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

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

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

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

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

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Tomáš Čikel
9. srpna 2021

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

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 bezpečnsotným hrozbám a únikom citlivých informácii. 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 [17]. 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 [13]. 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 su 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 su 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 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 sekcii sa zoznámime s prostredím IoT technológii, s IoT zariadeniami, ktoré sa nachádzajú v našej IoT sieti, porovnáme ich systémy a zoznámime sa s protokolmi, 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 iné, 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 možu užívatelia zdielať informácie tvorené buď samotným užívateľom (napr. manuálne nastavenie teploty na termostate) alebo informácie zozbierané z IoT zariadení napr senzorov. Hlavnou funkciou IoT je umožniť užívateľom jedinečnú identifikáciu, označenie, prístup, automatizáciu a kontrolu zariadení kedykoľvek a kdekoľvek pomocou Internetu [2].

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 a 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 [2].

2.1 Zariadenia v našej IoT sieti

Zariadenia v našej sieti pochádzajú od rôznych výrobcov. Ale môžeme ich rozdeliť do dvoch skupín. Zariadenia od spoločnosti Sonoff a zariadenia, ktoré sú od viacerých výrobcov, ale všetky su postavené na platforme Tuya. Všetky sú napájané pomocou 240 V, a okrem zásvuky 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 našej 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 mikročip ja 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. Mikročip taktiež podporuje nízko levelové komunikačmé protokoly ako I2C, SPI, sériovú komunikáciu pomocou UART a taktiež aktualizácie OTA [22]. ESP8266 dokáže operovať v rôznych režimoch:

- Aktívny režim: Mikročip 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 su 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 [22].

BHT-6000 Termostat

Séria termostatov navrhnutá na používanie v domácnostiach, slúži na ovládanie oteplovania vody, podlahy a samotného domu. Komunikujú cez Wifi, môžu podporovať Modbus alebo BACnet komunikačné protokoly. Presnosť meranie 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.

Obrázek 2.1: BHT-6000

[3] Real Clock Timer
[4] Infračervené žiarenie

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 tlačidla	Zobrazovanie informácií a ovladánie zariadenia

Tabulka 2.1: Komponenty zariadenia BHT-6000

BW-SHP6 Zásuvka

Smart zásvuka od spoločnosti BlitzWolf, na platforme Tuya, ktorú je možné kontrolovať cez Wifi alebo fyzicky pomocou tlačidla. Zbiera aktuálne štatistiky o spotrebe elektrickej energie pomocou BL0937 mikročipu. Zariadenie na rozdiel od ostatných zariadení v našej sieti obsahuje `ESP8285` mikročip, 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.

Obrázek 2.2: BW-SHP6 zásuvka

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

Tabulka 2.2: Rozdelenie pinov BW-SHP6

QS-WFI-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 už ovládame naše cielové zariadenie manuálne a iba chceme pridať podporu možnosti ovládať zariadenie aj vzdialene. Každý kanál má maximálne zaťaženie 5A.

Obrázek 2.3: QS-WFI-S03-2

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

Tabulka 2.3: Pripojenie komponent na ESP8266 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 našej sieti ponúka taktiež zapojenie na ochranné uzemnenie. Maximálne zaťaženie vypínača je až 16A, takže sa hodí na monitorovanie a spúšťanie energeticky náročných spotrebičov.

Obrázek 2.4: Sonoff POW

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

Tabulka 2.4: Pripojenie komponent na ESP8266 pre Sonoff POW

Sonoff T3US3C nastenný vypínač

Dotykový 3-kanálový vypínač od spoločnosti Sonoff. Ovládať sa dá pomocou aplikácie cez Wifi alebo fyzicky vďaka dotykovej plochy, ktorá obsahuje LED podsvietenie pre ľahkú navigáciu. Ako jediné zariadenie v našej 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.

Obrázek 2.5: Sonoff T3US3C

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

Tabulka 2.5: Pripojenie komponent na ESP8266 pre Sonoff T3US3C

KS-601 vypínač

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

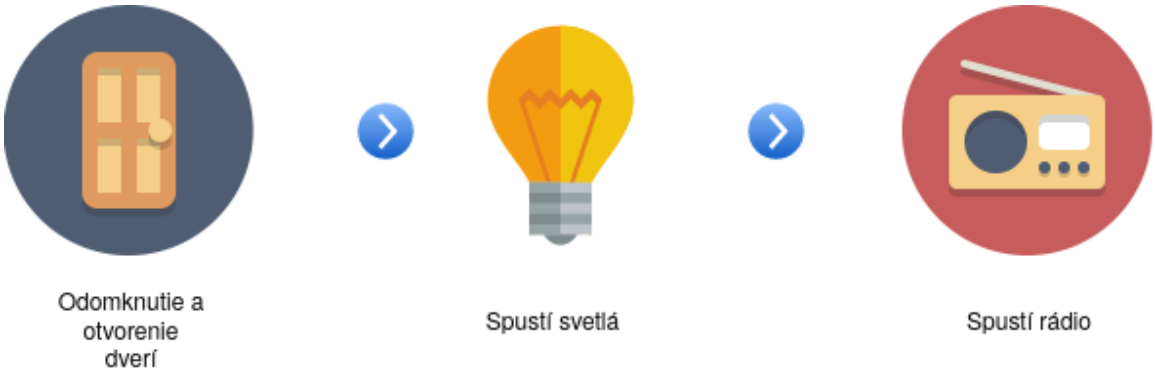
Obrázek 2.6: KS-601

ESP8266 piny	Komponenta
GPIO02	LED
GPIO05	Tlačidlo
GPIO12	Relé 1

Tabulka 2.6: Pripojenie komponent na ESP8266 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.



Obrázek 2.7: Ukážka jednoduchej "smart scéni"

Taktiež môžeme vďaka nim prepojiť zariadenia s inými systémami ako napr. IFTTT[5], Amazaon Alexa, Google Assistant ktoré nám ponúkajú možnosť ovládať zariadenia v našej 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 našej 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

[5]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 su 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 (pridať footnote) 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

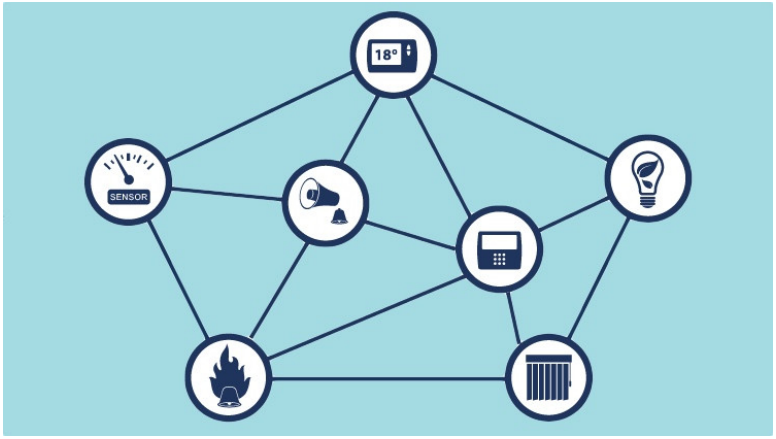
Každé Tuya-based zariadenie ma priradený automatizačný systém od svojho výrobcu. Systémy sú v podstate identické s jemne rozdieľným užívateľským prostredím. Podobne ako systém pre Sonoff zariadenia sa taktiež nachádzajú na cloude a tak isto ako Ewelink je klient týchto systémov podporovaný na platformách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

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 [16]. Preto sa v IoT sieťach používa niekoľko štandartov a technológii. Väčšina domácnosti



Obrázek 2.8: Z-Wave architektúra

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 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.

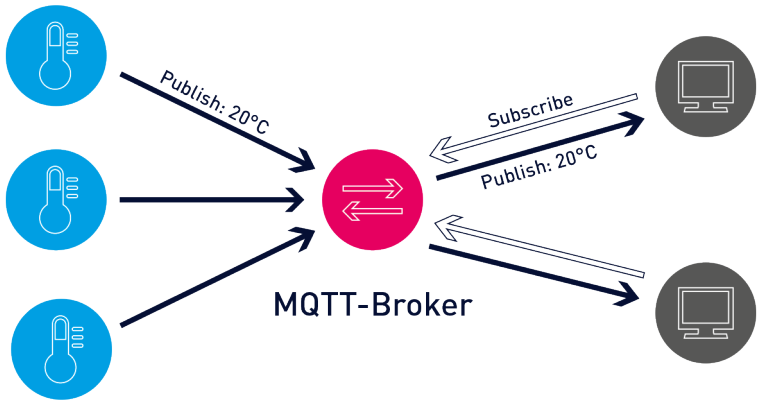
2.3.1 Protokoly na štandarte 802.15.4 a proprietarné protokoly

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 [7].

Zigbee je bezdrátový komunkačný protokol, ktorý na nízkolevelovú komunikáciu používa IEEE 802.15.4. 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, zatiaľ čo Zigbee poskytuje sieťovú a aplikačnú vrstvu [7].

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 [24].

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 protokol. Používa topológiu hviezdy, čo znamená, že zariadenia nemôžu komunikovať priamo medzi sebou [1].



Obrázek 2.9: Architektura MQTT

2.3.2 IoT protokoly v štandarte IEEE 802.11

MQTT protokol beží na TCP protokoli (existuje varianta MQTT-SN, ktorá može 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 [9].

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 [21].

CoAP je M2M[6] 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 [4]. Tento protokol podporuje multicast skupiny, asynchrónne výmeny správ, je komptabilný a ľahko preložiteľný do HTTP, kedže aj jeho metódy su 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. [18].

[6] 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[7] [6]. 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 [23]. V našej sieti je tento protokol používany zariadeniami, ktoré su 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 ".local". MDNS ma vyhradénu IPv4 adresu 224.0.0.251 z rozsahu pre multicast a pre IPv6 je to f02:fb.

[7] DNS over HTTP

Kapitola 3

Bezpečnosť

To čo robí internetom tak doležitým prvkom je prepojenie sieti a fakt že sa môžeme pripojiť do inej siete bez nutného fyzického prístupu, je aj to čo ho robí nebezpečným a to je že potencionálny útočník sa môže nachádzať na hocijakom mieste. IoT siete niesú moc rozdielne než klasické počitačové siete. S tohto dôvodu čelia tieto siete takmer všetkým hrozbám ktorým čeli klasická sieť [12]. V sekcii sa pozrieme na známe bezpečnostné hrozby, vykonáme analýzu komunikácie na to aké informácie posielajú naše zariadenie na cloud a či sú dostatočné zabezpečené, simuláciu mitm útoku ktorý overuje či zabezpečné informácie pomocou TLS sú skutočne zabezpečené a zanalyzujeme ako sa zariadenia správajú v prípade výpadku systému.

Internetovú bezpečnosť možeme rozdeliť na tri základné prvky na ktoré sa treba zamerať pri chápaní bezpečnosti[11].

Dôvernost spočíva v zachovaní súkromia údajov, takže iba oprávneni 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 sme priajali 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 uživatelia a aby nebolo bránené pristupovať autorizovaným uživateľom [11].

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 útočnikovi sa napr. vlámať do domu. Strata dôvernosti vecí, napr kľúče a heslá povedú k neoprávneným hrozbám prístupu do systému [11].

Hrozby autentifikácie môžu viesť k manipulácii so snímacími alebo kontrolnými informáciami. Napríklad neautentizované výstrahy o stave systému môžu zmiasť domového kontrolóra, aby si myslel že nastala mimoriadna situáci a umožní použiť núdzový východ, umožnujúci nezakonný vstup do domu [11].

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 topri pripojení neautorizovaných zariadení do sieťe. 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 záplavi požiadaviek môže viesť k DoS (Denial of Service) útoku [11].

3.2 Bezpečnostné hrozby v IoT

IoT siete obsahujú širokú škálu zariadení na ktorých sa môže bezpečnostné hrozby prejaviť na rôznych úrovniach. Nižšie sa zameriavame na hrozby ktoré sa týkaju hlavne štandartu 802.11, o hrozbách IoT sieťach ktoré používaju sieťe s dlhým dosahom (Lorawan,6LoWPAN) sa dozviete viacej tu [5, 10].

3.2.1 Fyzické hrozby

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.

Prerušením napájania

Napríklad pomocou výpadku elektrického prúdu, je možné odstaviť zariadenia ktoré sú závisle na napájeni z externého zdroja a tým zabrániť funkčnosti zariadenia alebo prípadne prepnutia zariadenia do iného režimu. Zaradenia ktoré používaju na napájanie batériu by nemohli nájsť AP na ktorý boli pôvodne pripojení a mohli by sa automaticky prepojiť do párovacieho 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

Rušíví útok

Je typ DoS útoku ktorý sa nezamierava na konkrétny protokol alebo komunikáciu ale zameriava sa na celkové frekvenčné pasmo ktoré zahlcuje svojim vysielaním a tak neumožnuje komunikovať zariadením v sieti. Narozdiel od ostatných DoS útokov, pri ktorých je výsledkom zahletenie jednej siete alebo zariadenia je možné týmto útokom zahltiť viaceo sietí naraz. Týmto zahltením dokáže útočník jednoducho znefunkčniť sieť čo je preukázané tu [15].

Zábranenie spánku

Tento útok sa zameriava na zariadenia ktoré sú napájané z batérie, neumožnením zariadeniu prístup ku "spánku"zariadenie môže rýchlo vyčerpať batériu minimalizovaný čas počas

ktorého je zariadenie funkčné. Tentot typ útoku je ťažké detekovať kvôli tažkému rozoznaniu medzi skutočnými požiadavkami a tými ktoré maju iba zabrániť spánku u zariadenia [3].

ARP spoofing

Pomocou tohto útoku sa da 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 uživaťeľa 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). [5].

Deautentifikačný útok

Je typ DoS útoku ktorý využíva že manažment rámce v 802.11 štandarte sú neni zabezpečené. Útočník takto dokáže prečítať tieto rámce a pomocou získaných informácii vytvoriť vlastné rámce ktoré vyzerajú ako keby pochádzali od obete útoku. Na tento útok sa vyuzíva deauthentication rámec, tento sa posiela na AP v prípade že klient chce ukončiť spojenie. Útočník posiela rámce ktoré vyzerajú že pochádzaju od klienta ku danému AP ktorý po prijatí ukonči spojenie [14].

3.2.3 Hrozby na vyšších vrstviach OSI modelu

Syn flood

Zatial čo predchádzajúce útoky sa zameriavali skor 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 aočakáva potvrdenie od klienta ktoré ale nikdy nepríde. Takto je port na nejakú dobuzarezervovaný tomuto spojeniu, zatiaľ čo utočník posiela dalšie žiadosti o pripojeniektorým budú priradené dalšie porty. Takto útočník zahltí všetky otvorené porty naserveri a ten už nebude mať porty pre legitímne pripojenia [19].

HTTP flood

Je DoS útok ktorý sa podobá klasickému prístupu na web server, kde útočník posiela opakovane na server HTTP žiadosti čo ma za výsledk zahltenie servera ktorý už nemá časť 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 náročne na výkon ktorého majú IoT zariadenia už aj tak málo. V prípade že sa neimplementujú tieto zabezpečenia sú posielané dáta ľahko viditeľne 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 iniciliazáciu komunikácie od klienta tzv. "handshake"a sám inicializuje komunikáciu

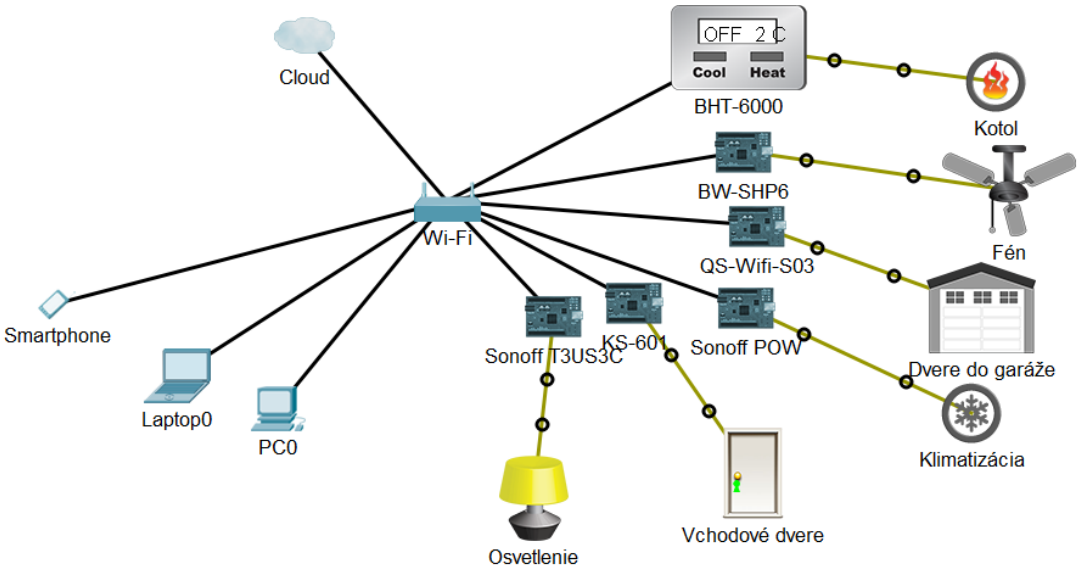
zo serverom. Takto dokáže dešifrovať obidve strany komunikácie, v prípade že klient neoveruje certifikát ktorý dostal od serveru komunikácia prebieha bez toho klient alebo server vedeli že sa medzi nimi nachádza útočník.

Nezabezpečný firmware

Nedostatočne zabezpečený/chybový firmware môže obsahovať spôsob akým útočník prenikne do zariadnia/siete, v prípade že firmware nedostáva aktualizácie ktoré zabezpečujú objavené hrozby môže ostať zariadenie nezabezpečené po celú životnosť zariadenia, tým ponúka útočnikovi veľa možností ako zneužiť hrozby ktorých počet sa bude počas životnosti zariadenia zvyšovať. Tak isto aktualizácie zariadení OTA, v prípade že sú neni zabezpečené ponúkaju útočníkovi možnosť prepísať samotný firmware na zariadení a tým prebrať kontrolu nad zariadením.

3.2.4 Analýza komunikácie

Analýza zariadení v našej IoT sieti bola vyknonaná na každom zariadení zvlášť, zariadená 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. Ďalej na komunikovanie používali rôzne protokoly zabezpečenie komunikácie sa líšilo podľa zariadenia.



Obrázek 3.1: Architekturá siete

Tabuĺka nižšie zobrazuje komunikačné protokoly ktoré naše zariadenia používali nakomunikovanie s automatizačným systémom a smartphonom.

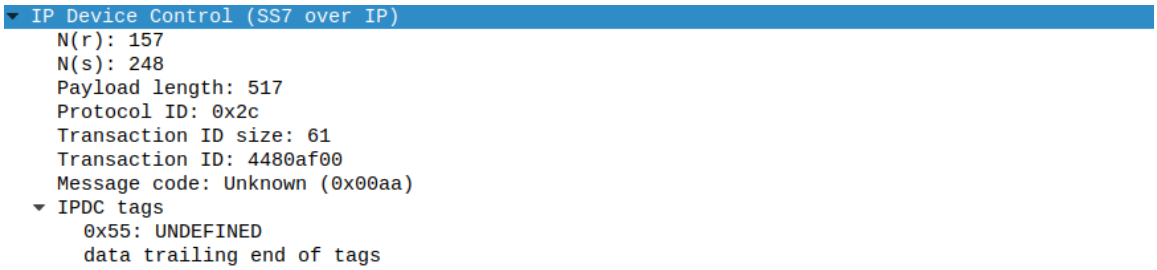
BHT-6000-Termostat, QS-Wifi-S03 Vypínač, BW-SHP6 zásuvka

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 komuni-

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)

Tabulka 3.1: Zariadenia a ich komunikačné protokoly

kácia umožnuje čítať dáta ktoré su posielané, dáta sú neni v čitatelnej forme a komunikácia s automatizačným systémom je šifrovaná.



Obrázek 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 sú neni šifrované, vďaka tomu je vidieť že komunikácia prebiea na HTTP a MQTT protokole. MQTT Hneď potom pošle *Subscribe Request* pripojí sa na topic s menom *smart/device/in/<id>*, zatiaľ čo všetky správy ktoré posiela ku serveru majú topic s menom *smart/device/out/<id>*. Pomcou will message povie zariadenie serveru aku správu a do akého topicu poslať v prípade že sa zariadenie odpojí.

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.

HTTP komunikácia obsahuje rôzne výmeny dát, od vypýtania si mien domén na ktoré by sa zariadenie malo pripojiť až po posielanie dát ako napríklad času, tieto dáta su kódované v neznámom formáte a serializované do *x-www-from-urlencoded*. Z HTTP komunikácie by mohol útočnik získať informácie napríklad a tom kedy sa ma dané zariadenie pripojené

```
▸ 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
```

Obrázek 3.3: Klient sa prihlasuje na MQTT broker

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

Obrázek 3.4: Klient sa prihlási k odoberaniu topicu

zapnúť vďaka tomu že sa nastavené časovače posielaju k serveru. Taktiež by mohol ziskať prístup k dátam ktoré si pýta zariadenie od serveru kedže je v komunikácii vidieť ako pristupuje zariadenie ku serveru, aký token používa a aký klúč 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
```

Obrázek 3.5: Parametre HTTP požiadavku

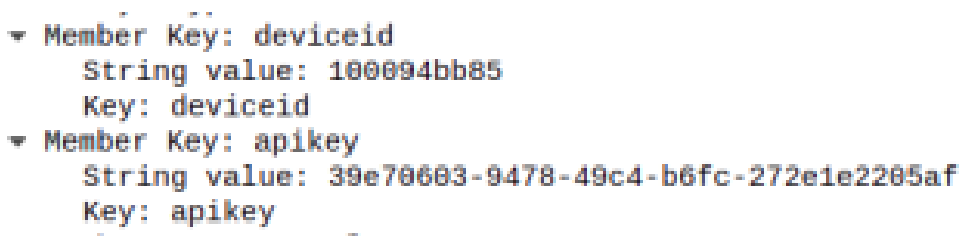
Sonoff T3US3C vypínač a Sonoff Pow vypínač s meraním spotreby energie

Tieto zariadenia si vymenia niekolko správ so smartphonom pri inicializácii zariadenia, kedže je táto komunikácia nešifrována je možné zachytiť citlivé infromácie ako *ID* IoT zariadenia, jeho *api klúč* 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 ktory preposiela komunikáciu medzi nimi. Táto komunikácia je HTTP ale je šifrovaná pomocou TLS1.2 takže zase nieje možné určiť o aký typ komunikácie ide.

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.

[[spravit nanovo screenshoty do spravnej velkosti]]

Obrázek 3.6: Server posiela serializované dát v JSON klientovi.



Obrázek 3.7: Komunikácia 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.

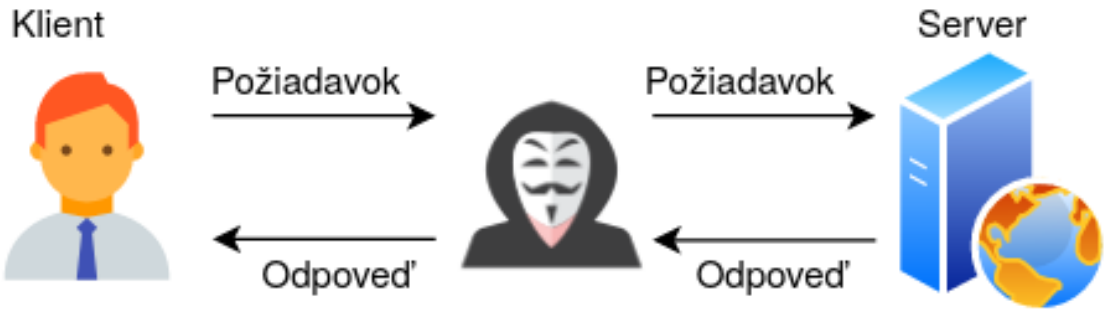
V prípade nešifrovanej komunikácie stačí iba presmerovať komunikáciu k útočnikovi ktorý prepošle dáta serveru, v pripáde komunikácie šifrovanej pomocou TLS útočník musí zachytiť inicializáciu komunikácie tzv. handshake. Počas tohto procesu si členovia komunikácie vymieňajú rôzne informácie, útočníka konkrétne zaujimajú 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 ktorého vytvorí vlastný podvrhnutý certifikát ktorý pošle klientovi. Aby klient dokázal overiť či neni certifikát podvrhnutý, je elektroniky 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 [20].

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 naše zariadenie nejde jednoducho pripojiť na proxy, nebolo možné použiť klasickú proxy kedže pri nej zariadenie treba nastaviť aby sa pripojilo na proxy, trebalo 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.

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

```
▾ Member Key: deviceid
    String value: 100094bb85
    Key: deviceid
▾ Member Key: selfApikey
    String value: 272e1460-aa5e-43ab-ba47-614e09bd1345
    Key: selfApikey
▾ Member Key: iv
    String value: NjU0NDUxMTI0ODg2MzkzOA==
    Key: iv
▸ Member Key: encrypt
▾ Member Key: data
    String value: X1jsIHnGKzjIvdWh+DKoRdoF6b1StJR9zJhhrUI+qlIGT9o2YGy58ZeO6JZ+C6Qa
    Key: data
```

Obrázek 3.8: Komunikácia v lokánom móde medzi smarthphone a Sonoff zariadením



Obrázek 3.9: MITM útok

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

```
10.42.0.63   10.42.0.1    TCP      30579 → 443 [SYN] Seq=0 Win=5840 Len=0 MSS=1460
10.42.0.1    10.42.0.63   TCP      443 → 30579 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
10.42.0.63   10.42.0.1    TLSv1.2  Client Hello
10.42.0.1    10.42.0.63   TCP      443 → 30579 [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 → 30579 [ACK] Seq=87 Ack=85 Win=64156 Len=1460 [TCP segment of a reassembled PDU]
10.42.0.63   10.42.0.1    TCP      30579 → 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      30579 → 443 [FIN, ACK] Seq=85 Ack=1999 Win=5388 Len=0
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 → 30579 [FIN, ACK] Seq=2006 Ack=86 Win=64155 Len=0
10.42.0.63   10.42.0.1    TCP      30579 → 443 [RST, ACK] Seq=86 Ack=2006 Win=5840 Len=0
10.42.0.63   10.42.0.1    TCP      30579 → 443 [RST, ACK] Seq=86 Ack=2007 Win=5840 Len=0
```

Obrázek 3.10: Ukážka komunikácie kde Sonoff overuje certifikát

V prípade Tuya-based zariadeni 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 na strane automatizačného systému ale môže ísť aj o DoS alebo DDoS útok na automatizačný systém alebo zariadenia a IoT sieť na ktorej prebieha komunikácia.

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

Tuya

V prípade Tuya zariadení keď nebol dostupný systém ovládanie pomocou aplikácie bolo ďalej funkčné vďaka ovládaniu pomocou IPDC protokolu, taktiež bol funknčný časovač o ktorý sa dal nastaviť na klientovi aj v prípade nedostupného systému. Nebolo možné nastaviť rozvrh pravidelného vypnutia/zapnutia zariadenií. Ak bol rozvrh nastavený už pred výpadkom spojenia zo sýstémom tak aj tak zariadenia nereagovali podla 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 našej sieti napr. zámok ktorý sa pravidelne zamýka na noc. Viac komplikované zariadenie ako zásuvka BW-SHP6 a termostat BHTW-6000 neposkytovali, 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, kedže zariadenie neponúka možnosť manuálne vypnúť/zapnúť cielové zariadenie, by potencionálny útočník mohol deaktivovať zariadenie na veľmi dlhú dobu keďže sa zariadenie zapája priamo do elektrickej siete a bežný uživateľ nemúsí mať schopnosti, znalosť alebo dostupné zariadenie na ktoré by reinštaloval cielové zariadenie. 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 zariadeni 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 odosť viacej pohodlný a rýchlejší kedže nepotrebujeme fyzický prístup ku zariadenu. Toto ale môže využiť útočník na nahratie vlastného firmware v prípade že táto funkcia neni zabezpečená. Podobne ako open-source Arduino knihovny ktoré ponúkaju minimálne zabezpečnie pre OTA aktualizácie [8], boli niektoré zariadenia nedostatočne zabezpečené.

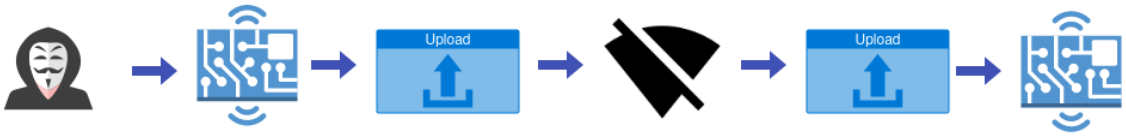
Sonoff

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 ssl útoku čo sme aj overili vyššie, ten umožnoval získať informácie ktoré umožnili tento útok.

[3] https://github.com/mirko/SonOTA

Tuya

Pre Tuya zariadenia bol použitý nástroj Tuya-convert[4] pomocou ktorého bolo možné úspešne nahráť vlastný firmware na naše zariadenia, nahrať firmware takýmto spôsobom bolo možné iba keď je 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 sa pri opätovnom spustení 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 4.2. Kde by útočník fyzickým prístupom k zariadenu prepol zariadenie do párovacieho módu, nahral škodlivý firmware[5] ktorý by vykonával deautentifikačný útok a po skončený by nahral pôvodný firmware ktorý predtým stiahol, takže uživateľ by si všimol iba to že zariadenie nebolo chvílu dostupné, v prípade že není často používane by si ani nevšimol že sa niečo so zariadením stalo.



Obrázek 3.11: Nahratie firmware pomcou OTA

3.2.8 Známe a objavené hrozby v našej sieti

Analýzou komunikácie sme objavili 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 sme neoverili mitm útok u Tuya zariadení môžeme predpokladať že pri OTA sa používa tento útok na získánie informácii k nahratiu firmware podobne ako sa používal pri SONOTA[6].

Niektoré s týchto hrozieb môže zabezpečiť samotný výrobca zariadenia pomocou aktualizácii, ale samotná aktulizácia nemusí nikdy prísť. Zatial 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], naša zásuvka nedostala žiadnu aktualizáciu ktorá by zabránila tomuto útoku. Preto na zabezpečenie siete a zariadení je treba použiť open-source systémy ktoré ponúkajú riešienia na najdené hrozby.

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

Útoky, konkrétne hrozby	**Navrhnuté riešenia**
Zdielanie citlivých informácii 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 automatizácii priamo na komunikáciu medzi zariadeniami a priamo na zariadenia

Tabulka 3.2: Nájdené hrozby v IoT zariadeniach

Kapitola 4

Open-source systémy

V sekcii sa porovnáme rôzne open-source systémy a firmware pre zariadenia. Implementujeme ich do našej siete a porovnáme výsledky z oficialnými automatizačnými systémami.

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

- Uživateľ 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 spolahlivosť už iba vďaka tomu že sa nachádzaju v lokálnej sieti a tým uberajú potencionálny bod zlyhania.
- Oproti oficiálnym systémom ponúkaju o dosť vicej prepojení a integracii so externými systémami.

Medzi nevýhody open-source systémov patrí:

- Nutné zariadenie v lokálnej sieti na ktorom bude systém bežat často sa jedná o jednodeskový počítač ako napr Raspberry Pie.
- Kedže sa sýstém 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

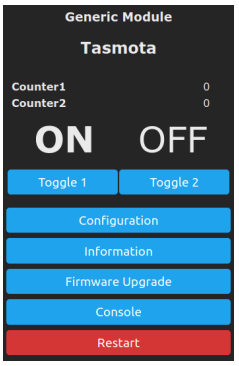
Oficiálny firmware na zariadeniach zvyčajne podporuje integráciu len pod oficiálny sýstém, z tohto dôsledku treba nahrať custom firmware na zariadenia vďaka ktorému ich integrujeme pod naše open-source systémy. Samotný firmware treba ale nakonfigurovať pre konkrétne zariadenie, preto je treba zistiť na ktoré konkrétne piny su pripojené jednotlivé komponenty zariadenia. Z tohto dôvodu je dobré si vybrať firmware z veľkou komunitou ktorá

medzi sebou zdieľa rozdelenie vstupno/výstupnich pinov zariadení. Pri výbere firmwaru sa zameriavame na nasledujúce na tieto 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
- Zdielanie 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 čí 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 aj 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 šabloni ktoré sú vo formáte JSON a ponúkaju jednoduchý spôsob vytvárania,upravovania a zdielania konfiguracii pre rôzne typy zariadenia. Vďaka týmto šablonám ponúka veľku databázu dopredu vytvorených konfigurácii hlavne pre komerčne dostupné IoT zariadenia. Automatizácie je možne vytvoriť pomocou časovačov a pravidiel ktoré majú jednoduchú syntax: `on <trigger> do <command> endon`, alebo komplikovanejšie automatizácie je možné vytvoriť pomocou skriptovacieho jazyka. 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 overvením certifikačnej autority alebo pomocou SHA256 odltačku vytvoreného z verejného klúča. Pre Tuya zariadenia ako BHT-6000 termostat TuyaMCU kde sa dajú mapovať Tasmota funkcie na sériovu komunikáciu ktorouo komunikuj ú Esp8266 a hlavný mikrokontrolér. Aktualizácie je možné vykonať buď pomocou nahratia frimware OTA alebo stiahnutím priamo na zariadení s web servera.



Obrázek 4.1: Tasmota GUI

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

Obrázek 4.2: Šablóna pre BW-SHP6

ESPHome

Esphome je open-source nástroj ktorý pomocou konfiguračných subórov dokáže vytvoriť vlastný firmware na ESP8266 a ESP32 mikročipy[1]. Esphome používa jednoduché yaml konfiguračne súbory vďaka ktorým si uživaťel môže vytvoriť firmware podla seba. Zaťial 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 zariadený, sa celá konfigurácia zariadenia deje ešte pred nahratím firmware na zariadenie. V prípade zmeny je nutné prepísanie celého firmware, zaťial čo pri tasmote sa iba reštartuje zariadenie. Zaťial čo Tasmota používa na automatizáciu pravidlá tak Esphome používa pre komplikovanejšie automatizácie Lambda funkcie, ktoré sa píšu v C++. Esphome podporuje komunikáciu pomocou MQTT 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[2]. Pri tomto 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 a viac zameriava na elektrické komponenty ako senzory, displeje atď.

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

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

api:

logger:

ota:
```

Obrázek 4.3: ESPHome konfigrácia

[1] https://esphome.io/
[2] https://esphome.io/components/api.html

	Tasmota	**Esphome**
Typ nástroja	Firmware	Nástroj na vytváranie firmware
Konfigurácia	GUI,Príkazy	Yaml súbory
Automatizácia	Pravidlá,skriptovanie	Akcie v yaml ,Lambda funkcie
Zmeny v konfiguracii	Priamo na zariadení	Nutná kompilácia a nahratie nového firmware
Aktualizácie	OTA bin súbory, URL	OTA bin súbory
Zdielanie konfigurácii	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

Tabulka 4.1: Porovnanie Tasmota a Esphome firmare

4.2 Open-source systémy

Tasmota a Esphome firmware nám umožnujú kontrolovať zariadenie ale v prípade že chceme kontrolovať viacero zariadeni a skutočne zautomatizovať domácnosť je potreba mať automatizačný systém. V prípade open-source automatizačných systémov sa zameriavame na tieto body:

- Správa systému: vytváranie uživaťelov, ich autentifikácia a zdielanie kontroly nad zariadeniami medzi nimi
- Konfigurácia: Akým spôsobom sa konfiguruje systém
- Aumatizácia: pomocou akych nástrojov sa vytváraju automatizácie a aké komplikované automatizácie je možné vytvoriť
- Komunikácia: aké protokoly a ich zabezpečenie systém ponúka
- Integrácie systémov: akú veľkú datbázu integrácii ponúka systém
- Vzdialený prístup: či systém ponúka cloud službu, alebo iný spôsob ako pristúpiť ku 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 jednodeskové počítače alebo Synology NAS server. OpenHAB ponúka balíky tie ponúkaju rôzne funkcie pre uživaťela podľa toho čo potrebuje, v balíkioch ponúka rôzne typy UI a rozšírení. Konfigurácia sa nastavuje v PaperUI v ktorom sa ale nedá všetko nastaviť pomocou GUI a niektoré funckie sa musia nakonfigurovať pomocou konfiguračných súborov. OpenHAB ponúka širokú škálu rozšírení (addonov) ktoré pridávaju ďalšiu funkcionalitu 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 zo zariadeniami je podporovaná pre MQTT, Z-Wave a ZigBee protokoly. Vzdialený pristúp je nutné použiť VPN alebo sprievodnú openHAB Cloud služby ktorá poskytuje bezplatný vzdialený prístup. Taktiež ponúka sprievodnú aplikáciu pre systém Android a IoS. Oproti oficiánlnym automatizačným systémom tu chýba možnosť vytvorenia viaceých uživateľov a zdieľanie automatizácie a zariadení s nimi. Jednoduché pravidlá v OpenHAB môžu mať takýto formát:

```
rule "<RULE_NAME>"
when
    <TRIGGER_CONDITION>
then
    <SCRIPT_BLOCK>
end
```

Výpis 4.1: Ukážka jednoduchého pravidla

Home Assistant

Home assistant (neskôr už iba HA) je open-source automatizačný systém napísany v Python, ktorý je optimalizovaný pre jednodeskové počítače ako napríklad Raspeberry Pi[3]. Systém poskytuje jednoduchý spôsob na aktualizovanie systému a vytvárania snapshotov pre uloženie momentálnej konfigurácie. Aj keď sa časť systému da nakonfigurovať podobne ako v openHAB v GUI, sa väčšina konfigurácie vykonáva podobne ako v Esphome pomocou yaml konfiguračných súborov. HA na rozdiel od OpenHAB ponúka systém pre manažovanie uživateľov 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 uživateľ ich jedným kliknutím prídá 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 4.2: Jednudchá automatizácia

- Pomocou grafického editoru, HA automaticky vytvárá yaml konfiguračný súbor.
- Automatickým konvertovaním textu ktorý zadá uživateľ, systém sa s neho snaží vyčítať čo presne uživateľ chcel zautomatizovať, táto funkcia sa dá použiť iba na jednoduché automatizácie a není spoľahlivá, v našom prípade sa nepodarilo vytvoriť ani jednoduchú automatizáciu.

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

- V prípade veľmi komplikovaných automatizácii sa dá použiť sandboxové prostredie AppDaemon, je to viacvláknové Python exekučné prostriedie na písanie komplikovaných automatizácii pre HA.

```python
import appdaemon.plugins.hass.hassapi as hass


class OutsideLights(hass.Hass):
    def initialize(self):
        self.run_at_sunrise(self.sunrise_cb)
        self.run_at_sunset(self.sunset_cb)

    def sunrise_cb(self, kwargs):
        self.turn_on(self.args["off_scene"])

    def sunset_cb(self, kwargs):
        self.turn_on(self.args["on_scene"])
```

Obrázek 4.4: Jednoduchá automatizácia v AppDaemon

Tak isto ako OpenHAB ponúka HA širokú škálu rozšírení, medzi ne patria aj špecifické rozšírenia pre vyššie spomenutý firmware. Ako TasmoAdmin ktorý nám umožnuje manažovať a zauktualizovať všetky Tasmota zariadenia naraz, alebo 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.

	Home Assistant	**OpenHAB**
Správa systému	Ponúk manažment uživaťelov, ich autentikáciu, multifaktorovú autentifikáciu	Nepnúka manažment uživaťelov a ich autentifikáciu
Konfigurácia systému	Yaml súbory,GUI	PaperUI, konfiguračné súbory
Aumatizácia	Yaml súbory, Grafický editor, konvertovaním 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 rozširenia	Rozšírenia pre špecifický firmware	Celkovo viacej integrácii
Cloud	30 dňová skúšobna verzia, DuckDNS a Tor rozšírenia	Zadarmo ale neobsahuje podporu

Tabulka 4.2: Porovnanie a Esphome firmare

4.3 Implementácia v našej IoT sieti

Pre našu implementáciu sme sa rozhodli 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 s tohto dôvodu sme použili špecifický Wthermostat firmware pre toto zariadenie. Zatial čo tento firmware ponúka plnú funkcionalitu oficiálneho firmware a dá sa ovládať pomocou MQTT, neponúka šifrovanie MQTT alebo HTTP ku prístupu na web server, taktiež nedá sa zaheslovať prístup ku web serveru alebo vypnúť web server ako je možné pri Tasmota. Pre automatizačný systém sme sa rozhodli pre HA vďaka ponuke použítelných rozšíreni a bezpečnostných opatrení. Pri vyriešení objavených hrozieb a konfigurácii našej sieti sme postupovali v tomto poradí:

1. Návrh siete: rozdielenie 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

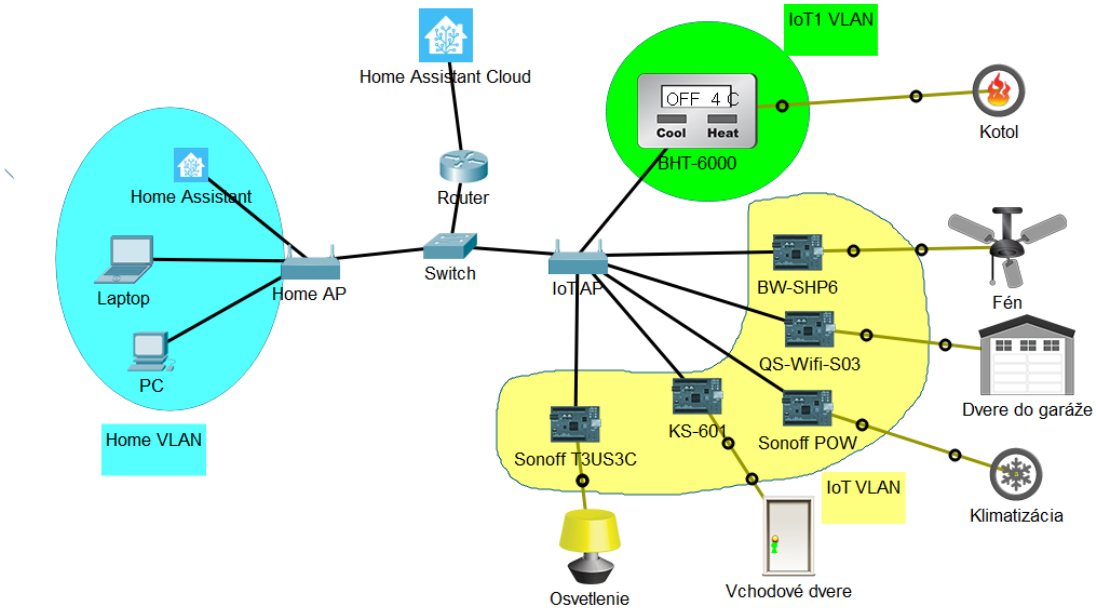
Návrh siete

Zatial čo IoT zariadenia ktoré používajú Tasmotu budú v jednej skupine, BHT-6000 bude mať samostatnú skupinu kedže v našej implemntácii používa iný firmware ktorý neni zabezpečený dostatočne, toto nám taktiež umožní že v prípade že dáme na toto zariadenie pôvodní firmware, možme povoliť komunikáciu s oficiálnym systémom bez vystavenia ostatných internetu. Ostatné zariadenia spolu s HA inštaláciou budú v jednej sieti kvôli jednoduchej integrácii zariadení v sieti pod automatizáci (HA môže pomocou ping zariadení určiť kto sa náchadza v dome).

Pri samotnej implementácii je nutné mať router/switch ktorý dokáže používať VLAN. Nastaviť rozsah IP adries pre špecifické VLANi a nastaviť Firewall pravídla ktoré umožnia/eliminujú komunikáciu ktorá buď prichádza alebo vchádza do VLAN. V našej implementácii povolujeme porty 1883,8883 pre MQTT a MQTT ktoré je zašifrované pomocou TLS. Taktiež port 123 ktorým pomocou NTP protokolu synchrozujeme čas na našich zariadeniach. Pri Tasmota zariadeniach otvoríme tento port taktiež pre pripojenie na internet, v prípade výpadku/poruchy HA tak zariadenia budú stále zariadenia schopné zosynchronizovať svoj čas čo je nutné pri automatizáciach.

	Home VLAN	**IoT1 VLAN**	**Internet**
IoT VLAN	8883,123	-	123

Tabulka 4.3: Povolené porty medzi IoT a ostantnými VLAN



Obrázek 4.5: Rozdelenie siete do VLAN

	Home VLAN	**IoT VLAN**	**Internet**
IoT1 VLAN	1883,123	-	-

Tabulka 4.4: Povolené porty medzi IoT1 a ostantnými VLAN

4.3.1 Konfigurácia Home Assistant

Inštalácia HA je jednoduchá vďaka image ktoré HA ponúka. V základe HA neobsahuje príliš veľa funkcii ktoré neskôr rozšírime pomocou inštalácie rozšírení, HA v základe dokáže komunikovať iba s Esphome zariadeniami pomocou jeho NativeAPI, a samotný systém obsahuje niekoľko bezpečnostných hrozieb ktoré je nutné odstrániť:

- Nezabezpečená HTTP komunikácia – útočník môže získať prihlasovacie údaje do systému
- V prípade útoku hrubou silou neexistujú žiadne opatrenia ako zabrániť tomuto útoku, takže potencionálny útočník by mohol neobmedzene skúšať prihlasovanie do systému

Konfigurácia systému prebiehala v týchto bodoch kde ako prvé sme eliminiovali hrozby ktoré systém mal, a postupn pridávali ďalšie funkcie do systému inštalovaním rozšírení:

- Zabezpečenie šifrovania pomocou TLS - pomocou openssl sme vygenerovali nami podpísane certifikáty, ktoré sme vložili do systému. 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 sme nastavili parameter *ip_ban_enabled*[4] ktorý automaticky odmietne pokusy o pirhlásenie z ip adresy z ktorej prekročil nami 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
```

Obrázek 4.6: Konfigurácia v configuration.yaml

- Vzdialený prístup sme zabezpečili pomocou Tor addonu ktorý nám umožní pristupovať ku HA ako "onion"stránke pomocou Tor služby pre skrývanie služieb[5]. Výhoda použitia tohto addonu je to že nepotrebujeme otvoriť žiadne porty na našom routri a nemusíme nastaviť žiadne TLS certifikáty. Stači nastaviť port na ktorý sa budeme pripájať a spustiť tento addon, v logu nášho addonu sa objavila doména na ktorú sa potom pripojíme 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: ------------------------------------------------------------
```

Obrázek 4.7: Tor log

- Pre dalšiu vrstvu zabezpečenia pri autentifikácii do systému sme pridali multifaktorovú autentifikáciu, k tomuto je nutné použiť aplikáciu ako Google authentificator alebo Authy, konkrétne použijeme TOTP[6] algoritmus. Po spárovaní sa každých 30 sekúnd generuje šest číslene heslo ktoré treba zadať do HA pre úspešné príhlasenie . Táto služba je dôležitá hlavne pri používani vzdialeného prístupu pomocou rozšíreni Tor alebo DuckDNS kde domény na prihlásanie do systému sú verejné dostupné a hocikto sa na ne môže pripojiť.

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

Obrázek 4.8: Konfigurácia pre spustenie viacfaktorovej autentifikácie

- Kedže naše zariadenia používaju MQTT potrebujeme MQTT server, v prípade že už máme server je možné ho priradiť do systému, jednoduchší spôsob je ale nainštalovať Mosquitto rozšírenie a nakonfigurovať ho aby používalo TLS. Taktiež sme nakonfigurovali prihlasovacie údaje pre naše zariadenia ktoré boli jedinečne pre každé zariadenia. V takomto prípade prí získani údajov pre jedno zariadenia útočník nezíska prístup ku všetkým zariadeniam.

[4] https://www.home-assistant.io/integrations/http/
[5] https://2019.www.torproject.org/docs/onion-services
[6] Time-Based One-Time Password-https://tools.ietf.org/html/rfc6238

```
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
```

Obrázek 4.9: Ukážka konfigurácie pre Mosquitto server

- Nainštalovnaím NTP rozšírenia sme sprevádzkovali lokálny NTP server ktorý umožní našim zariadeniam synchronizovať čas bez nutnosti pripojenia k internetu. Na ktorom iba špecifikujeme server z ktorého chceme získavať čas.

V tomto bode už je systém dostatočne zabezpečený a plne funkčný, ešte ale možeme použiť rozšírenia ktoré nám zlepšia samotnú administratívu systému a zariadení:

- Rozšírenie TasmoAdmin umožní efektívne manažovať naše tasmota zariadenia, aktualizovať zariadenia a konfigurovať ich z jedného miesta pomocou GUI. Rozšírenie používa vlastný server ktorý sme zabezpečili znova pomocou TLS. Toto rozšírenie ale komunikuje zo 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 aktualizáie zariadení.

Posledným bodom je už iba zálohovanie systému pomocou snapshotov ktoré HA ponúka, samotný snapshot sme zabezpečili heslom kedže by z neho útočník mohol získať citlivé informácie o našom systéme.

4.3.2 Firmware

Firmware bol nahratý vyššie spomenutým nástrojom Tuya-convert pre Tuya zariadenia a nástrojom Esptool ktorý používa UART komunikáciu pre Sonoff zariadenia. Wthermostat firmware ponúka možnosť nakonfigurovať zariadenia iba pomocou GUI podobného Tasmote, kde nastavíme MQTT server a NTP server na ip adresu pre náš HA pre časovu synchronizáciu. Firmware taktiež ponúka moznosť 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 ať.d...

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

3. 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 sme mohli zašifrovať komunikáciu bolo nutné skompilovať Tasmotu, a povoliť TLS šifrovanie, taktiež sme povolili skriptovanie ktoré nám umožní vytvárať silnejšie automatizácie a správne kontrolovať QS-WIFI-S02 vypínač. Tasmota momentálne nepodporuje počítadla frekvenciue 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 sme vykonali pomocou PlatformIO.

Konfigurácia je možná pomocou GUI alebo príkazov ktoré sa dajú použiť buď v konzoli, pomocu MQTT protokolu alebo piužítám sériovej komunikácie. Kedže konfigurácia GUI si vyžaduje časté retartovanie zariadenia je veľmi zdláhava preto je efektívnejšie konfigurovať priamo pomocou príkazov pri ktorých je možné použiť prikáz 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
```

Obrázek 4.10: Konfigurácia v *my _user_config.h*

Wi-Fi

V základných nastaveniach nastavíme SSID a heslo primárneho AP a sekundárneho redundaného AP a parametru Wificonfig na 4. Takto pri výpadku nášho primárneho AP, Tasmota nebude vytvárať vlastný AP ktorý neni zabezpečený heslom ale pripojí sa na náš redundantný AP. V prípade že nemáme k dispozíci 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 4.3: Wifi konfigurácia

MQTT

Po skompilovaní a nahratí firmware na zariadenia je nutné nastaviť IP adresu servera a port na ktorom sa bude komunikovať, ďalej autentifikáčné údaje pre príhlasenie 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 našej implementácii sme použili nami podpísane certifikáty, tým že je nutné sa naučiť odtlačok pri prvom pripojení je možné vykonať mitm útok pri prvom pripojení, kde by sa Tasmota naučila útokčnikov odtlačok, preto sme dopredu vypočítali odtlačok pomocou nástroja *Tasmota-fingerprint*[7] a vložili ho priamo do zariadenia. Nakoniec nastavili parameter *SetOption19* ktorý umožní automaticky detekovať naše zariadenie pomocou MQTT pre HA.

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

Výpis 4.4: MQTT konfigurácia

Hlavný NTP server sme zmenili na náš lokálny ktorý beží na HA a nastavili kedy sa posúva čas aby v prípade použiťia 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 4.5: Konfigurácia NTP a času

Posledným bodom je samotné nakonfigurovanie funkčnosti zariadenia pomocou šablón, kde z databázi[8] zariadení nájdeme šablony, 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 4.6: Vloženie šablony pre BW-SHP6

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 s tohto dôvodu by automatizácie mali prebiehať v automatizačnom systéme. V prípade že sa preruší z nejakého dôvodu (porucha,DoS útok atď) komunikácia z HA sme vytvorili jendnoduchý skript ktorý v prípade výpadku, spustí túto redundantnú kominkáciu a automatizácie ostanú funkčné. V prípade že sa pripojenie obnoví tieto zase vráti zariadenia do pôvodného stavu. Príkazom DevGroupShare špecifikujeme ktoré funkcie chceme zdielať a príjmať. Skript už iba povoluje a vypína funkciu pomocou SetOption85.

```
Backlog
Grouptopic redundanttopic
DevgroupShare 1,0
```

[7] https://github.com/issacg/tasmota-fingerprint
[8] https://templates.blakadder.com/

```
SetOption85 1
```

Výpis 4.7: Konfigurácia skupín

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

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

Výpis 4.8: Konfigurácia autentifkácie a vypnutie web servera

4.3.3 Testovanie a porovnanie s oficialnými automatyzačními systémami

Podobne ako pri oficiálnych automatizačných systémoch bola komunikácia nahravaná pomocou nástroja Wireshark. Všetky Tasmota zariadenia v sieti komunikovali pomocou TLS, zatiaľ čo BHT-6000 nemalo šifrovanú komunikáciu. Vykonaním mitm útoku sme overili že všetky naše zariadenia ktoré používaju TLS overujú certifikát.

1. BHT-6000 - nezabezpečená MQTT komunikácia Wthermostat firmware umožnuje čítať citlivé infomrácie s ktorých by útočník napríklad vedel vyčitať či sa nachádza obyvateľ domu v domácnosti podla nastavenej teploty. Taktiež by sa zo získania prihlasovacích údajov mohol prihlásiť na MQTT server a odoberať/posielať správy do sieťe.

```
▸ 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
```

Obrázek 4.11: MQTT prihlasovacie údaje na náš server

2. Deautentifikačný a rušívý DoS - kedže všetky naše zariadeni používaju Esp8266 ktorý podporuje iba komunikáciu ktorá nešifruje manažment rámce ktoré tento útok používa je stále možné vykonať tento DoS útok na našu sieť.

3. 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 spušťať rôzne automatizácie.

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

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

Obrázek 4.12: Nešifrované UDP správy do multicast skupiny

4. Fyzický prístup - aj keď majú naše zariadenia zabezepečené OTA aktulizácie v prípade že útočník získa fyzický prístup može prepísať firmware pomocu UART komunikácie alebo stiahnuť firmare a získať z neho citlivé informácie ako napr. prihlasovacie údaje na Wi-Fi, pomocou decode-config nástroja ktorý slúži na dekódovanie [9].

Oproti pôvodnej sieti s oficiálnými automatizačními systémami naša implementácia Pri porovnaní s pôvodnou sieťou s oficiálnými automatizačními systémami a oficiálnym firmware obsahuje naša sieť tieto vylepšenia:

- Zašifrovaná komunikácia - všetky naše zariadenia okrem BHT-600 majú zabezpečenú komunikáciu pomocou TLS, a overujú certifkát takže sa úspešne bránia mitm útoku.
- Automatizačný systém v lokálnej sieti - presunutím systému do lokálnej siete sme zarbánili úniku citlivých informácii mimo našu lokálnu sieť taktiež sme zabránili nefukčnosti automatizácii v prípade výpadku internetové pripojenia, zaťial čo sme zachovali schopnosť sa vzdialene a bezpečne pripojiť na HA. Taktiež sme tento systém zabezpečili proti útokom hrubou silou pomocou viacfaktorovej autentifkácie a zakazovním ip adries z ktorých príchadza útok.
- Rozdelenie siete do VLAN - rozdelním našej sieti do rozdielnych VLAN a povolením iba MQTT a NTP komunikácie sme zabránili k prístupu nezabezpečeného BHT-6000 zariadenia a efektívne obmedzili možnosti vykonania DoS a iných útokov na našu sieť v prípade že by útočník získal kontrolu nad nejakým zariadením.
- Redundance automatizácii - presunutím automatizácii ako spustenie zariadenia podľa času priamo na zariadenia, a vytvorením reduntaných automatizácii pre jednoduché automatizácie sme zabezpečili že niektoré automatizácie budú funkčné aj v prípade výpadku samotného HA v našej lokálanej sieti.
- Open-source firmware - využitím open-source firmware sme zabezpečili možnosť aktualizovať firmware OTA, ale zabránili možnému stiahnutiu alebo nahratia firmware OTA bez nášho povolenia. Taktiež sme pridali druhý AP na ktorý sa zariadenia pripojí, v prípade výpadku hlavného AP čim sme znova zvýšili redundanciu zariadení.

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

Kapitola 5

Diskusia a doporučenia na vytvorenie bezpečnej IoT siete

5.1 Diskusia

[[**spravit uvod a rozpisat**]] Aj keď je výsledkom našej implementácie sieť ktorá obsahuje odosť menej bezpečnostných hrozieb stále obsahuje nejaké hrozby ktoré by sa modali vyriešiť

- BHT-6000: aj keď samotný autor tohto firmware neplánuje pridať TLS podporu[1] je možné že niekto iný príde podporu kedže je tento firmware open-source a často modifikovaný komunitou [2]. Taktiež Tasmota postupne podporuje viacej zariadení ktoré používaju TuyaMCU takže je možné že sa v budúcnosti objaví konfigurácia pre toto zariadenie vďaka komunite. Alternatívním riešením môže byť používanie oficiálneho firmware a integrácia Tuya platformi do HA, kde by sme síce mali zabezpečenú komunikáciu pomocou TLS ale komunikácia by prebiehala cez cloud a mohla by byť napadnutá mitm útokom a zariadenie by nebolo možné automatizovať v prípadfe výpadku internetového pripojenia.

- Automatizácie medzi zariadeniami: Kedže táto komunikácia predstavuje riziko by mala byť použíta iba na redundanciu automatizácii. A aj v tomto prípade je nutné sa zamyslieť aké automatizácie chceme takto ovládať, zatial 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. Záleži od samotnej implementácie ako moc je redundancia konkrétnej automatizácie dôležitá a aké veľké riziko by predstavovalo spustenie tejto automatizácie v inom momente ako je naplánovane uživateľom.

- Deutetifikačný útok a rušivý Dos útok: Zabrániť takýmto útokom nie je možné zmenou firmware alebo automatyzačného 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 manažovateľne rámce ktoré sú na tento útok používane [25]. Tento štandart ale nepodporuje samotný Esp8266 mikrokontrolér a ani vetšina Wi-Fi zariadení.

- Fyzický prístup: Aj keď sa tejto hrozbe nedá úplne zabrániť, útočník s dostatkom času a hrubej sily by sa pravdepodobne dokázal dostať ku zariadeniu a nahrať/stiahnuť

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

firmware z tohto zariadenia. Skoro všetky naše zariadenia su navrhnuté na umiestnenie buď priamo do steny alebo elektrickej skrine okrem BW-SHP6 zásuvky. Toto je ale pri našich konkrétnych zariadeniach veľmi nepravdepodobné a v prípade že sú naše zariadenia zapnuté v elektrickej sieti je nutné tieto zariadenia odpojiť zo sieti aby sa bolo možné dostať ku UART komunikácii, už len toto samotné notifikuje uživateľa že zo zariadením neni niečo správne kedže z ničoho nič prestalo fungovať. U zariadení BHT-6000 a BW-SHP6 je dokonca nutné spájkovať samotné drôty ku UART pinom.

5.2 Doporučenia pre vytvorenie bezpečnostnej sieti

1. Použitie open-source systému a firmware - v prípade že chceme zabezpečiť sieť musíme zvoliť open-source firmware, ako sme preukázali oficiálne systémy posielajú mimo lokálnu sieť nezabezpečenú komunikáciu s ktorej je možné získať citlivé informácie, taktiež nekontrolujeme ake informácie o nás získava samotný výrobca zariadenia. Použitím open-soure systému presunieme systém do lokálnej siete kde máme vetšiu kontrolu nad zariadeniami a samotným systémom.

2. Výber systému a firmware - pri výbere firmware je nutné sa zamerať hlavne na aké zariadenia budeme firmware nahrávať a aké zabezpečenie firmware ponúka, zatial čo Tasmota je hlavne pre komerčné zariadenia Esphome je škôr zamerané na jednotlivé komponenty zariadení,ďalšie možnosti sú Espurna a EspEasy

3. VLAN - Zariadenia v sieti musia byť rozdelené do separátných VLAN podľa použitia a či potrebujú medzi sebou komunikovať. IoT zariadenia ktoré používaju ten istý firmware by mali byť v separátnej VLAN čo nám umožní ich jednoduché manažovanie pomocou firewall

4. Firewall -

5. Zašifrovanie komunikácie - všetká komunikácia medzi zariadeniami a automatizačným systémom musí byť zašifrovaná a zariadenia musia overovať certifikáty aby sa vyhli zverjenovania citlivých informácii do sieti a potencionálnym mitm útokom.

6. Vzdialený prístup - komunikácia do našej sieti musí byť dostatočne zabezpečená pomocou TLS, v prípade že je doména automatizačného systému verejná je nutné implementovať opatrenia proti útokom hrubou silou.

7. 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.

8. Fyzický prístup - fyzický prístup k zariadeniam musí byť zabezpečený, toto je možné spraviť umiestneným zariadenia na miesto ktoré neni viditeľné alebo neni tak ľahko dostupné aby útočník mohol získať prístup k zariadeniu a stiahnuť z neho firmware.

9. Automatizácie - Kontrola nad zariadeniami musí byť funkčná aj v prípade výpadku automatizačného systému, ak je to možné pre kritické automatizácie by mala existovať nejaka možnosť redundancie. Taktiež prepojenie systému zo smartphone je nutné aby v prípade výpadku systému alebo zariadnia zo sieti mohli byť obyvatelia notifikovaný o tomto výpadku.

10. Zálohovanie - Automatizačný sýstém 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. Samotná záloha musí byť umiestnená mimo systému, idéalne na zabezpečený cloud alebo verzovací systém ako Git.

11. Administratíva - nástroje ktoré nám umožnia jednoduchú administratívu su často neni zabezpečené preto je nutné aby sa používali iba v prípade administratívy a iných prípadoch boli tieto funkcie/nástroje vypnuté

Kapitola 6

Záver

Literatura

[1] 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.

[2] 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.

[3] 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.

[4] Bormann, C. Dostupné z: https://coap.technology/.

[5] 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.

[6] Fielding, R. et al. *Hypertext Transfer Protocol – HTTP/1.1* [online]. Dostupné z: https://tools.ietf.org/html/rfc2616.

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

[8] 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.

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

[10] 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.

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

[12] 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.

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

[14] 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.

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

[16] 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.

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

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

[19] 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.

[20] 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.

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

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

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

[24] 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.

[25] 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.

Příloha A

Konfigurácia open-source IoT sieťe

A.1 Konfigurácia sieťe

A.2 Home Assistant

Home Assistant je open-source automatizačný systém, inštalácia je jednoduchá stačí stiahnuť 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
```

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 existujú dve rozšírenia:

- 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 a umožní nám sa pripojiť do HA pomocou SSH. Toto rozšírenie je ale možné nainštalovať iba po zapnutí "Advanced"módu ktorý sa zapína v našom profile.

A.2.1 Zabezpečenie a vzdialený prístup

Pre zabezpečenie HTTP komunikácie je možné použiť nami podpísane certifikáty 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
```

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
```

Obrázek 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.

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

```
[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: ------------------------------------------------------------
```

Obrázek A.2: Caption

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.

Obrázek A.3: Tor log kde sa zobrazí doména

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

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

- Ďalším spôsobom ako zabezpečiť ochranu proti útokom hrubou silou je odmietať prihlasovania do systému od 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
```

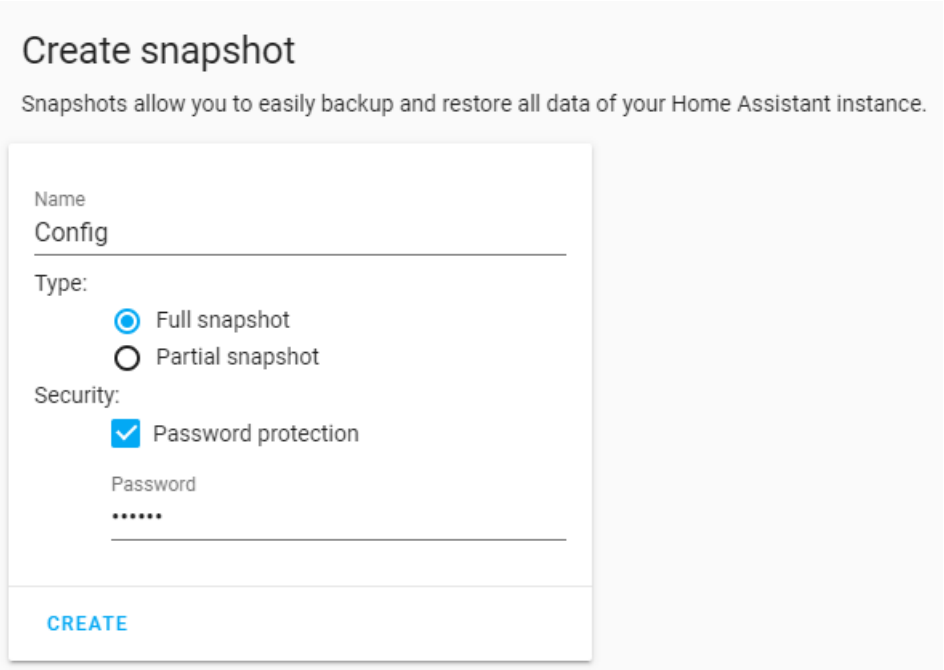
A.2.2 MQTT a zálohovanie systému

Posledným bodom konfigurácie je nainštalovať MQTT server pre komunikáciu s našimi zariadeniami. Na toto je možné použiť samostatný MQTT server na ktorý sa naše zariadenie pripojí, jednoduchšie je ale použiť Moqsuitto rozširenie. V konfiguračnom súbor treba 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, napríklad 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
```

Obrázek A.4: Caption

Po nakonfigurovaní je nutné taktiež zálohovať systém, HA ponúka vytváranie snapshotov do ktorých si možeme zvoliť ktorú časť systému chceme zálohovať



Obrázek A.5: Vytváranie snapshotu

Ďalej môžeme nainštalovať rozšírenia pre špecifický firmware, pre Esphome môžeme integrovať samotné Esphome pod 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 zapnuté web serveri na samotných zariadeniach ku ktorým pristupuje, kedže táto komunikácia neni šifrovaná by sa Tasmoadmin mal používať iba na administratívne účely.

A.3 Konfigurácia Esphome

Konfigurácia Esphome prebieha čisto iba pomocou yaml konfuračného súbora, pri konfigurácii je nutné zabezpečiť tieto body:

- Konfigurácia niekolkých AP na ktoré je možné sa pripojiť aby v prípade výpadku jedného existoval redundantný na ktorý sa zariadenie pripojí.

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

- 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>
```

- Pre efektívne aktualizácie OTA aktualizácie musia byť povolené, táto funkcia ale musí byť chránená heslom aby útočník nemohol

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

- 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 spúšťať 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>
```

- Esphome ponúka dva spôsobi komunikácie, MQTT a vlastné Esphom 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
```

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>
```

A.4 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 nájdete 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 funkcie. Taktiež nám skriptovanie umožní použiť aj zariadenia ktoré by sme s pravidlami nemohli používať lebo Tasmota nepodporuje 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
```

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 su 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 pomocou LetsEncrypt certifikačnej autority.

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 chcem overovať certifkačnú autoritu je nutné to špecifikovať pri kompilácii. Takže finálny *user_config_override.h* môže vyzerať nejako takto

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

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

```
#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"
```

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 tento Fallback AP, WifiConfig 4 v prípade výpadku pripojenia sa zariadenie pripojí na druhý AP bez reštartu.

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

Pri konfigurácii MQTT konfigurácia vyzerá podobne,vvložíme IP adresu MQTT brokera, Port na ktorý sa pripojíme (1883 bez TLS, pri použití 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. kde SetOption19 umožní HA automaticky detekovať Tasmota zariadenia:

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

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

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 možé 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 ;
```

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

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
GroupTopic <menoskupiny>;
DevGroupShare <IN>, <OUT>;
```

Použitím skriptu môžeme vytvoriť redundantné automatizácie ktoré sa spustia v prípade výpadku pripojenia na MQTT server, kde sekcia $>D$ sa vykoná pri štarte zariadenia, zatial čo sekcia $>S$ sa vykoná každú sekundu, skontroluje či je zariadenie pripojené na MQTT server, v prípade že zariadenie neni pripojené vyše 100 sekúnd zapne reduntatné automatizácie v prípade že sa obnoví pripojenie tak sa táto funkcia vypne.

```
>D
mqttcounter=0
devicegrouponline=false


>S
if devicegroup==false
if mqttc==false
then
mqttcounter=mqttcounter + 1
else
mqttcounter=0
```

```
endif
else
if mqttc==true
then
=> SetOption85 0
mqttcounter=0
endif


if mqttcounter>100
then
devicegroup=true
mqttcounter=0
=> Grouptopic <Topic>
=> DevGroupShare <x,y>
=> SetOption85 1
endif
```