nedostatkov a bezpečnostných hrozieb v našej sieti. V tretej časti sa čitatel zoznámi s open-source systémami a firmware. Implementujú a otestujú sa opatrenia, ktoré maju za úlohu znemožniť využitie objavených zraniteľností potencionálnym útočníkom. Za týmto účelom sa nahradia cloudové automatizačné systémy open-source ako je napríklad Home Assistant. V štrvtej časti sa nachádzajú diskusia a doporučenia, na čo sa zamerať pri vytváraní IoT siete z pohľadu bezpečnosti. Diskusia vychádza zo zistení z predošlých častí práce.

Kapitola 2

IoT

V tejto bude predstavené prostredie IoT technológii, a zariadeniami ktoré sa nachádzajú v našej IoT sieti, budú porovnané ich automatizačné systémy a v krátkosti budú predstavené protokoli, ktoré sa používaju v IoT sieťach.

Internet of Things je v posledných rokoch stúpajúcim trendom. IoT infraštruktúra je tvorená zo siete, ktorá obsahuje rôzne zariadenia ako počítače, senzory, vypínače, a technológie ako RFID[1], Bluetooth, NFC[2], Wi-Fi a iné. Cieľom IoT je rozšíriť funkcie internetu zvýšením schopnosti pripojiť rôzne zariadenia k sieti. Vďaka IoT môžu užívatelia zdielať informácie tvorené buď samotným užívateľom (napr. manuálne nastavenie teploty na termostate) alebo informácie zozbierané z IoT zariadení napr. senzor. Hlavnou funkciou IoT je umožniť užívateľom jedinečnú identifikáciu, označenie, prístup, automatizáciu a kontrolu zariadení kedykoľvek a kdekoľvek pomocou Internetu [10].

Smart Home

Smart Home konceptu sa rozumie domáca IoT sieť, do ktorej sú pripojené rôzne senzory, domáce spotrebiče, chytré zariadenia a je zabezpečená schopnosť ich vzdialeného sledovania, pripojenia a prístupu k zariadeniam v tejto sieti pomocou internetu. Hlavným cieľom Smart Home je zvýšiť automatizáciu v dome, zjednodušiť správu energií a ich spotrebu. Smart Home sa zameriava na automatizáciu a ovládanie služieb ako osvetlenie, kúrenie, ventilácia a klimatizácia, monitorovanie, ovládanie bezpečnosti domu (napr. zámok na dvere, alarm) a zvýšenie komfortu jeho uživatelom (napr. Amazon Echo). Technicky Smart Home pozostáva z 5 častí: ovládateľné zariadenia, senzory, sieť po ktorej zariadenia komunikúju, zariadenie ktoré zbiera informácie zo sensorov a ovláda ostatné zariadenia v sieti a neposlednom rade zariadenie pre vzdialený prístup ku sieti napr. pomocou aplikácie v smartphone [10].

2.1 Zariadenia v IoT sieti

Zariadenia v tejto konkrétnej IoT sieti pochádzajú od rôznych výrobcov. Ale je možné ich rozdeliť do dvoch skupín. Zariadenia od spoločnosti Sonoff a zariadenia, ktoré sú od viacerých výrobcov, ale všetky sú postavené na platforme Tuya. Všetky sú napájané pomocou 240 V, a okrem zásuvky BW-SHP6, ktorá sa iba zapojí do už existujúcej zásuvky sa všetky zariadenia musia zapojiť priamo do elektrickej siete.

[1] Vysokofrekvenčná identifikácia
[2] Krátkodosahové bezdotykové vysokofrekvenčné spojenie

ESP8266

Všetky zariadenia v sieti používaju buď *ESP8266* alebo *ESP8285* Wi-Fi mikrokontrolér, ktorý slúží na ovládanie zariadenia a vďaka ktorému sa pripájajú do siete. Tieto mikrokontroléri sú navrhnuté pre mobilné zariadenia, nositeľnú elektroniku a IoT zariadenia. ESP8285 mikrokontrolér je v podstate taký istý ako ESP8266 ale ma 1MB flash pamäte. Mikročipy bežia na 3.3 V a používajú L106 32 bit RISC procesor s maximálnou frekvenciou 160 Mhz. Tieto čipy podporujú Wi-Fi štandart 802.11 b/g/n, v prípade n štandartu podporuje iba 2,4 Ghz a rýchlosť až do 72.2 Mbps. Mikrokontroléry implementujú TCP/IP protokol a 802.11 b/g/n WLAN MAC protokol. Mikrokontrolér taktiež podporuje nízko levelové komunikačmé protokoly ako I2C, SPI, sériovú komunikáciu pomocou UART a taktiež aktualizácie OTA [31].

ESP8266 dokáže operovať v rôznych režimoch:

- Aktívny režim: Mikrokontrolér má zapnutý vysielač. ESP8266 môže posielať, prijímať alebo počúvať.
- Režim spánku: CPU beží v klasickom režime zatiaľ čo Wi-Fi s rádiom sú deaktivované.
- Režim ľahkého spánku: CPU a všetky periferné zariadenia sú pozastavené. Akékoľvek udalosti (MAC, RTC[3] timer, alebo externé prerušenia) čip prebudia.
- Režim hlbokého spánku: Všetky časti okrem RTC sú vypnuté.

Mikrokontroléry podporujú taktiež ovládanie pomocou IR[4], zároveň ponúkaju 17 univerzálnych vstupno/výstupných pinov, ktorým sa možu priradiť rôzne funkcie naprogramovaním príslušných registrov [31].

BHT-6000 Termostat

Séria termostatov navrhnutá na používanie v domácnostiach, slúži na ovládanie oteplovania vody, podlahy a samotného domu. Komunikujú cez Wi-Fi, môžu podporovať Modbus[5] alebo BACnet[6] komunikačné protokoly. Presnosť merania teploty je +- 0.5 °C. Termostat obsahuje pamäť, ktorá zachováva dáta v prípade výpadku napájania. Taktiež obashuje LCD display, na ktorom sa zobrazujú informácie ako napr. teplota,či vlhkosť. Náš konkrétny model obsahuje dva senzory `NTC39z0`. Interný, ktorý slúži na meranie teploty vzduchu a externý, ktorý slúži na meranie teploty podlahy. Termostat sa dá nastaviť pomocou aplikácie alebo aj priamo pomocou tlačidiel na samotnom termostate. Narozdiel od ostatných zariadení v sieti, používa ESP8266 iba na komunikáciu v sieti, samotnú kontrolu zariadenia zabezpečuje *Sonix SN8F57084SG* mikrokontrolér. ESP8266 komunikuje s týmto mikrokontrolérom UART sériovou komunikáciu, ktorá používa špecifický Tuya protokol.

Obr. 2.1: BHT-6000, prevzaté z [1]

[3] Real Clock Timer
[4] Infračervené žiarenie
[5] http://www.modbus.org/
[6] http://www.bacnet.org/

Komponenta	Špecifikácia
ESP8266	TYWE3S modul, slúži iba na komunikáciu v sieti
SN8F57084SG	Ovládanie termostatu, tlačidiel a displeja
2x NTC3950	Meranie teploty vzduchu a podlahy
LED display a tlačidlá	Zobrazovanie informácií a ovladánie zariadenia

Tabuľka 2.1: Komponenty zariadenia BHT-6000

BW-SHP6 Zásuvka

Smart zásuvka od spoločnosti BlitzWolf, na platforme Tuya, ktorú je možné kontrolovať cez Wi-Fi alebo fyzicky pomocou tlačidla. Zbiera aktuálne štatistiky o spotrebe elektrickej energie pomocou BL0937 mikročipu. Zariadenie na rozdiel od ostatných zariadení v sieti obsahuje `ESP8285` mikrokontrolér, ktorým sa pripája do systému. Maximálne zaťaženie zásuvky je 10A. Toto zariadenie sa narozdiel od ostatných veľmi ľahko zapojí do elektrickej siete prostým zapojením do existujúcej zásuvky.

Obr. 2.2: BW-SHP6, zásuvka prevzaté z [2]

ESP8266 piny	Komponenta
GPIO00	LED pre Relé
GPI002	LED
GPIO05	BL0937
GPIO13	Tlačidlo
GPIO15	Relé

Tabuľka 2.2: Rozdelenie pinov BW-SHP6

QS-Wifi-S03-2 vypínač

Dvojkanálový smart vypínač na platforme Tuya, ktorý neobsahuje žiadne tlačidlá na manuálne ovládanie, ale má terminály, do ktorých sa dajú zapojiť iné manuálne vypínače. Tie ale nie sú do ESP8266 zapojené ako vypínače ale ako počitadlá frekvencie pripojenia. Toto zariadenie sa hodí tam kde je už cielové zariadenie ovládané manuálne a iba chceme pridať podporu možnosti ovládať zariadenie aj vzdialene. Každý kanál má maximálne zaťaženie 5A.

Obr. 2.3: QS-Wifi-S03-2, prevzaté z [3]

ESP8266 piny	Komponenta
GPI002	Tlačidlo (reset)
GPIO04	Bzučiak
GPIO12	Počitadlo 2
GPIO13	Počitadlo 1
GPIO14	Relé 1
GPIO15	Relé 2

Tabuľka 2.3: Rozdelenie pinov pre QS-WFI-S03-2

Sonoff POW

Jednokanálový vypínač od spoločnosti Sonoff. Meria spotrebu elektrickej energie pomocou HLW8012 mikročipu. Zariadenie sa dá ovládať pomocou vypínača na samotnom zariadeni, ale nedá sa doňho zapojiť externý vypínač ako pri QS-WIFI-S03S-2. Takže, v prípade že by užívateľ chcel ovládať zariadenie aj manuálne, tento vypínač by nebol vhodný. Ale ako jediný vypínač v sieti ponúka taktiež zapojenie na ochranné uzemnenie. Maximálne zaťaženie vypínača je až 16A, takže sa hodí na monitorovanie a spúšťanie energeticky náročných spotrebičov.

Obr. 2.4: Sonoff POW, prevzaté z [4]

ESP8266 piny	Komponenta
GPIO00	Tlačidlo
GPIO12	Relé
GPIO14	HLW8012
GPIO15	LED 1

Tabuľka 2.4: Rozdelenie pinov pre Sonoff POW

Sonoff T3US3C nastenný vypínač

Dotykový 3-kanálový vypínač od spoločnosti Sonoff. Ovládať sa dá pomocou aplikácie alebo fyzicky vďaka dotykovej plochy, ktorá obsahuje LED podsvietenie pre ľahkú navigáciu. Ako jediné zariadenie v sieti sa dá taktiež ovládať pomocou RF komunikácie, konkrétne signálmi s frekvenciou 433.92 Mhz. Vypínač používa na pripojenie do siete ESP8266 mikrokontrolér a jeho maximálne zataženie pre jeden kanál je 2A.

Obr. 2.5: Sonoff T3US3C, prevzaté z [5]

ESP8266 piny	Komponenta
GPIO00	Tlačidlo 1
GPIO04	Relé 3
GPIO05	Relé 2
GPIO09	Tlačidlo 2
GPIO10	Tlačidlo 3
GPIO12	Relé 1
GPIO13	LED

Tabuľka 2.5: Rozdelenie pinov pre Sonoff T3US3C

KS-601 vypínač

Dotykový vypínač, ktorý nám dovoľuje ovládať iba jedno zariadenie, buď pomocou samotného vypínača alebo pomocou aplikácie. Obsahuje LED ktorá indikuje či je zariadenie vypnuté alebo zapnuté. Maximálne zaťaženie vypínača je 10A. Zariadenie používa ESP8266 mikrokontrolér na jeho ovládanie a pre komunikáciu so sieťou.

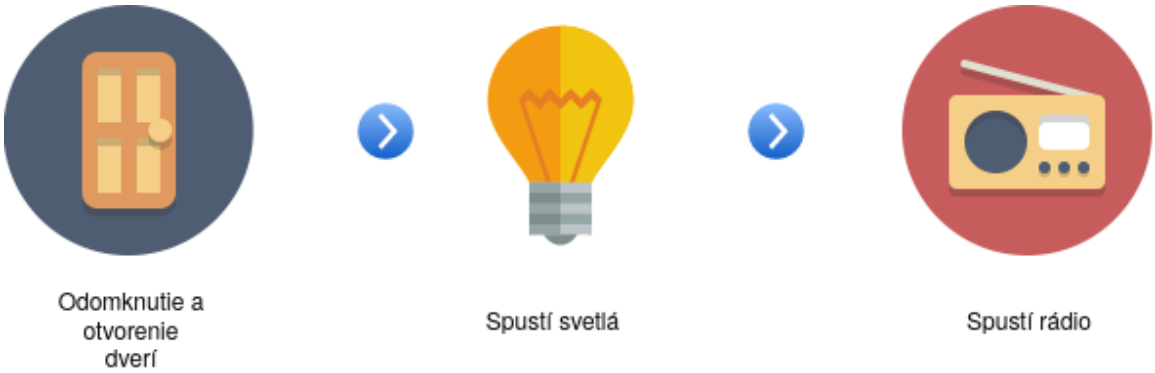
Obr. 2.6: KS-601, prevzaté z [6]

ESP8266 piny	Komponenta
GPIO02	LED
GPIO05	Tlačidlo
GPIO12	Relé

Tabuľka 2.6: Rozdelenie pinov pre KS-601

2.2 Automatizačné systémy

Na ovládanie IoT zariadení a ich automizáciu slúžia automatizačné systémy a zvyčajne aplikácie, ktoré tvoria ich uživaťelské prostredie. Vďaka nim dokážeme plánovať kedy presne sa má zariadenie zapnúť/vypnúť, nastaviť časovač počas ktorého bude zariadenie zapnuté/vypnuté napr. zapnutie zásuvky cez ktorú si budeme nabíjať telefón. Môžeme spojiť zariadenia do skupín. Toto sa využije napríklad pri zapínaní viacerých svetiel naraz. Dokážeme zdielať kontrolu zariadeni s inými uživateľmi, ale hlavne skutočne zautomatizujeme náš dom pomocou „smart scén“. V nich dokážeme prepojiť zariadenia medzi sebou a vytvoriť „scénu“, v ktorej sa budú funkcie jedného zariadenia spúšťať vykonaním nejakej funkcie v inom zariadení 2.7.



Obr. 2.7: Ukážka jednoduchej „smart scéni“

Taktiež je možné vďaka nim prepojiť zariadenia s inými systémami ako napr. IFTTT[7], Amazaon Alexa, Google Assistant ktoré ponúkajú napr. možnosť ovládať zariadenia v sieti pomocou hlasu.

2.2.1 Automatizačné systémy dodávané výrobcami

Automatizačné systémy dodávané výrobcami sa zvyčajne nachádzaju na cloude mimo lokálnu sieť. Výhoda týchto systémov je veľmi ľahká implementácia pre užívateľa. Ten sa nemusí zaoberať aktualizovaním systému alebo zariadení v prípade, že je vydaná firmware aktualizácia, ktorá buď prebieha automaticky alebo jedným kliknutím. Problémom týchto systémov je ten, že samotný systém sa nachádza mimo lokálnej siete, čiže v prípade straty internetového pripojenia, zariadenia nemusia fungovať tak ako by mali. Taktiež, užívateľ nemá kontrolu nad tým ako veľmi je tento systém zabezpečený a aké informácie o ňom zbiera výrobca. Každý výrobca má vlastný systém, nastáva teda ešte problém intergrity zariadení pod jeden systém v prípade rôznych výrobcov. V tejto IoT sieti nebolo možné integrovať zariadenia na Tuya platforme pod systém pre Sonoff zariadenia a tak isto Sonoff zariadenia pod Tuya systémy.

Ewelink

Automatizačný systém, ktorý sa nachádza na cloude. Integruje pod seba niekoľko značiek, medzi ktorými je aj Sonoff. Klient systému je podporovaný na platformách Android a IoS. Umožnuje kontrolovať zariadenia vzdialene, ale v prípade výpadku pripojenia na cloud sa

[7] If This, Than That je služba na zautomatizovanie úloh

dajú niektoré zariadenia kontrolovať lokálne prepnutím systému do „Lan módu“. V tomto móde ale nejde spárovať nové zariadenia so systémom a nie všetky zariadenia podporujú tento mód. Ewelink ponúka niekoľko spôsobov ako spárovať zariadenia so systémom:

- Rýchle spárovanie: Aplikácia automaticky vyhľadá zariadenia v lokálnej sieti ktoré sú v párovaciom režime a spáruje ich zo systémom.
- Manuálne spárovanie: Užívateľ si vyberie aké konkrétne zariadenie chce spárovať.
- Spárovanie pomocou QR kódu: Niektoré zariadenia podporujú GSM[8] sa dajú spárovať pomocou oskenovania QR kódu.
- Spárovanie pomocou Bluetooth: V prípade, že zariadenie podporuje iba Bluetooth, systém automaticky vyhľadá Bluetooth zariadenia v blízkosti.
- Spárovanie pomocou zvuku: System prehráva vysokofrekvenčný zvuk a v prípade, že to zariadenie podporuje, spáruje sa so systémom.

Systém ponúka možnosť automatizácie vytvorením „smart scén“, nastavenia časovača na konkrétne zariadenia a zdieľanie kontroly zariadenia s iným užívateľom. Taktiež sa dá zobraziť história daného zariadenia a je možné zaktualizovať firmware na zariadeniach, alebo zauktualizovať samotnú aplikáciu. Ewelink ponúka možnosť integrovať rôzne externé systémy ako napríklad Amazon Alexa, Google Home, IFTT, Google Nest, MI Home a Ezviz.

Tuya-based

Každé Tuya-based zariadenie malo priradený automatizačný systém od svojho výrobcu. Systémy sú v podstate identické s jemne rozdieľným užívateľským prostredím. Podobne ako systém pre Sonoff zariadenia sa taktiež nachádzajú na cloude a tak isto ako Ewelink je klient týchto systémov podporovaný na platformách IoS a Android. Tuya-based systémy ponúkaju iba tri možnosti spárovania zariadení so systémom:

- Rýchle spárovanie: Aplikácia automaticky vyhladá zariadenia v lokálnej sieti, ktoré sú v párovaciom režime a spáruje ich so systémom.
- Manuálne spárovanie: Užívateľ si vyberie aké konkrétne zariadenie chce spárovať.
- Spárovanie pomocou Bluetooth: V prípade, že zariadenie podporuje iba Bluetooth, systém automaticky vyhľadá Bluetooth zariadenia v blízkosti.

Tak isto ako systém Ewelink ponúka možnosť vytvorenia smart scén, ale v týchto systémoch sú rozdelené na dve časti: „Scenária“, v ktorých sa určí čo sa má vykonať a „Automatizácie“, v ktorých sa určia podmienky spustenia a ktorá scéna sa má spustiť v prípade splnenia tejto podmienky. Podobne sa dájú nastaviť aj časovače na zariadeniach a ich zdieľanie s iným užívateľom. Taktiež ponúkaju možnosť integrovať systémy, ale je ich menej než v príade Ewelink. Sú to Amazon Alexa, Google Assistant a IFTT.

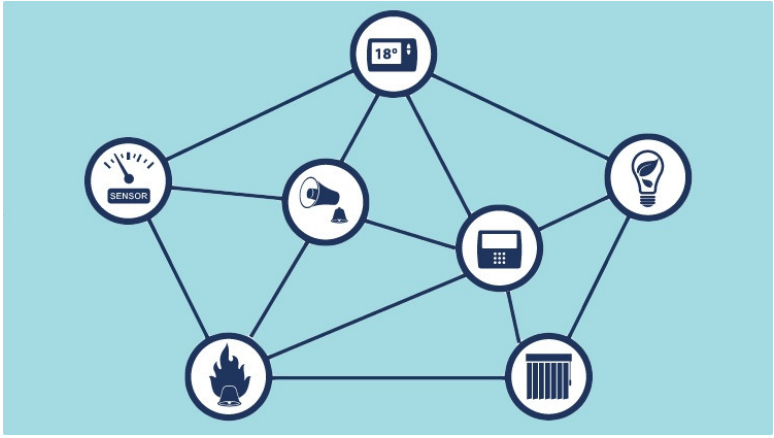
[8] Global System for Mobile Communications

2.3 Komunikácia IoT sieti

IoT sieť je tvorená z viacerých zariadení, každé s rôzným výkonom a rôznou podporou pre protokoly. Z tohto dôvodu neexistuje štandartný protokol a zabezpečenie pre IoT sieť [25]. Preto sa v IoT sieťach používa niekoľko štandartov a technólogii. Väčšina domácnosti používa Wi-Fi ktorá je postavená na štandarte IEEE 802.11. V prípade, že zariadenia nepotrebujú posielať až toľko dát a potrebujú komunikovať na väčšiu vzdialenosť, sa v posledných rokoch čoraz viac používa štandart IEEE 802.15.4 alebo proprietárne protokoly ako Z-Wave alebo Lorawan. Tieto protokoly sa začínajú objavovať čoraz viac vďaka skupinám výrobcov, ktorí sa spoločne snažia presadiť dané protokoly.

2.3.1 Protokoly na štandarte 802.15.4 a proprietarné protokoly

Z-Wave je bezdrátový proprietárny komunikačný protokol navrhnutý tak aby poskytoval spoľahlivý prenos dátovych paketov rýchlosťou do 100kbit/s. Pracuje v rozsahu 800 - 900 Mhz. Používa zmiešanú topológiu, ktorú kontroluje jeden ovládaci uzol. V praxi to znamená, že ak sú chcú dva uzly komunikovať, tak ak sú dostatočne blízko seba tak komunikujú medzi sebou priamo, inak si zariadenia nájdu uzol, na ktorý sa môžu obidve strany pripojiť a komunikujú pomocou tohto uzlu. Celkovo sieť podporuje maximálne 232 uzlov [15].

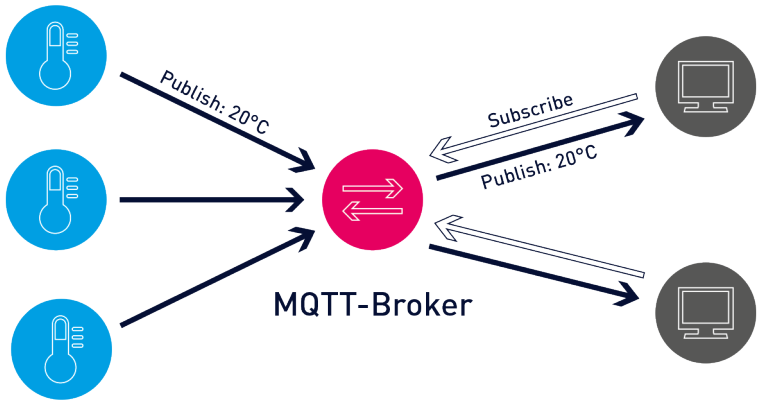


Obr. 2.8: Z-Wave architektúra, prevzaté z [19]

Zigbee je bezdrátový komunikačný protokol, ktorý na nízkolevelovú komunikáciu používa IEEE 802.15.4 štandart. Podobne ako Z-Wave používa zmiešanú topológiu a ten istý systém komunikovania. Zigbee protokol podporuje až 65 tisíc uzlov. Najčastejšie operuje na frekvencii 2.4GHz ale dokáže operovať aj na nižších frekvenciách. PAN štandart sa stará o fyzickú a linkovú vrstvu, zaťiaľ čo Zigbee poskytuje sieťovú a aplikačnú vrstvu [15].

Thread protokol vydaný v roku 2015 je priamym konkurentom pre vyššie spomenuté protokoly. Podobne ako Zigbee je postavený na štandarte 802.15.4, nad ktorým ešte ale používa 6LoWPAN protokol, ktorý umožnuje používať IPv6 adresy v jeho sieti. Tak isto používa zmiešanú topológiu a na transport sa používa UDP protokol čo umožnuje použitie napr. CoAP protokolu v aplikačnej vrstve [33].

LoRaWAN je nizkofrekvenčný proprietárny komunikačný protokol, ktorý slúži na komunikáciu na veľké vzdialenosti, zatiaľ čo ale 6LoWPAN dokáže komunikovať iba do 100m, v prípade LoRaWAN sa jedná až o vzdialenosť do 15km. Narozdiel od 6LoWPAN, ktorý používa IEEE 802.15.4 štandart, ktorý sa stará o fyzickú vrstvu, LoRaWAN používa LoRA



Obr. 2.9: Architektúra MQTT, prevzaté z [7]

protokol. Používa topológiu hviezdy, čo znamená, že zariadenia nemôžu komunikovať priamo medzi sebou [9].

2.3.2 IoT protokoly v štandarte IEEE 802.11

MQTT protokol na TCP protokoli (existuje varianta MQTT-SN, ktorá może bežať cez UDP), používa komunikáciu štýlu publish/subscribe a skláda sa z klientov a spostredkovateľa (MQTT broker). Sprostredkovateľ je server, na ktorý prichádzajú všetky správy od klientov. Tieto správy obsahuju „topic“, sprostredkovateľ ich potom preposiela konkrétnym klientom, ktorí sa prihlásili na odber toho konkrétného „topicu“.

MQTT ponúka 3 rôzne kvality služieb pre doručenie správ.

- **Maximálne jedna** — môže dôjsť k strate alebo duplicite správ. Používa sa napríklad pri senzore, ktorý posiela správy tak často, že strata jedného údaja nebude mať dopad na výsledné meranie.
- **Aspoň jedna** — správa ma garantované doručenie, ale môžu nastať duplicity. Použije sa napr. pri vyslaní signálu na zapnutie zariadenia.
- **Presňe jedna** — kde je zaručené že správa príde presne jedenkrát. Táto možnosť sa môže použiť napr. vo fakturačných systémoch, kde by duplicitné správy viedli k zlému zúčtovaniu [17].

MQTT-SN je varianta MQTT protoklu navrhnutá pre bezdrátove siete. Táto verzia ponúka efektívnejší prenos dát a spotrebu energií. Protokol nepotrebuje na jeho funkčnosť TCP/IP stack, ale stačí keď sieť poskytuje obojsmerný prenos medzi akýmkoľvek uzlom a konkrétnym uzlom (bránou), jeho prenos je taktiež možný pomocou UDP protokolu [30].

CoAP je M2M[9] protokol postavený na REST modely podobne ako HTTP, beží na UDP protokoli, používa komunikáciu typu request/response. Protokol bol navrhnutý tak aby mohol bežať na mikročipoch, ktoré majú k dispozícii iba 10kB RAM a 100kB kódového priestoru [12]. Tento protokol podporuje multicast skupiny, asynchrónne výmeny správ, je komptabilný a ľahko preložiteľný do HTTP, kedže aj jeho metódy sú podobné tým, ktoré používa HTTP ako napríklad GET. Na preklad CoAP komunikácie na HTTP sa používa CoAP proxy server. Tento umožňuje zariadeniam, ktoré nemajú dostatočné prostriedky na implementovanie IP stacku komunikovať s webovým serverom. [27].

[9] Machine to Machine

2.3.3 Ostatné protokoly používane v IoT

HTTP je pravdepodobne najznámejši protokol v aplikačnej vrstve, je používaný vo WWW od 1990. Je to bezstatový protokol, používaný hlavne na distribúciu webových stránok, ale može byť použitý aj na iné úlohy vďaka jeho rozšíreniam ako napríklad DoH[10] [14]. Dáta prenášané pomocou HTTP sú zvyčajne v textovom, XML alebo JSON formáte.

IPDC je séria protokolov, ktoré sa používali vo VoIP sieťach. IPDC výtváral interface medzi bránou medií, ktorá prepajá verejnú telefónnu sieť (PTSN) a ovládačom brány médií. IPDC bolo navrhnuté tak aby dokázalo vytvoriť spojenie, riadiť ho, kontrolovať média a prenášať signalizáciu pre prostredia, kde je logika riadenia služieb oddelená od siete. IPDC vidí bránu medií ako aplikáciu, ktorá može kontrolovať jeden až viacero fyzických zariadení. Taktiež može byť tento protokol použitý na ovladánie iných zariadení ako oznamovacie servery a servery slúžiace na digitálne prepojenie. Používajú sa dva rôzne typy správ: správy iba s hlavičkou a správy, ktoré okrem hlavičky obsahujú páry Atribút-Hodnota. Správy iba s hlavičkou sa používaju iba na potrvdzovanie [32]. V sieti je tento protokol používany zariadeniami, ktoré sú postavené na Tuya platforme.

DNS protokol používa UDP, beží na porte 53 a slúži na preklad doménovych mien na IP adresy a opačne. Je to decentralizovaný systém, v ktorom sa klient pýta DNS serveru či vie ip adresu k danej doméne. Ak nevie, DNS server sa postupne pýta ostatných serverov (začína od TLD v strome, ktorý obsahuje domény), či poznajú odpoveď až pokiaľ nedostane správnu ip adresu alebo poprípadne chybu, že k danej doméne neexistuje ip adresa.

MDNS protokol sa používa v malých sieťach, je to v podstate DNS v lokálnej sieti. Hlavnou výhodou MDNS je to, že systém nepotrebuje v podstate žiadnu konfiguráciu a administratívu. Domény v MDNS sa nachádzaju v TLD[11] „.local“. MDNS ma vyhradénu IPv4 adresu 224.0.0.251 z rozsahu pre multicast a pre IPv6 je to f02:fb.

[10] DNS over HTTP
[11] Top Level Domain

Kapitola 3

Bezpečnosť

To čo robí internetom tak dôležitým prvkom je prepojenie siete a fakt, že sa môžeme pripojiť do inej siete bez nutného fyzického prístupu. Zároveň ho to robí aj nebezpečným, vzhľadom na to, že potencionálny útočník sa môže nachádzať na hocijakom mieste. IoT siete nie sú veľmi rozdielne v porovnaní s klasickými počítačovými sietami. Z tohoto dôvodu čelia tieto siete takmer všetkým hrozbám ktorým čelí klasická sieť [21]. V tejto sekcii budú predstavené známe bezpečnostné hrozby, bude vykonaná analýza komunikácie na to aké informácie posielajú zariadenia na cloud a či sú dostatočné zabezpečené, MITM útok, ktorý overuje či zabezpečené informácie pomocou TLS sú skutočne zabezpečené a bude zanalyzované ako sa zariadenia správajú v prípade výpadku systému.

Internetovú bezpečnosť môžeme rozdeliť na tri základné prvky, na ktoré sa treba zamerať pri chápaní bezpečnosti [20].

Dôvernost spočíva v zachovaní súkromia údajov, takže iba oprávnení používateľia majú prístup k údajom. Na zachovanie dôvernosti sa používa kryptografia.

Autentifikácia overuje či s údajmi nebolo manipulované niekým iným a preukazuje, že údaje ktoré boli prijaté boli odoslané skutočne od toho zariadenia s ktorým chceme komunikovať.

Prístup k údajom a komunikácii iba oprávneným používateľom, ide o zabezpečenie infraštruktúry a zdrojov, aby nemali k nim prístup neopravnený užívatelia a aby nebolo bránené pristupovať autorizovaným uživateľom. [20].

3.1 Bezpečnosť v domácej IoT sieti

Aj keď je Smart Home veľmi odlišné prostredie, samotná IoT sieť je podobná klasickej počítačovej sieti takže hrozby v klasických sieťach sa vyskytujú taktiež v IoT sieťach.

Hrozby dôvernosti sú výsledkom nechceného zverejnenia citlivých informácií. Aj zdanlivo neškodné údaje, ako napríklad vnútorná teplota domu, spolu so znalosťami prevádzkových parametrov klimatizačného systému by sa mohli použiť na určenie či je dom obývaný alebo nie, tieto informácie by mohli pomôc útočnikovi sa napr. vlámať do domu. Strata dôvernosti vecí, napr kľúče a heslá povedú k neoprávneným hrozbám prístupu do systému [20].

Hrozby autentifikácie môžu viesť k manipulácií so snímacími alebo kontrolnými informáciami. Napríklad neautentizované výstrahy o stave systému môžu zmiasť domového

kontrolóra aby si myslel, že nastala mimoriadna situácia a umožní použiť núdzový východ, umožnujúci nezakonný vstup do domu [20].

Prístupové hrozby sú pravdepodobne najväčšie hrozby. Neoprávnený prístup k radiacemu systému, primárne na úrovni administrátora spôsobuje, že celý systém je v ohrozený. Táto hrozba môže nastať pri zlom spravovaní hesiel alebo pri pripojení neautorizovaných zariadení do siete. Aj keď nie je možné získať kontrolu nad systémom, neoprávnené pripojenie k sieti môže zabrať šírku pásma siete alebo viesť k odmietnutiu služby oprávneným používateľom. Pretože veľa inteligentných domov používa zariadenia, ktoré môžu byť napájané z batérie a bezdrôtovo prepojené do siete s nízkym výkonom, sieť do ktorej putujú záplavy požiadaviek môže viesť k DoS (Denial of Service) útoku [20].

3.2 Bezpečnostné hrozby v IoT

IoT siete obsahujú širokú škálu zariadení, na ktorých sa môžu bezpečnostné hrozby prejaviť na rôznych úrovniach. Nižšie sa zameriavame na hrozby, ktoré sa týkaju hlavne štandartu 802.11. O hrozbách v IoT sieťach, ktoré používaju siete s dlhým dosahom (Lorawan,6LoWPAN) sa dozviete viacej tu [13, 18].

3.2.1 Fyzické hrozby

Sú hrozby pri ktorých útočník nevyužuje komunikáciu na samotnom zaridadení ale skôr fyzický pristup k zariadeniu, tieto hrozby nemusí vyložene ovplyvnovať samotné zariadenie ale môže aj prostredie v akom sa zariadene nachádza.

Nezabezpečný prístup

Nezabezpečný prístup ku zariadeniu a jeho komponentom dáva útočníkovi možnosť získať podrobnejšie informácie o zariadení, prístup ku debugovaciemu interface pomocou ktorého môže zmazať firmware na zariadení a tak ho znefunčniť alebo nahrať vlastný firmware a tak prebrať kontrolu nad zariadením.

Prerušenie napájania

Napríklad pomocou výpadku elektrického prúdu je možné odstaviť zariadenia, ktoré sú závisle na napájani z externého zdroja a tým zabrániť funkčnosti zariadenia alebo prípadne prepnutia zariadenia do iného režimu. Zaradenia, ktoré používaju na napájanie batériu by nemohli nájisť AP, na ktorý boli pôvodne pripojení a mohli by sa automaticky prepojiť do párovacieho režimu, ktorý by útočnik mohol využiť na spárovanie z jeho systémom a tak prevziať kontrolu nad zariadením.

3.2.2 Hrozby na nižších vrstvách OSI modelu

Sú hrozby pri ktorých sa využívaju nižšie vrstvy OSI modelu, pri väčšinu týchto útokov je nutné aby sa útočník nachádal blízko sieti alebo aby mal dočasný prístup ku sieti čo by mohol neskôr použiť na vykonanie útoku.

Rušivý útok

Je typ DoS útoku, ktorý sa nezamierava na konkrétny protokol alebo komunikáciu, ale zameriava sa na celkové frekvenčné pásmo, ktoré zahlcuje svojím vysielaním a tak neumožnuje

komunikovať zariadeniam v sieti. Narozdiel od ostatných DoS útokov, pri ktorých je výsledkom zahltenie jednej siete alebo zariadenia, je možné týmto útokom zahltiť viacero sietí naraz. Týmto zahltením dokáže útočník jednoducho znefunkčniť sieť čo preukázal Guevara Noubir vo svojej práci kde sa zameriaval na nízko energitcké DoS útoky [24].

Zabránenie spánku

Tento útok sa zameriava na zariadenia, ktoré sú napájané z batérie, neumožnením zariadeniu prístup ku „spánku“. Zariadenie môže rýchlo vyčerpať batériu a tým sa minimalizuje čas, počas ktorého je zariadenie funkčné. Tentot typ útoku je ťažké detekovať kvôli ťažkému rozoznaniu medzi skutočnými požiadavkami a tými ktoré majú iba zabrániť spánku u zariadenia [11].

ARP spoofing

Pomocou tohto útoku sa dá vykonať MITM útok ale aj DoS útok. Útočník využíva ARP protokol, pomocou ktorého sa posielajú ARP správy, vďaka ktorým sa snaží priradiť MAC adresu útočnika ku validnej IP adrese nejakého užívateľa v sieti. Často sa jedná o bránu čo spôsobí, že všetka komunikácia, ktorá bude smerovaná mimo lokálnu sieť sa pošle útočníkovi, ten potom môže preposlať komunikáciu (MITM útok) alebo ju proste zahodiť (DoS) [13].

Deautentifikačný útok

Je to typ DoS útoku, ktorý využíva, že manažment rámce v IEEE 802.11 štandartne nie sú zabezpečené. Útočník takto dokáže prečítať tieto rámce a pomocou získaných informácií vytvoriť vlastné rámce, ktoré vyzerajú ako keby pochádzali od obete útoku. Na tento útok sa využíva deautentifikačný rámec, tento sa posiela na AP v prípade, že klient chce ukončiť spojenie. Útočník posiela rámce, ktoré vyzerajú ako keby pochádzajú od klienta ku danému AP, ktorý po prijatí ukončí spojenie [23].

3.2.3 Hrozby na vyšších vrstviach OSI modelu

Narozdiel od predchádzajucích hrozieb pri ktorých bol často nutný fyzický pristup ku sieti a hrozby boli iba v lokálnej sieti, je možné niektoré s týchto útokov vykonať aj priamo na automatizačný systém.

Syn flood

Zatiaľ čo predchádzajúce útoky sa zameriavali skôr na samotné zariadenia, tento typ útoku sa naopak zameriava na automatizačný systém. Pri ktorom sa klient pripojí na server, ktorý mu príradí port, na ktorom bude komunikovať a očakáva potvrdenie od klienta, ktoré ale nikdy nepríde. Takto je port na nejakú dobu zarezervovaný tomuto spojeniu, zatiaľ čo utočník posiela dalšie žiadosti o pripojenie, ktorým budú priradené dalšie porty. Takto útočník zahltí všetky otvorené porty na serveri a ten už nebude mať porty pre legitímne pripojenia [28].

HTTP flood

Je DoS útok, ktorý sa podobá klasickému prístupu na web server, kde útočník posiela opakovane na server HTTP žiadosti, čo ma za výsledok zahltenie servera, ktorý už nemá prostriedky odpovedať na legitímne požiadavky.

Autentifikácia a zabezpečenie komunikačných protokolov

Protokoly ako MQTT, CoAP, HTTP neobsahujú v základe zabezpečenie samotného protokolu. Je na samotnom výrobcovi zariadenia či implementuje zabezpečenie pomocou TLS alebo DTLS v prípade, že protokol používa UDP. To či je vôbec možné implementovať dané zabezpečenia ale záleži aj na výkone zariadenia, kedže sú tieto protokoly náročné na výkon, ktorého majú IoT zariadenia už aj tak málo. V prípade, že sa neimplementujú tieto zabezpečenia, sú posielané dáta ľahko viditeľné a môžu byť použité k ďalším útokom.

MITM útok na TLS

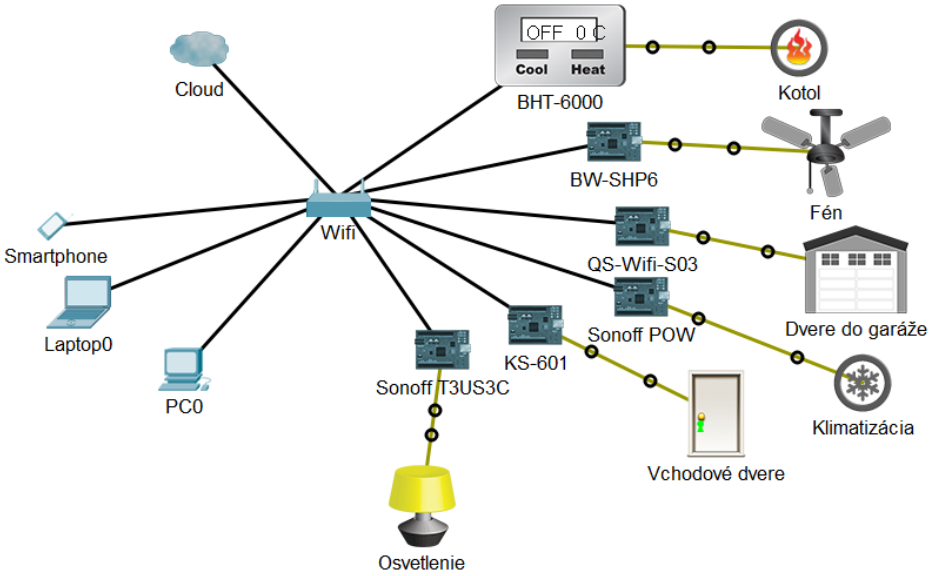
Tento útok sa používa v prípade, že sú zariadenia zabezpečene pomocou TLS. Útočník odchýtava inicializáciu komunikácie od klienta tzv. „handshake“ a sám inicializuje komunikáciu so serverom. Takto dokáže dešifrovať obidve strany komunikácie. V prípade, že klient neoveruje certifikát, ktorý dostal od serveru, komunikácia prebieha bez toho aby klient alebo server vedeli, že sa medzi nimi nachádza útočník.

Nezabezpečný firmware

Nedostatočne zabezpečený/chybový firmware ponúka možnosť útočníkovi preniknúť do zariadenia/siete. V prípade, že firmware nedostáva aktualizácie, ktoré zabezpečujú objavené hrozby, môže ostať zariadenie nezabezpečené počas celej svojej životnosti, tým ponúka útočnikovi veľa možností ako zneužiť hrozby, ktorých počet sa bude počas životnosti zariadenia zvyšovať. Tak isto aktualizácie zariadení OTA, v prípade že nie sú zabezpečené, ponúkaju útočníkovi možnosť prepísať samotný firmware na zariadení a tým prebrať kontrolu nad zariadením.

3.2.4 Analýza komunikácie

Analýza zariadení v IoT sieti bola vykonaná na každom zariadení zvlášť. Zariadenia boli ovládane cez Wi-Fi pomocou klientských aplikácii oficiálnych systémov od výrobcov zariadení. Komunikácia bola zachytená pomocou nástroja Wireshark. Zariadenia používali klasické protokoly ako DNS, MDNS na získanie ip adries serverov a DHCP protokol na vypýtanie adresy od servera. Na komunikovanie používali rôzne protokoly a zabezpečenie komunikácie sa líšilo podľa zariadenia.



Obr. 3.1: Architektúra siete

Tabuľka nižšie 3.1 zobrazuje komunikačné protokoly, ktoré zariadenia používali na komunikovanie s automatizačným systémom a smartphonom.

IoT Zariadenie	Lokálna sieť	Internet	Cielová adresa
BHT-6000	IPDC	TLS1.2	3.121.131.36 (a3.tuyaeu.com)
			52.57.38.165 (m2.tuyaeu.com)
QS-Wifi-S03	IPDC	TLS 1.2	3.121.131.36 (a3.tuyaeu.com)
			3.121.210.75 (m2.tuyaeu.com)
BW-SHP6	IPDC	TLS 1.2	3.121.210.75 (m2.tuyaeu.com)
KS-601	IPDC	MQTT,HTTP	3.121.250.46 (mq.gw.tuyaeu.com)
			3.122.186.171 (a.gw.tuyaeu.com)
			3.122.68.187 (a.gw.tuyaeu.com)
Sonoff POW	HTTP	TLS 1.2	3.122.122.135
			52.57.6.180 (eu-disp.coolkit.cc)
Sonoff T3US3C	HTTP	TLS 1.2	52.57.6.180 (eu-disp.coolkit.cc)
			52.57.118.192 (eu-api.coolkit.cc)

Tabuľka 3.1: Zariadenia a ich komunikačné protokoly

BHT-6000-Termostat, QS-Wifi-S03 Vypínač, BW-SHP6 zásuvka

Vyššie uvedené zariadenia mali ten istý princíp komunikácie. Zariadenia v lokálnej sieti komunikovali so smartphonom pomocou nešifrovaného IPDC protokolu. Komunikácia s automatizačným systémom bola šifrovaná pomocou TLS1.2. Aj keď nešifrovaná IPDC komunikácia umožnuje čítať dáta, ktoré sú posielané, dáta nie sú v čitateľnej forme a komunikácia s automatizačným systémom je šifrovaná.

```
▼ IP Device Control (SS7 over IP)
    N(r): 157
    N(s): 248
    Payload length: 517
    Protocol ID: 0x2c
    Transaction ID size: 61
    Transaction ID: 4480af00
    Message code: Unknown (0x00aa)
  ▼ IPDC tags
      0x55: UNDEFINED
      data trailing end of tags
```

Obr. 3.2: Ukážka formátu IPDC packetu z nahranej komunikácie

KS-601 vypínač

Zariadenie zase komunikuje so smartphonom lokálne pomocou IPDC protokolu. Dáta smerujúce mimo lokálnu sieť nie sú šifrované, vďaka tomu je vidieť, že komunikácia prebieha na HTTP a MQTT protokole. Zariadenie za prihlásí na MQTT server a hneď potom pošle *Subscribe Request*, pripojí sa na topic s menom *smart/device/in/<id>*, zatiaľ čo všetky správy, ktoré posiela serveru majú topic s menom *smart/device/out/<id>*. Pomocou will message povie zariadenie serveru aku správu a do akého topicu poslať, v prípade že sa zariadenie odpojí.

```
▶ Connect Flags: 0xce, User Name Flag, Password Flag, QoS Level: At least once delivery
  Keep Alive: 60
  Client ID Length: 20
  Client ID: 3807107868c63ae4252d
  Will Topic Length: 15
  Will Topic: tuya/smart/will
  Will Message Length: 107
  Will Message: {"clientId":"3807107868c63ae4252d","deviceType":"GATEWAY","message":"11
  User Name Length: 20
  User Name: 3807107868c63ae4252d
  Password Length: 16
  Password: b3a3a28c1d8ec06c
```

Obr. 3.3: Klient sa prihlasuje na MQTT broker

Taktiež je možné vidiet do akého topicu sa zariadenie prihlásilo, takže potencionálny útočník môže využiť túto informáciu aby dostával, spravý ktoré posiela dané zariadenie alebo aby poslal informácie do zariadenia a tým ho donútil vykonať nejakú akciu.

```
▾ MQ Telemetry Transport Protocol, Subscribe Request
  ▸ Header Flags: 0x82, Message Type: Subscribe Request, QoS Level: At least once delivery
    Msg Len: 41
    Message Identifier: 1
    Topic Length: 36
    Topic: smart/device/in/3807107868c63ae4252d
    Requested QoS: At most once delivery (Fire and Forget) (0)
```

Obr. 3.4: Klient sa prihlási k odoberaniu topicu

HTTP komunikácia obsahuje rôzne výmeny dát, od vypýtania si mien domén na ktoré by sa zariadenie malo pripojiť až po posielanie dát ako napríklad časové údaje. Tieto dáta sú kódované v neznámom formáte a serializované do *x-www-from-urlencoded*. Z HTTP komunikácie by mohol útočnik získať informácie napríklad o tom, kedy sa ma dané pripojené zariadenie zapnúť, vďaka tomu, že sa nastavené časovače posielajú na server. Taktiež by mohol získať prístup k dátam, ktoré si pýta zariadenie od serveru, kedže je v komunikácii vidieť ako pristupuje zariadenie ku serveru, aký token a aký kľúč používa na prihlásenie sa ku API.

```
▾ Request URI Query: a=s.gw.dev.pk.active&gwId=3807107868c63ae4252d&other=
    Request URI Query Parameter: a=s.gw.dev.pk.active
    Request URI Query Parameter: gwId=3807107868c63ae4252d
    Request URI Query Parameter: other={"token":"rIbAm2NN"}
    Request URI Query Parameter: t=1573424351
    Request URI Query Parameter: v=3.0
    Request URI Query Parameter: sign=949c56b084663e9e7dcfc63e5286aa94
```

Obr. 3.5: Parametre HTTP požiadavku

```
▾ Member Key: date
    String value: 20191110
    Key: date
▾ Member Key: dps
  ▸ Object
    Key: dps
▾ Member Key: loops
    Number value: 0
    Key: loops
▾ Member Key: time
    String value: 22:20
    Key: time
```

Obr. 3.6: Server posiela serializované dáta v JSON formáte klientovi

Sonoff T3US3C vypínač a Sonoff Pow vypínač s meraním spotreby energie

Tieto zariadenia si vymenia niekoľko správ so smartphonom pri inicializácii zariadenia. Kedže je táto komunikácia nešifrována je možné zachytiť citlivé informácie ako *ID* IoT zariadenia, jeho *api kľúč* na pripojenie k API, meno servera a port, na ktorý sa má pripojiť a dokonca aj meno a heslo na pripojenie k Wi-fi sieti. Všetky tieto informácie sú posielané v json formáte, v textovej forme. Zariadenie a smartphone sa pripoja na vzdialený server, ktorý preposiela komunikáciu medzi nimi. Táto komunikácia je ale šifrovaná pomocou TLS1.2 takže zase nie je možné určiť o aký typ komunikácie ide.

```
▾ Member Key: deviceid
    String value: 10004703da
    Key: deviceid
▾ Member Key: apikey
    String value: 94b84df9-2bf2-446b-aa61-7f9f2996dc4b
    Key: apikey
```

Obr. 3.7: Komunikácia medzi smarthphone a Sonoff zariadením

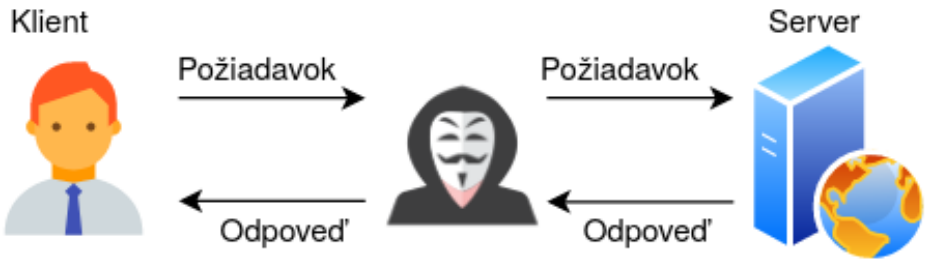
V prípade vypínača Sonoff T3US3C je ešte možná lokálna komunikácia, ak by došlo k výpadku pripojenia na cloud. V tomto móde sa zariadenie stáva serverom, na ktorý sa pripojí smarthphone. Táto komunikácia prebieha síce v nešifrovanom HTTP protokole ale dáta, ktoré sa preposielajú sú šifrované pomocou neznámeho šifrovacieho algoritmu.

```
▾ Member Key: deviceid
    String value: 100094bb85
    Key: deviceid
▾ Member Key: selfApikey
    String value: 272e1460-aa5e-43ab-ba47-614e09bd1345
    Key: selfApikey
▸ Member Key: iv
▸ Member Key: encrypt
▾ Member Key: data
    String value: Narlx49F2WLoEHLrLukjzoC+TMYeVz8DNcm26FPLnrlILlxlVDsXo6EXHwMzV52c
    Key: data
```

Obr. 3.8: Komunikácia v lokálnom móde medzi smarthphone a Sonoff zariadením

3.2.5 Simulácia MITM útoku

Pri tomto type útoku útočník preposiela alebo mení komunikáciu medzi klientom a serverom s tým, že obidve strany komunikácie si myslia, že komunikujú priamo medzi sebou.



Obr. 3.9: MITM útok

V prípade nešifrovanej komunikácie stačí útočníkovi iba presmerovať komunikáciu, v prípade komunikácie šifrovanej pomocou TLS, útočník musí zachytiť inicializáciu komunikácie tzv. handshake. Počas tohto procesu si členovia komunikácie vymieňajú rôzne informácie. Útočníka konkrétne zaujímajú iba SNI[1] pomocou ktorého klient identifikuje meno serveru na ktorý sa chce pripojiť, a digitálni certifikát serveru pomocou ktorého klient dokáže autentifikovať server. Vďaka SNI útočník dokáže požiadať o konkrétny certifikát pomocou

[1]Server Name Indication - umožnuje mať niekoľko domén na jednej IP adrese

ktorého vytvorí vlastný podvrhnutý certifikát, ktorý pošle klientovi. Aby klient dokázal overiť či nie je certifikát podvrhnutý, je elektronicky podípísaný dôveryhodnou certifikačnou autoritou. V prípade že útočník vytvorí certifikát, ktorý vyzerá, že prichádza od servera, a certifikát nebude podpísany certifikačnou autoritou tak klient preruší spojenie. Nie všetky IoT zariadenia ale overujú certifikát dodaný serverom [29].

Na vykonanie útoku sa používa proxy, na našu konkrétnu simuláciu bol použitý nástroj *mitmproxy*. Tento nástroj ponúka jednoduchú inštaláciu CA na zariadenia, ktoré to podporujú, generuje certifikáty za behu alebo sa dajú použiť dopredu vygenerované certifikáty. Kedže zariadenie nejde jednoducho pripojiť na proxy, nebolo možné použiť klasickú proxy kedže pri nej zariadenie treba nastaviť aby sa pripojilo na proxy, bolo treba teda použiť tzv. *transparent* proxy. V takom prípade sa zvyčajne routri presmeruje komunikácia na port, na ktorom daná proxy počúva, bez toho aby zariadenie vedelo, že komunikácia prebieha cez proxy.

Obidve Sonoff zariadenia v sieti overujú certifikát a tým sa efektívne chránia proti tomuto konkrétnemu MITM útoku.

10.42.0.63	10.42.0.1	TCP	13405 → 443 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
10.42.0.1	10.42.0.63	TCP	443 → 13405 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 M
10.42.0.63	10.42.0.1	TLSv1.2	Client Hello
10.42.0.1	10.42.0.63	TCP	443 → 13405 [ACK] Seq=1 Ack=85 Win=64156 Len=0
10.42.0.1	10.42.0.63	TLSv1.2	Server Hello
10.42.0.1	10.42.0.63	TCP	443 → 13405 [ACK] Seq=87 Ack=85 Win=64156 Len=1460 [
10.42.0.63	10.42.0.1	TCP	13405 → 443 [ACK] Seq=85 Ack=1547 Win=5840 Len=0
10.42.0.1	10.42.0.63	TLSv1.2	Certificate, Server Hello Done
10.42.0.63	10.42.0.1	TCP	13405 → 443 [FIN, ACK] Seq=85 Ack=1999 Win=5388 Len=
10.42.0.1	10.42.0.63	TLSv1.2	Alert (Level: Fatal, Description: Handshake Failure)
10.42.0.1	10.42.0.63	TCP	443 → 13405 [FIN, ACK] Seq=2006 Ack=86 Win=64155 Len

Obr. 3.10: Ukážka komunikácie kde Sonoff overuje certifikát

V prípade Tuya-based zariadení sa nepodarilo vykonať simuláciu z dôvodu, že server s ktorým komunikovali zariadenia podporoval iba šifru `TLS_PSK_WITH_AES_128_CBC_SHA256`, mitmproxy zatiaľ ešte nepodporuje TLS s PSK[2]. To isté sa týka aj iných nástrojov ako SSLsplit alebo BurpSuite.

3.2.6 Simulácia nedostupného systému

V prípade, že je automatizačný systém pre zariadenia nedostupný, môže ísť o chybu v sieti alebo o chybu na strane automatizačného systému. Môže ísť ale aj o DoS alebo DDoS útok na automatizačný systém, zariadenia, alebo IoT sieť, na ktorej prebieha komunikácia.

Tuya

V prípade Tuya zariadení, pre ktoré nebol dostupný automatizačný systém, ovládanie pomocou aplikácie bolo ďalej funkčné vďaka ovládaniu pomocou IPDC protokolu. Časovač bol taktiež funkčný, ktorý sa dal nastaviť na klientovi aj v prípade nedostupného systému. Nebolo možné nastaviť rozvrh pravidelného vypnutia/zapnutia zariadení. Ak bol rozvrh nastavený už pred výpadkom spojenia so systémom, tak aj tak zariadenia nereagovali podľa neho a ostali v stave v akom boli pred výpadkom. Potencionálny útočník by mohol využiť túto hrozbu na deaktivavnie rôznych zariadení, ktoré by kontrolovali vypínače v sieti napr. zámok ktorý sa pravidelne zamýka na noc. Viac komplikované zariadenie ako zásuvka BW-SHP6 a termostat BHT-6000 neposkytovali namerané štatistiky ako teplotu alebo spotrebu

[2] https://github.com/mitmproxy/mitmproxy/issues/3742

elektrickej energie, a na BHT-6000 termostate sa nedali vykonať zmeny nastavenia režimov a zmeny teploty.

Sonoff

Sonoff POW zariadenie bez prístupu na cloud ostalo v podstate deaktivované, nedalo sa ovládať lokálne, ostalo v tom režime v akom bolo v momente vypadnutia spojenia. Taktiež, zariadenie neposkytovalo žiadne štatistiky o spotrebe energií. Rozvrh a časovače sa nedali nastaviť ale v prípade, že boli nastavené pred výpadkom siete, zariadenie fungovalo podla nich správne, narozdiel od Tuya zariadení sa tieto informácie nachádzajú priamo na zariadení a nie iba na cloude.

Sonoff T3US3C vypínač sa podobne ako Tuya zariadenia dal ovládať lokálne, ale neponúkal možnosť nastaviť časovač alebo rozvrh na zariadenie. Ak mal dopredu nastavené tieto funkcie tak podobne ako pri POW vypínači bol schopný sa podľa nich riadiť.

3.2.7 OTA nahratie firmware

Na ESP8266 sa firmware nahráva buď pomocou UART komunikácie alebo poprípade, že to podporuje dopredu nahratý firmware tak sa dá nahrať aj OTA. Update pomocou OTA je oproti nahratiu pomocou UART o dosť viac pohodlný a rýchlejší, kedže nepotrebujeme fyzický prístup ku zariadeniu. Toto ale môže využiť útočník na nahratie vlastného firmware v prípade, že táto funkcia nie je zabezpečená. Podobne ako open-source Arduino knihovny, ktoré ponúkaju minimálne zabezpečenie pre OTA aktualizácie [16], boli niektoré zariadenia nedostatočne zabezpečené.

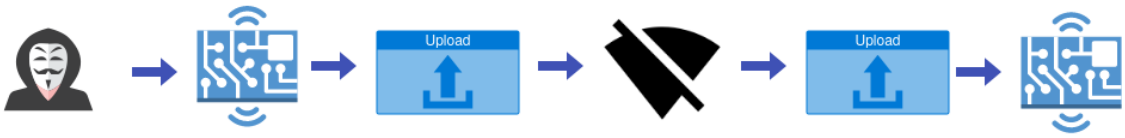
Sonoff

V prípade Sonoff zariadení existuje SONOTA[3] nástroj, ktorý ale už nefungoval na aktuálnu verziu firmware keďže Sonoff aktualizoval svoj firmware a zabezpečil svoje zariadenia proti MITM tls útoku, čo bolo overené aj nami vyššie 3.2.5. MITM tls útok umožnoval získať informácie, ktoré umožnili tento útok.

Tuya

Pre Tuya zariadenia bol použitý nástroj Tuya-convert[4] pomocou ktorého bolo možné úspešne nahrať vlastný firmware na zariadenia. Nahrať firmware takýmto spôsobom bolo možné iba keď bolo zariadenie v párovaciam režime, do ktorého sa zaradenia dostali iba fyzicky podržaním tlačidla na zariadení. Ale v prípade, že bolo zariadenie odpojené od napájania v párovaciam režime, pri opätovnom spustení sa zapne v tomto režime. Tento nástroj aj automaticky zálohuje firmware, ktorý je momentálne na zariadení. Ako by mohol toto využiť útočník je znázornené na obrázku 3.11, kde útočník fyzickým prístupom k zariadenu prepne zariadenie do párovacieho módu, nahrá škodlivý firmware[5], ktorý by vykonáva deautentifikačný útok a po skončený nahrá pôvodný firmware, ktorý predtým stiahol. Takže uživateľ si všimne iba to, že zariadenie nebolo chvíľu dostupné. V prípade, že nie je často používane by si ani nevšimol, že sa niečo so zariadením stalo.

[3] https://github.com/mirko/SonOTA
[4] https://github.com/ct-Open-Source/tuya-convert
[5] https://github.com/spacehuhn/esp8266_deauther



Obr. 3.11: Nahratie firmware pomcou OTA

3.2.8 Známe a objavené hrozby v IoT sieti

Analýzou komunikácie bolo zistené, že nie všetky zariadenia majú zabezpečenú komunikáciu a posielaju citlivé informácie o zariadeniach a samotnej sieti buď do automatizačného systému alebo do smartphone. Aj keď nebol úspešne vykonaný MITM útok u Tuya zariadení, môžeme predpokladať, že pri OTA sa používa tento útok na získanie informácií k nahratiu firmware podobne ako sa používal pri SONOTA[6].

Ochranu proti niektorým z týchto hrozieb môže zabezpečiť samotný výrobca zariadenia pomocou aktualizácii, ale samotná aktulizácia nemusí nikdy prísť. Zatiaľ, čo niektoré nové Tuya zariadenia implementujú už zmeny aby zabránili možnosti nahratia nového firmware OTA pomocou tuya-convert, konkrétne aj nové verzie BW-SHP6 [7], BW-SHP6 zásuvka nedostala žiadnu aktualizáciu, ktorá by zabránila tomuto útoku. Preto na zabezpečenie siete a zariadení je treba použiť open-source systémy, ktoré ponúkajú riešeenia na najdené hrozby.

Útoky, konkrétne hrozby	**Navrhnuté riešenia**
Zdieľanie citlivých informácií na cloud	Vlastný open-source automatizačný systém v lokálnej sieti
Nezabezpečená komunikácia	Nahratie open-source firmware a nakonfigurovanie TLS
MITM útoky	Open-source firmware, ktorý bude overovať certifikáty
OTA aktualizácie	Open-source firmware, ktorý poskytne možnosť akutualizovať zariadenie, ale zabezpečí túto možnosť autentifikáciu
DDoS útoky a nedostupný systém	VLAN, Open-source lokálny automatizačný systém, presunutie časť automatizácií priamo na komunikáciu medzi zariadeniami a priamo na zariadenia

Tabuľka 3.2: Hrozby a ich navrhnuté riešenia

[6] https://blog.nanl.de/2017/05/sonota-flashing-itead-sonoff-devices-via-original-ota-mechanism/
[7] https://github.com/ct-Open-Source/tuya-convert/issues/483

Kapitola 4

Open-source systémy

V tejto sekcii budú prestavené a porovnané rôzne open-source systémy a firmware pre zariadenia. Budu implementované do IoT siete a kde po ich testovani budú porovnané výsledky s oficialnými automatizačnými systémami.

Open-source automatizačné systémy poskytujú možnosť nahradiť oficiálne automatizačné systémy, a mať automatizačný systém na lokálnej sieti bez potreby pripojenia internetu. Medzi ich výhody patrí:

- Užívateľ má o dosť vyššiu kontrolu nad zariadeniami a samotným systémom.
- Automatizácia zariadení je funkčná aj v prípade výpadku internetového pripojenia.
- Lepšie riešia problém integrity zariadení od viacerých výrobcov pod jeden systém, zvyčajne vďaka aktívnej komunite.
- Zlepšujú bezpečnosť a spoľahlivosť už iba vďaka tomu, že sa nachádzaju v lokálnej sieti a tým odoberajú potencionálny bod zlyhania.
- Oproti oficiálnym systémom ponúkaju o dosť viac prepojení a integrací so externými systémami.

Medzi nevýhody open-source systémov patrí:

- Nutné zariadenie v lokálnej sieti, na ktorom bude systém bežat. Často sa jedná o jednodoskový počítač ako napr. Raspberry Pie.
- Keďže sa systém nachádza v lokálnej sieti, nie je možná vzdialená kontrola, nie všetky systémy ponúkaju cloud riešenie.
- Nasadenie a konfigurácia je oproti oficiálnym systémom o dosť viac komplikovanejšia.
- V prípade, že systém neponúka integráciu oficiálneho automatizačného systému je nutné nahrať custom firmware na zariadenia.

4.1 Firmware

Oficiálny firmware na zariadeniach zvyčajne podporuje integráciu len pod oficiálnym systémom, z tohto dôvodu treba nahrať custom firmware na zariadenia, vďaka ktorému ich integrujeme pod naše open-source systémy. Samotný firmware treba ale nakonfigurovať pre

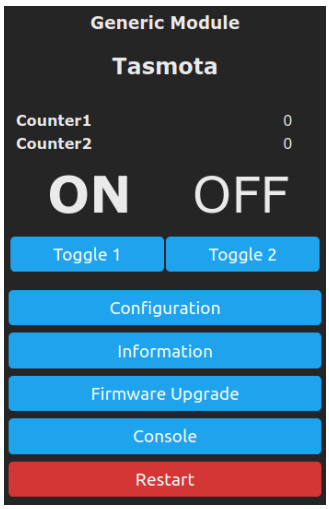
konkrétne zariadenie, preto je treba zistiť na ktoré konkrétne piny sú pripojené jednotlivé komponenty zariadenia. Z tohto dôvodu je dobré si vybrať firmware z veľkou komunitou, ktorá medzi sebou zdieľa rozdelenie vstupno/výstupných pinov zariadení. Pri výbere firmwaru bolo potrebné sa zamerať na nasledujúce vlastnosti:

- Konfigurácia: Možnosti a prebieh konfigurácie.
- Automatizácia: Aké nástroje a možnosti automatizácie ponúka firmware.
- Aktualizácie: Možnosti aktualizovania firmware.
- Zmeny v konfigurácii: Ako sa vykonávaju a konfigurujú zmeny keď je firmware už na zariadení.
- Zdieľanie konfigurácii: Ako jednoducho importovať konfigurácie vytvorené komunitou.
- Databázy zariadení: Akú veľkú datábazu konfigruácii zariadení daný firmware ponúka a na aké zariadenia sa zameriava.
- Komunikácia: Akými protokolmi umožnuje firmware komunikovať a či ponúka ich zabezpečenie.

Tasmota

Tasmota je open-source firmware pre zariadenia postavené na ESP8266 mikročipe. Tento firmware je štandartne dopredu vytvorený bez nutnosti kompilácie, kompilácia je nutná iba pri povolení niektorých funkcii ako napr. šifrovanie pomocou TLS.

Konfigurácia je možná buď pri samotnej kompiláci alebo prebieha priamo na zariadení, ktoré po spustení vytvorí AP, na ktorom ponúka GUI cez ktoré je možné konfigurovať a ovládať zariadenie. Pre samotnú konfigurácia pinov funkčnosti GPIO pinov sa využívajú šablóny, ktoré sú vo formáte JSON a ponúkaju jednoduchý spôsob vytvárania, upravovania a zdieľania konfigurácií pre rôzne typy zariadenia. Vďaka týmto šablónam ponúka veľku databázu dopredu vytvorených konfigurácií hlavne pre komerčne dostupné IoT zariadenia.



Obr. 4.1: Tasmota GUI

Automatizácie je možné vytvoriť pomocou časovačov a pravidiel, ktoré majú jednoduchú syntax: `on <trigger> do <command> endon` alebo komplikovanejšie automatizácie je možné vytvoriť pomocou skriptovacieho jazyka[1]. Zariadenia môžu medzi sebou komunikovať pomocou UDP multicast komunikácie. Ďalšie komunikačné protokoly sú MQTT a Zigbee. MQTT komunikáciu je možné zašifrovať pomocou TLS, kde Tasmota overuje certifikáty dvoma spôsobmi, buď overením certifikačnej autority alebo pomocou SHA256 odltačku vytvoreného z verejného klúča. Pre Tuya zariadenia ako BHT-6000 termostat ponúka Tasmota TuyaMCU funkciu, v ktorej sa dajú mapovať Tasmota funkcie na príkazy, ktorými sa komunikuje v

[1] https://tasmota.github.io/docs/Scripting-Language/

sériovej komunikácii medzi ESP8266 a hlavným mikrokontrolérom. Aktualizácie je možné vykonať buď pomocou nahratia firmware OTA alebo stiahnutím priamo na zariadenia z web servera.

```
{"NAME":"BW-SHP6 10A","GPIO":[158,255,56,255,0,134,0,0,131,17,132,21,0],"FLAG":0,"BASE":45}
```

Obr. 4.2: Šablóna pre BW-SHP6

ESPHome

ESPHome je open-source nástroj, ktorý pomocou konfiguračných súborov dokáže vytvoriť vlastný firmware na ESP8266 a ESP32 mikrokontroléri. ESPHome používa jednoduché YAML konfiguračne súbory, vďaka ktorým si užívateľ môže vytvoriť firmware podľa seba. Zatiaľ čo Tasmota vyžaduje externý nástroj na nahratie firmware na zariadenia, samotné ESPhome ponúka túto funkciu. ESPhome taktiež ponúka možnosť web servera, v ktorom sa ale nedá modifikovať konfigurácia. Oproti Tasmota, kde sa samotný firmware konfiguruje priamo na zariadení, sa celá konfigurácia zariadenia deje ešte pred nahratím firmware na zariadenie. V prípade zmeny je nutné prepísanie celého firmware, zatiaľ čo pri tasmote sa iba reštartuje zariadenie.

```
esphome:
  name: <NAME_OF_NODE>
  platform: ESP8266
  board: esp01_1m

wifi:
  ssid: <YOUR_SSID>
  password: <YOUR_PASSWORD>

api:

logger:

ota:
```

Obr. 4.3: ESPHome konfigurácia

Zatiaľ čo Tasmota používa na automatizáciu pravidlá tak ESPHome vytvára automatizácie priamo v YAML súboroch.

```
switch:
  - platform: gpio
    pin: GPIO3
    id: light1

binary_sensor:
  - platform: gpio
    pin: GPIO4
    on_press:
        then:
            - switch.toggle: light1
```

Výpis 4.1: Automatizácia kde senzor detekuje vstup a zapína svetlo

Pre komplikovanejšie automatizácie používa ESPHome Lambda funkcie, ktoré sa píšu v C++[2]. ESPHome podporuje komunikáciu pomocou MQTT protokolu alebo pomocou vlastného Native API TCP protokolu, ktorý kóduje informácie do optimalizovaného formátu pomocou vďaka tomu má odosť menšiu réžiu v niektorých správach než MQTT[3]. Pri tomto

[2] https://esphome.io/guides/automations.html?highlight=automation#templates-lambdas
[3] https://esphome.io/components/api.html

protokole sa systém pripája na samotné zariadenia takže nie je nutné mať server ako pri MQTT, momentálne tento protokol ale podporuje iba samotné ESPHome a Home Asisstant automatizačný systém. Tento protokol ale nieje možné zašifrovať pomocou TLS, je iba možné zabezpčiť prístup k API heslom čo ale neni dostatočné kedže komunikácie neni šifrovaná a heslo je možné odchytiť.

MQTT je možné zabezpečiť pomocou TLS, kde ESPHome overuje certifikát vytvorením SHA1 odtlačku, ktorý je ale možné s dostatočným výkonom sfalšovať. Databáza dopredu vytvorených konfigurácii obsahuje menej konfigurácii pre komerčne dostupné IoT zariadenia, viac sa zameriava na elektrické komponenty ako senzory, displeje atď.

	Tasmota	**Esphome**
Typ nástroja	Firmware	Nástroj na vytváranie firmware
Konfigurácia	GUI, príkazy	YAML súbory
Automatizácia	Pravidlá, skriptovanie	Akcie v YAML, Lambda funkcie
Zmeny v konfigurácii	Priamo na zariadení	Nutná kompilácia a nahratie nového firmware
Aktualizácie	OTA bin súbory, URL	OTA bin súbory
Zdieľanie konfigurácií	Šablony v JSON formáte	YAML konfigurácie
Databázy zariadení	Komerčne dostupné IoT zariadenia	Jednotlivé komponenty zariadení
Komunikácia	MQTT (TLS-SHA256), Zigbee, Multicast UDP	MQTT (TLS-SHA1), Native API

Tabuľka 4.1: Porovnanie Tasmota a ESPhome firmware

4.2 Open-source systémy

Tasmota a ESPhome firmware nám umožnujú kontrolovať zariadenie, ale v prípade že je nutné kontrolovať viacero zariadení a skutočne zautomatizovať domácnosť je potreba mať automatizačný systém. Pri výbere automatizačného systému bolo potrebné sa zamerať na nasledujúce vlastnosti:

- Správa systému: vytváranie užívateľov, ich autentifikácia a zdieľanie kontroly nad zariadeniami medzi nimi
- Konfigurácia: akým spôsobom prebieha konfigurácie systému
- Aumatizácia: pomocou akých nástrojov sa vytváraju automatizácie a aké komplikované automatizácie je možné vytvoriť
- Komunikácia: aké protokoly a ich zabezpečenie systém ponúka
- Integrácie systémov: akú veľkú databázu integrácií ponúka systém
- Vzdialený prístup: či systém ponúka cloud službu alebo iný spôsob ako pristúpiť k lokálnemu systému

OpenHAB

OpenHAB je open-source automatizačný systém napísany v Jave, je podporovaný pre veľkú škálu systémov od klasických ako Windows alebo MacOS až po ARM jednodoskové počítače alebo Synology NAS server. OpenHAB ponúka balíky, ktoré ponúkajú rôzne funkcie pre užívateľa. Podľa toho čo užívateľ potrebuje, OpenHAB ponúka v balíkoch rôzne typy UI a rozšírení. Konfigurácia sa nastavuje v PaperUI, v ktorom sa ale nedá všetko nastaviť pomocou GUI, niektoré funckie sa musia nakonfigurovať pomocou konfiguračných súborov. OpenHAB ponúka širokú škálu rozšírení (addonov), ktoré pridávajú ďalšie funkcionality a integrácie do systému. Automatizácia prebieha pomocou pravidiel, ktorých syntax je postavená na *Xbase* a podobná *Xtend*. Od verzie 2.4 OpenHAB ponúka možnosť vytvárať pravidlá v grafickom editore, táto funkcia ale neni spoľahlivá. Komunikácia so zariadeniami je podporovaná pre MQTT, Z-Wave a ZigBee protokoly. Pre vzdialený prístup je nutné použiť VPN alebo sprievodnú OpenHAB Cloud službu, ktorá poskytuje bezplatný vzdialený prístup. Taktiež ponúka sprievodnú aplikáciu pre systém Android a IoS. Oproti oficiánlnym automatizačným systémom tu chýba možnosť vytvorenia viacerých užívateľov a zdieľanie automatizácie a zariadení medzi nimi. Jednoduché pravidlá v OpenHAB môžu mať takýto formát:

```
rule Lights
when
    LightSwitch1 changed
then
    sendCommand(LightSwitch2,ON)
end
```

Výpis 4.2: Jednoduché pravidlo kde zautomatizujeme spúštanie viacerých svetiel

Home Assistant

Home assistant (neskôr už iba HA) je open-source automatizačný systém napísany v Pythone, ktorý je optimalizovaný pre jednodoskové počítače ako napríklad Raspeberry Pi[4]. Systém poskytuje jednoduchý spôsob pre aktualizovanie systému a vytvárania snapshotov pre uloženie momentálnej konfigurácie. Aj keď sa časť systému dá nakonfigurovať podobne ako v OpenHAB v GUI, väčšina konfigurácie sa vykonáva podobne ako v ESPhome pomocou YAML konfiguračných súborov. HA na rozdiel od OpenHAB ponúka systém pre manažovanie užívateľov a ich autentifikáciu, dokonca ponúka aj modul pre viacfaktorovú autentifikáciu. Systém podporuje protokoly ako MQTT, Z-Wave, Zigbee a taktiež Native API protokol od ESPhome, ktorý podporuje v základe bez nutnosti niečo konfigurovať alebo inštalovať rozšírenia. HA automaticky detekuje všetky zariadenia v sieti, ktoré používajú tento protokol a užívateľ ich jedným kliknutím pridá do systému.

Automatizácia sa v HA dá konfigurovať rôznymi spôsobmi:

- Podobne ako v ESPhome pomocou YAML konfiguračných súborov.

```
- alias: 'Rule 2 - Away Mode'
    trigger:
        plafrom:state
        to: 'not_home'
    action:
        service: light.turn_off
        entity_id: all
```

Výpis 4.3: Automatizácia, ktorá vypne svetlá v prípade že sa nikto nenachádza doma

- Pomocou grafického editoru, HA automaticky vytvárá YAML konfiguračný súbor.
- Automatickým konvertovaním textu, ktorý zadá užívateľ. Systém sa z neho snaží vyčítať čo presne užívateľ chcel zautomatizovať. Táto funkcia sa dá použiť iba na jednoduché automatizácie a není spoľahlivá, v našom prípade sa nepodarilo vytvoriť ani jednoduchú automatizáciu.
- V prípade veľmi komplikovaných automatizácií sa dá použiť sandboxové prostredie AppDaemon. Je to viacvláknové Python exekučné prostredie na písanie komplikovaných automatizácii pre HA.

Tak isto ako OpenHAB ponúka HA širokú škálu rozšírení, medzi ne patria aj špecifické rozšírenia pre vyššie spomenutý firmware. Rozšírenie TasmoAdmin, ktorý nám umožnuje manažovať a zauktualizovať všetky Tasmota zariadenia naraz, či rozšírenie ESPhome rozšírenie, ktoré integruje ESPhome do HA a umožnuje vytvárať a aktualizovať ESPhome firmware priamo v HA. Z oficiálnych systémov sa dá prepojiť iba s Tuya systémom. Podobne ako OpenHAB, HA ponúka sprievodný cloud systém pre vzdialený prístup (v prípade HA sa ale nejedná o bezplatný systém), ale taktiež DuckDNS alebo Tor rozšírenia, ktoré taktiež umožnujú vzdialený prístup k lokálnej inštalácii. Tak isto ponúka sprievodné aplikácie pre systém IoS a Android.

[4]https://www.home-assistant.io/hassio/

	Home Assistant	**OpenHAB**
Správa systému	Ponúka manažment užívateľov, ich autentifikáciu, multifaktorovú autentifikáciu	Neponúka manažment užívateľov a ich autentifikáciu
Konfigurácia systému	YAML súbory,GUI	PaperUI, konfiguračné súbory
Automatizácia	YAML súbory, Grafický editor, konvertovanie textu (nespoľahlivé), NodeRed, AppDaemon (Python)	Pravidlá postavené na Xtend a Xbase, Grafický editor (nespoľahlivé), NodeRed
Komunikácia	MQTT, Zigbee, Z-Wave, ESPhome API	MQTT, Zigbee, Z-Wave
Integrácie a rozšírenia	Rozšírenia pre špecifický firmware	Celkovo viac integrácií ale menej využitelných ako pri HA
Cloud	30-dňová skúšobná verzia, DuckDNS a Tor rozšírenia	Zadarmo ale neobsahuje podporu

Tabuľka 4.2: Porovnanie Home Assistant a OpenHAB

4.3 Implementácia v našej IoT sieti

Pre našu implementáciu bolo rozhodnuté použiť Tasmota firmware pre všetky zariadenia okrem BHT-6000, na ktoré je síce možné nahrať Tasmotu a používať TuyaMCU funkciu na komunikáciu s mikrokontrolérom, ktorý ovláda samotné zariadenia, ale Tasmota momentálne neponúka funkcie pre termostat, takže ovládanie zariadenia nebolo plnohodnotné a niektoré funkcie by sa nedali ovládať. Z tohto dôvodu bol použitý špecifický Wthermostat[5] firmware pre toto zariadenie. Zatiaľ čo tento firmware ponúka plnú funkcionalitu, ktorú ponúka aj oficiálny firmware, a dá sa ovládať pomocou MQTT, neponúka šifrovanie MQTT alebo HTTP ku prístupu na web server. Taktiež sa nedá zaheslovať prístup ku web serveru alebo vypnúť web server ako je to možné pri Tasmota. Ako automatizačný systém bol vybraný HA vďaka ponuke použiteľných rozšírení a bezpečnostných opatrení. Pri vyriešení objavených hrozieb a konfigurácii open-source bolo postupované v tomto poradí:

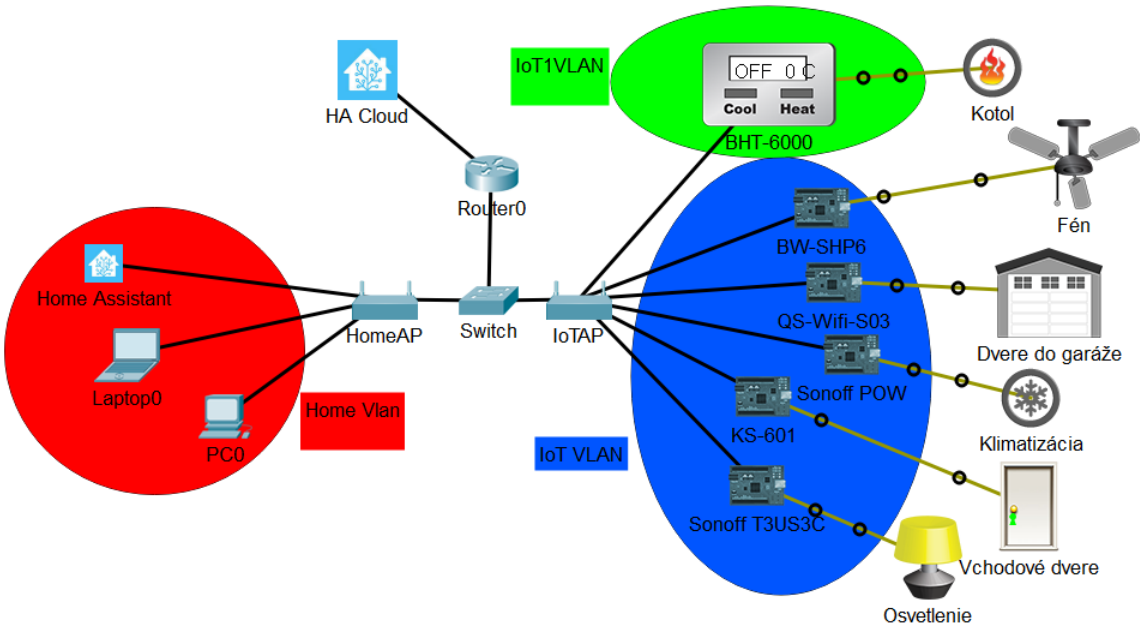
1. Návrh siete: rozdelenie zariadení do špecifických VLAN a konfigurácia firewall
2. Inštalácia a konfigurácia open-source systému
3. Nahratie firmware na zariadenia a konfigurácia pod systém
4. Testovanie a porovnanie s oficialnými automatizačnými systémami

Návrh siete

Zatial čo IoT zariadenia, ktoré používajú Tasmotu boli priradené do spoločnej VLAN, BHT-6000 bolo priradené do sepáratnej VLAN kedže používa iný firmware, ktorý nie je zabezpečený dostatočne. Toto taktiež umožní, že v prípade že sa na zariadenie nahrá pôvodný firmware, bude možné povoliť komunikáciu s oficiálnym systémom bez vystavenia ostatných IoT zariadení internetu. Ostatné zariadenia spolu s HA inštaláciou boli priradené v jednej

[5] https://github.com/klausahrenberg/WThermostatBeca/tree/master/WThermostat

sieti kvôli jednoduchej integrácii zariadení v sieti pod automatizáciu (HA môže napríklad pomocou ping zariadení určiť kto sa náchadza v dome).



Obr. 4.4: Rozdelenie siete do VLAN

Pri samotnej implementácii bolo nutné nastaviť rozsah IP adries pre špecifické VLANi a nastaviť firewall pravidlá, ktoré umožnia/eliminujú komunikáciu, ktorá buď prichádza alebo vchádzať do VLAN. V tejto implementácii boli povolené porty 1883, 8883 pre MQTT a MQTT, ktoré je zašifrované pomocou TLS. Port 123 bol povolený pre NTP protokol, pomocou ktorého synchrozujeme čas na našich zariadeniach. Pri Tasmota zariadeniach bol otvorený tento port taktiež pre pripojenie na internet, v prípade výpadku/poruchy HA tak zariadenia budú zariadenia stále schopné zosynchronizovať svoj čas, čo je nutné pri automatizáciach.

Zdroj	**Destinácia**	**Port**	**Akcia**
IoTVLAN	HomeVLAN	8883,123	Povolené
IoT1VLAN	HomeVLAN	1883,123	Povolené
IotVLAN	Internet	123	Povolené
HomeVLAN	IoTVlan	8883,123	Povolené
HomeVLAN	IoT1Vlan	1883,123	Povolené
IoT1VLAN	ANY	ANY	Zakázané
IoTVLAN	ANY	ANY	Zakázané

Tabuľka 4.3: Povolená komunikácia medzi VLAN

4.3.1 Konfigurácia Home Assistant

Inštalácia HA bola jednoduchá vďaka image, ktoré HA ponúka. V základe HA neobsahuje príliš veľa funkcií. Tieto funkcie boli neskôr expandované pomocou inštalácie rozšírení. HA v základe dokáže komunikovať iba s ESPhome zariadeniami pomocou jeho NativeAPI a samotný systém obsahuje niekoľko bezpečnostných hrozieb ktoré bolo nutné odstrániť:

- Nezabezpečená HTTP komunikácia — útočník môže získať prihlasovacie údaje do systému.
- Útok hrubou silou — v prípade útoku hrubou silou neboli implementované žiadne opatrenia ako zabrániť tomuto útoku, takže potencionálny útočník by mohol neobmedzene skúšať prihlasovanie do systému.

Konfigurácia systému prebiehala v nasledujúcich bodoch kde ako prvé boli eliminované hrozby, ktoré systém mal a postupne boli pridávané ďalšie funkcie do systému inštalovaním rozšírení:

- Zabezpečenie šifrovania pomocou TLS - pomocou openssl boli vygeneroané nami podpísane certifikáty, ktoré sa vložili do systému do priečinku */ssl/*. Do *configuration.yaml* sme do sekcie *http* vložili cestu k certikátom. HA automaticky detekuje certifikáty a zabezpečí pomocou nich komunikáciu.
- Na eliminovanie pokusov o útoky hrubou silou bol nastavený parameter *ip_ban_enabled*[6], ktorý automaticky odmietne pokusy o prihlásenie z ip adresy z ktorej prekročil dopredu daný povolený počet pokusov o prihlásenie do systému.

```
http:
  ssl_certificate: /ssl/fullchain.pem
  ssl_key: /ssl/privkey.pem
  ip_ban_enabled: True
  login_attempts_threshold: 3
```

Obr. 4.5: Konfigurácia v configuration.yaml

- Vzdialený prístup bol zabezpečený pomocou Tor rozšírenia, ktorý umožní pristupovať ku HA ako „onion“ stránke, aj keď samotný Tor neponúka end-to-end šifrovanie, tak Tor služba pre skrývanie služieb[7] zbezpečí end-to-end šifrovanie. Výhoda použitia tohto rozšírenia je to že nebolo potrebné otvoriť žiadne porty na routri alebo nastaviť TLS certifikáty. Bol nastavený iba port na ktorý sa bude pripájať užívateľ, v logu rozšírenia sa objavila doména na ktorú sa potom pripájalo pomocou Tor prehliadača.

```
[14:13:09] INFO: ------------------------------------------------------------
[14:13:09] INFO: Your Home Assistant instance is available on Tor!
[14:13:09] INFO: Address: defk5viwasi2fby7.onion
[14:13:09] INFO: ------------------------------------------------------------
```

Obr. 4.6: Tor log

[6] https://www.home-assistant.io/integrations/http/
[7] https://2019.www.torproject.org/docs/onion-services

- Pre ďalšiu vrstvu zabezpečenia pri autentifikácii do systému bola pridaná multifaktorová autentifikácia. K tomuto bolo nutné aplikáciu ako Google authentificator alebo Authy a implementácia tejto služby bola vykonaná pomoco TOTP[8] algoritmu. Po spárovaní sa každých 30 sekúnd generuje šest číselne heslo ktoré treba zadať do HA pre úspešné príhlasenie. Táto služba bola dôležitá hlavne pri používaní vzdialeného prístupu pomocou rozšírenia Tor alebo DuckDNS, kde domény na prihlásanie do systému sú verejné dostupné a umožnujú ľubovoľný prístup.

```
homeassistant:
  auth_mfa_modules:
    - type: totp
```

Obr. 4.7: Konfigurácia pre spustenie viacfaktorovej autentifikácie

- Kedže zariadenia používaju MQTT bolo nutné mať v sieti MQTT server, v prípade že sa už server nachádza v sieti je možné ho priradiť do systému. Jednoduchší spôsob bol ale nainštalovať Mosquitto rozšírenie a nakonfigurovať ho aby používalo TLS. Taktiež boli nakonfigurované prihlasovacie údaje pre zariadenia, ktoré boli jedinečné pre každé zariadenie. V takomto prípade prí získani údajov pre jedno zariadenie útočník nezíska prístup ku všetkým zariadeniam.

```
logins:
  - username: QS-WIFI-S02 #prihlasovacie údaje
    password: nQj54K
  - username: BW-SHP6
    password: HZHueS
anonymous: false        #zakázanie pripojenia bez autentifikácie
customize:
  active: false
  folder: mosquitto
certfile: fullchain.pem #vygenerované klúče a certifikát pre TLS
keyfile: privkey.pem
```

Obr. 4.8: Ukážka konfigurácie pre Mosquitto server

- Nainštalovnaím NTP rozšírenia bol sprevádzkovaný lokálny NTP server, ktorý umožní našim zariadeniam synchronizovať čas bez nutnosti pripojenia k internetu. Na rozšírení bol iba špecifikovaný server pomocou ktorého HA synchronizoval čas.

V tomto bol už systém dostatočne zabezpečený a plne funkčný. Ešte bolo potrebné použiť rozšírenia, ktoré nám zlepšia samotnú administratívu systému a zariadení:

- Rozšírenie TasmoAdmin umožní efektívne manažovať Tasmota zariadenia, aktualizovať a konfigurovať ich z jedného miesta pomocou GUI. Rozšírenie používa vlastný server, ktorý bol zabezpečený znova pomocou TLS. Toto rozšírenie ale komunikuje so zariadeniami pomocou HTTP nešifrovaného protokolu a vyžaduje aby bolo Web UI spustené na zariadeniach. Taktiež vyžaduje otvoriť port 80 vo firewall na routri. Preto je nutné toto rozšírenie využívať iba v prípade administratívnych záležitostí ako aktualizácie zariadení.

[8] Time-Based One-Time Password

Posledným bodom bolo už iba zálohovanie systému pomocou snapshotov, ktoré HA ponúka. Samotný snapshot bol zabezpečený heslom kedže by z neho útočník mohol získať citlivé informácie o našom systéme. Snapshot sa ukladá do */backup* súbora. Pre pravidelné vytváranie snapshotov bolo použité *Google Drive Backup* rozšírenie, vďaka ktorému bolo špecifikované pravidelné vytváranie snapshotov a ich zálohovanie na Google Drive.

```
max_snapshots_in_hassio: 4
max_snapshots_in_google_drive: 4
days_between_snapshots: 3
use_ssl: true
send_error_reports: false
snapshot_password: SnapSQS3
```

Obr. 4.9: Konfigurácia rozšírenia pre vytváranie snapshotov

4.3.2 Firmware

Firmware bol nahratý Tuya-convert nástrojom spomenutým v 3.2.7 pre Tuya zariadenia a nástrojom Esptool, ktorý používa UART komunikáciu pre Sonoff zariadenia. Wthermostat firmware ponúka možnosť nakonfigurovať zariadenia iba pomocou GUI podobnému TasmotA, kde bol nastavený MQTT server a NTP server na IP adresu pre HA. Firmware taktiež ponúka možnosť nastavenia rozvrhu, čo je možné použiť ako redundanciu v prípade výpadku systému.

Štandartne dopredu skompilovaný Tasmota firmware neponúka niektoré funkcie z dôvodu šetrenia pamäte a výkonu. Z tohto dôvodu obsahuje tento firmware v základe bezpečnostné hrozby:

1. Nezabezpečená autentifikácia — útočník sa môže pripojiť na zariadenia, zmeniť nastavenia zariadenia, prebrať nad nim kontrolu, prepísať firmware atd...

2. Nezabezpečená komunikácia pomocou TLS — potencionálny útočník môže čítať MQTT komunikáciu so serverom z ktorej dokáže vyčítať citlivé informácie, taktiež HTTP komunikácu v prípade že sa pripojíme na GUI zariadenia, takže v prípade že aj zabezpečíme prístup k GUI, dokáže útočník vyčítať prihlasovacie údaje s HTTP komunikácie.

3. Nezabezpečený AP — v prípade výpadku AP na ktorý je zariadenie pripojené, Tasmota sama vytvorí AP na ktorý sa dá pripojiť bez nutnosti hesla a tým kontrolovať zariadenie.

Na to aby bolo možné zašifrovať komunikáciu bolo nutné skompilovať Tasmotu a povoliť TLS šifrovanie. Taktiež bolo povolené skriptovanie, viz 4.10, ktoré umožní vytvárať silnejšie automatizácie a správne kontrolovať QS-WIFI-S02 vypínač. Tasmota momentálne nepodporuje počítadlo frekvencie na vstupe ako vypínač, takže je nutné použiť skript, ktorý nám umožní toto zariadenie používať. Tieto funkcie sa povolujú v *my_user_config.h* súbore a samotná kompiláciu bola vykonaná pomocou PlatformIO.

Konfigurácia je možná pomocou GUI alebo príkazov, ktoré sa dajú použiť buď v konzoli, pomocu MQTT protokolu alebo použitím sériovej komunikácie. Kedže konfigurácia GUI si vyžaduje časté reštartovanie zariadenia, je veľmi zdĺhava a preto je efektívnejšie konfigurovať priamo pomocou príkazov pri ktorých je možné použiť príkaz Backlog, ktorý umožnuje zadať až do 30 príkazov naraz bez nutnosti reštartovania zariadenia.

```
#ifndef USE_SCRIPT
#define USE_SCRIPT  //  Povolenie skriptovnia
#endif
#ifdef USE_RULES
#undef USE_RULES    // Tasmota podporuje buď iba skriptovanie alebo pravidlá
#endif
#ifndef USE_MQTT_TLS
#define USE_MQTT_TLS  // Povolenie TLS pre MQTT
#endif
```

Obr. 4.10: Konfigurácia v *my _user_config.h*

Wi-Fi

V základných nastaveniach nastavíme SSID, heslo primárneho AP a sekundárneho redundaného AP a parameter Wificonfig na 4[9]. Takto pri výpadku nášho primárneho AP nebude Tasmota vytvárať vlastný AP, ktorý nie je zabezpečený heslom ale pripojí sa na náš redundantný AP. V prípade, že nemáme k dispozícii redundatný AP je nutné nastaviť WifiConfig na parameter 5, pri ktorom zariadenie iba počká kým sa objaví znova pôvodný AP bez vytvárania vlastného AP.

```
Backlog
SSID <SSID1> ;
Password1 <HESLOPRESSID1> ;
SSID2 <SSID2> ;
Password2 <HESLOPRESSID2> ;
Wificonfig 4
```

Výpis 4.4: Wi-Fi konfigurácia

MQTT

Po skompilovaní a nahratí firmware na zariadenia je nutné nastaviť IP adresu servera a port, na ktorom sa bude komunikovať. Ďalej je potrebné nastaviť autentifikáčné údaje pre prihlasenie sa na MQTT server, kde pre každé zariadenie je nutné mať jedinečné prihlasovacie údaje, aby v prípade že útočník získa prihlasovacie údaje pre jedno zariadenie nezískal kontrolu nad všetkými zariadeniami.

V implementácii bol použítý nami podpísaný certifikát, tým že bolo nutné naučiť sa odtlačok pri prvom pripojení, je možné vykonať MITM útok pri prvom pripojení, kde by sa Tasmota naučila útočníkov odtlačok, preto bol dopredu vypočítaný odtlačok pomocou nástroja *Tasmota-fingerprint*[10] a bol vložený priamo do zariadení. Nakoniec bol nastavený parameter *SetOption19*[11], ktorý umožní automaticky detekovať zariadenie pomocou MQTT pre HA.

[9] https://tasmota.github.io/docs/Commands/#wi-fi
[10] https://github.com/issacg/tasmota-fingerprint
[11] https://tasmota.github.io/docs/Commands/#setoptions

```
Backlog
MqttHost <BROKERIP> ;
MqttPort <PORT> ;
MqttUser <MENO> ;
MqttPassword <HESLO> ;
MqttFingerprint <SHA256FINGERPINT>;
SetOption19 1
```

Výpis 4.5: MQTT konfigurácia

Hlavný NTP server bol zmenený na lokálny, ktorý beží na HA a bolo nastavené kedy sa posúva čas, aby v prípade použitia časovačov na zariadení automatizácie fungovali správne.

```
Backlog
NtpServer1 <NTPSERVER>
TimeZone 99
TimeDST 0,0,3,1,2,120
TimeSTD 0,0,10,1,3,60
```

Výpis 4.6: Konfigurácia NTP a času

Posledným bodom je samotné nakonfigurovanie funkčnosti zariadenia pomocou šablón, kde z databáze[12] zariadení nájdeme šablóny, prípadne skript, ktoré ako v prípade QS-Wifi-S03.

```
Template {"NAME":"QS-Wifi-S03","GPIO":[0,0,17,0,160,0,0,0,43,42,21,22,0],"
    FLAG":0,"BASE":18}
```

Výpis 4.7: Vloženie šablóny pre QS-Wifi-S03

Pre samotné automatizácie je možné v prípade jendoduchých automatizácii dať zariadenia do multicast skupín, v ktorých budú medzi sebou zdielať informácie pomocou UDP protokolu. Táto komunikácia ale neni šifrovaná, z tohto dôvodu by automatizácie mali prebiehať v automatizačnom systéme. V prípade, že sa z nejakého dôvodu preruší komunikácia s HA (porucha, DoS útok atď), skript, ktorý bol vytvorený spustí túto redundantnú komunikáciu a automatizácie ostanú funkčné. V prípade, že sa pripojenie obnoví skript vráti zase zariadenia do pôvodného stavu. Príkazom DevGroupShare špecifikujeme, ktoré funkcie chceme zdielať a prijímať. Skript iba povoluje a vypína funkciu pomocou SetOption85.

```
Backlog
DeviceGroup redundanttopic
DevgroupShare 1,0
SetOption85 1
```

Výpis 4.8: Konfigurácia skupín

[12] https://templates.blakadder.com/

Prístup ku Tasmota GUI je možné zabezpečiť pomocou hesla, kedže ale Tasmota neponúka možnosť zašifrovať komunikáciu s web serverom tak je toto veľmi tenká vrstva zabezpečenia. Z tohto dôvodu je nutné vypnúť toto GUI a tým zabrániť prístupu k zariadeniu pomocou HTTP. Po vypnutí sa zariadenia dajú ovládať iba pomocou MQTT protokolu.

```
Backlog
WebPassword <HESLO> ;
WebServer 0 ;
```

Výpis 4.9: Konfigurácia autentifikácie a vypnutie web servera

4.3.3 Testovanie a porovnanie s oficialnými automatizačnými systémami

Podobne ako pri oficiálnych automatizačných systémoch bola komunikácia nahrávaná pomocou nástroja Wireshark. Všetky Tasmota zariadenia v sieti komunikovali pomocou TLS, zatiaľ čo BHT-6000 nemalo šifrovanú komunikáciu. Vykonaním MITM útoku bolo overené, že všetky naše zariadenia ktoré používajú TLS overujú certifikát pomocou kontroly odtlačku.

192.168.1.218	192.168.1.113	TCP	58	51776 → 8883 [SYN] Seq=0 Win=5840 Len=0
192.168.1.113	192.168.1.218	TCP	60	8883 → 51776 [SYN, ACK] Seq=0 Ack=1 Win
192.168.1.218	192.168.1.113	TCP	54	51776 → 8883 [ACK] Seq=1 Ack=1 Win=5840
192.168.1.218	192.168.1.113	TLSv1.2	146	Client Hello
192.168.1.113	192.168.1.218	TCP	60	8883 → 51776 [ACK] Seq=1 Ack=93 Win=641
192.168.1.113	192.168.1.218	TLSv1.2	1065	Server Hello, Certificate, Server Hello
192.168.1.218	192.168.1.113	TCP	54	51776 → 8883 [RST, ACK] Seq=93 Ack=1012

Obr. 4.11: Komunikácia kde Tasmota overuje certifikát

Testovaním a porovnaním s pôvodnou implementáciou boli objavené v open-source tieto zmeny:

1. BHT-6000 — nezabezpečená MQTT komunikácia Wthermostat firmware umožňuje čítať citlivé informácie z ktorých by útočník napríklad vedel vyčítať či sa nachádza obyvateľ domu v domácnosti podľa nastavenej teploty. Taktiež by sa zo získania prihlasovacích údajov mohol prihlásiť na MQTT server a odoberať/posielať správy do siete.

```
▸ Connect Flags: 0xc2, User Name Flag, Password Flag, QoS Level: At most once delivery
  Keep Alive: 15
  Client ID Length: 23
  Client ID: thermostatbeca_11482448
  User Name Length: 8
  User Name: BHT-6000
  Password Length: 6
  Password: HZHueS
```

Obr. 4.12: Zachytená MQTT komunikácia medzi BHT-600 a serverom

```
thermostat/schedules{"w1h":"07:06","w1t":20.00,"w2h":"08:00","
22.00,"w6h":"22:00","w6t":15.00,"a1h":"06:00","a1t":20.00,"a2h
20.00,"a5h":"17:00","a5t":20.00,"a6h":"22:00","a6t":15.00,"u1h
20.00,"u4h":"13:30","u4t":20.00,"u5h":"17:00","u5t":20.00,"u6h
```

Obr. 4.13: Zachytená MQTT komunikácia kde zariadenie posiela HA rozvrh

2. Deautentifikačný a rušívý DoS — kedže všetky naše zariadenia používaju ESP8266 alebo ESP8285, ktoré podporujú štandart ktorý nešifruje manažment rámce, ktoré tento útok používa je stále možné vykonať tento DoS útok na našu sieť.

3. TasmoAdmin komunikácia — v prípade že chceme vykonať administratívu na zariadeniach, by bolo nutné otvoriť dalšie firewall porty a komunikácia medzi VLAN by nebola šifrovaná.

4. Automatizácie medzi zariadeniami — útočník by mohol z komunikácie získať informácie o zariadeniach a samotných automatizáciach. Taktiež by mohol ovládať zariadenia posielaním správ do mutlicast skupiny a tým spúšťať rôzne automatizácie.

```
+...M-TASMOTA_DGR/redundantgroup HTTP/1.1

+...M-TASMOTA_DGR/redundantgroup HTTP/1.1
```

Obr. 4.14: Nešifrované UDP správy do multicast skupiny

5. Fyzický prístup — aj keď majú naše zariadenia zabezepečené OTA aktulizácie, v prípade že útočník získa fyzický prístup, môže prepísať firmware pomocu UART komunikácie alebo stiahnuť firmware a získať z neho citlivé informácie ako napr. prihlasovacie údaje na Wi-Fi pomocou decode-config[13] nástroja, ktorý slúži na dekódovanie Tasmota firmware.

Pri porovnaní s pôvodnou sieťou s oficiálnymi automatizačnými systémami a oficiálnym firmware obsahuje open-source sieť tieto vylepšenia:

- Zašifrovaná komunikácia — všetky zariadenia okrem BHT-600 majú zabezpečenú komunikáciu pomocou TLS a overujú certifkát takže sa úspešne bránia MITM útoku a úniku dát.

- Automatizačný systém v lokálnej sieti — presunutím systému do lokálnej siete bolo zabránené úniku citlivých informácií mimo lokálnu sieť. Taktiež bolo zabránené nefukčnosti automatizácii v prípade výpadku internetové pripojenia, zaťial čo bola zachovaná schopnosť sa vzdialene a bezpečne pripojiť na HA. Taktiež bol tento systém zabezpečený proti útokom hrubou silou pomocou viacfaktorovej autentifkácie a zakazovaním ip adries, z ktorých príchadza útok.

- Rozdelenie siete do VLAN — rozdelním sieti do rozdieľnych VLAN a povolením iba MQTT a NTP komunikácie bolo zabránené k prístupu nezabezpečeného BHT-6000 zariadenia a efektívne obmedzené možnosti vykonania DoS a iných útokov na sieť v prípade, že by útočník získal kontrolu nad nejakým zariadením.

[13] https://github.com/tasmota/decode-config

- Redundancia automatizácii — presunutím automatizácií pri ktorých sa vykonávajú akcie podľa času, priamo na zariadenia, a vytvorením reduntaných automatizácii pre jednoduché automatizácie bolo zabezpečené že časť automatizácii bude funkčná aj v prípade výpadku samotného HA v lokálnej sieti.
- Open-source firmware — využitím open-source firmware bola zabezpečená možnosť aktualizovať firmware OTA, ale bolo zabránené možnému stiahnutiu alebo nahratia firmware OTA bez povolenia užívateľa. Taktiež bol pridaný druhý AP, na ktorý sa zariadenia pripojia v prípade výpadku hlavného AP, čím bola znova zvýšená redundancia zariadení.

Kapitola 5

Diskusia a doporučenia na vytvorenie bezpečnej IoT siete

V tejto sekcii sa nachádza diskusia o nájdených hrozbách, ktoré boli objavené v zariadeniach a o konkrétných riešeniach pomocouu použitia open-source automatizačného systému a firmware. Taktiež sú v nej predstavené možné riešenia pre hrozby, ktoré obsahuje aj open-source implementácia a zhrnuté body, na ktoré sa treba zamerať pri budovaní bezpečnej IoT siete.

5.1 Diskusia

Aj keď zariadenia s pôvodním firmware a oficialnými automatizačnými systémami obsahovali bezpečnostné hrozby, je vidieť že samotný výrobcovia neponúkaju pravidelné aktualizácie, ktoré by zábránili týmto hrozbám. Z použitých zariadení iba Sonoff aktualizovalo svoje zariadena už proti známym hrozbám ako MITM útok a za čas robenia tejto práce iba Sonoff T3US3C zariadenie dostalo firmware aktualizáciu.

Taktiež bolo zistené , že zariadenia, ktoré ponúkajú rôzne informácie ako spotrebu elektrickej energie alebo teplotu, neposkytujú tieto informácie v prípade výpadku pripojenia na automatizačný systém aj keď je ich možné ovládať lokálne pomocou smartphone. Zatiaľ čo na poskytovanie dlhodobých štatistík je nutný automatizačný systém z dôvodu limitovaných zdrojov ktoré ESPP8266 ponúka, zariadenia by mali poskytovať aktuálne informácie v prípade že je ich možné ovládať.

Taktiež tým že zariadenia posielajú dáta na cloud sa objavuje otázka či a aké dáta zbierajú výrobcovia o užívateľoch a ich sietiach, napr. Tuya zariadenia podľa samotného výrobcu môžu zbierať rôzne dáta ako fotky alebo videá zo smart kamery [8].

Zatiaľ čo výsledkom open-source implementácie je sieť, ktorá obsahuje o dosť menej bezpečnostných hrozieb, tak aj samotné open-source zariadenia v základe obsahujú hrozby, a je na užívateľovi či ich vyrieši alebo nie, a teda aj v prípade správne nakonfigurovanej open-source implementácie v tejto sieti stále existujú hrozby:

- BHT-6000: Aj keď samotný autor tohto firmware neplánuje pridať TLS podporu[1] je možné, že niekto iný pridá podporu kedže je tento firmware open-source a je často modifikovaný komunitou [2]. Taktiež Tasmota postupne podporuje viacej zariadení, ktoré používajú TuyaMCU takže je možné, že sa v budúcnosti objaví konfigurácia pre

[1] https://github.com/klausahrenberg/WThermostatBeca/issues/69
[2] https://github.com/klausahrenberg/WThermostatBeca/network/members

toto zariadenie vďaka komunite. Alternatívním riešením môže byť používanie oficiálneho firmware a integrácia Tuya platformi do HA. Takto by sice bola zabezpečená komunikáciu pomocou TLS, ale komunikácia by prebiehala cez cloud a mohla by byť napadnutá MITM útokom a zariadenie by nebolo možné automatizovať v prípade výpadku internetového pripojenia.

- Automatizácie medzi zariadeniami: Kedže táto komunikácia predstavuje riziko mala by byť použitá iba na redundanciu automatizácii. A aj v tomto prípade je nutné zamyslieť sa aké automatizácie chceme takto ovládať. Zaťial čo pri automatizácii svetiel nevzniká až také veľké riziko, tak pri automatizácii zariadení ako sú napr. garážové dvere by útočník mohol získať prístup do domu. Záleží od samotnej implementácie ako veľmi je redundancia konkrétnej automatizácie dôležitá a aké veľké riziko by predstavovalo spustenie tejto automatizácie v inom momente ako je naplánované užívateľom.

- TasmoAdmin komunikácia: Kedže sa toto rozšírenie využíva hlavne pri administrácii, tak táto hrozba nemusí byť častá ale pri používaní rozšírenia môže uživateľ zabudnúť vypnúť webserver na Tasmota zariadeniach čím by potencionálnému útočnikovi dal možnosť prebrať kontrolu nad zariadeniami. Riešenie by bolo dať TasmoAdmin rozšírenie na samostatné zariadenie a umiestniť ho priamo do VLAN pre Tasmota zariadenia. Takto by bolo možné pristupovať pomocou šifrovanej kuminikácii ku TasmoAdmin zaťial čo nešifrovaná kumunikácia medzi TasmoAdmin a Tasmota zariadeniami ostala izolovaná vo VLAN. Ďalšim možným riešením by bolo umiestnenie „reverse proxy“[3] do IoT VLAN a priamo pristupovať na zariadenia pomocou domén ktoré by sa zariadeniam pridelili.

- Deautetifikačný útok a rušivý Dos útok: Zabrániť takýmto útokom nie je možné zmenou firmware alebo automatizačného systému kedže fungujú na veľmi nízkom leveli OSI modela. Proti deautentifikačnému útoku, ktorý je možný vykonať aj s ESP8266 mikrokontrolérom existuje riešenie: IEEE 802.11W štandart, ktorý implementuje šifrovanie pre manažovateľné rámce, ktoré sú na tento útok používané [34]. Tento štandart ale nepodporuje samotný ESP8266 mikrokontrolér a momentálne ani väčšina Wi-Fi zariadení.

5.2 Doporučenia pre vytvorenie bezpečnej IoT siete

1. Použitie open-source systému a firmware — v prípade, že chceme zabezpečiť sieť je nutné použiť open-source riešenia pre automatizačný systém a firmware. Ako bolo preukázané oficiálne systémy posielajú mimo lokálnu sieť nezabezpečenú komunikáciu, z ktorej je možné získať citlivé informácie. Taktiež nekontrolujeme aké informácie o nás získava samotný výrobca zariadenia. Použitím open-soure systému presunieme systém do lokálnej siete kde máme vyššiu kontrolu nad zariadeniami a samotným systémom.

2. Výber systému a firmware — pri výbere automatizačného systému je nutné zamerať sa na to aké rozšírenia a integrácie systém ponúka a aké možnosti zabezpečenia daný systém ponúka. Zaťial čo Home Assistant ponúka možnosť zabezpečenia GUI

[3] https://github.com/jc21/nginx-proxy-manager

prihlasovacími údajmi a manažment užívateľov, OpenHAB neponúka túto funkciu a je nutné použiť externý nástrok ako NGINX Reverse Proxy [4]

Pri výbere firmware je nutné zamerať sa hlavne na aké zariadenia budeme firmware nahrávať a aké zabezpečenie firmware ponúka. Zatiaľ čo Tasmota je hlavne pre komerčné zariadenia a ponúka zabezpečenie pomocou TLS, ESPhome je skôr zamerané na jednotlivé komponenty zariadení a jeho implementácia TLS nie je dostatočne zabezpečená. Ďalšie možnosti pre firmware sú Espurna[5] a ESPEasy[6]

3. VLAN a Firewall — Zariadenia v sieti musia byť rozdelené do separátných VLAN. Okrem rozdelenia IoT siete a klasickej počítačovej siete, je dobré rozdeliť IoT zariadenia do separátnych VLAN v prípade, že používaju rozdielny firmware alebo komunikáciu. Nastavením firewall pravidiel je nutné povoliť iba konkrétnu komunikáciu, ktorá je nutná na správne používanie firmware, aby útočník nemal napríklad prístup ku zariadeniam z internetu.

4. Zašifrovanie komunikácie — všetká komunikácia medzi zariadeniami a automatizačným systémom musí byť zašifrovaná pomocou TLS a zariadenia musia overovať certifikáty aby sa vyhli zverejnovaniu citlivých informácií do sieti a potencionálnym MITM útokom.

5. Vzdialený prístup — komunikácia do sieti musí byť dostatočne zabezpečená pomocou TLS, v prípade že je doména automatizačného systému verejná je nutné implementovať možnosť autentifkovať uživateľov a opatrenia proti útokom hrubou silou.

6. Autentifikácia — prístup k zariadeniam a systému musí byť vždy pomocou autentifikácie, prihlasovacie údaje musia byť vždy unikátne aby v prípade, že potencionálny útočník získa údaje k jednému zariadeniu nezískal prístup ku všetkým zariadeniam.

7. Fyzický prístup — fyzický prístup ku zariadeniam a ku interfacu UART musí byť zabezpečený iba pre oprávnených užívateľov. Toto je možné dosiahnuť umiestneným zariadenia na miesto, ktoré nie je viditeľné alebo nie je tak ľahko dostupné aby útočník mohol získať prístup k zariadeniu a stiahnuť z neho firmware.

8. Automatizácie — kontrola nad zariadeniami musí byť funkčná aj v prípade výpadku automatizačného systému. Ak je to možné, pre kritické automatizácie by mala existovať nejaká možnosť redundancie. Taktiež je nutné prepojenie systému so smartphone aby v prípade výpadku systému alebo zariadenia zo sieti mohli byť obyvatelia notifikovaný o tomto výpadku.

9. Zálohovanie a aktualizácie — automatizačný systém a aj samotný firmware na zariadeniach musí byť zálohovaný. Toto umožní v prípade nejakej chyby v systéme alebo zariadení rýchlo a efektívne znova obnoviť zariadenia do predchádzajúceho funkčného stavu. Samotna záloha musí vznikať pravidelne aby boli zachované aktuálne zmeny a umiestnená mimo systému, idéalne na zabezpečený cloud alebo verzovací systém ako Git. Taktiež je nutné pravidelne aktualizovať firmware a automatizačný systém aby bol systém chráneny proti aktuálnym hrozbám.

[4] https://www.openhab.org/docs/installation/security.html#nginx-auth
[5] https://github.com/xoseperez/espurna
[6] https://www.letscontrolit.com/wiki/index.php/ESPEasy

10. Administratíva — nástroje ktoré nám umožnia jednoduchú administratívu sú často nezabezpečené ako napríklad Tasmoadmin, preto je nutné zabezpečiť tieto nástroje, inak je nutné aby sa používali iba v prípade administratívy a iných prípadoch boli tieto funkcie/nástroje vypnuté.

Kapitola 6

Záver

Témou tejto bakálarskej práce bola bezpečnosť IoT siete. Na začiatku bolo potrebné oboznámiť sa s IoT zariadeniami v navrhnutej sieti a ich funkciami a porovnať ich automatizačné systémy. Ďalej bolo nutné informovať sa, ktoré technológie a protokoly sa bežne používajú v IoT sieťach.

Okrem toho bolo nevyhnutné zoznámiť sa s bezpečnostnými hrozbami v IoT. Bola vykonaná analýza bezpečnostni siete a jej IoT zariadení. Analýza bezpečnosti prebiehala nahratím komunikácie, ktorá prebiehala medzi zariadeniami a automatizačným systémom, pomocou nástroja WireShark. V nej bolo zistené, že nie všetká komunikácia medzi zariadeniami, ktorá ponúka možnosť útočníkovi získať citlivé informácie ako prihlasovacie údaje alebo čas kedy by sa zariadenie malo vypnút/zapnuť, je šifrovaná pomocou TLS. Toto by napríklad mohol útočník využiť na to aby zistil kedy sa užívateľ nachádza doma.

Následne bola vykonaná simulácia MITM útoku, ktorá ukázala, že zariadenia od spoločnosti Sonoff overujú certifikát a tým sa efektívne bránia proti MITM útoku.

Sledovaním chovania zariadení v prípade nedostupného systému sa objavilo, že zatiaľ čo Sonoff zariadenia pri nedostupnom systéme boli stále schopné vykonať automatizácie podľa času, Tuya zariadenia neboli schopné vykonávať tieto automatizácie.

Ďalej sa testovala schopnosť nahrať/stiahnuť firmware na zariadenia pomocou OTA z iného zdroja ako samotného automatizačného systému. Počas tohto testu bolo zistené, že iba Tuya zariadenia nie sú zabezpečené proti tomuto útoku a teda dávajú možnosť takto prepísať firmware a prebrať nad zariadením kontrolu.

Nakoniec boli uvedené možné riešenia pre objavené hrozby a v krátkosti boli predstavené a porovnané open-source firmware a automatizačné systémy.

Pri implementácii navrhnutej siete, bola najprv sieť a zariadenia v sieti rozdelené do rôznych VLAN a bola povelená iba nutná komunikácia medzi nimi pomocou firewallu. Home Assitant bol naištalovaný ako náš automatizačný systém a bol zabezpečený proti potencionálnym hrozbám. Nainštalovaním rôznych rozšírení bol umožnený vzdialený prístup k Home Assistant a boli rozšírené funkcie systému napríklad o MQTT a NTP server. Nahratím Tasmota firmware na zariadenia a jeho nakonfigurovaním bola zabezpečená ich funkčnosť a šifrovaná komunikácia s MQTT serverom, taktiež boli vytvorené redundantné automatizácie v prípade, že by zariadenia stratili pripojenie ku HA. Ďalej bola otestovaná a porovnaná navrhnutá open-source implementácia a s pôvodným firmware na zariadeniach a oficiálnymi automatizačnými systémami od výrobcov zariadení, a dospelo sa k záveru, že open-source implementácia obsahuje nejaké hrozby ale celkovo je bezpečnejšia.

Na záver prebehla diskusia o objavených hrozbách nájdených aj v pôvodnej a open-source sieti a vytvorili súbor doporučení na ktoré sa treba zamerať v prípade vytvárania IoT sieťe.

Literatúra

[1] [cit. 2020-01-20]. Dostupné z: https://www.amazon.com/MoesHouse-BHT-6000-GBLW-Underfloor-Thermostat-Compatible/dp/B07VLF2MJS.

[2] [cit. 2020-01-20]. Dostupné z: https://www.blitzwolf.com/BlitzWolf-BW-SHP6-2300W-WIFI-Smart-Socket-EU-Plug-Works-with-Alexa-Remote-Control-Time-Switch-Electricity-Monitoring-p-300.html.

[3] [cit. 2020-01-20]. Dostupné z: http://www.wifismartplugfactory.com/product/Wifi-Smart-Switch.html.

[4] [cit. 2020-01-20]. Dostupné z: https://www.smartsense.co.za/33-large_default/sonoff-pow2.jpg.

[5] [cit. 2020-01-20]. Dostupné z: https://sonoff.tech/product/wifi-smart-wall-swithes/tx-series.

[6] [cit. 2020-01-20]. Dostupné z: https://www.amazon.co.uk/MOES-Control-Anywhere-Compatible-Required/dp/B07P2L2DV3/ref=pd_sbs_60_img_1/260-9280068-2773949?_encoding=UTF8&pd_rd_i=B07P2L2DV3&pd_rd_r=9cfb3d8d-9b1a-4bf7-9f5a-1b41674a00e2&pd_rd_w=F9SEc&pd_rd_wg=qut4x&pf_rd_p=e44592b5-e56d-44c2-a4f9-dbdc09b29395&pf_rd_r=1QBYY975MPMZB7NK1PHE&psc=1&refRID=1QBYY975MPMZB7NK1PHE.

[7] *MQTT*. [cit. 2020-03-22]. Dostupné z: https://www.paessler.com/it-explained/mqtt.

[8] *Tuya Mobile Privacy Notice* [online]. [cit. 2020-05-20]. Dostupné z: https://images.tuyacn.com/app/html/SmartLife_Privacy_Policy_en.html.

[9] Al Kashoash, H. A. A. a Kemp, A. H. Comparison of 6LoWPAN and LPWAN for the Internet of Things. *Australian Journal of Electrical and Electronics Engineering*. Taylor & Francis. 2016, roč. 13, č. 4, s. 268–274. Dostupné z: http://www.tandfonline.com/doi/abs/10.1080/1448837X.2017.1409920. ISSN 1448-837X.

[10] Ali, B. a Awad, A. I. Cyber and Physical Security Vulnerability Assessment for IoT-Based Smart Homes. *Sensors (Basel, Switzerland)*. 2018, roč. 18, č. 3. Dostupné z: http://search.proquest.com/docview/2012915309/. ISSN 1424-8220.

[11] Bhattasali, T., Chaki, R. a Sanyal, S. Sleep Deprivation Attack Detection in Wireless Sensor Network. *ArXiv.org*. Ithaca: Cornell University Library, arXiv.org. 2012, roč. 40, č. 15. Dostupné z: http://search.proquest.com/docview/2086228293/. ISSN 09758887.

[12] Bormann, C. [cit. 2020-03-20]. Dostupné z: https://coap.technology/.

[13] Butun, I., Osterberg, P. a Song, H. Security of the Internet of Things: Vulnerabilities, Attacks, and Countermeasures. *IEEE Communications Surveys & Tutorials*. IEEE. 2020, roč. 22, č. 1, s. 616–644. ISSN 1553-877X.

[14] Fielding, R. et al. *Hypertext Transfer Protocol – HTTP/1.1* [online]. [cit. 2020-03-22]. Dostupné z: https://tools.ietf.org/html/rfc2616.

[15] Frenzel, L. *What's The Difference Between ZigBee And Z-Wave?* [online]. [cit. 2020-03-20]. Dostupné z: https://www.electronicdesign.com/technologies/communications/article/21796052/whats-the-difference-between-zigbee-and-zwave.

[16] He, X., Papa, M. a Gamble, R. Extending Over-the-Air Libraries to Secure ESP8266 Updates. In: *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 2019, s. 1–6. ISBN 9781728150925.

[17] (IBM), I. B. M. C. a Eurotech. *MQTT V3.1 Protocol SPecification* [online]. [cit. 2020-03-20]. Dostupné z: http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/MQTT_V3.1_Protocol_Specific.pdf.

[18] Khan, M. A. a Salah, K. IoT security: Review, blockchain solutions, and open challenges. *Future Generation Computer Systems*. Elsevier B.V. 2018, roč. 82, s. 395–411. ISSN 0167-739X.

[19] Lamkin, P. *Z-Wave explained: What is Z-Wave and why is it important for your smart home?* [online]. 2020 [cit. 2020-03-22]. Dostupné z: https://www.the-ambient.com/guides/zwave-z-wave-smart-home-guide-281.

[20] Lin, H. a Bergmann, N. W. *IoT Privacy and Security Challenges for Smart Home Environments* [online]. [cit. 2020-02-20]. Dostupné z: https://www.mdpi.com/2078-2489/7/3/44/htm.

[21] Mbanaso, U. M. a Chukwudebe, G. A. Requirement analysis of IoT security in distributed systems. In: *2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON)*. IEEE, 2017, s. 777–781. ISBN 9781509064229.

[22] Michael, M. *Attack Landscape H1 2019: IoT, SMB traffic* [online]. [cit. 2020-03-18]. Dostupné z: https://blog.f-secure.com/attack-landscape-h1-2019-iot-smb-traffic-abound/.

[23] Milliken, J., Selis, V., Yap, K. M. a Marshall, A. Impact of Metric Selection on Wireless DeAuthentication DoS Attack Performance. *IEEE Wireless Communications Letters*. IEEE. 2013, roč. 2, č. 5, s. 571–574. ISSN 2162-2337.

[24] Noubir, G. a Lin, G. Low-power DoS attacks in data wireless LANs and countermeasures. *Mobile Computing and Communications Review*. Júl 2003, roč. 7, s. 29–30.

[25] Oh, S. a Kim, Y. Security Requirements Analysis for the IoT. In: *2017 International Conference on Platform Technology and Service (PlatCon)*. 2017, s. 1–6.

[26] Phil, E. *Over 100 Million IoT Attacks Detected in 1H 2019* [online]. [cit. 2020-03-18]. Dostupné z: https://www.infosecurity-magazine.com/news/over-100-million-iot-attacks/.

[27] Shelby, Z. et al. *RFC 7252 - The Constrained Application Protocol (Coap)* [online]. [cit. 2020-03-20]. Dostupné z: https://tools.ietf.org/html/rfc7252#section-1.

[28] Shuminoski, T. a Risteski, A. Analysis of the SYN Flood DoS Attack. *International Journal of Computer Network and Information Security*. Hong Kong: Modern Education and Computer Science Press. 2013, roč. 5, č. 8, s. 1–11. Dostupné z: http://search.proquest.com/docview/1623861730/. ISSN 20749090.

[29] Simpson, A. K., Roesner, F. a Kohno, T. Securing vulnerable home IoT devices with an in-hub security manager. In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017, s. 551–556. ISBN 9781509043385.

[30] Stanford Clark, A. a Truong, H. L. *MQTT For Sensor Networks (MQTT-SN)* [online]. [cit. 2020-03-20]. Dostupné z: http://www.mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf.

[31] Systems, E. *ESP8266EX Datasheet* [online]. [cit. 2020-03-20]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[32] Taylor, P. T., Calhoun, P. R. a Rubens, A. C. [cit. 2020-03-22]. Dostupné z: https://tools.ietf.org/pdf/draft-taylor-ipdc-00.pdf.

[33] Unwala, I., Taqvi, Z. a Lu, J. Thread: An IoT Protocol. In: *2018 IEEE Green Technologies Conference (GreenTech)*. IEEE, 2018, s. 161–167. ISBN 9781538651834.

[34] Wang, W. a Wang, H. Weakness in 802.11w and an improved mechanism on protection of management frame. In: *2011 International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2011, s. 1–4. ISBN 9781457710094.

Príloha A

Konfigurácia open-source IoT sieťe

A.1 Home Assistant

Home Assistant je open-source automatizačný systém, inštalácia je jednoduchá stačí sťiahnuť image pre konkrétne zariadenie z oficiálnej stránky Home Assistant https://www.home-assistant.io/hassio/installation/, vložiť daný image na naše zariadenie a spustiť. V prípade že chceme aby HA bežal v systéme Linux ako Docker container použijeme nasledujúci príkaz kde *CESTA_KU_CONFIG* je treba nahradiť cestou konfiguračného súboru v systéme Linux

```
docker run --init -d --name="home-assistant" -e "TZ=Europe/Bratislavad" -v
    /CESTA_KU_CONFIG:/config --net=host homeassistant/home-assistant:stable
```

Výpis A.1: Inštalácia HA ako Docker container

Pri prvom spustení bude HA sťahovať aktualizácie čo môže trvať okolo 20 minút. Po aktualizovaní bude HA dostupný na doméne http://homeassistant.local:8123 ak router v sieti podporuje mDNS, v prípade že nepodporuje bude systém dostupný na lokálnej IP adrese ktorá mu bola pridelená čiže napríklad http://192.168.1.45:8123. Po nastavení uživateľa, polohy a metrického systému, HA automaticky oskenuje sieť pre zariadenia ktoré je možné pridať do systémyu, potom už dostaneme prístup samotné GUI pre HA, kde ďalej nainštalujeme rozšírenia ktoré priadjú do HA daľšie funkcie a efektívne zabezpečia systém.

Pre pridanie rozšíreni existuje *Add-on Store* do ktorého sa dostaneme cez *Superviser*. Každé nainštalované rozšírenie sa objaví v *Superviser*, kde každé obsahuje dokumentáciu, konfiguračný súbor a log.

Ako prvé nainštalujeme rozšírenie ktoré nám umožní editovať konfiguračné súbory priamo v HA GUI. Na túto funkciu si je možné vybrať s troch rôznych rozšírení:

- File Editor: Ktorý nám umožnuje vytvárať, čítať, editovať a nahravať súbory do našej HA inštalácii a taktiež umožnuje prepojenie z verzovacím systémom Git
- Terminal & SSH: Toto rozšírenie nám pridá plne funkčný terminál do HA GUI a umožní nám sa pripojiť do HA pomocou SSH. Toto rozšírenie je ale možné nainštalovať iba po zapnutí Advanced módu, ten je možné zapojiť v našom profile.
- Visual Studio Code: Rozšírenie ktoré ponúka plne funkčné Visual Studio Code prostredie a umožnuje editovať konfiguračné súbory, rozšírenie ale neni podporované pre

všetky zariadenia kedže vyžaduje veľa pamäte RAM, preto sa neodporučuje využívať toto rozšírenie ak máme menej ako 4 GB pamäte.

A.1.1 Zabezpečenie a vzdialený prístup

Pre zabezpečenie HTTP komunikácie je možné použiť nami podpísane certifikátý ktoré vygenerujeme pomocou openssl, a vložíme do /ssl/ súboru:

```
openssl req -sha256 -addext "subjectAltName = IP:X.X.X.X" -newkey rsa:4096
    -nodes -keyout privkey.pem -x509 -days 730 -out fullchain.pem
```

Výpis A.2: Vygenerovanie klúčovv pre TLS šifrovanie

Kde "x.x.x.x."je adresa HA v lokálnej sieti. Po ich vygenerovaní stačí pridať do konfiguračného súboru *configuration.yaml* cestu k samotným certifikátom.

```
http:
    ssl_certificate: /ssl/fullchain.pem
    ssl_key: /ssl/privkey.pem
```

Tento postup je možné použiť len v prípade že nechceme použiť tieto certifikáty pre zabezpečenie komunikácie so vzdialeným prístupom.

Ďalším možným spôsobom ako zabezpečiť http komunikáciu je použitím DuckDNS rozšírenia ktoré nám umožní vzdialený prístup ku HA a zabezpečí našu komunikáciu pomocou Lets Encrypt. Po nainštalovaní DuckDNS je nutné sa zaregistrovať na https://www.duckdns.org/. Kde si zvolíme našu subdoménu a vygenerujeme token, tieto dva údaje je nutné vložiť do našeho konfiguračného súboru kde je taktiež nutné povoliť Lets encrypt ktoré pri prvom spustení vygeneruje certifikáty ktoré sa budú používať. Pri použití tohto addonu je nutné na routri presmerovať port 443 na port 8123 pre našu špecifickú adresu.

```
lets_encrypt:
  accept_terms: true
  certfile: fullchain.pem #zabezpečenie TLS pomocou Lets Encrypt
  keyfile: privkey.pem
token: 9ab72a12-cfe1-4047-89f8-2807d8affe4a # vygenerovaný token
domains:
  - myhadomain123.duckdns.org #subdoména pre HA
```

Obr. A.1: Duckdns konfigurácia

Ďalšie možnosti ako zabezpečiť vzdialený prístup je použiť oficiálny Cloud pre HA, ktorý je dostupný na https://www.nabucasa.com/ jeho inštalácia je veľmi jednoduchá, stačí ísť do **Configuration -> HA Assistant Cloud** kde je nutné sa zaregistrovať po úspešnej registrácii stačí povoliť **Remote Acess** a automaticky sa vygenerujú TLS certfikáty znova pomocou Lets Encrypt ktoré zabezpečia komunikáciu, prístup je možný vďaka tomu že HA je pripojený ku proxy daného cloudu, viacej infromácii https://www.nabucasa.com/config/remote/.

Poslednou možnosťou je použitie Tor rozšírenia ktorý sa oplatí používať v prípade že nám ISP neposkytuje verejnú adresu, nechceme otvárať port na našom routri, nechceme konfigurovať TLS certifikáty a chceme ostať úplne anonymní. Po inštalácii stačí nastaviť

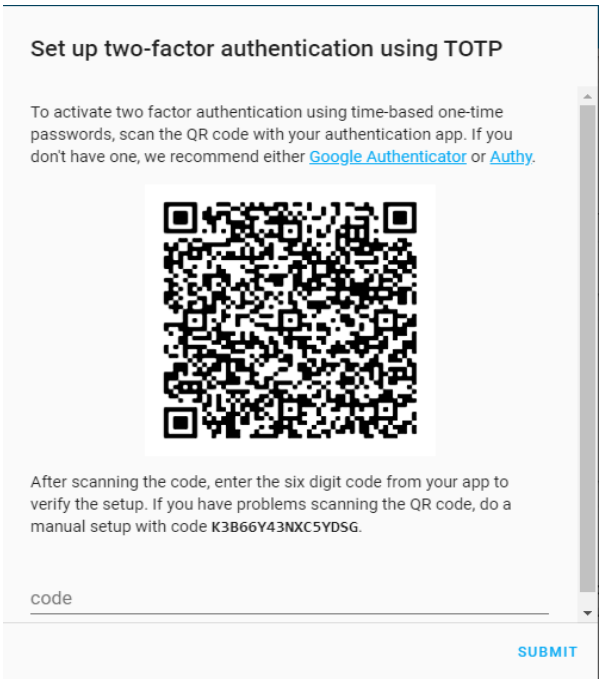
port na ktorý sa budeme pripájať a spustiť tento addon, v logu nášho addonu sa objaví doména na ktorú sa potom pripojíme pomocou ľubovolného Tor prehliadača.

```
[14:13:09] INFO: ----------------------------------------------------------
[14:13:09] INFO: Your Home Assistant instance is available on Tor!
[14:13:09] INFO: Address: defk5viwasi2fby7.onion
[14:13:09] INFO: ----------------------------------------------------------
```

Obr. A.2: Tor log v ktorom sa zobrazí onion doména

Ďalším bodom zabezpečenia systému je ochrana proti útokom hrubou silou, to je možné zabezpečiť dvoma spôsobmi:

- Multifaktorová autentifikácia: kde HA používa TOTP modul ktorý vygeneruje časovo obmedzené unikátne heslo. Táto funkcia vyžaduje autentifikačnú aplikáciu na našom Smartphone ako https://authy.com/. V prípade že už máme zapnutý „Advanced mód“ stačí ísť do nášho profilu kde po zapnutí funkcie je nutné oskenovať QR kód ktorý sa nám zobrazí a zadať kód ktorý nám ponúkne aplikácia.



Obr. A.3: Dvojfaktorová autentifikácia

V prípade že neni zapnutý „Advanced mód“ je nutné povoliť túto funkciu v konfiguračnom súbore

```
homeassistant:
    auth_mfa_modules:
        - type: totp
```

Výpis A.3: Konfiguráia multifaktorovej autentifikácie

- Ďalším spôsobom ako zabezpečiť ochranu proti útokom hrubou silou, je odmietať prihlasovania do systému od IP adries ktoré prekročili nami daný limit pokusov o prihlásanie: Toto sa znová konfiguruje v konfiguračnom súbore v *http* sekcii

```
http:
    ip_ban_enabled: true
    login_attempts_treshold: 5
```

Výpis A.4: Konfigurácia ochrany proti útokom hrubou silou

A.1.2 MQTT a zálohovanie systému

Posledným bodom konfigurácie je nainštalovať MQTT server pre komunikáciu s našimi zariadeniamia a zálohovať systém. Na toto je možné použiť samostatný MQTT server na ktorý sa naše zariadenie pripojí, jednoduchšie je ale použiť Mosquitto rozširenie. Kde po nainštalpovaní rozšírenia je nutné v konfiguračnom súbore nakonfigurovať prihlasovacie údaje pre naše zariadenia, ktoré z dôvodu bezpečnosnti musia byť jedinečne. Taktiež treba použiť certifikáty pre zabezpečenia komunikácie pomocou TLS, je treba zistiť v dokumentácii firmware ktorý chceme použiť ake certifikáty podporujú, zaťial čo Tasmota podporuje aj overovanie certifikačnej autority Lets Encrypt https://tasmota.github.io/docs/TLS/, ESPHome podporuje iba overovanie pomocou odtlačku verejného klúča https://ESPHome.io/components/mqtt.html#ssl-fingerprints.

```
logins:
  - username: QS-WIFI-S02 #prihlasovacie údaje
    password: nQj54K
  - username: BW-SHP6
    password: HZHueS
anonymous: false         #zakázanie pripojenia bez autentifikácie
customize:
  active: false
  folder: mosquitto
certfile: fullchain.pem #vygenerované klúče a certifikát pre TLS
keyfile: privkey.pem
```

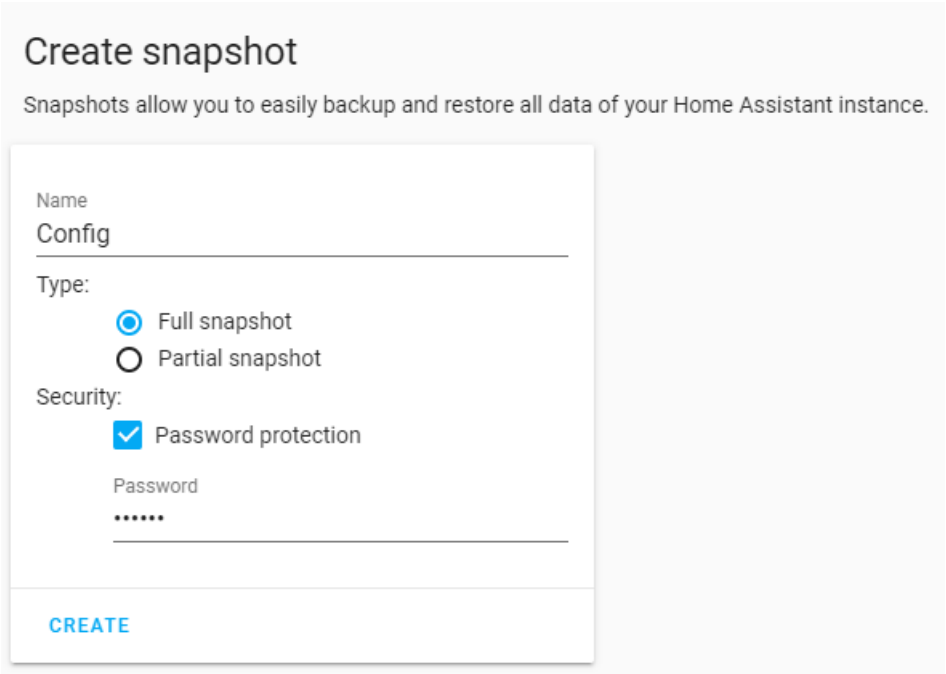
Obr. A.4: Ukážka MQTT konfigurácie

Po nakonfigurovaní je nutné taktiež zálohovať systém, HA ponúka vytváranie snapshotov do ktorých si možeme zvoliť ktorú časť systému chceme zálohovať. Samotný snapshot sa uloží do */backup* súboru.

Pre zautomatizovanie vytvárania možeme vytvoriť jednoduchú automatizáciu ktorú vložíme do *automations.yaml* ktorá nám bude pravideľne vytvárať snapshot.

```
- alias: Snapshot 1.00AM
  trigger:
    platform: time
    at: '01:00:00'
  action:
  - service: hassio.snapshot_full
    data_template:
    name: Aut Backup {{ now().strftime('%Y-%m-%d_%H:%M') }}
```

Výpis A.5: Automatizácia pre vytváranie snapshotov



Obr. A.5: Vytváranie snapshotu

Pre efektívne vytváranie snapshotov a aj ich zálohovanie je lepšie použiť špecifické rozšírenie ktoré nám umožní zároveň aj zálohovať snapshoty na Google Drive. Kedže sa toto rošírenie nenachádza v Ädd-on Store"ju nutné ho tam pridať vložením jeho github repozitára `https://github.com/sabeechen/hassio-google-drive-backup`. Potom si už iba prepojíme rozšírenie s našim účtom na Google Drive a nakonfigurujeme ako často a koľko snapshotov chceme vytvárať.

```
max_snapshots_in_hassio: 4
max_snapshots_in_google_drive: 4
days_between_snapshots: 3
use_ssl: true
send_error_reports: false
snapshot_password: SnPsSQS23
```

Obr. A.6: Ukážka konfigurácie rozšírenia pre snapshoty

Ďalej môžeme nainštalovať rozšírenia pre špecifický firmware, pre ESPHome môžeme integrovať samotné ESPHome pod HA. Podobne ako pri rozšírení pre automatizáciu snapshotov je nutné pridať jeho repozitár do HA. Toto rozšírenie nám ponúkne všetky možnosti ESPHome a ešte ponúkne možnosť vytvárania konfigurácii jednoducho pomocou GUI ktoré ponúka. V tomto rozšíreni není nutné nič konfigurovať kedže beží priamo v systéme HA.

Pre Tasmota môžeme nainštalovať TasmoAdmin rozširenie, to nám umožní manažovať a aktualizovať všetky zariadenia z jedného miesta. Kedže Tasmoadmin ponúka web server je nutné zapnúť TLS šifrovanie pre komunikáciu s ním, rozšírenie použije tie isté certifikáty ktoré sme použili na šifrovanie HA komunikácie. Toto rozšírenie ale vyžaduje aby boli zapnutý web server na samotných zariadeniach ku ktorým pristupuje, kedže táto komunikácia neni šifrovaná by sa Tasmoadmin mal používať iba na administratívne účely.

A.2 Konfigurácia ESPHome

Konfigurácia ESPHome prebieha iba pomocou konfiguračných súborov,samotný nástroj ponúka aj GUI kde sa dá vykonať časť konfigurácie ale väčšina konfigurácie sa stále vykoná pomocou YAML konfiguračných súborov.

Ako prvé nastavíme AP na ktoré sa chceme pripojiť, je dobré nastaviť viacero AP aby v prípade výpadku jedného AP sa zariadenie pripojilo na redundantný AP.

```
wifi:
    networks:
    - ssid: <SSID1>
        password: <HESLOPRESSID1>
    - ssid: <SSID2>
        password: <HESLOPRESSID2>
```

Výpis A.6: Konfgiurácia Wi-Fi

V prípade iba jedného AP na ktorý je možné sa pripojiť je možné použiť Fallback AP kde v prípade výpadku pôvodného pripojenia zariadenie prejde do AP režimu, ten musí byť zabezpečený heslom aby sa útočník nedostal jednoducho na zariadenia

```
wifi:
    ssid: <SSID1>
    password: <HESLOPRESSID1>
    ap:
        ssid: <SSIDAP>
        password: <HESLOPREAP>
```

Výpis A.7: Konfgiurácia Wi-Fi a AP

Pre efektívne aktualizácie, OTA aktualizácie musia byť povolené, inak by sa firmware musel vždy nahrávať pomocou UART komunikácie, OTA funkcia ale musí byť chránená heslom aby útočník nemohol nahrať firmware bez nášho vedomia.

```
ota:
    password: <OTAHESLO>
```

Výpis A.8: Konfgiurácia OTA aktualizácii

ESPHome ponúka možnosť webserveru, komunikáciu s ním ale nieje možné zašifrovať preto by mal byť webserver vypnutý, v prípade že je nutné mať zapnutý webserver je nutné zabezpečiť prístup autentifikáciu, toto je ale veľmi tenká vrstva zabezpečenia kedže prihlasovacie údaje je možné vidieť v odchytenej komunikácii. S tohto dôvodu by sa OTA aktualizácie nemali spuštať pomocou web serveru kedže útočník by mohol odchytiť heslo na OTA aktualizáciu a nahrať svoj firmware na zariadenie.

```
web_server:
    port: 80
    auth:
        username: <MENO>
        password: <HESLO>
```

Výpis A.9: Konfgiurácia Web servera

ESPHome ponúka dva spôsobi komunikácie, MQTT a vlastné ESPHome Native API ktoré používa TCP protkol, tento protokol sa ale nedá zabezpečiť pomocou TLS preto je nutné použiť MQTT ktoré je možné zabezpečiť pomocou TLS. ESPHome overuje iba odlačok ktorý je SHA1 hash z certifikátu, takže vždy ked sa zmení certifikát je nutné zmeniť aj samotný odltačok, ďalej SHA1 neni úplne bezpečné a s dostatočným vykonom je možné tento odtlačok sfalšovať. Samotný odtlačok sa získa daním mqtt servera a portu do konfigurácie a spustením:

```
    ESPHome myconfig.yaml mqtt-fingerprint
```

Výpis A.10: Vypočítanie odtalčku

Výstupom bude odltačok ktorý vložime do našej konfigurácie ktorá pre MQTT vyzerá takto:

```
mqtt:
    broker: <IPBROKER>
    port: 8883
    username: <Meno>
    password: <Heslo>
    discover: True #Aby HA automaticky objavil zariadenie
    ssl_fingerprints: <SHA1FINGERPRINT>
```

Výpis A.11: Konfgiurácia MQTT

V prípade že potrebujeme aby ESPHome dokázalo spustiť automatizácie podla času je nutné nakonfigurovať NTP. Ak používame Native Api protokol si zariadenia dokáže samo zosynchronizovať čas s našou Home Assistant inštaláciou.

```
time:
  - platform: homeassistant
    id: homeassistant_time
```

Výpis A.12: Konfgiurácia synchronizácie času

V prípade že používame MQTT je možné nakonfigurovať NTP serveri pomocou ktorých bude ESPHome synchronizovať čas.

```
time:
  - platform: sntp
    servers:
        - "192.168.0.1"
        - "1.pool.ntp.org"
        - "2.pool.ntp.org"
```

Výpis A.13: Konfgiurácia NTP

Nakoniec treba nakonfigurovať samotné piny na zariadení, dopredu vytvorené konfigurácie nájdete buď v databáze zariadení ktorú spravuje komunita https://esphome-configs.io/ alebo na oficiálnej stránke pre ESPHome https://esphome.io/

A.3 Tasmota

Tasmota oproti ESPHome ponúka možnosť konfigurovať zariadenia priamo na zariadení, nevyžaduje kompiláciu pri každej zmene, stále je ale nutné pred kompiláciou povoliť niektoré funkcie ktoré sá nenachádzajú v základnej verzii. Informácie o tom ako skompilovať Tasmotu sú dostupné tu https://tasmota.github.io/docs/Compile-your-build/

V konfiguračnom súbore *user_config_override.h* je nutné povoliť TLS a taktiež je dobré povoliť skriptovanie, ktoré nám umožní vytvárať skripty a pomocou nich zložitejšie automatizácie, zároveň je ale nutné zakázať povôdné pravidlá kedže Tasmota nepodporuje to aby boli povolené obidve tieto funkcie. Taktiež nám skriptovanie umožní použiť aj zariadenia ktoré by sme s pravidlami nemohli používať v prípade že Tasmota nepodporuje nejaké ich funkcie.

```
#ifndef USE_MQTT_TLS
#define USE_MQTT_TLS
#define MQTT_PORT 8883

#ifndef USE_SCRIPT
#define USE_SCRIPT
#endif
#ifdef USE_RULES
#undef USE_RULES
#endif
```

Výpis A.14: Špecifikovanie parametrov kompilácie

V prípade že chceme skompilovaný firmware nahrať na viacero zariadeni je možné pridať parametre ktoré budú u všetkých zariadeniach také isté ako napríklad MQTT server, viacej infromácii o parametroch ktoré sú možné takýmto spôsobom zadať je v *my_user_config.h* súbore zdrojového kodu.

Tasmota implementuje TLS pre MQTT pomocou BearSSL[1] a overovanie certifikátov od serveru je možné dvoma spôsobmi.

1. Overovaním certifikačnej autority ktorá vydala certifikát, táto možnosť sa používa v prípade že certifikáty serveru boli vytvorené pomocu LetsEncrypt. Tasmota skontroluje certifikát ktorý dostala od serveru s certifikačnou autoritou LetsEncrypt.

2. V prípade že sa používaju nami podpísane certifikáty, je používaný koncept TOFU[2], Tasmota pri prvom pripojení vyráta odtlačok z verejného klúča pri prvom pripojení, tento odtlačok potom používa na overenie či komunikácia skutočne prišla od servera.

V prípade že chceme overovať certifkačnú autoritu je nutné to špecifikovať pri kompilácii. V prípade že chceme overovať iba odlačok to nieje nutné špecifikovať pri kompilácii. Takže finálny *user_config_override.h* môže vyzerať nejako takto:

[1] https://bearssl.org/
[2] Trust on First Use

```
#ifndef USE_SCRIPT
 #define USE_SCRIPT
#endif
#ifdef USE_RULES
 #undef USE_RULES
#endif

#ifndef USE_MQTT_TLS
 #define USE_MQTT_TLS
#endif
#define MQTT_PORT 8883
#ifndef USE_MQTT_TLS_CA_CERT
 #define USE_MQTT_TLS_CA_CERT
#endif
#ifdef MQTT_HOST
  #undef MQTT_HOST
#endif
#define MQTT_HOST "brokerip"
```

Výpis A.15: Ukážka finálneho user_config_override.h

Ďalej je konfigurácia možná buď pomocou GUI alebo príkazov, kedže konfigurácia pomocou GUI je docela jednoduchá a zaberie viacej času (zariadenie sa pri každej zmene reštartuje) budeme používať *Backlock* prikáz ktorý nám umožní zadať až do 30 prikázov naraz.

V prípade že nenastavíme žiadne parametre pri kompilácii, Tasmota spustí svoj AP na ktorý sa pripojíme. Ako prvé nastavíme AP na ktoré sa má zariadenie pripájať, kedže samotný Tasmota AP neni možné zabezpečiť pomocou hesla, je nutné vypnúť toto Fallback AP, WifiConfig 4 v prípade výpadku pripojenia sa zariadenie pripojí na druhý AP bez reštartu.

V prípade, že nemáme k dispozícii redundatný AP je nutné nastaviť WifiConfig na parameter 5, pri ktorom zariadenie iba počká kým sa objaví znova pôvodný AP bez vytvárania vlastného AP.

```
    Backlog
    SSID <SIID1> ;
    Password1 <HESLOPRESSID1> ;
    SSID2 <SSID2> ;
    Password2 <HESLOPRESSID2> ;
    Wificonfig 4
```

Výpis A.16: Konfigurácia Wi-Fi

Pri konfigurácii MQTT konfigurácia vyzerá podobne, vložíme IP adresu MQTT brokera, port na ktorý sa pripojíme (1883 bez TLS, pri použítí TLS 8883), prihlasovacie údaje pre náš server, v prípade že používame overovanie certifikátov pomocou odtlačkov, sa Tasmota sama naučí odtlačok pri prvom pripojení na server, alternatívne môžeme pomocou nástroja https://github.com/issacg/tasmota-fingerprint získať odltačok a priamo ho vložiť do

zariadenia. Nastavením SetOption19 umožníme HA aby automaticky zdetekoval príslušné zariadenie v sieti.

```
Backlog
MqttHost <> ;
MqttPort <> ;
MqttUser <> ;
MqttPassword <> ;
MqttFingerprint <>;
SetOption19 1
```

Výpis A.17: Konfigurácia MQTT

V prípade že na našich zariadeniach je nutné mať prehľad o čase je nutné špecifikovať NTP server na ktorý sa chcem pripojiť,

Kedže v HA sme nainštalovali NTP server je nutné zmeniť základný NTP server na adresu násho servera, je možné špecifikovať 3 rôzne serveri. Taktiež je nutné nastaviť prechod na letný/zimný čas a samotný posun času, kedže samotné NTP nevykoná zmenu času. TimeZone 99 nastaví časovú zónu podľa parametrov v TimeDST a TimeSTD, ktoré sme nastavili na zmenu času na Slovensku, viac o parametroch o zmene času nájdete na https://tasmota.github.io/docs/Commands/#management

```
Backlog
NtpServer1 <>
NtpServer2 <>
NtpServer3 <>
TimeZone 99
TimeDST 0,0,3,1,2,120
TimeSTD 0,0,10,1,3,60
```

Výpis A.18: Konfigurácia NTP

Prístup ku samotnému Web serveru cez ktorý konfigurujeme zariadenie je nutné zabezpečiť pomocou autentifkácie, podobne ako pri ESPHome je toto len tenká vrstva zabezpečenia kedže prihlasovacie údaje je možné vidieť v odchytenej komunikácii. Pre tento dôvod je nutné vypnúť samotný webserver a zapínať ho iba v prípade administratívnych záležitostí.

```
Backlog
WebPassword <heslo> ;
WebServer 0 ;
```

Výpis A.19: Konfigurácia web servera

Po vypnutí webserveru bude možné zadávať príkazy iba pomocou MQTT takže je dôležité aby bolo zariadenie úspešne pripojené na MQTT server, inak by bolo nutné nahrať nový firmware pomocou UART komunikácie.

Nakoniec aby samotné zariadenie správne fungovalo je nutné nakonfigurovať samotné piny na zariadení, na to slúžia šablóny ktoré sú vo JSON formáte. Tie je možné získať zo širokej databázy zariadení na https://templates.blakadder.com/. A vložiť ich do zaridenia pomocou *Template* príkazu.

```
Template {"NAME":"QS-Wifi-S03","GPIO":[0,0,17,0,160,0,0,0,43,42,21,22,0],"
    FLAG":0,"BASE":18}
```

Výpis A.20: Vloženie šablony

V prípade že pre zariadenie neexistuje už dopredu vytvorená šablóna v databáze zariadení je zistiť funkcie jednotlivých pinov pomocou tohto návodu:

https://tasmota.github.io/docs/Configuration-Procedure-for-New-Devices/

V prípade jednoduchých automatizácii je možné vykonávať automatizácie pomocou UDP multicast komunikácie, táto komunikácia ale neni šifrovaná takže by tieto automatizcáie mali slúžiť skôr iba na redundanciu, pomocou DevGroupShare dokážeme určiť ktoré konkrétne funkcie dokážeme zdielať medzi zariadeniami, viac informácii o tejto funkcii nájdete na https://tasmota.github.io/docs/Device-Groups/.

```
Backlog
DevGroupName <menoskupiny>;
DevGroupShare <IN>, <OUT>;
SetOption85 1
```

Výpis A.21: Konfigurácia UDP multicast komunikácie

Použitím skriptu môžeme spúšťať redundantné automatizácie ktoré sa spustia v prípade výpadku pripojenia na MQTT server, kde sekcia $>D$ obsahuje inicializácie premmených, zaťial čo sekcia $>S$ sa vykoná každú sekundu, skontroluje či je zariadenie pripojené na MQTT server, v prípade že zariadenie neni pripojené vyše 100 sekúnd zapnú reduntantné automatizácie. Viac o skriptovacom jazyku a jeho možnostiach nájdete na https://tasmota.github.io/docs/Scripting-Language/

```
>D
mqttcounter=0
p:devicegrouponline=0

>S
if devicegrouponline==0
then
if mqtts==0
then
mqttcounter+=1
else
mqttcounter=0
endif

if mqttcounter>100
then
devicegrouponline==1
mqttcounter=0
=>Backlog DevGroupName redundanttopic; DevGroupShare 1,0; SetOption85 1
mqttcounter=0
endif

else
if mqtts>0
then
=> SetOption85 0
mqttcounter=0
endif
endif
```

Výpis A.22: Skript pre zapnutie UDP mutlicast v prípade výpadku MQTT