VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE ANOMÁLIÍ V IOT SÍTÍCH

ANOMALY DETECTION IN IOT NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JOZEF HALAJ**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. Ing. JAN KOŘENEK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Halaj Jozef, Bc.**
Program: Informační technologie Obor: Bezpečnost informačních technologií
Název: **Detekce anomálií v IoT sítích**
Anomaly Detection in IoT Networks
Kategorie: Vestavěné systémy

Zadání:

1. Nastudujte protokoly, které se používají pro komunikaci s bezdrátovými IoT senzory.
2. Pro vybranou sadu senzorů analyzujte charakteristiku síťového provozu. Zaměřte se na časové a volumetrické parametry komunikace.
3. Navrhněte pro analyzovaný provoz vhodný detektor anomálií.
4. Proveďte implementaci tak, aby bylo možné provozovat detektor anomálií na domácích směrovačích se systémem OpenWRT nebo na vhodné vestavěné hardwarové platformě s procesorem ARM.
5. Pokuste se simulovat výskyt různých anomálií a analyzujte reakci vytvořeného detektoru.
6. Implementovaný detektor anomálií ověřte a vyhodnoťte nejen pomocí simulovaných útoků, ale i v reálném provozu.
7. V závěru diskutujte dosažené výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

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

Vedoucí práce: **Kořenek Jan, doc. Ing., Ph.D.**
Konzultant: Krejčí Radek, RNDr., CESNET
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 20. května 2020
Datum schválení: 25. října 2019

Abstrakt

Cieľom práce bola analýza komunikačných protokolov IoT, ich zraniteľností a vytvorenie vhodného detektora anomálií. Detektor musí byť možné prevádzkovať na smerovačoch so systémom OpenWRT. Na vytvorenie výsledného riešenia bolo potrebné analyzovať komunikačné protokoly BLE a Z-Wave so zameraním na ich bezpečnosť a zraniteľnosti. Ďalej bolo potrebné analyzovať možnosti detekcie anomálií, navrhnúť a implementovať detekčný systém. Výsledkom je modulárny detekčný systém postavený na NEMEA frameworku. Detekčný systém je schopný odhaliť opakované párovanie BLE zariadení predstavujúce potenciálny útok na párovanie. Systém umožňuje odchytávanie Z-Wave komunikácie pomocou SDR, detekciu skenovania Z-Wave siete a niekoľkých útokov na smerovanie v sieti. Systém rozširuje existujúci detektor nad štatistickými IoT dátami o podrobnejšie štatistiky so širším pohľadom na sieť. Pôvodné riešenie malo k dispozícii len Z-Wave štatistiky s obmedzeným pohľadom na sieť získané zo Z-Wave kontroléra. Modulárne riešenie systému poskytuje flexibilitu nasadenia a jednoduchú rozšíriteľnosť systému. Funkčnosť riešenia bola overená experimentami a sadou automatizovaných testov. Systém bol taktiež úspešne otestovaný na smerovači so systémom OpenWRT a v reálnej prevádzke. Výsledky práce boli použité v rámci projektu SIoT.

Abstract

The goal of the thesis was an analysis of IoT communication protocols, their vulnerabilities and the creation of a suitable anomaly detector. It must be possible to run the detector on routers with the OpenWRT system. To create the final solution, it was necessary to analyze the communication protocols BLE and Z-Wave with a focus on their security and vulnerabilities. Furthermore, it was necessary to analyze the possibilities of anomaly detection, design and implement the detection system. The result is a modular detection system based on the NEMEA framework. The detection system is able to detect re-pairing of BLE devices representing a potential pairing attack. The system allows interception of Z-Wave communication using SDR, detection of Z-Wave network scanning and several attacks on network routing. The system extends the existing detector over IoT statistical data with more detailed statistics with a broader view of the network. The original solution had only Z-Wave statistics with a limited view of the network obtained from the Z-Wave controller. The modular solution of the system provides deployment flexibility and easy system scalability. The functionality of the solution was verified by experiments and a set of automated tests. The system was also successfully tested on a router with OpenWRT and in the real world enviroment. The results of the thesis were used within the SIoT project.

Kľúčové slová

Internet vecí, IoT, Inteligentná domácnosť, Brána, BeeeOn, Bezpečnosť, SIoT, Detekcia anomálií, NEMEA, OpenWRT, Bluetooth Low Energy, BLE, Z-Wave, SDR

Keywords

Internet of Things, IoT, Smart Home, Gateway, BeeeOn, Security, SIoT, Anomaly Detection, NEMEA, OpenWRT, Bluetooth Low Energy, BLE, Z-Wave, SDR

Citácia

HALAJ, Jozef. *Detekce anomálií v IoT sítích*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Jan Kořenek, Ph.D.

Detekce anomálií v IoT sítích

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Doc. Ing. Jana Kořenka, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Jozef Halaj
3. júna 2020

Poďakovanie

V prvom rade by som sa chcel poďakovať rodine za podporu pri štúdiu, pretože bez nich by som to do tohto bodu určite nedotiahol. Veľké ďakujem patrí vedúcemu diplomovej práce Doc. Ing. Janovi Kořenkovi, Ph.D. za odborné vedenie, cenné rady a čas venovaný konzultáciám. Takisto sa chcem poďakovať konzultantovi RNDr. Radkovi Krejčímu za cenné rady a celému tímu SIoT za výbornú spoluprácu na projekte.

Obsah

Kapitola 1

Úvod

Spolu s technickým pokrokom sa zvyšujú požiadavky ľudí na komfort, čo najkvalitnejší životný štandard a uľahčenie si najbežnejších úkonov. Bežné úkony, ktoré nám ešte donedávna boli prirodzené nahradzujú elektronické zariadenia. Internet už dávno neslúži len ako zdroj informácií, ale čím ďalej viac preniká do nášho života, spotrebičov okolo nás, priemyslu, či zdravotníctva.

Obklopujú nás rôzne senzory schopné merať vlastnosti prostredia a neustále získavať dáta, a naopak aktívne prvky umožňujúce ovplyvňovať stav okolia na podnet riadiacich príkazov. Vzniká Internet vecí (IoT) zlučujúci tieto zariadenia a umožňujúci komunikovať s nimi pomocou bezdrôtových technológií.

Záujem o Internet vecí sa čím ďalej viac a viac zvyšuje. Spolu s ním sa zvyšuje aj počet rôznych útokov. Keďže zariadenia komunikujú bezdrôtovými technológiami sú náchylné na odpočúvanie a zraniteľné. Preto je nutné dbať na zabezpečenie IoT, detekciu rôznych anomálií a prípadných útokov.

Táto práca sa zaoberá práve detekciou anomálií a útokov v IOT sieťach, konkrétne sa venuje zraniteľnostiam komunikačných protokolov Bluetooth Low Energy a Z-Wave. U protokolu Bluetooth Low Energy tvorí základ práce detekcia potenciálneho útoku na párovací proces, ktorý je najzraniteľnejšou časťou tohto protokolu. Pri Z-Wave sa venuje získavaniu sieťovej komunikácie pomocou softvérovo definovaného rádia (SDR) a detekcii útokov nad touto komunikáciou. Detekuje sa skenovanie siete útočníkom a útoky na smerovanie v sieti. Ďalej rozširuje existujúci detektor anomálií nad štatistickými dátami o podrobnejšie štatistiky so širším pohľadom na sieť, zatiaľ čo pôvodné riešenie malo k dispozícií len štatistiky s obmedzeným pohľadom na sieť získané zo Z-Wave kontroléra. Celkovo je cieľom práce navrhnúť a implementovať detekčný systém tak, aby bolo možné prevádzkovať systém na domácich smerovačoch so systémom OpenWRT alebo na vhodnej vstavanej hardvérovej platforme s procesorom ARM.

Text práce je rozdelený na niekoľko častí. Prvá časť predstavuje architektúru IoT sietí. Druhá časť sa zaoberá analýzou komunikačných IoT protokolov Bluetooth Low Energy a Z-Wave, kde sa venuje hlavne ich bezpečnosti a zraniteľnostiam. Tretia časť popisuje možnosti detekcie anomálií v IoT sieťach, modulárny detekčný systém NEMEA a Zabezpečenú bránu IoT (SIoT). Ďalšou časťou je analýza sieťovej prevádzky vybraných senzorických zariadení Z-Wave. Nasleduje návrh a realizácia detekčného systému. Poslednou časťou je popis experimentov a testovania vytvoreného detekčného systému.

Kapitola 2

Architektúra IoT sietí

Existuje niekoľko definícií pojmu Internet of Things (IoT). Väčšina z nich však IoT popisuje ako myšlienku spojenia každodenných objektov s internetom, čo má viaceré účely. Senzory slúžia na získavanie množstva informácií (dát) z okolitého prostredia. Pomocou aktívnych prvkov (aktuátorov) je možné naopak ovplyvňovať stav prostredia.

Každý deň je k internetu pripojených viac a viac zariadení. Podľa odhadov by ich počet mal celosvetovo v roku 2025 presiahnuť hranicu 75 miliárd [14]. Takéto množstvo zariadení generuje nebývalý objem dát a reprezentuje nespočetne nových typov zariadení a dátových variácií. Nie je možné, aby všetky tieto zariadenia komunikovali priamo s dátovým centrom, pretože nároky na šírku pásma by boli obrovské [4]. Navyše v čase keď by boli dáta presunuté a ich analýza na cloude dokončená, môže byť už neskoro na prípadnú reakciu. Ďalším problémom je obmedzený výkon pripojených prvkov, ktorý je dôležitý pre použitie bezpečnostných funkcií umožňujúcich kompletne zabezpečenú komunikáciu.

2.1 Fog computing model

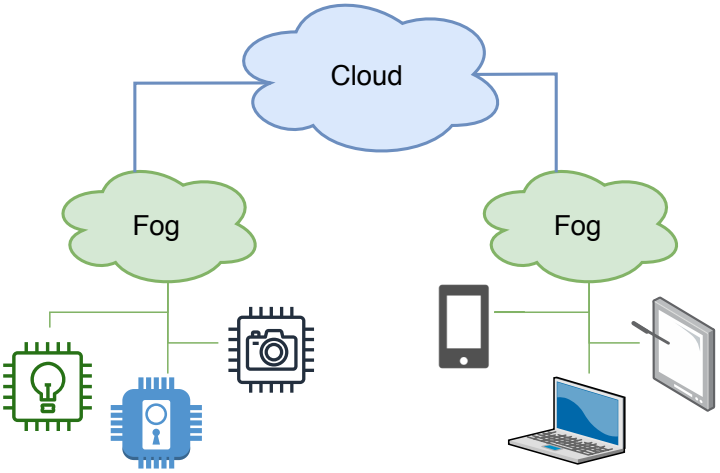
Fog computing je rozšírenie cloud computingu, ktoré spočíva v pridaní niekoľký vrstviev pre presunutie výpočetného výkonu bližšie ku koncovým zariadeniam [4]. Rozšírenie cloud computingu je realizované pridaním sieťových zariadení (fog uzlov), ktoré okrem pripojenia k internetu ponúkajú úložisko dát a výpočetný výkon pre beh externých programov.

Úlohou týchto programov je monitorovanie a analýza dát v reálnom čase od zariadení pripojených do siete a iniciovanie akcií (napr. zamknutie dverí alebo zaslanie alertu). Základná myšlienka modelu je zobrazená na obrázku 2.1. Model pozostáva z troch vrstiev: vrstva koncových zariadení, fog vrstva a cloud.

Použitie fog computing modelu prináša nasledovné výhody:

- **Nižšie nároky na latenciu a šírku pásma** – Dáta z koncových zariadení sú spracovávané lokálne miesto odoslania na cloud. Tým je možné rýchlejšie reagovať na kritické podnety.
- **Jednotná správa koncových zariadení** – Vďaka sieťovým zariadeniam v jednotlivých fog vrstvách sa už nemusí pristupovať priamo na koncové prvky, ktoré často komunikujú rôznymi protokolmi. Sieťové zariadenia odtieňujú rôznorodosť týchto protokolov a vytvárajú abstraktnú vrstvu na prístup ku koncovým prvkom. Vďaka tejto abstrakcii je tiež zjednodušené spracovanie získaných dát.

- **Zlepšenie zabezpečenia a súkromia** – Citlivé dáta nie je nutné prenášať po sieti na spracovanie v cloude, čím sa znižuje šanca ich úniku. Okrem toho sú sieťové prvky trvalo napájané a pripojené k internetu, čím sa znižuje šanca nespracovania časovo-citlivých dát kvôli výpadku spojenia.



Obr. 2.1: Fog computing model

Nevýhodou môže byť jeho distribuovaná topológia a s tým spojené väčšie nároky na údržbu. Samotný fog computing tak, ako popisuje štruktúru IoT siete a distribúciu spracovania dát neobsahuje bezpečnostné slabiny. Komunikačný model IoT môžeme však rozdeliť do troch vrstiev, kde každá obsahuje špecifické zraniteľné miesta [13].

Senzorová vrstva zahŕňa všetky koncové zariadenia získavajúce informácie zo svojho okolia alebo vykonávajúce určitú akciu. Tieto zariadenia sú väčšinou pripojené k IoT bráne (bližšie popísanej v 2.2), pomocou rôznych bezdrôtových komunikačných protokolov. Princíp komunikácie a možnosti topológie závisia na použitej technológii.

Najväčšou slabinou tejto vrstvy sú bezdrôtové komunikačné protokoly. Ak nie je použité zabezpečenie, komunikácia môže byť jednoducho odpočúvateľná alebo modifikovateľná. Taktiež tu môžeme nájsť zariadenia označené ako zabezpečené, ktoré však vďaka používaniu staršej verzie komunikačného protokolu využívajú zastaralé bezpečnostné funkcie alebo obsahujú implementačné chyby. Tento prípad je o to nebezpečnejší, pretože vyvoláva falošný pocit bezpečia. Batériou napájané zariadenia sa môžu stať ľahkým terčom útoku, kde môžu byť vybité nadmernou komunikáciou.

Sieťová vrstva je prepojená so senzorovou pomocou IoT brány. Vrstva slúži na preposlanie spracovaných senzorických dát z brány ďalším službám a taktiež na vzdialenú správu brány. Komunikácia na sieťovej vrstve typicky používa TCP/IP protokolové riešenie. Najčastejšie sa jedná o protokol HTTPS (Hypertext Transfer Protocol Secure) a jeho variantu WebSocket alebo technológiu VPN (Virtual Private Network). Taktiež sú využívané technológie na výmenu správ ako napr. MQTT (Message Queuing Telemetry Transport) alebo COAP (Constrained Application Protocol).

Kým sieťová vrstva využíva tradičnú TCP/IP komunikáciu, bezpečnostné hrozby sú rovnaké ako v klasických sieťach. Je potrebné dodržať princípy dôvery, integrity a do-

stupnosti. Takto je možné predísť útokom ako: DoS (Denial of Service), MITM (Man In The Middle) a podvrhnutiu informácií.

Aplikačná vrstva sa stará o spracovanie a ukladanie dlhodobých dát. Vrstva sa typicky nachádza v dátovom centre, poskytuje užívateľovi vzdialené užívateľské rozhranie o spracovaných dátach a umožňuje mu konfigurovať sieť. Dôležité je taktiež poskytovanie automatizácie nad senzorickou sieťou.

Bezpečnostné problémy vrstvy je možné prirovnať k hrozbám cloud computingu. Bežnými útokmi na túto vrstvu sú Buffer Overflow, SQL Injection alebo DoS.

Vo všeobecnosti je problém aj spôsob nasadenia IoT siete. Nasadenie mnoha koncových prvkov do jedného spoločného segmentu komplikuje bezpečnostné pravidlá a zvyšuje dopad potenciálneho útoku. Pri úspešnom napadnutí jedného prvku útočník získa možnosť rozšírenia prístupu na ďalšie uzly v rovnakom segmente.

2.2 IoT brána

IoT brána je fyzické sieťové zariadenie (prípadne softvér, ktorý môže bežať na rôznych hardvérových zariadeniach) slúžiace ako prepojovací bod medzi koncovými prvkami a fog vrstvou [13]. Ak je brána reprezentovaná výkonnejším sieťovým prvkom, môže v nej prebiehať aj nejaké spracovanie dát. V tomto prípade je tiež považovaná za fog prvok.

Hlavnou úlohou brány je teda získavať dáta z pripojených koncových prvkov a poskytovať ich vyšším vrstvám. Pre IoT siete je typické, že obsahujú veľké množstvo prvkov komunikujúcich rôznymi technológiami. Z dôvodu diverzity, brána vyžaduje niekoľko rozhraní pre komunikačné protokoly ako napr. Z-Wave, Bluetooth Low Energy alebo Zigbee. Detailný popis vybraných komunikačných protokolov je možné nájsť v nasledujúcej kapitole.

2.2.1 Brána BeeeOn

V súčasnosti existuje množstvo rôznych brán od rozdielnych výrobcov. Ich parametre sa líšia na základe spôsobu nasadenia a nárokov na prevádzku. Veľkým problémom v tejto oblasti je malý dôraz na bezpečnostné funkcie, ktoré umožňujú vzdialené riadenie brány, kontrolu prevádzky a aktualizáciu jej softvéru. Z tohto dôvodu vznikol na Fakulte Informačných technológií VUT opensource projekt BeeeOn[1]. Jeho hlavným cieľom je vytvorenie softvérovej IoT brány, ktorú je možné spustiť na viacerých hardvérových platformách s dôrazom na bezpečnosť a širokú škálu podporovaných IoT protokolov a rôznorodých koncových zariadení (napr. BLE, Z-Wave, IQRF alebo Jablotron).

[1] http://www.beeeon.org/

Kapitola 3

Bezdrôtové komunikačné protokoly IoT

Chytré zariadenia využívajú na komunikáciu rôzne bezdrôtové protokoly. Medzi najznámejšie patrí Bluetooth Low Energy, Z-Wave, ZigBee, LoRaWAN a mnoho iných. Táto kapitola predstavuje vybrané bezdrôtové komunikačné protokoly IoT a hlbšie sa venuje protokolom Bluetooth Low Energy a Z-Wave, ktoré tvoria hlavnú časť tejto práce. Popisuje spôsob komunikácie, zabezpečenie a zraniteľnosti týchto technológií.

ZigBee je otvorený štandard založený na štandarde IEEE 802.15.4 pre nízku spotrebu energie a komunikáciu v nízkom prenosovom pásme na krátku vzdialenosť, ktorý spravuje ZigBee aliancia. Umožňuje však aj komunikáciu na väčšie vzdialenosti. Zariadenia využívajúce ZigBee štandard vyielajú vo frekvenčnom pásme 868 MHz, 915 Mhz, ale najčastejšie v pásme 2.4 GHz kde dosahujú rýchlosť 250 kbit/s. ZigBee bol navrhnutý a optimalizovaný pre zariadenia, ktoré využívajú batériu a jej výdrž je jednou z hlavných priorít. Aby bola zaistená nízka spotreba, strávia zariadenia väčšinu času v spánkovom režime. ZigBee poskytuje niekoľko rôznych topológií ako mesh, hviezda alebo stromová topológia a je použiteľný v senzorových sieťach, domácnostiach, ale taktiež aj v priemysle [1].

LoRaWAN je bezdrôtový protokol pre komunikáciu na veľké vzdialenosti vo WAN sieťach. Zariadenia komunikujú priamo s jednou alebo viacerými bránami, ktoré preposielajú dáta na servery cez bežnú IP sieť. Prenosová rýchlosť je úmerná vzdialenosti od 0.3 kbit/s do 50 kbit/s v nelicencovanom frekvenčnom pásme. Najväčšia výhoda LoRaWAN je vzdialenosť, na ktorú dokážu zariadenia komunikovať dosahujúca až 20km s čím súvisí flexibilita a mobilita technológie a ďalej je dôraz kladený taktiež na šetrenie energie [9].

3.1 Bluetooth Low Energy

Bluetooth Low Energy (ďalej BLE, tiež známy ako Bluetooth 4 alebo Bluetooth Smart) je technológia navrhnutá na použitie v senzorických sieťach s dôrazom na nízku spotrebu energie. Je určená pre zariadenia prenášajúce len malé množstvo dát so zníženou prenosovou rýchlosťou oproti Bluetooth Classic. Hlavnou výhodou je práve nízka spotreba energie kde batéria veľkosti mince môže vydržať mesiace až roky. Technológia bola predstavená ako časť Bluetooth 4.0 špecifikácie a nie je spätne kompatibilná s Bluetooth Classic. Tak isto

ako klasický Bluetooth komunikuje vo frekvenčnom pásme 2.4 GHz, ale inou sadou kanálov s frekvenčnými skokmi. BLE komunikácia sa nachádza vo frekvenčnom rozsahu 2.4-2.4835 GHz a používa 40 2-MHz kanálov. Klasický Bluetooth pracuje v rovnakom spektre avšak používa 79 1-MHz kanálov čo je hlavný dôvod nekompability s BLE. Maximálna prenosová rýchlosť BLE 4.0 je 1 MBit/s a dosah maximálne 100m v ideálnych podmienkach [9, 12].

3.1.1 Spôsob komunikácie

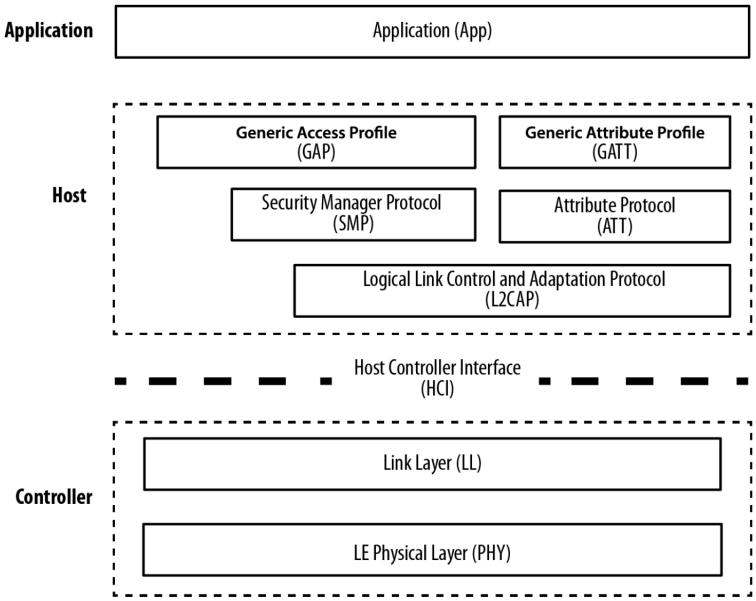
BLE zariadenia sú dvoch typov. Prvým typom je centrálne zariadenie (Master), čo je typicky počítač alebo chytrý telefón. Druhý typ predstavujú periférne zariadenia, čo sú senzory a iné zariadenia s nízkym výkonom (Slave). Periférne zariadenia kontinuálne broadcastujú takzvané advertising pakety na troch vyhradených kanáloch, vďaka ktorým môžu byť objavené centrálnymi zariadeniami. Rozlišujeme 2 druhy komunikácie a to Broadcasting (všesmerové vysielanie) a Connection (spojenie) [16].

Broadcasting je jediná možnosť ako môže zariadenie posielať dáta viacerým zariadeniam naraz. Definujeme tu 2 roly: Broadcaster a Observer. Broadcaster periodicky posiela nepripojiteľné advertising pakety každému kto počúva, zatiaľčo Observer periodicky skenuje pre príjem týchto paketov. Keď Observer príjme advertising paket, môže si ešte vyžiadať takzvané Scan Response Data. Po ustanovení spojenia medzi centrálnym a periférnym zariadením sa automaticky advertising pakety prestanú posielať.

Connection je spojenie dvoch zariadení kde si môžu navzájom vymieňať dáta. Periférne zariadenie môže byť naraz pripojené len k jednému centrálnemu zariadeniu, ale centrálne zariadenie môže byť pripojené k viacerým periférium. Centrálne zariadenie skenuje frekvencie pre nájdenie pripojiteľných advertising paketov a keď ich nájde môže iniciovať spojenie. Periférne zariadenie periodicky vysiela advertising pakety a akceptuje nadviazanie spojenia. Po nadviazaní spojenia si môžu zariadenia vymieňať dáta a môže sa dohodnúť na intervale výmeny dát a teda ďalšom nadviazaní spojenia. Týmto spôsobom sa šetrí energia, pretože zariadenia sa môžu uspať a v určitých intervaloch sa prebúdzať a komunikovať.

3.1.2 Protokolový zásobník BLE

Obrázok 3.1 zobrazuje protokolový zásobník BLE technológie od najnižšej fyzickej vrstvy až po vrstvu aplikačnú. Z najvyššej úrovne abstrakcie ho tvoria 3 časti. Aplikačný blok tvorí užívateľská aplikácia, ktorá využíva protokolový zásobník BLE. Host (hostiteľ) zastrešuje vrchnú časť zásobníka nachádzajúcu sa typicky v operačnom systéme a Controller (kontrolér) spodnú časť predstavujúcu hardverové komponenty BLE (napr. pripojiteľný adaptér). Medzi nimi je rozhranie HCI (Host Controller Interface). Jeho úlohou je prepojiť tieto 2 časti štandardizovanou cestou kde prenáša dáta sériovým rozhraním napr. USB alebo UART.



Obr. 3.1: Protokolový zásobník Bluetooth Low Energy [16]

Každá vrstva zásobníka má ako v iných technológiách špecifickú úlohu [16].

Fyzická vrstva sa stará o fyzickú komunikáciu prostredníctvom rádiových vĺn a prevod analógových dát do digitálnej podoby.

Linková vrstva je tvorená kombináciou hardvéru a softvéru a celkovo riadi prenos fyzickou vrstvou. Je zodpovedná za nadviazanie a ukončenie spojenia, zaisťuje šifrovanie, uchováva Bluetooth Device Address atď.

Logical Link Control and Adaption Protocol (L2CAP) zapúzdruje protokoly z vyšších vrstiev do štandardného BLE paketu a naopak, a stará sa o fragmentáciu.

Attribute Protocol (ATT) je klient/server protokol založený na atribútoch poskytovaných zariadením. Klient si môže vypýtať alebo poslať dáta a server ich poskytne alebo uloží na základne UUID atribútu.

Security Manager Protocol (SMP) poskytuje bezpečnostné mechanizmy BLE ako autentifikáciu, autorizáciu a párovanie. Je zodpovedný za proces párovania dvoch zariadení a za autentifikačné mechanizmy keď spárované zariadenia nadviažu spojenie.

Generic Attribute Profile (GATT) pridáva dátový model a hierarchiu nad ATT a definuje ako budú dáta organizované a vymieňané medzi rôznymi aplikáciami. Dáta sú organizované do služieb kde každá služba obsahuje jednu alebo viacero charakteristík a každá charakteristika obsahuje dáta spolu s metadátami. GATT služby sú organizované do GATT profilov. Služby sú rozlišované pomocou UUID.

Generic Access Profile (GAP) je základným kameňom umožňujúcim BLE zariadeniam medzi sebou komunikovať. Špecifikuje ako sa majú zariadenia chovať pri vyhľadávaní ostatných zariadení, vytváraní spojenia, odosielaní dát, atď.

3.1.3 Zabezpečenie a zraniteľnosti

Potenciálny útok na BLE je komplikovaný vďaka rozloženiu komunikácie v rôznych kanáloch. Existuje však niekoľko nie drahých možností na odpočúvanie komunikácie (napr. open-source platforma Ubertooth[1]).

Zabezpečenie BLE je realizované pomocou šifrovania dát AES medzi dvoma zariadeniami, ktoré nadviazali spojenie. Aby mohli byť dáta šifrované je potrebné, aby vykonali proces zvaný párovanie. Pri párovaní si zariadenia dohodnú párovaciu metódu a vygenerujú kľúče, pomocou ktorých šifrujú prenášané dáta po nadviazaní zabezpečeného spojenia. Tieto kľúče si uložia a následne sú ďalšie spojenia vytvárané ako šifrované. Bezpečnosť šifrovania závisí na výmene kľúčov počas párovania. BLE 4.0 využíva Legacy Pairing náchylný na odpočúvanie. Až verzia 4.2 prináša výrazné zvýšenie bezpečnosti využitím Elliptic-Curve Diffie-Hellman (ECDH) pre ochranu výmeny kľúčov pred odpočúvaním. Aktuálne existujú 4 metódy párovania [9].

Just Works je najmenej bezpečná metóda, pretože kľúče sú automaticky vymenené a nie je potrebný žiadny vstup od užívateľa. V prípade Legacy Pairing je jednoducho odpočúvateľná a neexistuje tu žiadna ochrana proti útoku Man-in-the-Middle (MITM).

Passkey Entry používa 6 ciferný kód pre dočasný kľúč. Na jednom zariadení sa vygeneruje a zobrazí 6 ciferný kód, ktorý užívateľ musí zadať cez klávesnicu na druhom zariadení. Rovnako ako v prípade Just Works je v prípade Legacy Pairing jednoducho odpočúvateľná.

Out Of Band využíva na výmenu párovacích informácií iné komunikačné rozhranie (napr. NFC). Bezpečnosť je teda plne závislá na zabezpečení vybraného komunikačného rozhrania.

Numeric Comparison je metóda pridaná v BLE 4.2 rozširujúca metódu Just Works o kontrolu 6 ciferného čísla poskytnutého užívateľovi oboma zariadeniami na kontrolu.

Ako už bolo spomínané tak bezpečnosť šifrovania závisí na ochrane generovaných a vymieňaných šifrovacích kľúčoch pri párovaní. Do BLE verzie 4.2 však šifrované spojenie nie je bezpečné, pretože neexistuje ochrana pred odpočúvaním párovacieho procesu. Väčšina bežných zariadení žiaľ stále používa BLE verzie 4.0. Napriek zvýšeniu bezpečnosti procesu párovania je šifrovanie voliteľné a niektorí výrobcovia ho neimplementujú.

Ďalším problémom BLE bezpečnosti je komplexnosť BLE špecifikácie, čo môže viesť k implementačným chybám a s tým spojeným zraniteľnostiam. Niektorí výrobcovia implementujú šifrovanie a ďalšie bezpečnostné mechanizmy na aplikačnej vrstve, ale ich implementácia môže bežne obsahovať nedostatky alebo slabú ochranu proti špecifickým hrozbám (napr. MITM alebo replay útoky).

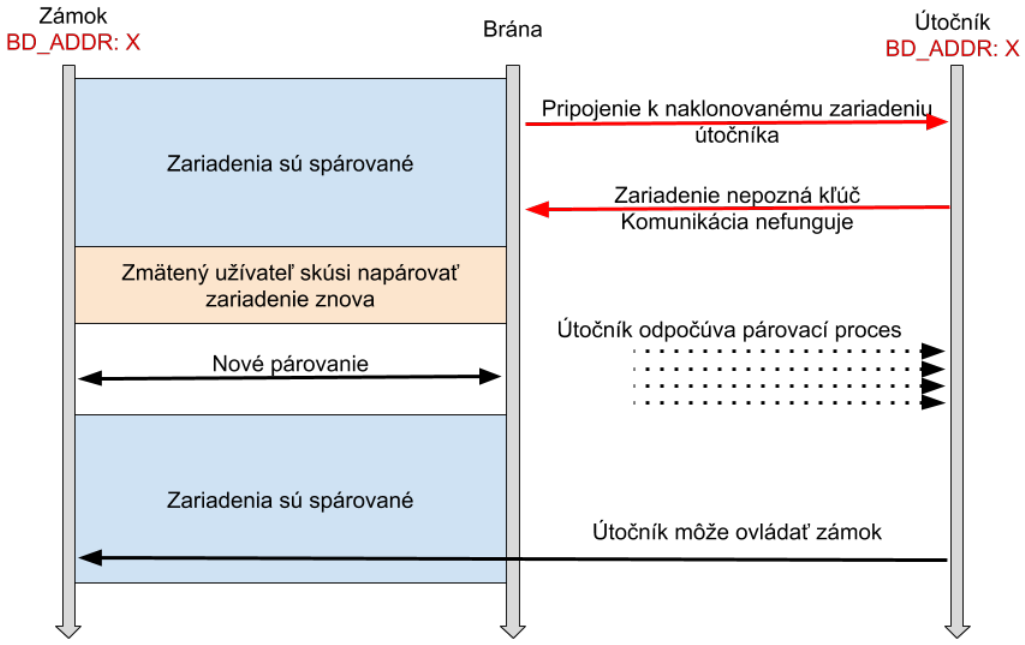
Útok na párovanie

Na obrázku 3.2 je zobrazený princíp útoku na nezabezpečené párovanie. Útočník môže vynútiť nové párovanie a odchytiť výmenu kľúčov z čoho môže vypočítať hlavný šifrovací kľúč. So znalosťou kľúča môže odpočúvať komunikáciu a taktiež ovládať zariadenia, ku ktorým by nemal mať prístup. Pri naklonovaní zariadenia si prisvojí jeho adresu a vydáva sa za neho. Užívateľská aplikácia sa pokúsi pripojiť k zariadeniu, ale nepodarí sa jej to, pretože

[1] https://github.com/greatscottgadgets/ubertooth

útočník nepozná šifrovací kľúč. Zmätený užívateľ pravdepodobne potom zmaže párovanie a zaháji procedúru znova. V prípade Just Works sa môže užívateľ spárovať s útočníkom, ktorý bude schopný okamžite zachytiť prevádzku a môže vystupovať ako MITM medzi užívateľskou aplikáciou a zariadením. V prípade Passkey Entry prestane vystupovať ako klonované zariadenie, odchytí nový párovací proces a vypočíta hlavný šifrovací kľúč [12].

Opakované párovanie je preto nežiadúce a predstavuje potenciálny útok. Existuje ešte teoretická možnosť injektovania príslušnej správy linkovej vrstvy (LL_REJECT_IND) v správny moment pri zahajovaní šifrovaného spojenia. Táto správa vynúti pregenerovanie kľúčov, ako keby išlo o nové párovanie [11].



Obr. 3.2: Útok na párovanie

3.2 Z-Wave

Z-Wave je proprietárny bezdrôtový komunikačný protokol primárne určený na automatizáciu domácnosti. Z-Wave sieť má podobu mesh topológie a dokáže prepojiť až 232 zariadení. Uzly sú rozdelené na kontroléry, kde jeden z nich musí byť primárny kontrolér schopný pridávať a odoberať zariadenia v topológií, a sluhov predstavujúcich ovládanie osvetlenia, termostat, zámky atď. Uzly potom buď odpovedajú na požiadavku kontroléra alebo ju širia sieťou ďalej. Vzdialenosť medzi komunikujúcimi uzlami môže byť približne 30 metrov. Prenosová rýchlosť dosahuje 100 kbit/s. Keďže Z-Wave pracuje v pásme nižšom ako 1 GHz je odolný proti rušeniu prostredníctvom WiFi a iných bezdrôtových technológií ktoré pracujú v pásme 2.4 GHz ako Bluetooth alebo ZigBee.

Všetky komunikačné prvky sú certifikované Z-Wave alianciou, ktorá poskytuje licenciu na vývoj a technickú dokumentáciu. Zatiaľ čo plná podpora a technická dokumentácia je poskytnutá len členom aliancie existuje verejne dostupná vrstva predstavujúca rozhranie k sieti Z-Wave. Otvorená implementácia tejto vrstvy sa nazýva OpenZWave a je vyvíjaná komunitou [9].

3.2.1 Spôsob komunikácie

Ako už bolo spomenuté v jednej sieti Z-Wave sa môže nachádzať 232 rôznych zariadení. Sieť je identifikovaná 32 bitovým unikátnym identifikátorom nazývaným HomeID. Primárny kontrolér, ktorý je vlastne vlastníkom siete si pri prvom spustení vygeneruje HomeID a tým identifikuje sieť. Ďalšie uzly môžu byť potom pridané do siete len spárovaním s primárnym kontrolérom. Pri párovaní dostanú nové prvky 8 bitový identifikátor nazývaný NodeID. Po pridaní do siete komunikujú uzly s kontrolérmi, ale taktiež trvalo napájané prvky môžu správy šíriť ďalej čím sa zvyšuje rozloha siete. Pre odobranie uzlu zo sieti je potrebné zariadenie odpárovať.

Z-Wave špecifikácia určuje triedu funkcionalit nazývanú Command Class pre každé Z-Wave zariadenie. Zariadenie môže mať týchto tried viac podľa svojej komplexnosti. V rámci každej triedy je definovaná sada príkazov. Napríklad inteligentná elektrická zásuvka má Command Class BinarySwitch, obsahujúci príkazy SET, GET a REPORT. SET odosiela kontrolér pre nastavenie hodnoty, GET odosiela kontrolér pre získanie hodnoty a REPORT odosiela senzor ako odpoveď na príkaz GET. Väčšina správ v sieti vyžadaduje potvrdenie. V prípade neobdržania potvrdenia je správa zasielaná znova.

Z-Wave protokol je rozdelený do niekoľkých vrstviev. Nasledujúce sekcie popisujú PHY, MAC a transportnú vrstvu Z-Wave založené na štandarde ITU-T G.9959 [8].

3.2.2 Fyzická a MAC vrstva

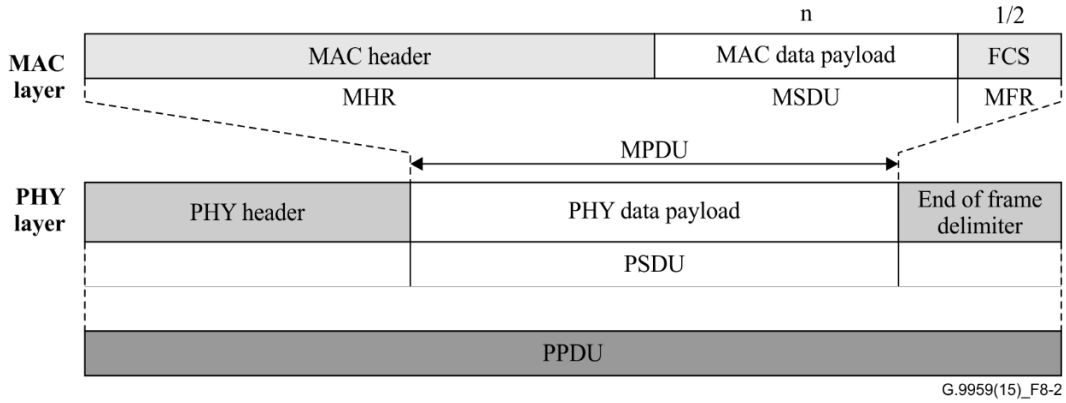
Fyzická vrstva definuje ako presunúť bity z vysielača k prijímaču [10, 8]. Z-Wave nie je priamo viazaný na určitú frekvenciu avšak využíva doporučené frekvencie v štandarde podľa lokality. Používajú sa 3 komunikačné kanály zobrazené v tabuľke 3.1, ktoré sa líšia frekvenciou, prenosovou rýchlosťou, typom modulácie a kódovania.

Kanál	**1**	**2**	**3**
Prenosová rýchlosť	9.6 kB/s	40 kB/s	100kB/s
Modulácia	FSK	FSK	GFSK
Kódovanie	Manchester	NRZ	NRZ
Frekvencia (EU)	868.4 MHz	868.4 MHz	869.85 MHz

Tabuľka 3.1: Z-Wave kanály

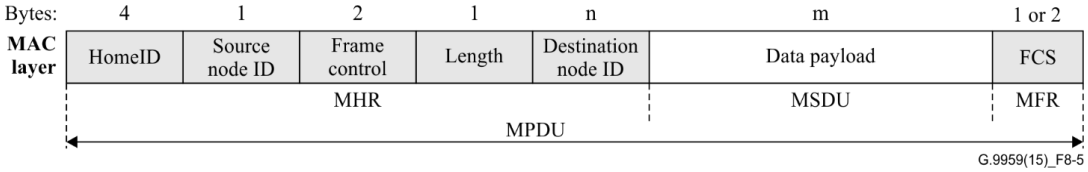
Z-Wave dokáže prepínať frekvencie ak je jedna frekvencia zahltená. Použitý kanál sa vyjednáva medzi uzlami a použije sa kanál s najvyššou možnou prenosovou rýchlosťou. Z tohto dôvodu môže jedno zariadenie komunikovať s rôznymi zariadeniami rôznymi kanálmi. Časť komunikácie ako napr. párovanie alebo niektoré koordinačné správy sieťového manažmentu prebieha len na prvom kanáli.

Na obrázku 3.3 je zobrazené zapúzdrenie rámca MAC vrstvy MPDU (MAC protocol data unit) do dátovej jednotky fyzickej vrstvy PPDU (PHY protocol data unit) ako PSDU (PHY service data unit). Hlavička obsahuje preambulu na synchronizáciu, položku start of frame (oddeľovač začiatku rámca) a na koniec sa pridáva položka end of frame (oddeľovač konca rámca).



Obr. 3.3: Z-Wave PHY/MAC rámec [8]

MAC vrstva definuje ako môže byť celý dátový stream prenesený z jedného zariadenia na druhé zariadenie [10, 8]. Obsahuje algoritmus pre vyhýbanie sa kolíziám, ktorý zabraňuje uzlom začať vysielať, kým vysielajú iné uzly. To je dosiahnuté tým, že keď uzly nevysielajú sú stále v móde prijímania a potom odložia vysielanie ak je MAC vrstva v dátovej fáze prijímača.



Obr. 3.4: Z-Wave MAC rámec [8]

Na obrázku 3.4 je zobrazený generický Z-Wave MAC rámec. Hlavnými položkami, ktoré zaujímajú MAC vrstvu sú HomeID, source NodeID a destination NodeID. Ako už bolo spomenuté HomeID reprezentuje identifikátor jednej logickej Z-Wave siete. Všetky uzly pridané do siete zdieľajú tento identifikátor a tak je možné odlíšiť komunikáciu kolidujúcej siete. Každý uzol je unikátne identifikovaný pomocou NodeID, ktoré mu je priradené po spárovaní s kontrolérom. Na základe týchto identifikátorov je potom možné posielať správy medzi jednotlivými zariadeniami.

3.2.3 Transportná vrstva

Transportná vrstva sa stará o to ako sú prenášané dáta v sieti [10]. Je zodpovedná za potvrdenie doručenia rámca, znovuzaslanie, prebudenie batériovo napájaných zariadení a autentifikáciu zdroja paketu. Vrtva má k dispozícií maximálne 64 bajtov (MAC rámec) na prenos sieťovo relevantných dát plus zapúzdrenie aplikačného rámca. Všetky rámce v sieti sú identifikované pomocou HomeID a NodeID odosielateľa a príjemcu. Ostatné položky slúžia buď na kontrolu samotného rámca (dĺžka a kontrolný súčet) alebo na riadenie toku rámcov v sieti. Pre tento účel slúžia 2 bajty Frame Control zobrazené na obrázku 3.5. Typ rámca je určený položkou Header type:

Singlecast je základný typ, kde rámec je posielaný len jednému príjemcovi. Na základe položky ACK Req. buď je alebo nie je vyžadované potvrdenie doručenia rámca. V prípade, že je vyžadované potvrdenie rámca no to však nepríde, odosielateľ čaká náhodný

čas v rozmedzí 20ms a 100ms a vyskúša zaslať rámec ešte 2 krát. Po troch neúspešných pokusoch sa vzdáva a reportuje chybu. Zariadenie potom hľadá inú cestu zaslania rámca napr. pomocou smerovania.

Multicast umožňuje jednému odosielateľovi zaslať správu viacerým príjemcom. Tento rámec však v sieti Z-Wave nie je bežne využívaný.

Broadcast slúži na zaslanie správy všetkým dostupným uzlom v sieti. Jeho Header type je však Singlecast a odlišuje sa špeciálnym cieľovým NodeID (255 = 0xFF) v rámci.

Ack je posledný typ a slúži na potvrdenie prijatej singlecast správy v prípade, že je potvrdenie vyžadované. Payload rámca je pri tomto type prázdny.

Byte \ Bit	7	6	5	4	3	2	1	0
5	Routed	ACK Req.	Low power	Speed modified	Header type			
6	Reserved	Beaming Info.		Reserved	Sequence number			

G.9959(15)_FA.20

Obr. 3.5: Formát frame control bajtov [8]

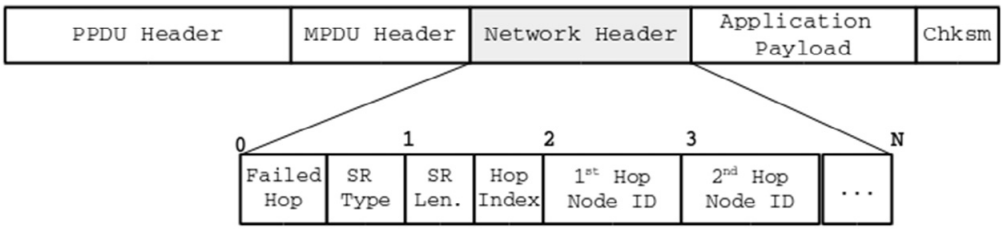
Ďalšou dôležitou položkou je routed špecifikujúce, či je do rámca zapúzdrená sieťová hlavička a má byť smerovaný. O to sa stará sieťová vrstva detailne popísaná v ďalšej sekcii. Sekvenčné číslo je v rozsahu 0x1 až 0xF a mení ho vyššia vrstva. V prípade znovuposielania používa transportná vrstva stále to isté číslo, ktoré dostala. Pri generovaní potvrdzovacieho Ack rámca je použité sekvenčné číslo z prijatej správy. Položk

Na verifikáciu, že rámec bol prijatý správne a nie je poškodený slúži posledná položka rámca v MPDU kontrolný súčet. Pre kanály 1 a 2 je to 8-bitový checksum. Kontrolný súčet je počítaný nad celým MPDU okrem seba samotného. Ide o jednoduchý algoritmus kombinácie všetkých bajtov pomocou XOR funkcie. Pre ochranu 63 bajtov rámca je však značne nedostatočný. V komunikačnom kanáli 3 je použitá už bezpečnejšia verifikácia pomocou CRC-16 (2 bajty). Generujúci polynóm je označovaný ako CRC CCITT a je inicializovaný hodnotou 0x1D0F.

3.2.4 Sieťová vrstva a smerovanie

Zatiaľ čo špecifikácia PHY, MAC, transportnej a časti aplikačnej vrstvy Z-Wave sú verejne dostupné, existuje len niekoľko verejne dostupných detailov týkajúcich sa sieťovej vrstvy [2].

Smerované rámce Z-Wave obsahujú sieťovú hlavičku, ktorá sa nachádza medzi MPDU hlavičkou a aplikačnou vrstvou. Položky hlavičky sú zobrazené na obrázku 3.6. Prvé 2 bajty sú zložené zo štyroch štvorbitových častí (nibble). Prvá časť je Failed Hop, použitý v prípade chybovej smerovanej správy, označujúci skok trasy kde nastala chyba. V inom prípade je jeho hodnota 0. Druhá časť označuje typ zdrojovej trasy (SR) a smerovacie uzly podľa neho zistia ako preposlať smerovanú správu. V tretej časti je uložená dĺžka SR v bajtoch a štvrtý nibble uchováva hop index na zistenie stavu SR pri jej preposielaní. V ďalších bajtoch hlavičký je uložená SR, konkrétne node ID jednotlivých uzlov. SR obsahuje len vnútorné uzly trasy a ich počet je maximálne 4. Zdrojový a cieľový uzol sú uložené v MPDU hlavičke. Po prirátaní cieľového uzlu je teda maximálna veľkosť Z-Wave trasy 5.



Obr. 3.6: Formát sieťovej hlavičky Z-Wave [2]

Preposielanie smerovaných správ vnútornými uzlami trasy prebieha nasledovne. Prítomnosť sieťovej hlavičky v rámci je určená pomocou bitového flagu routed v MPDU hlavičke. V prípade, že uzol obdržal správu s nastaveným týmto bitom, rozhodne či je zodpovedný za preposlanie správy. Na základe položky hop index sieťovej hlavičky, označujúcej offset trasy je zistený nasledujúci hop. Ak sa node ID uzlu nachádza na tejto pozícii, uzol preposiela správu. Aktualizuje hop index na základe typu smerovanej správy, prepočíta kontrolný súčet a odošle správu. Tabuľka 3.2 zobrazuje funkciu určujúcu, ktorý uzol je zodpovedný za preposlanie správy.

Hodnota hop indexu	**Cieľ ďalšieho hopu**
0 až N - 1	V SR, indexovaný hop indexom
N	Cieľový uzol v MPDU hlavičke
0xf	Zdrojový uzol v MPDU hlavičke

Tabuľka 3.2: Cieľ preposielania pre SR o dĺžke N, $1 \leq N \leq 4$

Tabuľka 3.3 zobrazuje 3 typy smerovaných rámcov spolu so správaním hop indexu. Hlavným typom je aplikačný rámec a potom sú to smerované rámce určené na informovanie zdrojového uzlu o stave aplikačného smerovaného rámca. ACK sa zasiela od cieľového uzlu v prípade úspešného doručenia správy, NACK posiela vnútorný uzol SR, ak je detekovaná chyba v preposielaní.

Hodnota typu SR	**Popis**	**Chovanie hop indexu**
0x00	Aplikačný rámec	Inkrementácia
0x03	ACK	Dekrementácia
0x05	NACK	Dekrementácia

Tabuľka 3.3: Známe typy smerovaných rámcov

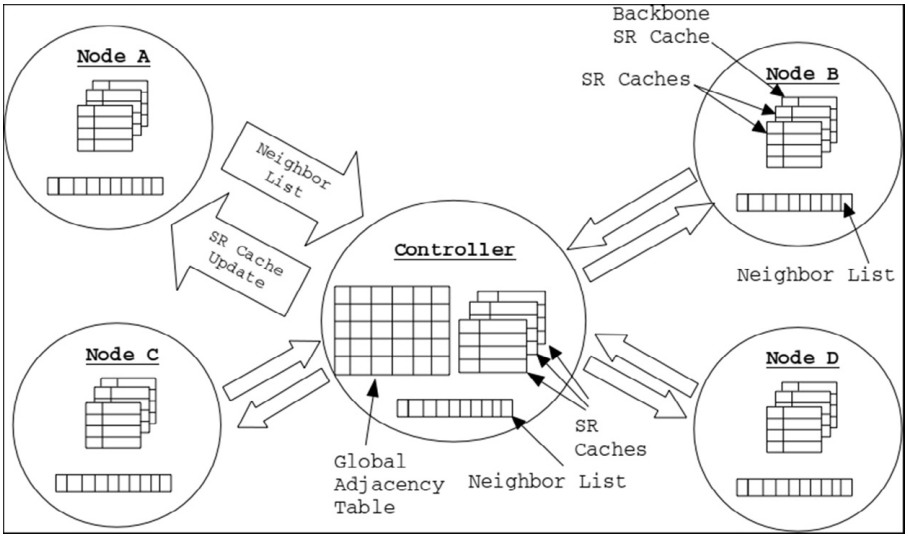
Ak chce zdrojový uzol poslať správu uzlu, ktorý nie je jeho sused vzdialený o 1 uzol v sieti, využije na to smerovaný aplikačný rámec. Medzi MPDU hlavičku a aplikačný payload vloží sieťovú hlavičku a nastaví bitový flag routed. Uzol vyberie zo svojej SR cache trasu, ktorú vloží do sieťovej hlavičky a nastaví jej ďalšie položky (typ smerovaného rámca, dĺžku trasy, hop index na 0). Po vytvorení rámca ho odošle. Následne uzly, ktoré sa identifikujú, že sú zodpovedné za preposlanie správy postupne prepošlú správu cieľovému uzlu. Preposielanie končí keď cieľový uzol zistí, že je správa určená pre neho (hop index odpovedá dĺžke trasy).

Zdrojový uzol je oboznámený o doručení správy do cieľa prijatím smerovanej ACK správy. Odosielajúci uzol skúša posielať smerovanú správu 3 krát. Pokusy sa opakujú po neobdržaní potvrdzovacej ACK správy do učitého časového limitu. Cieľový uzol po obdržaní

smerovanej správy vytvorí potvrdzovaciu ACK správu. Z prijatej správy skopíruje MPDU hlavičku spolu so sieťovou hlavičkou a vymení zdrojový a cieľový uzol. V sieťovej hlavičke zmení položku typ správy na ACK. Pred odoslaním dekrementuje hop index a prepočíta kontrolný súčet. Preposielacie uzly sa znovu identifikujú na základe hop indexu ako offsetu trasy a postupne dekrementujú hop index a preposielajú správu ku zdroju. Zdrojový uzol obdrží ACK keď je hop index nastavený na 0xf a použije sekvenčné číslo v MPDU hlavičke na identifikáciu potvrdenia pôvodnej smerovanej aplikačnej správy.

Uzly preposielajúce smerovaný aplikačný rámec sú zodpovedné za detekovanie a hlásenie zistených smerovacích chýb zdrojovému uzlu. Uzol po preposlaní rámca čaká na potvrdenie, že rámec bol prijatý ďalším uzlom. Vo všetkých prípadoch okrem posledného skoku je potvrdenie zistené odpočutím, že nasledujúci uzol preposlal správu. Posledný hop pred preposlaním nastaví v MPDU hlavičke značku ACK Req. Potvrdenie je teda realizované prijatím ACK MAC vrstvy od cieľa. Ak nie je zistené potvrdenie do časového limitu, uzol skúša posielať správu znova. V prípade, že nedosiahne úspech aspoň po 3 pokusoch vzdáva sa a hlási chybu. Vygeneruje smerovanú NACK správu podobne ako sa generuje ACK s tým rozdielom, že typ správy je NACK a hop index uzlu, ktorý nepotvrdil prijatie správy je skopírovaný do položky Failed Hop. Následne je správa preposlaná k zdroju rovnako ako ACK správa.

Na obrázku 3.7 sú zobrazené dátové štruktúry a činnosti potrebné na údržbu sieťovej topológie a zisťovanie trás. Lokálna topológia siete sa nachádza v každom smerovacom uzle ako zoznam susedov vzdialených o 1 uzol v sieti (NL - Neighbor List). Globálna topológia sa nachádza v tabuľke susednosti (Adjacency Table) spravovanej kontrolérom. Tabuľka je aktualizovaná vypýtaním si NL od každého uzlu v sieti. Každý uzol taktiež obsahuje niekoľko SR cache záznamov. Nové cesty sú generované kontrolérom na požiadanie, kde uzol požiada o SR k cieľovému uzlu. Kontrolér môže vygenerovať niekoľko SR a poskytnúť ich žiadajúcemu uzlu kde sú cachované. Okrem NL a niekoľkých SR cache záznamov obsahujú uzly taktiež uloženú Backbone SR cache kvôli dostupnosti kontroléra. Z-Wave smerovací protokol je považovaný za hybridný smerovací protokol.



Obr. 3.7: Architektúra manažmentu Z-Wave siete [2]

Na správu topológie slúži niekoľko príkazov (Command) zlúčených po neverejnou triedou príkazov (Command class) 0x01 označovanou tiež System CC. Jednotlivé príkazy je vidieť

v tabuľke 3.4. Príkazy využívajú 2 formáty primitívnej dátovej štruktúry, ktoré sú popísané v prílohe A. Prvou štruktúrou je tzv. NL primitív používaný na prenos stavu topológie v koordinačných správach na manažment smerovania. Druhou štruktúrou je záznam v SR cache predstavujúci smerovaciu cestu.

Hodnota	**Príkaz**	**Parametre**
0x04	Do NL Test	NL primitív
0x05	Get NL	Cieľové node ID
0x06	Report NL	NL primitív
0x07	NL Test Done	Žiadny
0x0C	SR Cache Assignment	Záznam v SR cache primitív
0x14	Backbone Cache Assignment	Záznam v SR cache primitív
0x15	SR Request	Cieľové node ID
0x18	NL Test	Neznámy parameter

Tabuľka 3.4: Zistené sieťové príkazy System CC

Každý uzol obsahuje zoznam susedných uzlov a ten si aktualizuje po prijatí správy Do NL Test od iného uzlu. V payloade správy sa nachádza NL primitív označujúci ktoré uzly majú byť otestované na susednosť. Test susednosti je realizovaný zasielaním správy NL Test uzlom využitím prenosového kanála 1. Uzly, ktoré na túto správu odpovedajú sú pridané do zoznamu susedov (test skončí neúspešne ak nie je obdržané potvrdenie po 3 pokusoch). Po otestovaní susednosti uzol posiela správu NL Test Done odosielateľovi Do NL Test.

Tabuľka susednosti je aktualizovaná keď kontrolér požiada uzol o jeho NL pomocou správy Get NL. Uzol odpovedá pomocou správy Report NL. Tabuľka sa aktualizuje po reštarte kontroléra alebo užívateľským vyžiadaním. Nevyžiadané Report NL by mali byť kontrolérom ignorované.

Uzol môže požiadať o smerovaciu cestu k cieľu pomocou správy SR Request. Kontrolér odpovedá sekvenciou možných ciest správami SR Cache Assignment. Taktiež mu môže poslať aj sekvenciu správ Backbone Cache Assignment pre obnovenie cache uzlu o kriticky dôležité cesty ku kontroléru. Ak je SR Request prijatá uzlom, ktorý nie je kontrolér, uzol odpovedá s jeho Backbone cache sekvenciou správ Backbone Cache Assignment. Žiadajúci uzol sa potom môže dotazovať kontroléra. Toto správanie poskytuje obnovovací mechanizmus pre uzly so zastaralými alebo poškodenými Backbone SR cache záznamami.

Ak je na poslanie správy vyžadovaná viac-skoková trasa, cieľový uzol sa nenachádza v tabuľke susedov vzdialených o 1 uzol v sieti, uzol si vyberá trasu zo svojej SR cache. V prípade, že je k dispozícii viac ako jedno riešenie, zdrojový uzol musí vykonať výber. Presná metóda výberu nie je známa a líši sa pre rôzne zariadenia. Pre účely tejto práce však nie je podstatná. Ak vybraná trasa zlyhá, uzol musí zvoliť alternatívnu trasu. Proces výberu sa opakuje až kým sa nevyčerpajú všetky trasy v cache a uzol sa nevzdá. Uzol môže taktiež vyskúšať poslať správu bez smerovania.

3.2.5 Aplikačná vrstva

Aplikačná vrstva rieši realizáciu konkrétnej funkcie [10]. Každá správa, ktorá je vymieňaná medzi dvoma zariadeniami je nazývaná príkaz (Command). Príkazy môžu byť klasifikované do 3 skupín.

- **SET** – požiadanie zariadenia, aby niečo vykonalo.

- **GET** – požiadanie zariadenia, aby niečo poskytlo.
- **REPORT** – reportovanie nejakej hodnoty alebo stavu zariadeniu.

Prihliadnuc na to, že zariadenia sú rozdielnych typov, jednotlivé príkazy nemusia vždy mať rovnaký význam a musia byť bližšie špecifikované. Z-Wave organizuje všetky príkazy do tried príkazov (Command Classes). Triedy príkazov popisujú funkciu zariadenia a zahrňujú všetky potrebné príkazy pre túto funkciu. Napríklad batériovo napájané zariadenia poskytujú funkciu zistenia úrovne nabitia batérie. To zahrňuje trieda príkazov Battery, ktorá obsahuje len príkazy GET a REPORT, pretože nie je čo nastavovať, ale je možné vyžiadať stav batérie a taktiež ho reportovať. Jednoduchý prepínač napr. v podobe inteligentnej zásuvky poskytuje funkcionalitu v podobe triedy príkazov Binary Switch CC. Táto trieda príkazov obsahuje aj príkaz SET, pretože prepínač je možné prepnúť. Zoznam všetkých tried príkazov je možné nájsť v [10] alebo priamo vo verejnej časti dokumentácie spolu s jednotlivými príkazmi. Trieda príkazov 0x01 označovaná tiež ako System nie je zverejnená. Niektoré jej príkazy však boli získané reverzným inžinierstvom.

Okrem tried príkazov existujú aj triedy zariadení (Device Classes). Pre zaistenie interoperability medzi rôznymi Z-Wave zariadeniami od rozličných výrobcov, určité zariadenia musia mať určité dobre definované funkcie nad rámec základnej triedy príkazov. Trieda zariadení popisuje určitý typ zariadení a definuje, ktoré triedy príkazov sú pre tieto zariadenia povinné. Rozlišuje 3 levely triedy príkazov.

- **Basic Device Class** rozlišuje či je zariadenie kontrolér alebo slave (sluha). Každé zariadenie músi mať špecifikovanú triedu zariadenia.
- **Generic Device Class** definuje, ktorú základnú funkciu musí zariadenie spĺňať ako kontrolér alebo sluha, ale je voliteľná.
- **Specific Device Class** je voliteľná a bližšie špecifikuje ďalšie funkcie zariadenia.

3.2.6 Zabezpečenie a zraniteľnosti

Zabezpečenie v sieti Z-Wave je realizované na niekoľkých úrovniach. Senzorické zariadenia, ktoré chcú komunikovať v konkrétnej Z-Wave sieti musia byť do siete pridané spárovaním s kontrolérom. Ak to zariadenia podporujú môže byť párovanie vykonané v zabezpečenom režime a následná sieťová komunikácia bude šifrovaná. Ďalšie zabezpečenie predstavoval fakt, že Z-Wave je proprietárny protokol a jeho detaily spočiatku neboli dostupné verejnosti.

Pôvodne bola Z-Wave komunikácia šifrovaná voliteľne pomocou AES. Kľúče potrebné na šifrovanie komunikácie sú generované počas párovacieho procesu z dát, ktoré zasiela primárny kontrolér uzlom. Táto časť komunikácie je žiaľ šifrovaná pomocou východzieho kľuča (samé nuly), ktorý je uložený v trvalej pamäti kontroléru. To je problém, pretože v prípade, že útočník odchytí túto komunikáciu môže si dopočítať hlavné šifrovacie kľúče. V roku 2016 predstavila aliancia Z-Wave nový S2 (Security 2) framework, ktorý vylepšuje bezpečnosť zariadení a ich komunikácie a od roku 2017 je povinný pre všetky novo certifikované zariadenia. S2 zvyšuje bezpečnosť komunikácie, pretože šifrovanie je povinné a prináša zabezpečenie procesu výmeny kľúčov pomocou mechanizmu ECDH [9].

Hlavným bezpečnostným problémom je, že väčšina dnešných zariadení bola certifikovaná ešte pred zavedením frameworku S2. Vzhľadom na to, že šifrovanie komunikácie bolo voliteľné a výrobcovia väčšinou uprednostňujú funkcionalitu na úkor bezpečnosti sú tieto zariadenia jednoducho napadnuteľné. Žiaľ aj v prípade, že zariadenia využívajú šifrovanie

je párovací proces napadnuteľný. Z-Wave sa spoliehalo na to, že je proprietárny protokol avšak ten bol postupne s využítím softvérovo definovaného rádia (SDR) a reverzného inžinierstva preskúmaný natoľko, že je dnes už jednoduché s využitím zariadenia akým je napr. HackRF One[2] ovládať zariadenia, v ktorejkoľvek nezabezpečenej sieti Z-Wave. Zariadenie je zobrazené na obrázku 3.8.

Šifrovanie pomocou S2 frameworku bolo spočiatku bezpečné avšak už sa tu vynoril tzv. Downgrade útok, kde vďaka spätnej kompatibilite dokáže útočník za určitých okolností vynútiť spárovanie zariadenia bez použitia S2 frameworku a tak kompletne obísť zabezpečenie. Útok spočíva v tom, že útočník vstúpi do komunikácie pri párovaní zariadenia s kontrolérom a presvedčí kontrolér, že zariadenie nepodporuje bezpečnostný štandard S2. Komunikujúce zariadenia sa tak dohodnú na použití napadnuteľnej verzie štandardu S0 [15]. Open-source implementácia OpenZWave, ktorú využíva aj BeeeOn brána, však k dnešnému dňu nepodporuje zabezpečenie S2 a preto naň v tejto práci nie je braný ohľad.

Obr. 3.8: HackRF One

Z-Wave poskytuje dôvernosť, integritu zdroja a integritu dát pomocou Security Command Class (zabezpečenie S0) [2]. Aplikačné rámce môžu byť zapúzdrené v zabezpečených rámcoch, ktoré sú šifrované a podpísané. Rámec je zabezpečený pomocou symetrickej kryptografie AES a troch zdieľaných kľúčov, ktoré pozná každý uzol v sieti, ktorý vyžaduje zabezpečenie. Pri procese párovania uzol špecifikuje, ktoré z jeho podporovaných tried príkazov budú zapúzdrené v Security CC. Použitie zabezpečenia sa viaže na účel zariadení. Napríklad zariadenia typu zámok dverí, detektor pohybu alebo alarm používajú zabezpečenie častejsie ako rôzne senzorické zariadenia, vypínače, žiarovky, termostat atď.

Security CC má však niekoľko obmedzení a zraniteľností. Za prvé je šifrovaná len aplikačná vrstva správ. Všetky položky MPDU hlavičky, sieťová vrstva a hlavička Security CC nie sú šifrované. Dátová integrita je poskytovaná pomocou kontroly podpisu aplikačnej vrstvy a niektorých položiek MPDU hlavičky (ID zdrojového uzlu, cieľového uzlu a dĺžky rámca). Žiadna položka sieťovej hlavičky nie je zahrnutá do kontroly podpisu. Keďže všetky spárované zariadenia zdieľajú rovnaký autentifikačný kľúč zabezpečenie integrity zdroja na

[2] https://greatscottgadgets.com/hackrf/one/

základe ID uzlu nie je možné. Zariadenia môžu rozlišovať len medzi správami, ktoré pochádzajú z dôveryhodných a nedôveryhodných zdrojov. Trieda príkazov takzvaná System CC, zahrňujúca príkazy na manažment siete, nevyžaduje šifrovanie.

Veľa kritických operácii neumožňuje spustenie len zo siete, ale vyžaduje nejakú interakciu človeka, napr. fyzickú interakciu so zariadeniami v sieti. Pri párovaní zariadenia je potrebný reset zariadenia, stlačenie tlačidla alebo znovuvloženie batérií pre uvedenie do párovacieho režimu. V tomto okamžiku zariadenie posiela tzv. Node Info správu do siete. Ďalšou operáciou je aktualizácia tabuľky susednosti. Keďže nevyžiadané NL Report správy sú ignorované, útočník pokúšajúci sa nakaziť topológiu musí reagovať na požiadavku z kontroléra.

Kontrolér v Z-Wave sieti je privilegovaný, pretože má globálny pohľad na sieť ako aj informácie o každom uzle. Mal by preto identifikovať a reagovať na integritné útoky. Napr. zahodiť správu poslanú do siete pomocou SDR, kde je ako zdrojový uzol uvedené id, ktoré nikdy nebolo spárované, zatiaľ čo ostatné uzly nie sú schopné identifikovať takýto vonkajší uzol.

Podvrhnutie komunikácie

Ako už bolo povedané, pomocou technológie SDR a špeciálneho zariadenia HackRF One je útočník okrem odpočúvania Z-Wave komunikácie schopný aj vysielať ním vytvorené rámce do siete. Podvrhnutie komunikácie alebo vydávanie sa za nejaký uzol porušuje zdrojovú integritu protokolu. Z-Wave zariadenia s výnimkou kontroléra implicitne dôverujú položkám zdroja a cieľa v MPDU hlavičke. Vďaka tomu je jednoduché vydávať sa za rámce pochádzajúce z kontroléra alebo iného uzlu. Zariadenia používajúce Security CC majú čiastočnú ochranu proti podvrhnutiu komunikácie vďaka kontrole podpisu. Útočník, ktorý nemá k dispozícii autentifikačný a šifrovacie kľúče nedokáže skonštruovať rámec zabezpečený pomocou Security CC, tak aby ho zariadenia nezahodili. Môže však stále previesť útok pomocou správ z podporovaných tried príkazov, ktoré nevyžadujú zapúzdrenie v Security CC, ako napr. System CC.

Skenovanie siete

Okrem pasívneho odpočúvania komunikácie v sieti môže útočník previesť aj aktívne skenovanie siete [6]. To je možné priamo pomocou existujúceho nástroja EZ-Wave [7].

Útočník môže najprv zoskenovať, ktoré uzly s príslušnými NodeID sa nachádzajú v sieti. Vyšle do siete broadcast správu kde ako zdrojové NodeID použije ID kontroléra teda 1, nastaví bitový flag ACK_REQ značiaci vyžiadanie potvrdenia pomocou ack správy a ako aplikačný payload použije NOOP CC 0x00. Uzly v dosahu útočníka odpovedia pomocou ack správy a tak útočník len zozbiera ich zdrojové NodeID. Z dôvodu nemožnosti overiť pôvod správy bez použitia zabezpečenia, uzly v sieti odpovedajú útočníkovi na všetky dotazy. Pre zistenie informácií o konkrétnom zariadení je potom možné použiť nasledujúce triedy príkazov a ich príkaz GET:

Device Manufacturer and Device Type poskytuje identifikačné číslo výrobcu, typ produktu a identifikačné číslo produktu. Z OpenZwave databázy zariadení je potom možné zistiť informácie o zariadení v human-readable forme.

Device Software Version poskytuje verziu firmvéru zariadenia. Je možné zistiť typ Z-Wave knižnice, verziu Z-Wave protokolu, subverziu Z-Wave protokolu, aplikačnú verziu a aplikačnú subverziu.

Basic Operational Status poskytuje status najbežnejších funkcií zariadenia.

Configuration Settings vyžaduje ďalší parameter pre adresovanie konkrétnych konfiguračných možností zariadenia.

Uzly odpovedajú príkazom REPORT príslušnej triedy príkazov. Výnimku tvorí správa pre získanie podporovaných tried príkazov (trieda príkazov System, príkaz Get Supported CC), kde ako odpoveď príde správa Node Info.

Tento útok nespôsobuje v sieti záškodnícku činnosť, ale poskytne útočníkovi odrazový mostík v podobe dôležitých informácií o zraniteľnostiach zariadení na použitie ďalších bežných útokov ako napríklad podvrhnutie komunikácie, DoS útok, atď..

Zraniteľnosti smerovania

Táto sekcia popisu zraniteľnosti sieťovej vrstvy a princíp akým je možné napadnúť smerovanie v sieti, ktoré môže vyeskalovať k útoku Black hole [2].

Modifikácia zoznamu susedov – Tento útok môže byť použitý na pridanie falošných uzlov do siete bez ich spárovania. Útočník sa môže votrieť do zoznamu susedov nejakého uzlu a postupne sa tak dostať až to tabuľky susednosti kontroléra. Pravdepodobnosť, že nakazený NL skončí v tabuľke susednosti kontroléra závisí na jeho správaní. Ak kontrolér najpr zašle správu Do NL Test a následne správu Get NL tak po vymaskovaní dostane len tie uzly, ktoré si vyžiadal, typicky spárované. Útočník počúva na kanále 1 a odpovedá na správu NL Test. Z iného procesu zašle uzlu správu Do NL Test kde vloží aj svoje ID. Uzol následne testuje útočníka pomocou správy NL Test a ten mu odpovedá, takže si falošný uzol vloží do svojho zoznamu susedov. Napadnutý uzol po vyžiadaní pomocou správy Get NL zasiela Report NL aj s útočníkovým ID.

Preskúmanie topológie siete – Útočník môže pasívne spracovávať smerované rámce a tak po dlhom čase zistiť topológiu siete, to je však nevýhodné. Lepšie riešenie je rozposlať správu Get NL a prijímať správy NL Report susedov vzdialených o 1 uzol v sieti. Následne skonštruovať validnú SR správu a zaslať Get NL susedom vzdialeným o 2 uzly v sieti(všetkým ostatným, ale len 2 hop odpovedajú) atď. až získa celú topológiu siete. Algoritmus je presne popísaný v [2].

Modifikácia SR cache – Ako už bolo spomenuté smerovacie cesty sú uzlom distribuované pomocou správ SR Cache Assignment a Backbone Cache Assignment. Útočník môže jednoducho využitím týchto správ zmeniť SR cache a Backbone cache všetkým uzlom v sieti okrem kontroléra a tak smerovať komunikáciu tak ako uzná za vhodné.

Modifikácia smerovaných rámcov – S využitím útoku modifikácie SR cache môže útočník presmerovať aplikačné rámce tak, aby boli posielané cez Man-In-The-Middle (MITM) uzol, kde môžu byť modifikované.

Rozsah modifikácie správy však musí spĺňať určité podmienky tak, aby Watchdog (uzol, ktorý generuje chybovú správu v momente keď pri preposielaní smerovanej správy zistí, že ďalší uzol správu nepreposlal) sledujúci MITM uzol nevygeneroval routovaciu NACK správu. Zdrojový uzol by potom mohol vybrať inú trasu a tak obísť MITM uzol. Preto sa musí útočník vyhnúť tomu, aby bola vygenerovaná NACK správa a on si tak zachoval anonymitu a maximalizoval životnosť svojej výhody. MITM uzol musí pri úprave rámca inkrementovať hop index a prepočítať kontrolný súčet. Môže

upraviť aplikačný payload správy, kde zmenou prvých dvoch bajtov reprezentujúcich triedu príkazov a príkaz môže drasticky zmeniť účel rámca. Prípadne môže tieto bajty nastaviť na 0, čím premení ktorýkoľvek rámec na neškodnú hello správu. MITM môže ďalej bez detekovania zmeniť id cieľa alebo ďalší skok. Nemôže zmeniť počet skokov nad dĺžku SR ani dĺžku rámca. Ďalšie povolené a nepovolené položky nie sú zatiaľ známe [2]. Pri zabezpečených rámcoch už z podstaty Security CC a kontroly podpisu nemôže modifikovať aplikačný payload šifrovaných rámcov, id zdrojového a cieľového uzlu, a dĺžku rámca.

Útok Black hole – Využitím spomenutých zraniteľností je v Z-Wave sieti uskutočniteľný tzv. Black hole (čierna diera) útok, pre daný pár zdrojového a cieľového uzlu. Útok spočíva v tom, že uzol pod vplyvom útočníka BHN (Black Hole Node) v tichosti zahadzuje aplikačné rámce miesto ich preposlania do cieľa. Black hole útok v Z-Wave sieti je možný, ak existuje aspoň jedna cesta medzi BHN a zdrojovým uzlom, ktorá neobsahuje cieľový uzol. Kvôli limitu dĺžky smerovacej cesty nesmie byť táto cesta dlhšia ako 4 skoky, inak BHN nemôže pripojiť cieľový uzol na koniec trasy. Je však potrebné zaručiť, aby watchdog BHN uzlu nevygeneroval smerovanú NACK správu. To je možné vložením fiktívneho uzlu medzi BHN a cieľový uzol do správy Cache Assignment, ktorý zaručí, že watchdog BHN uzlu odchytí správu preposlanú z BHN do fiktívneho uzlu a tak nevygeneruje NACK. Fiktívny uzol však už správu nedoručí do cieľa. Kvôli tomuto obmedzeniu musí teda existovať aspoň jedna cesta medzi BHN a zdrojovým uzlom, ktorá neobsahuje cieľový uzol a nie je dlhšia ako 3 skoky, čo dovoľuje vloženie BHN a fiktívneho uzlu ako vnútorné skoky priradenej SR do cieľa.

Útočník uskutoční útok nasledovným spôsobom. Preskúma sieť tak ako bolo popísané, čím si odvodí tabuľku susednosti. Tabuľka je použitá na zistenie, či existuje cesta medzi zdrojom a BHN, ktorá nie je dlhšia ako 3 skoky a neobsahuje cieľový uzol. Ak je takáto cesta objavená, pošle SR Cache Assignment správu cieľovému uzlu kde vnútorné skoky cesty v dátovej štruktúre záznamu cache obsahujú všetky vnútorné skoky SR od zdroja po BHN, BHN a fiktívny uzol, položka cieľ je nastavená ako cieľový uzol, index na nulu, a status na 0x10, aby bola cesta braná s najvyššou prioritou.

Rozsah útoku je obmedzený na sieťovú vrstvu. Zatiaľ čo rámce sú zahadzované bez toho, aby to bolo v sieti spozorované, aplikačná vrstva očakáva odpoveď na svoju požiadavku a môže sa opakovane pokúsiť o odoslanie požiadavky znova. Týmto spôsobom však nie sú zistené žiadne chyby smerovania, zdrojový uzol pokračuje v používaní trasy obsahujúcej BHN. Zatiaľ čo BHN úspešne zahadzuje opakované odpovede na požiadavky, používateľské rozhranie kontroléra môže eventuálne notifikovať užívateľa o komunikačných problémoch so vzdialeným uzlom, ktorý môže zahájiť zotavovanie siete v snahe napraviť zlyhanie. Útok je však možné ďalej opakovať a tak odstaviť komunikáciu od tohto uzlu.

Kapitola 4

Detekcia anomálií v IoT sieťach

Táto kapitola predstavuje možnosti detekcie anomálií v IoT sieťach, detekčný systém pre analýzu sieťovej prevádzky NEMEA, projekt SIoT (zabezpečená brána IoT), v rámci ktorého je táto práca riešená a existujúci detektor anomálií nad štatistickými dátami z bezdrôtových sietí.

4.1 Možnosti detekcie anomálií v IoT sieťach

Detekčné systémy z pohľadu spracovania dátového toku sieťovej prevádzky a detekcie incidentov môžeme rozdeliť na dve základné kategórie [13]:

Detekcia anomálií využíva štatistické modelovanie. Najprv sa z dátového toku vytvorí profil bežného chovania siete a ten sa následne porovnáva s aktuálnym chovaním. Ak správanie siete vybočí zo stanoveného profilu nad definované limity, tak je detekovaný incident. Výhodou tejto metódy je aj detekcia dosiaľ neznámych útokov. Nevýhodou však je väčšie množstvo falošných poplachov.

Detekcia signatúr využíva profil (signatúry) bežne známych útokov a je teda zameraná, na detekciu konkrétnych útokov. Výhodou tejto metódy je, že je pomerne presná a detekuje málo falošných poplachov. Nedokáže však detekovať nové formy útoku.

Aby bola vôbec možná nejaká detekcia v sieťach IoT je potrebné zabezpečiť zber dát na analýzu. Pre zber informácií je možné využiť nasledujúce prístupy:

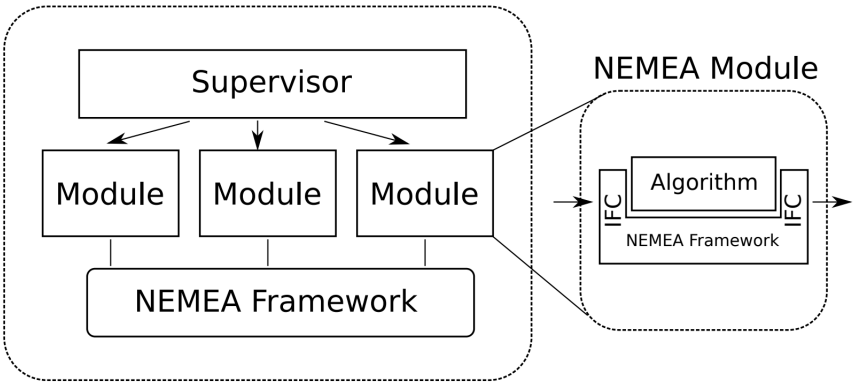
Externá sonda je samostatné zariadenie umiestnené v sieti, schopné sledovať prebiehajúcu komunikáciu a odosielať získané dáta na spracovanie. Výhodou je veľké množstvo možných rôznych získaných dát. Komplikáciou je šifrovaná sieťová komunikácia a cena kvalitnej sondy.

Prevádzkové štatistiky sú získavané priamo na bráne IoT z jej rozhraní. Nie je potrebné žiadne externé zariadenie, ale brána to musí umožňovať. Výhodou je pomerne nenáročné získavanie dát o sieťovej prevádzke, avšak štatistiky nie sú tak podrobné ako u externej sondy.

Testbed je metóda kedy sa testované a meracie zariadenie umiestnia do špeciálneho prostredia s cieľom overiť, či novo pripojovaný senzor spĺňuje všetky bezpečnostné požiadavky a neobsahuje známe zraniteľnosti. Meracie prvky sú nástroje schopné odpočúvať komunikáciu. Táto metóda sa využíva hlavne pri penetračných testoch.

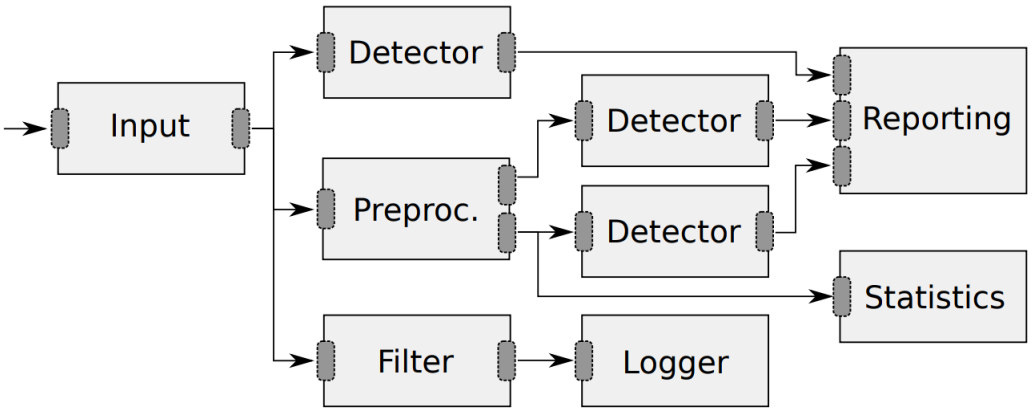
4.2 Systém NEMEA

NEMEA (Network Measurements Analysis) je tokovo orientovaný, modulárny detekčný systém pre analýzu sieťovej prevádzky [3]. Skladá sa z nezávislo bežiacich heterogénnych modulov, ktoré sú prepojené jednosmernými komunikačnými rozhraniami a kontinuálne spracovávajú tok dát vo forme prúdu správ. Správy zvyčajne predstavujú informácie o sieťovom toku, detekované bezpečnostné incidenty, výsledky nejakej analýzy alebo čokoľvek iné.



Obr. 4.1: Architektúra NEMEA modulov [3].

Obrázok 4.1 zobrazuje architektúru NEMEA systému. Jedná sa o moduly, ktoré môžu byť monitorované a kontrolované nástrojom zvaným Supervisor. Každý modul je program implementujúci nejaký algoritmus spracovania dát alebo detekcie (napr. zber sieťového toku, filtrovanie, spracovanie, detekciu anomálií, ukladanie logovacích záznamov alebo hlásenie výsledku nejakej analýzy). Všetky moduly využívajú NEMEA framework, ktorý implementuje funkcionalitu komunikácie medzi modulmi, dátový formát správ, bežne potrebné algoritmy alebo dátové štruktúry.



Obr. 4.2: Príklad prepojenia niekoľkých NEMEA modulov [3].

Framework je vyvinutý v jazyku C a systém prináša podporu pre vyvíjanie modulov v jazykoch C/C++ a Python. Najdôležijejšou časťou frameworku je TRAP knižnica (Traffic Analysis Platform) implementujúca komunikačné rozhrania a základnú funkcionalitu potrebnú pre každý modul a UniRec knižnica implementujúca dátový formát pre binárnu reprezentáciu prenášaných správ medzi komunikačnými rozhraniami. Systém NEMEA môže bežať na ľubovoľnom unixovom operačnom systéme a jednotlivé moduly môžu byť rozmiestnené na viacerých fyzických strojoch. Systém je teda vhodný na použitie na domácich smerovačoch so systémom OpenWRT alebo na bráne IoT.

Obrázok 4.2 zobrazuje príklad inštancie NEMEA systému s jedným modulom zaisťujúcim zber sieťového toku a niekoľkými modulmi na spracovanie sieťového toku, vytváranie štatistík, detekciu incidentov a ich hlásenie.

Komunikácia medzi modulmi

Na prijímanie a posielanie správ využívajú NEMEA moduly jednosmerné vstupné a výstupné TRAP komunikačné rozhrania (IFC). Každý modul môže používať niekoľko vstupných aj výtupných rozhraní. Výstupné rozhranie modulu môže byť pripojené k niekoľkým vstupným rozhraniam iných modulov. Všetci príjemcovia dostávajú potom rovnaké dáta od toho istého posielajúceho. Je tak možné tieto moduly za seba reťaziť a nad tými istými dátami vykonávať rôznu analýzu, filtrovanie alebo detekciu.

Rozhrania sú vlastne abstrakciou nad rôznymi metódami medziprocesovej komunikácie. Hlavné typy rozhraní sú unixový socket používaný na komunikáciu na jednom fyzickom stroji a TCP socket používaný na komunikáciu po sieti. Ďalej je to rozhranie používané na uloženie výstupu do súboru, z ktorého môže byť následne načítané vstupným rozhraním alebo špeciálne výstupné rozhranie, ktoré zahadzuje správy do neho poslané.

Typy rozhraní a ich parametre (názov socketu, IP adresa a port, názov súboru) sú definované pri štarte programu a spracované pomocou TRAP knižnice, ktorá sa stará o komunikáciu. Vývojár sa teda môže zamerať výlučne na implementáciu účelu vyvíjaného modulu.

Dátový formát

Dáta prenášané medzi modulmi skrz komunikačné rozhrania majú takmer výhradne binárny formát UniRec (možnosť neštrukturovaných dát alebo JSON). UniRec je výkonný binárny formát pre ukladanie a prenos dátových záznamov podobný prostej štruktúre jazyka C. Oproti štruktúre jazyka C podporuje položky premenlivej veľkosti. Je to generická dátová štruktúra, ktorej formát je definovaný pomocou takzvanej šablóny. Oproti iným formátom ako JSON je navrhnutá na veľmi rýchly prístup k jednotlivým položkám s takmer rovnakou rýchlosťou ako štruktúra jazyka C, ale narozdiel od C štruktúry môže byť šablóna definovaná za behu programu.

Všetky správy prenášané jedným komunikačným rozhraním (IFC) majú rovnakú šablónu. Pre každé IFC rozhranie sa teda špecifikuje šablóna a v momente kedy sa má vstupné rozhranie pripojiť na výstupné sa automaticky skontroluje ich kompatibilita. Ak formát výstupného rozhrania obsahuje všetky položky vyžadované vstupným rozhraním spojenie je nadviazané inak je odmietnuté.

4.3 SIoT – Zabezpečená brána IoT

Hlavným cieľom projektu SIoT je posilniť bezpečnosť v neustále sa rozširujúcich sieťach Internetu vecí, ktoré obsahujú často nezabezpečené a ľahko napadnuteľné prvky. Zabezpečená brána IoT predstavuje sadu softvérových nástrojov umožňujúcich prevádzkovať bezdrôtové IoT siete, monitorovať ich chovanie a detekovať narušenie bezpečnosti prevádzky. Zabezpečená brána je obvykle nasadzovaná na domácich smerovačoch so systémom OpenWRT. Z dôvodu nedostatočného výkonu týchto smerovačov je riešenie postavené na modulárnej architektúre, aby bolo možné časť funkcionality presunúť na zariadenia s ďalšími výpočetnými prostriedkami.

Funkciu IoT brány zaisťuje BeeeOn Gateway 2.2.1, ktorá prepája zariadenia z rôznych IoT sietí s aplikačným serverom. Okrem toho brána zhromažďuje dáta o prevádzke a predáva ich ďalej na vyhodnotenie detekčnými modulmi zabezpečenej brány IoT. Okrem detekčných modulov môžu byť súčasťou nasadenia brány aj moduly zaisťujúce zber dodatočných informácií. To je výhodné, pretože samotná IoT brána neumožňuje zber všetkých potrebných dát pre detekciu incidentov. Všetky moduly sú postavené na NEMEA frameworku, ktorý zaisťuje komunikáciu medzi nimi. Výstupy modulov môžu byť ukladané a preposielané na ďalšie spracovanie alebo zaistenie reakcie na udalosti. Jednou z možností je aj ukladanie a následná vizualizácia dát v nástroji Coliot[1], ktorý je ďalšou časťou softvéru zabezpečenej brány IoT.

4.3.1 WSN Anomally detector

V rámci projektu SIoT vznikol detektor anomálií nad štatistickými dátami z bezdrôtových sietí IoT [13]. Nástroj sa skladá z niekoľných NEMEA modulov a rozširuje funkcionalitu IoT brány BeeeOn a možnosti detekcie. Hlavnými modulmi sú kolektor, ktorý zbiera dáta z brány a posiela ich na analýzu detektoru. Detektor je hlavnou komponentou celého riešenia a vyhodnocuje dáta z kolektoru. Nad konfiguráciou zvolenými dátami, ktoré si ukladá do časových rád umožňuje vytvoriť profil a v prípade, že sa objavia nejaké anomálie vygeneruje na výstup hlásenie. Môže sa jednať o kontrolu limitov kde napr. určité namerané hodnoty zo senzorov by nemali presiahnuť určitú hodnotu. Ďalej to môže byť periodicita dát alebo množstvo prenášaných správ atď.

Kolektor má niekoľko výstupných rozhraní, nad ktorými môže detektor robiť analýzu:

- **onExport** – reprezentuje namerané dáta zo senzorov, ktoré sú odoslané vždy po prijatí.
- **onDispatch** – umožňuje získavať zadané užívateľské príkazy.
- **onHciStats** – získava v pravidelnom intervale štatistiky z Bluetooth adaptéra o pripojenej sieti.
- **onDriverStats** – v pravidelnom intervale generuje štatistiky o Z-Wave sieti z pripojeného Z-Wave kontroléru.
- **onNodeStats** – podobne ako onDriverStats generuje v pravidelnom intervale štatistiky z pripojeného Z-Wave kontroléru, ale pre jednotlivé uzly v sieti.

Štatistiky zo Z-Wave siete sú získavané z knižnice OpenZWave umožňujúcej ovládať kontrolér. Môže sa jednať o celkový počet správ v sieti, počet zahodených správ alebo počet

[1] https://github.com/CESNET/coliot

odoslaných správ ku konkrétnemu uzlu v sieti atď. Nevýhodou týchto dát je, že sú to dáta len z pohľadu daného kontroléra. Hlavne pre konkrétne uzly v sieti to nemusia byť skutočné hodnoty a rada útokov tak nemusí byť vôbec odhalená. Tento problém je možné vyriešiť vytvorením externej sondy, ktorá bude získavať dáta priamo zo siete a počítať štatistiky nad nimi.

Kapitola 5

Analýza sieťovej prevádzky senzorických zariadení Z-Wave

Táto kapitola je venovaná vykonanej analýze sieťovej prevádzky vybraných Z-Wave zariadení, ktoré sú použité pri riešení práce. Najprv sú predstavené vybrané senzorické zariadenia. Nasleduje popis spôsobu získavania dát na analýzu, ktorý je použitý aj pre detekčný systém. Poslednou časťou je analýza sieťovej prevádzky s ohľadom na časové a volumetrické parametre komunikácie. Samotná analýza je vykonaná nad komunikáciou senzorických zariadení v sieti vytvorenej Z-Wave kontrolérom pripojeným k BeeeOn bráne.

5.1 Vybrané zariadenia

AeoTec Z-Stick Gen5 – ZW090-C je Z-Wave kontrolér pripojiteľný cez USB. Umožňuje vytvorenie Z-Wave siete identifikovanej pomocou HomeID a pripojenie senzorických zariadení. V zariadení sa nachádza batéria, ktorá spolu s tlačidlom umožňuje bez použitia hostiteľského systému pridať alebo odstrániť zariadenia zo Z-Wave siete. Stav zariadenia je indikovaný pomocou LED diody. Zariadenie je možné vrátiť do výrobných nastavení pomocou tlačidla umiestneného na zadnej strane. Po vyresetovaní sa vygeneruje nový náhodný identifikátor siete. Po pripojení kontroléra pomocou USB k hostiteľskému systému sa kontrolér prepne do režimu SerialApi. V tomto režime je vždy prebudený a prijíma rámce v sieti jemu určené. Zároveň je ho možné ovládať pomocou sériového API (v našom prípade skrz BeeeOn bránu s využitím OpenZWave knižnice). Pomocou hostiteľského systému je možné pridávať a odoberať senzorické zariadenia, ovládať ich a prijímať od nich dáta. Kontrolér nepodporuje bezpečnostné štandardy S2 ani S0 a tak žiadna komunikácia vo vytvorenej sieti nie je šifrovaná [18].

Obr. 5.1: AeoTec Z-Stick Gen5 – ZW090-C [18]

AeoTec MultiSensor 6 – ZW100-C predstavuje 6 samostatných senzorov, ktoré sa nachádzajú v jednom Z-Wave zariadení. Multisenzor umožňuje merať teplotu, vlhkosť,

intenzitu svetla, UV žiarenie a detekovať pohyb a otras. Stav zariadenia je indikovaný pomocou LED diódy. Pomocou tlačidla na zadnej strane je ho možné pridať alebo odobrať zo Z-Wave siete a tiež vyresetovať do výrobných nastavení. Zariadenie môže byť napájané batériovo alebo pomocou USB adaptéra, čo určuje v akom režime zariadenie pracuje. Pri napájaní z batérie sa zariadenie uspáva a neumožňuje preposielanie smerovaných rámcov v sieti. V prípade napájania zo siete sa zariadenie neuspáva a tak umožňuje preposielanie smerovaných rámcov. Zariadenie podporuje zabezpečenie S0. Ďalej podporuje aj asociáciu v sieti a tak napr. pri detekcii pohybu dokáže zaslať správu inteligentnej žiarovke a tým ju zapnúť [19].

Obr. 5.2: AeoTec MultiSensor 6 – ZW100-C [19]

Climax Door Contact – DC-23ZW umožňuje detekovať otvorenie dverí alebo okna pomocou magnetického senzoru. Zariadenie je možné pridať a odobrať zo Z-Wave siete pomocou tlačidla na prednej strane. Pomocou tohto tlačidla je možný taktiež reset zariadenia. Aktuálny stav je indikovaný LED diódou. Zariadenie je napájané batériovo, väčšinu času je uspané a tak nie je schopné preposielať smerované správy v sieti. Podporuje zabezpečenie S2 a asociáciu [20].

Obr. 5.3: Climax Door Contact – DC-23ZW [20]

Fibaro Wall Plug E/F - FGWPE/F-102 je bezdrôtovo ovládaná zásuvka, do ktorej je možné pripojiť elektrické zariadenia s výkonom do 2500W. Inteligentnú zásuvku je možné zapínať a vypínať bezdrôtovo alebo pomocou zabudovaného tlačidla. Tlačidlo slúži aj na pridanie alebo odobranie zariadenia zo Z-Wave siete a vrátenie zariadenia do výrobných nastavení. Inteligentná zásuvku dokáže merať výkon a spotrebu elektrickej energie. Aktuálny výkon ako aj stav zariadenia je indikovaný viacfarebnou LED diódou. Zariadenie je napájané priamo z elektrickej siete a tak dokáže preposielať smerované rámce v Z-Wave sieti. Podporované je zabezpečenie S0 aj asociácia [21].

Obr. 5.4: Fibaro Wall Plug E/F - FGWPE/F-102 [21]

D-Link Z-Wave Motion Sensor – DCH-Z120 predstavuje senzor pohybu. Umožňuje teda informovať o neoprávnenom vstupe do miestnosti. Okrem pohybového senzoru obsahuje aj senzor teploty a svetelnosti. Zariadenie je napájané batériou. Pridanie a odobranie zo siete, ako aj reset zariadenia sa realizuje stláčaním tlačidla na jeho zadnej strane a aktuálny stav je indikovaný LED diódou. Zariadenie podporuje asociáciu. Nepodporuje zabezpečenie S2 ani S0 [17].

Obr. 5.5: D-Link Z-Wave Motion Sensor – DCH-Z120 [17]

5.2 Odchytávanie sieťovej prevádzky pomocou SDR

Ako už bolo spomenuté z BeeeOn brány je možné získať štatistiky o komunikácií v Z-Wave sieti. Štatistiky sú získavané z pripojeného kontroléra pomocou OpenZWave knižnice. Okrem štatistík je z brány podobne možné získať aj niektoré udalosti z kontroléra. Tieto udalosti reprezentujú napr. inicializáciu kontroléra, spárovanie nového zariadenia, alebo časť aplikačnej vrstvy prijatých a odoslaných správ kontrolérom. Štatistiky spolu s udalosťami však neposkytujú dostatočný pohľad na sieť pre analýzu sieťovej prevádzky a detekciu niektorých útokov.

Vhodným riešením sa javí vytvorenie sondy, ktorá bude odchytávať komunikáciu v sieti a poskytovať dáta na analýzu. Z-Wave je však proprietárny protokol a bežné zariadenia to nepodporujú. Nie je tak možné priamou cestou získavať potrebné dáta o sieťovej komunikácií. Existuje riešenie s využitím softvérovo definovaného rádia (SDR). Rádio komponenty ako filtre, modulátory a demodulátory, atď. sú typicky implementované v hardvéri. Moderné výpočetné prostriedky však umožňujú väčšinu týchto komponentov implementovať aj softvérovo. To umožňuje jednoduché spracovanie signálu a tým aj výrobu lacných rádií.

Na obrázku 5.6 je zobrazený USB adaptér, pôvodne DVB-T tuner, ktorý obsahuje čip RTL2832U vhodný na použitie ako SDR. Keďže jeho pôvodné použitie ako DVB-T tuner zabezpečilo masovú produkciu jeho cena je okolo 20 $. Vznikol projekt RTL-SDR[1], kde vďaka možnosti priameho prístupu k surovým I/Q dátam na čipe RTL2832U bol tento DVB-T tuner konvertovaný na širokopásmové softvérovo definované rádio pomocou vlastného softvérového ovládača.

Obr. 5.6: RTL-SDR USB adaptér

S využitím RTL-SDR je tak možné vytvoriť sondu na získavanie sieťovej prevádzky, ktorá prinesie kompletný pohľad na dianie v sieti. Na použitie sa ponúka niekoľko existujúcich nástrojov. Nástroj EZ-Wave [7] je postavený nad GNU Radio a Scapy Radio projektom. Vylúčil som ho však z dôvodu, že dokáže odchytiť len rámce kanála 2 a jeho následné použitie pre detekčný systém by bolo nevhodné. Nasadenie nástroja na OpenWRT by bolo pravdepodobne nemožné. Rozhodoval som sa teda medzi nástrojom Waving-Z[2], čo je fork nástroja rtl-zwave[5] a samotným rtl-zwave. Rozhodol som sa jednotlivé nástroje otestovať, z čoho mi vyšlo, že rtl-zwave je spoľahlivejší. Kanály 2 a 3 dosahovali približne rovnakú úspešnosť odchytávania rámcov, ale na kanále 1 rtl-zwave jasne dominoval. Jeho výhodou je aj to, že ide pomerne o jednoduchý kód bez žiadnych ďalších závislostí a je teda vhodný na ďalšie použitie v detekčnom systéme.

5.3 Analýza sieťovej prevádzky

Analýza sieťovej prevádzky je vykonaná nad komunikáciou senzorických zariadení v sieti vytvorenej Z-Wave kontrolérom pripojeným k BeeeOn bráne. Použité zariadenia ako aj spôsob získavania dát na analýzu je popísaný v predchádzajúcich častiach kapitoly. BeeeOn brána v aktuálnej verzii nepodporuje všetku funkcionalitu vybraných zariadení. Podporuje pridanie a odobranie zariadenia do resp. zo siete. Ďalej podporuje príjem nameraných senzorických dát a ovládanie zariadení ako napr. prepnutie inteligentnej zásuvky. Nepodporuje

[1] https://www.rtl-sdr.com
[2] https://github.com/baol/waving-z

pokročilejšie funkcie ako napr. vytvorenie asociácie medzi zariadeniami alebo zmenu konfigurácie zariadení. Nie je tak možné nastavenie vlastného intervalu pre zasielanie dát od batériovo napájaných zariadení. Keďže použitý kontrolér nepodporuje bezpečnostné štandardy S2 ani S0 žiadna komunikácia vo vytvorenej sieti nie je šifrovaná a vytvorená sieť môže byť jednoducho napadnuteľná. Z tohto dôvodu je dôležité poskytnúť zabezpečenie formou systému na detekciu anomálií a čo najrýchlejšie odhalenie útočníka.

Ako už bolo spomenuté senzorické zariadenia sú pridávané do siete procesom zvaným párovanie. Najprv je potrebné zapnutie párovacieho režimu na bráne, čím začne kontrolér periodicky všesmerovo vysielať do siete správy Transfer Presentation triedy príkazov System. Kontrolér v párovacom režime prijíma rámce všetkých Z-Wave sietí. V dobe kedy je kontrolér v párovacom režime je potrebné 3 krát stlačiť tlačidlo na zariadení, aby sa taktiež preplo do párovacieho režimu. Zariadenia pri prepnutí do párovacieho režimu pošle všesmerovo správú NodeInfo triedy príkazov System. V správe používa náhodné HomeID a zdrojové NodeID 0. Ďalej špecifikuje svoj typ a zoznam podporovaných tried príkazov. Ostatné položky správy mi nie sú známe. Kontrolér odpovedá rámcom s HomeID, ktoré prijal a cieľovým NodeID 0. Ide o správu AssignID triedy príkazov System a uzlu zasiela pridelené NodeID spolu s HomeID siete kontroléra. V tomto momente je zariadenie pridané do siete a je v nej unikátne identifikované. Kontrolér si od neho následne vyžiada ďalšie informácie a zariadenie mu ich posiela. Medzi tieto informácie patrí výrobca zariadenia a jeho typ, verzia firmvéru, prípadne konfigurácia a základný prevádzkový stav.

Po spárovaní pošlú zariadenia namerané hodnoty. Následne zariadenia nekomunikujú až do momentu keď sú buď prebudené na vonkajší podnet alebo sa naplní čas pravidelného prebudenia. Východzí interval pravidelného prebudenia použitých senzorov je jedna hodina. Senzor otvorenia dverí pri detekovaní otvorenia a zatvorenia posiela správu Alarm. To isté sa týka senzoru pohybu, ktorý túto správu posiela pri detekovaní pohybu v miestnosti. Inteligentná zásuvka zas posiela zmenu stavu pri jej manuálnom zapnutí alebo vypnutí. Pri pravidelnom prebudení posielajú senzory správu s nameranými veličinami. Okrem toho posielajú úroveň nabitia batérie, ak sú napájané batériovo. Z použitých zariadení je možné z BeeeOn brány ovládať len inteligentnú zásuvku. Tú je možné zapnúť a vypnúť. Vtedy sú kontrolérom posielané správy na zmenu stavu a zásuvka odpovedá nový stavom.

Batériovo napájané zariadenia po prebudení najpr posielajú notifikáciu o tom, že sa prebudili na určitý čas a tak im môže kontrolér zaslať nazhromaždené požiadavky. Môže ísť napr. o správy pre manažment smerovania v sieti. Ak už kontrolér nemá žiadne ďalšie požiadavky zasiela im pokyn značiaci, že sa môžu znova uspať.

Trvalo napájané zariadenia (inteligentná zásuvka a multisenzor pri napájaní z USB) dokážu preposielať smerované správy a tak umožňujú využitie smerovania v Z-Wave sieti. Celé smerovanie prebieha na pozadí na úrovni kontroléra a BeeeOn brána o tom vôbec nevie. Odsledoval som, že preskúmanie siete kontrolérom a vytvorenie sieťovej topológie prebieha pri štarte kontroléra, spárovaní nového zariadenia, ale spozoroval som ho aj v pre mňa náhodnom čase. Samotné smerovanie v sieti funguje rovnako ako bolo naštudované. Použitie smerovania generuje veľké množstvo takmer rovnakých správ líšiacich sa len v sieťovej hlavičke. Smerovanie samo o sebe vyžaduje potvrdzovanie správ. Mimo smerovaných správ som odsledoval, že väčšina správ posielaných v sieti vyžadujú potvrdenie. V prípade problémov v komunikácií a nedoručení potvrdenia sú rovnaké správy zasielané znova tak ako bolo naštudované, čo generuje taktiež väčšie množstvo správ v sieti.

Kapitola 6

Návrh detekčného systému

Táto kapitola sa zaoberá návrhom detekčného systému schopného odhaliť vybrané anomálie a útoky na Bluetooth Low Energy a najmä na Z-Wave sieť, ktorej som sa venoval najväčšiu časť práce. Pri Bluetooth Low Energy ide o detekciu nezvyčajného opakovaného párovania senzorických zariadení s kontrolérom, čo predstavuje potenciálny útok vynútenia nového párovacieho procesu, kde pri jeho odpočutí je útočník schopný napadnúť následnú šifrovanú komunikáciu so zariadením. Detekcia bude realizovaná priamo nad odchytávanými dátami z HCI rozhrania stroja, na ktorom bude detektor nasadený.

Pri Z-Wave ide o komplexnejšiu prácu. Za prvé je to spôsob získavania sieťovej komunikácie Z-Wave všetkých prenosových kanálov pomocou SDR. Detekované bude skenovanie siete, z čoho môže útočník zistiť informácie o senzorických zariadeniach, na základe ktorých dokáže presne určiť typ daného zariadenia a následne zaútočiť iným spôsobom. Ďalej budú detekované útoky na smerovanie v sieti, pomocou ktorých môže začať útočník vystupovať v komunikácií ako MITM a tak zahadzovať alebo pozmeňovať komunikáciu. Poslednou časťou je rozšírenie existujúceho detektora WSN Anomaly o štatistiky získané priamo zo Z-Wave siete a identifikácia scenárov anomálií, ktoré bude možné detekovať.

V nasledujúcom texte sú ako prvé špecifikované požiadavky na detekčný systém. Nasleduje popis princípu jednotlivých detekcií a na záver je popísaná celková navrhnutá architektúra detekčného systému spolu s jednotlivými modulmi.

6.1 Požiadavky na detekčný systém

Ešte pred samotným návrhom riešenia je potrebné špecifikovať a ujasniť si požiadavky na výsledný systém. Požiadavky vyplývajú zo zadania diplomovej práce, z výskumného projektu SIoT, v rámci ktorého je práca riešená a v neposlednom rade zo zvolených detekcií útokov a anomálií na základe vykonanej analýzy.

Detekčný systém ako celok musí byť nasaditeľný na domácich smerovačoch so systémom OpenWRT alebo vhodnej vstavanej platforme s procesorom ARM, kde bude bežať aj IoT brána. Tieto smerovače majú obmedzený výkon a je preto potrebné, aby systém spotrebovával čo najmenej výpočetných a pamäťových prostriedkov. Okrem toho je nutné brať ohľad na možnosť použitia zvolených technológií na tejto platforme. Celé riešenie zabezpečenej brány SIoT vyžaduje rozšíriteľnosť a flexibilitu nasadenia v podobe modulárnej architektúry detekčného systému. To umožní nasadenie len potrebných komponentov a prípadné fyzické oddelenie komponentu zaisťujúceho zber dát a komponentu na analýzu a vyhodnocovanie dát. Vďaka tomu bude možné prevádzkovať detekčný systém aj na zariadeniach s veľmi

obmedzenými výpočetnými prostriedkami, kde bude bežať len to najnutnejšie a ostatné sa môže presunúť na zariadenie s vyšším výkonom.

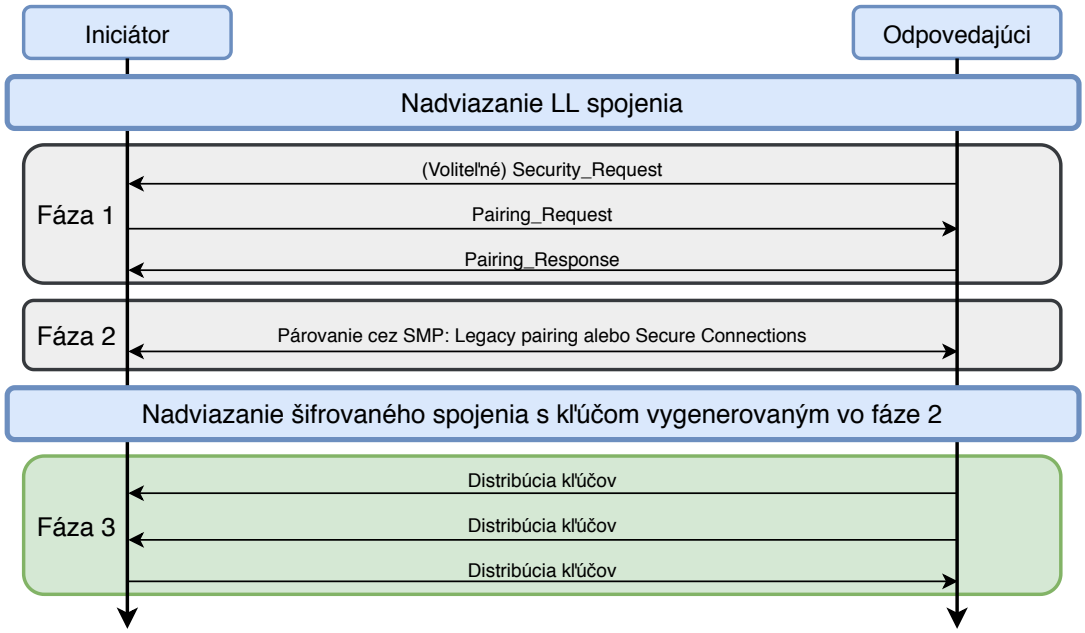
Pre samotné detekcie je potrebné zaistiť vhodný zber dát z IoT sietí. Z týchto dát musia byť detekčné algoritmy schopné odhaliť vybrané útoky a anomálie. Dôležitá je presnosť detekcie a teda schopnosť odhaliť čo najviac prevedených útokov a zároveň čo najmenší počet hlásení o útokoch kedy skutočne neprebehli. Hlásenia o útokoch by mali mať jednotný formát v rámci všetkých detekčných modulov projektu SIoT a je preto potrebné na to brať ohľad.

6.2 Detekcia opakovaného párovania Bluetooth Low Energy

Aby bolo vôbec možné nejakú detekciu vykonávať je potrebné nájsť spôsob odchytávania BLE komunikácie. Vhodným riešením sa javí vytvorenie RAW socketu nad HCI rozhraním. Týmto socketom je možné odchytávať všetku komunikáciu prúdiacu cez vybrané HCI rozhranie stroja, na ktorom beží príslušný detekčný modul. Nejedná sa len o BLE komunikáciu, ale taktiež aj Bluetooth Classic komunikáciu. Z rozhrania je možné odchytávať nasledujúce typy paketov, ktoré sú definované v Bluetooth špecifikácii, spolu s časovou značkou a smerom paketu:

- **Command** predstavuje príkazy zasielané od hostiteľa ku kontroléru, ktorý ich vykonáva. Môže ísť o spustenie skenovania zariadení v okolí, nadviazanie spojenia s objaveným zariadením, príkaz na začatie šifrovania spojenia a mnoho iných.
- **Event** slúži na informovanie hostiteľa o udalostiach na nižších vrstvách a zasiela ho kontrolér. Môže sa jednať o informáciu o vykonaní zaslaného príkazu, objavenie zariadenia pri skenovaní okolitého prostredia, informáciu o vytvorení spojenia so vzdialeným zariadením a mnoho iných. Narozdiel od príkazov kde sú BLE príkazy definované rovnako ako Bluetooth Classic sú BLE eventy zabalené v špeciálnom evente zvanom LE Meta.
- **ACL/SCO Data** sú dátové pakety L2CAP protokolu. Môže sa jednať o pakety získaných senzorických dát a iné, ale pre detekciu párovania sú dôležité pakety SMP protokolu, ktorý riadi párovanie.

Odchytené HCI pakety je následne možné analyzovať a vyhľadať v nich párovací proces. Párovací proces BLE sa skladá z niekoľkých krokov ako vidieť na obrázku 6.1 a samotné párovanie pomocou SMP má 3 fázy. Prvým krokom je nadviazanie spojenia medzi BLE zariadeniami. Nasleduje fáza 1 párovacej procedúry SMP kde si zariadenia vymenia informácie a svoje možnosti, na základe ktorých sa zvolí párovacia metóda. Nasleduje fáza 2 kde prebieha samotné párovanie podľa zvolenej metódy či už pre Legacy pairing alebo Secure Connections. Ďalším krokom je nadviazanie zabezpečeného spojenia pomocou kľúča vygenerovaného vo fáze 2. V poslednej fáze sa cez zabezpečené spojenie distribuujú kľúče na šifrovanie pri ďalšom pripojení.



Obr. 6.1: Proces párovania BLE

Keďže k centrálnemu zariadeniu môže byť pripojených naraz niekoľko periférnych zariadení, musí si detektor v pamäti uchovávať zoznam jednotlivých spojení a ich stav. Detektor potom pracuje ako stavový automat. Drží si spojenie v pamäti až kým nie je ukončené a v prípade, že spracuje paket, ktorý sa týka párovacieho procesu prejde do ďalšieho stavu pre dané spojenie. Po skončení procesu párovania vygeneruje hlásenie na výstup.

6.3 Detekcia skenovania Z-Wave siete

Podobne ako analýza sieťovej komunikácie Z-Wave je aj zber sieťovej prevádzky pre detekčný systém realizovaný pomocou SDR. Slúži na to bude sonda, ktorá bude odchytávať komunikáciu v sieti a poskytovať dáta na analýzu. Zodpovedný za to bude samostatný modul na odchytávanie sieťovej prevádzky Z-Wave. Monitorované budú všetky prenosové kanály.

Princíp skenovania siete a triedy príkazov, ktoré sú na to využívané už bolo popísané v sekcii 3.2.6. Mohlo by sa zdať, že stačí odchytávať komunikáciu a hľadať v nej rámce príslušných tried príkazov. Tieto rámce sa však vyskytujú aj v bežnej komunikácii kontroléra so zariadeniami. Je preto potrebné odlíšiť kedy sa jedná o skenovanie siete útočníkom a kedy sa jedná o bežnú komunikáciu. Analýzou zachytenej bežnej komunikácie testovaných zariadení v sieti som zistil, že tieto správy sa typicky vyskytujú len pri párovaní nového zariadenia s kontrolérom.

Odchytávanie komunikácie pomocou SDR nie je sto percentné a odchytávanie na kanále 1 je najmenej úspešné. Tam prebieha vyslanie NodeInfo rámca a následné priradenie NodeID pri párovaní. Z tohto dôvodu sa nemôžeme spoľahnúť, že detektor zachytí v komunikácii párovanie a tak bude vedieť odlíšiť, že sa nejedná o skenovanie siete útočníkom. Je potrebný iný spôsob zistenia, že sa v danom čase spárovalo nové zariadenie.

Túto informáciu je možné získať z BeeeOn brány, kde je už podpora zachytávania notifikácií z kontroléra pomocou OpenZwave knižnice. NEMEA kolektor na bráne je rozšírený

o nové rozhranie, cez ktoré sa budú posielať notifikácie okrem iného o pridaní a odobratí zariadenia do resp. zo siete spolu s časovou značkou, HomeID a NodeID.

Navrhovaný detektor má 2 vstupy, ktorými bude prijímať dáta v reálnom čase. Prvým vstupom prúdia odchytené rámce zo siete a druhým notifikácie z BeeeOn brány. Keďže sa nejedná o jeden dátový tok je potrebná určitá časová synchronizácia. Časovú značku prijatého rámca je možné zapísať až po jeho získaní pomocou rtl-zwave, čo má svoju réžiu. Otestovaním som zistil, že vygenerovaná notifikácia o spárovaní má timestamp priemerne o sekundu menší ako odchytené rámce získavania informácií o novom uzle. Notifikácia môže byť však opozdená a rámec prijatý skôr.

Detektor by si mohol rámce ukladať, spracovávať ich až po určitej dobe a tak hlásiť alerty. Toto riešenie však nepovažujem za vhodné, pretože by sa zbytočne plnila pamäť, ktorá môže byť pri nasadení na domácom smerovači obmedzená. Lepšie riešenie sa mi javí ukladať si pre každý uzol podozrenie zo skenovania s časovou značkou kedy vzniklo. Hlásenia o útokoch potom generovať pravidelne v určitých intervaloch po skontrolovaní, či daný uzol v tom intervale bol alebo nebol spárovaný. V tomto intervale sa teda po prijatí prvého podozrivého rámca (jeho Command a Command Class sa používa na skenovanie) poznačí pre zainteresovaný uzol (pre príkazy GET je to cieľový a pre príkazy REPORT zdrojový uzol), že je podozrivý zo skenovania a uloží sa časová značka spolu použitou triedou príkazov. Pre ďalšie podozrivé rámce k tomu istému uzlu sa potom ukladajú, ktoré ďalšie triedy príkazov boli použité. Pri generovaní alertov v pravidelných intervaloch sa prejdú všetky podozrivé uzly, a tie ktoré boli spárované v intervale sa vylúčia. V alerte sa spolu s identifikátorom skenovaného uzla posiela časová značka začiatku skenovania a triedy príkazov, ktoré boli použité.

6.4 Detekcia útokov na smerovanie v Z-Wave sieti

Princíp útokov na smerovanie bol detailne popísaný v sekcii 3.2.6. Všetky útoky majú spoločný menovateľ a to neznámy uzol pokúšajúci sa preniknúť do smerovania siete. Dôležitou podmienkou pre odhalenie týchto útokov je teda potreba poznania, ktoré zariadenia do siete patria a ktoré nie. Spôsob získania tejto informácie je obdobný ako pri detekcii skenovania siete. Z tohto dôvodu je vhodné, aby bola detekcia útokov na smerovanie a skenovanie siete realizovaná jedným spoločným detektorom. Detektor bude reportovať niekoľko rôznych alertov podľa typu útoku.

Modifikácia zoznamu susedov – Pri modifikácií NL posiela útočník uzlu správu Do NL Test, kde payloadom správy je NL primitív. Bity nastavené na 1 označujú, ktoré uzly má napadnutý uzol otestovať pomocou správy NL Test. Útočník nastaví bit svojho falošného NodeID a uzol mu potom posiela správu NL Test kde ho uvedie ako cieľové NodeID. Po otestovaní všetkých zadaných uzlov môže poslať správu Report NL, kde v payloade tvoreného NL primitívom, bude nastavený bit pre falošné NodeID.

Detektor podobne ako pre skenovanie siete generuje alerty v určitom intervale. Pri spracovaní prijatého rámca Do NL test skontroluje, či sú všetky uzly v payloade spárované. Ak niektorý nie je, poznačí si k cieľovému NodeID potenciálny útok modifikácie NL a uloží si nespárované uzly. Pre rámec typu Report NL pracuje rovnako s rozdielom, že si potenciálny útok značí k zdrojovúmu NodeID. Pri spracovaní NL test si poznačí pre zdrojové NodeID nespárované cieľové NodeID. Pri genenerovaní alertu ešte skontroluje, či zaznamenané uzly neboli spárované v intervale generovania alertu a ak boli alert nie je reportovaný.

Modifikácia SR cache – Pre dosiahnutie modifikácie SR cache posiela útočník napadnutému uzlu správu SR Cache Assignment alebo Backbone Cache Assignment, kde v payloade správy tvorenej záznamom SR cache, vloží ako hop trasy falošné NodeID. Detektor si teda pri spracovaní prijatého rámca skontroluje, či sú všetky hopy trasy spárované. Ak niektoré nie sú, poznačí si k cieľovému NodeID potenciálny útok modifikácie SR cache a zaznamená si nespárované uzly. Pri generovaní alertov v intervale, skontroluje či neboli uzly spárované a ak neboli reportuje alert.

MITM uzol v smerovacej ceste – Po využití modifikácie SR cache môže útočník presmerovať smerovanú komunikáciu cez svoj falošný uzol a vystupovať v komunikácie ako MITM. Môže modifikovať rámce tak ako bolo popísané v sekcii 3.2.6 alebo ich zahadzovať. Samotná detekcia útoku black hole nie je možná z dôvodu, že pomocou SDR nie sú odchytené všetky rámce komunikácie v sieti. Bolo by potrebné sledovať a zaznamenávať si každý smerovaný rámec cez MITM a v momente keď by nebola detekovaná očakávaná nasledujúca smerovaná správa vygeneroval by sa alert. Tieto očakávané rámce však nemusia byť pomocou SDR odchytené a tak by sa generovalo obrovské množstvo falošných poplachov

Môžme však všeobecne detekovať MITM uzol v komunikácií. Detektor po prijatí smerovaného rámca skontroluje, či sú spárované všetky hopy smerovacej cesty v sieťovej hlavičke. V prípade, že niektorý nie je, poznačí si k záznamu príslušného uzlu, že je to potenciálny MITM uzol a ukladá si smerovaciu cestu, v ktorej bol objavený. Pri generovaní alertov skontroluje, či uzol v intervale nebol spárovaný a ak nebol, generuje alert spolu so všetkými smerovacími cestami, v ktorých bol nájdený.

Útoky neodhaliteľné navrhnutým spôsobom – Útočník môže vykonávať záškodnícku činnosť aj iným spôsobom. Nemusí sa pokúšať preniknúť do smerovania ako neznámy uzol, ale môže kaziť smerovanie zasielaním koordinačných rámcov pre manažment smerovania kde bude využívať len spárované uzly v sieti. Môže napríklad zasielať uzlom správy SR Cache a Backbone Cache Assignment s nesprávnymi smerovacími cestami. Taktiež môže preskúmať sieť pomocou algoritmu Outsider topology discovery popísaného v [2]. Spomínanú činnosť je možné detekovať zo štatistík o počte odchytení takýchto rámcov v sieti a odchylky od bežnej prevádzky. Pre tieto účely a mnoho ďalších bude existujúci detektor WSN Anomaly rozšírený o ďalšie štatistiky priamo zo Z-Wave siete. Spôsob rozšírenia je popísaný v ďalšej časti.

6.5 Štatistiky zo Z-Wave siete

Existujúci detektor WSN Anomaly nad štatistickými dátami získanými pomocou OpenZwave knižnice z kontroléra BeeeOn brány je možné rozšíriť o podrobnejšie štatistiky získané priamo zo siete Z-Wave pomocou SDR. Výhodou týchto štatistík je, že budú vytvárané z celkového pohľadu na sieť a nie z pohľadu kontroléra. Tiež je možné generovať štatistiky, ktoré kontrolér neposkytuje.

Navrhovaným riešením je vytvorenie NEMEA modulu schopného zo zachytenej komunikácie v zadaných intervaloch generovať štatistiky pre sieť a taktiež pre každý uzol v sieti. Štatistiky budú posielané ďalej do detektora WSN Anomaly. Existuje obrovské množstvo rôznych štatistík, ktoré je možné generovať a preto je vhodné stanoviť scenáre útokov a možných anomálií, ktoré z týchto štatistík bude možné určiť. Detektor anomálií nad štatistickými dátami bude potrebné upraviť a to rozšírením zoznamu položiek, ktoré podporuje.

6.5.1 Scenáre anomálií

V tejto sekcii sú popísané navrhnuté možné scenáre anomálii na základe poznatkov získaných teoretickou analýzou Z-Wave protokolu a analýzou zachytenej bežnej komunikácie v sieti. Scenáre detekcie anomálií môžme rozdeliť podľa toho, či sú postavené nad štatistikami pre Z-Wave sieť alebo nad konkrétnym uzlom.

Kvalita komunikačných kanálov – Hlavne u bezdrôtových komunikačných protokolov je veľmi dôležitá kvalita komunikačného kanálu, ktorý môže byť útočníkom zarušený. V tomto prípade sa bude očakávať nárast počtu poškodených správ, či už kvôli zlej veľkosti prijatých rámcov alebo nesprávneho kontrolného súčtu. Sledovať bude možné tieto štatistiky aj pre jednotlivé komunikačné kanály, na ktorých boli rámce odchytené.

Kvalita smerovania – Zhoršenie kvality smerovania v sieti môže znamenať, že útočník napadol sieť napríklad vytvorením a vnútením vadných smerovacích ciest jednotlivým uzlom. Môžeme ho detektovať na základe zvýšenia počtu smerovaných NACK správ oproti bežnej prevádzke.

Pre jednotlivé uzly, môžeme sledovať počet NACK správ kde sa uzol vyskytuje ako cieľové NodeID, čo znamená, že je problém v smerovaní k tomuto uzlu alebo naopak keď sa vyskytuje ako zdrojové NodeID. Taktiež môžme sledovať počet správ kde uzol vystupuje ako failed hop v NACK správe a smerovanie zlyháva práve na tomto uzle.

Skenovanie a zásahy do topológie siete – Ako už bolo spomenuté pri popise detekcie útokov na smerovanie, útočník sa nemusí pokúšať votrieť do smerovania ako neznámy uzol, ale môže ho napadnúť zasielaním koordinačných správ pre manažment smerovania v sieti tak, že bude využívať len spárované uzly.

Môžeme teda generovať štatistiky o počte týchto správ v sieti a sledovať nadmerné zýšenie počtu takýchto správ, čím je možné odhaliť neoprávnené zásahy do topológie siete.

Periodicita komunikácie od sieťových prvkov – Senzorické prvky typicky zasielajú namerané hodnoty periodicky v určitých časových intervaloch. Takto fungujú všetky batériou napájané zariadenia z dôvodu šetrenia energie.

Pre jednotlivé prvky sa teda bude sledovať timestamp poslednej správy kde uzol vystupuje ako zdrojové NodeID a počet správ od tohto uzlu. V prípade nárastu počtu správ a menšieho rozdielu v timestampe je možné odhaliť podvrhnutie komunikácie z tohto uzlu útočníkom, prípadne vadný/napadnutý uzol. V opačnom prípade kedy počet správ poklesne prípadne prestanú chodiť úplne správy od tohto uzlu a rozdiel v timestampe poslednej správy sa neúmerne zväčší môže mať senzor vybitú batériu, mohol stratiť konektivitu alebo byť odcudzený.

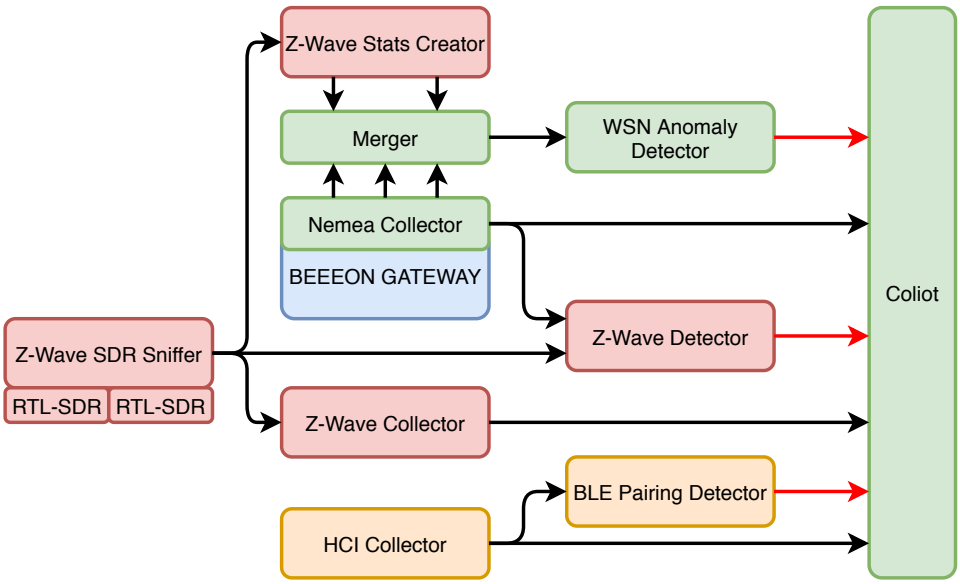
Množstvo prenášaných správ v sieti – Posledný scenárom je neobvyklý nárast správ v sieti, čo môže znamenať DoS (Denial of Service) útok. Sledovať sa bude celkový počet správ v sieti a pre každý uzol počet správ kde uzol vystupuje ako zdrojové NodeID.

6.6 Architektúra detekčného systému a jednotlivé moduly

V tejto časti práce je popísaná navrhnutá architektúra a jednotlivé moduly detekčného systému. Navrhovaným riešením je rozšírenie zabezpečenej brány SIoT o niekoľko NEMEA modulov. Na obrázku 6.2 je zobrazená architektúra jednotlivých modulov a ich prepojení. Všetky moduly sú postavené na NEMEA frameworku, ktorý medzi nimi zaisťuje komunikáciu. Moduly sú medzi sebou prepojené pomocou UniRec rozhraní a je ich teda možné nasadiť na rôznych fyzických strojoch. Taktiež je možné nasadiť len niektoré moduly podľa potreby.

Zelenou farbou sú vyznačené existujúce moduly. Oranžovou farbou sú vyznačené vytvorené BLE moduly. Jedná sa o HCI Collector, ktorý zbiera komunikáciu z HCI rozhrania brány a jednotlivé pakety posiela na výstup. Na tento výstup je napojený Ble Pairing Detector, ktorý v paketoch hľadá párovací proces a hlási alerty na výstup. Vytvorené Z-Wave moduly sú označené červenou farbou. Zber komunikácie zo siete pomocou SDR zaisťuje modul Z-Wave SDR Sniffer, ktorý poskytuje zachytené rámce v podobe sekvencie bajtov pre všetky ostatné Z-Wave moduly. Z-Wave Collector tieto rámce parsuje na výstup a niektoré položky prevádza do ľudsky čitateľnej formy, kvôli jednoduchšej manuálnej analýze. Z-Wave Detektor okrem rámcov prijíma aj udalosti z BeeeOn brány. Detekuje skenovanie siete a útoky na smerovanie. Detekované incidenty hlási na výstup. Posledným vytvoreným modulom je Z-Wave Stats Creator, ktorý generuje štatistiky zo zachytených rámcov. Tieto štatistiky je možné použitím modulu Merger spolu so štatistikami z modulu Nemea Collector bežiaceho v rámci BeeeOn brány zlúčiť na jedno výstupné rozhranie. Na toto rozhranie sa môže napojiť detektor WSN Anomaly a tak detekovať anomálie nad všetkými počítanými štatistikami.

Zozbierané pakety a rámce je možné logovať a exportovať do systému Coliot kde je nad nimi možné robiť ďalšiu analýzu. Detekované útoky a anomálie sú v podobe alertov (výstražných hlásení) taktiež zasielané do Coliotu alebo je možné si napríklad pomocou NEMEA modulu email-reporter nastaviť zasielanie na email.



Obr. 6.2: Architektúra detekčného systému

6.6.1 HCI kolektor

Úlohou kolektora (HCI Collector) je získavanie BLE paketov z HCI rozhrania brány. Má jedno výstupné rozhranie UniRec, ktorého šablóna je definovaná v-tabuľke 6.1. Na toto výstupné rozhranie sa pripája detektor svojim vstupným rozhraním. Modul po štarte vytvorí Raw HCI Socket na rozhraní, ktorého číslo bolo predané ako parameter pri spustení. Následne periodicky číta pakety, z prvého bytu zistí typ paketu a spolu s ďalšími informáciami ho bez prvého bytu posiela výstupným UniRec rozhraním. Výhodou kolektora je, že získava a posiela všetky pakety aj protokolu Bluetooth Classic a bude ho teda možné použiť aj na iné účely, ako napr. vytváranie štatistík z komunikácie a iné detekcie. Pre účely projektu SIoT a ďalšiu možnú analýzu zisťuje kolektor aj adresu zariadenia pri dátových ACL paketoch, ktorú posiela na výstup. V inom prípade posiela na výstup nulovú adresu.

Dátový typ	**Názov položky**	**Popis**
time	TIMESTAMP	Timestamp odchytenia paketu.
macaddr	DEV_ADDR	Adresa zariadenia pre dátové ACL pakety.
macaddr	HCI_DEV_ADDR	Adresa kontroléra.
uint8	PACKET_TYPE	Typ paketu.
uint8	DATA_DIRECTION	Smer paketu.
size	SIZE	Veľkosť paketu v bajtoch.
bytes	PACKET	Paket v podobe sekvencie bajtov.

Tabuľka 6.1: Šablóna výstupného rozhrania modulu HCI kolektor a vstupného rozhrania detektora párovania BLE

6.6.2 Detektor párovania BLE

Druhým modulom je detektor párovania BLE (BLE Pairing Detector), ktorý v paketoch hľadá párovací proces a generuje alerty. Má jedno vstupné a jedno výstupné rozhranie. Vstupným rozhraním sa pripája na výstup HCI kolektora a na výstupné rozhranie generuje alerty o párovaní. Šablóna výstupného rozhrania je definovaná v tabuľke 6.2. Pre zistenie, že párovanie bolo opakované si musí detektor perzistentne uchovávať zoznam spárovaných zariadení, aby mal túto informáciu aj po reštarte. Vhodným riešením sa javí ukladanie zariadení v podobe prázdnych súborov pomenovaných adresou zariadenia. Pomocou vstupného parametru je možné špecifikovať zložku, do ktorej budú ukladané spárované zariadenia. Detektor zisťuje aj aká verzia a metóda párovania bola použitá a taktiež, či párovanie bolo úspešné.

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp začiatku párovania.
macaddr	INCIDENT_DEV_ADDR	Adresa párovaného zariadenia.
uint32	ALERT_CODE	Kód hlásenia: prvé párovanie (0), opakované párovanie (1).
string	CAPTION	Popis.
macaddr	HCI_DEV_ADDR	Adresa kontroléra.
uint8	SUCCESS	Párovanie dopadlo úspešne: áno (1), nie (0).
uint8	VERSION	Verzia párovania BLE: Legacy Pairing (0), Secure Connection (1), neznáme (255).
uint8	METHOD	Párovacia metóda: Just Works (0), Passkey Entry (1), OOB (2), Numeric Comparison (3), neznáme (255).

Tabuľka 6.2: Šablóna výstupného rozhrania detektora párovania BLE

6.6.3 Modul pre odchytávanie Z-Wave rámcov

Modul pre odchytávanie Z-Wave rámcov (Z-Wave SDR Sniffer) využíva rtl-sdr na prijímanie signálu. Na získanie rámcov v podobe sekvencie bajtov je použitý nástroj rtl-zwave. Modul využíva 2 inštancie rtl-sdr a rtl-zwave pre simultánne odchytávanie všetkých kanálov. Prvá inštancia odchytáva kanály 1 a 2 na frekvencii 868.4 MHz a druhá kanál 3 na frekvencii 869.85 MHz. Úlohou modulu je teda odchytávať rámce pomocou SDR a posielať ich na výstupné UniRec rozhranie, ktorého šablóna je definovaná v tabuľke 6.3. Na toto rozhranie sa pripájajú všetky ostatné Z-Wave moduly. Na výstup sa posielajú všetky rámce (aj poškodené) kvôli ďalšej analýze napr. pri vytváraní štatistík. Pomocou vstupných parametrov je možné špecifikovať gain (dB) prijímania rtl-sdr.

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp odchytenia rámca.
uint8	CHANNEL	Kanál, na ktorom bol odchytený rámec.
bytes	FRAME	Rámec v podobe sekvencie bajtov.

Tabuľka 6.3: Šablóna výstupného rozhrania modulu pre odchytávanie Z-Wave rámcov, vstupného rozhrania Z-Wave kolektora, prvého vstupného rozhrania pre príjem rámcov Z-Wave detektora a vstupného rozhrania modulu pre vytváranie štatistík zo Z-Wave siete

6.6.4 Z-Wave kolektor

Ďalším modulom je Z-Wave kolektor (Z-Wave Collector). Úlohou kolektora je prezentácia rámcov na analýzu. Kolektor prevádza niektoré položky rámca akými sú napr. bitové polia, typ rámca, triedu príkazov, príkazy System CC, skoky cesty v smerovanej správe atď. do ľudsky čitateľnej formy a takto ich posiela na výstup do Coliotu. Kolektor má jedno vstupné a jedno výstupné rozhranie. Vstupným rozhraním prijíma rámce v podobe sekvencie bajtov z modulu pre odchytávanie Z-Wave rámcov. Tie spracuje a na výstupné rozhranie posiela jednotlivé položky rámca v ľudsky čitateľnej forme. Šablóna výstupného rozhrania je zobrazená v tabuľke 6.4.

Dátový typ	**Názov položky**	**Popis**
time	TIMESTAMP	Timestamp odchytenia rámca.
uint64	DEV_ADDR	Zdrojové NodeID.
uint8	CHANNEL	Kanál, na ktorom bol odchytený rámec.
uint32	HOME_ID	HomeId siete.
uint8	DST_ID	Cieľové NodeID.
uint8	SIZE	Veľkosť rámca v bajtoch.
string	TYPE	Typ rámca.
uint8	ACK_REQ	Označuje, či je vyžadované potvrdenie rámca.
uint8	SEQ_NUM	Sekvenčné číslo rámca.
uint8	ROUTED	Označuje, či má rámec sieťovú hlavičku.
string	ROUTED_TYPE	Typ smerovaného rámca.
uint8	SRC_HOP	Aktuálne zdrojové node id pri smerovaní.
uint8	DST_HOP	Aktuálne cieľové node id pri smerovaní.
uint8	FAILED_HOP	Hop na ktorom zlyhalo smerovanie.
bytes	HOPS	Skoky cesty.
bytes	PAYLOAD	Aplikačný payload v bajtoch.
string	CMD_CLASS_STR	Trieda príkazov konvertovaná na string.
string	CMD_STR	System CC príkaz konvertovaný na string.

Tabuľka 6.4: Šablóna výstupného rozhrania modulu Z-Wave kolektor

6.6.5 Z-Wave detektor

Úlohou detektora je odhalenie skenovania siete a útokov na smerovanie. Princíp jednotlivých detekcií už bol popísaný v tejto kapitole. Detektor má 2 vstupné a jedno výstupné rozhranie. Prvým vstupným rozhraním prijíma rámce z modulu pre odchytávanie Z-Wave rámcov a druhým udalosti z BeeeOn brány potrebné na inicializáciu detektora s HomeID siete a zistenie spárovania nových zariadení. Šablóna rámca pre príjem eventov je zobrazená v tabuľke 6.5. Rámce sú postupne analyzované. V prípade detekcie niektorého útoku sa ukladá podozrenie o útoku spolu s potrebnými informáciami o ňom. V intervaloch, ktoré je možné špecifikovať parametrom pri spustení sú v prípade potvrdenia útoku generované alerty. Tie sú zasielané výstupným rozhraním, ktorého šablóna je popísaná v tabuľke 6.6. Pomocou vstupných parametrov je možné ďalej špecifikovať veľkosť časového okna na synchronizáciu medzi prijatými rámcami a eventami z brány, a veľkosť časového okna kedy sa nemajú hlásiť alerty o skenovaní siete v dobe párovania zariadenia, ktorého sa to týka.

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp vygenerovania eventu.
double	EVENT_TYPE	Typ eventu.
double	HOME_ID	HomeID siete.
double	NODE_ID	NodeID uzla.

Tabuľka 6.5: Šablóna druhého vstupného rozhrania pre príjem OpenZwave udalostí z BeeeOn brány modulu Z-Wave detektor

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp vzniku útoku.
uint64	INCIDENT_DEV_ADDR	Napadnutý uzol alebo MITM uzol.
uint32	ALERT_CODE	Kód hlásenia útoku: skenovanie siete (1), modifikácia NL (2), modifikácia SR Cache (3), MITM uzol (4).
string	CAPTION	Popis: skenovanie siete (použité CC), modifikácia NL (neznáme uzly), modifikácia SR Cache (neznáme uzly), MITM uzol (smerovacie cesty).

Tabuľka 6.6: Šablóna výstupného rozhrania modulu Z-Wave detektor

6.6.6 Modul pre vytváranie štatistík zo Z-Wave siete

Modul pre vytváranie štatistík zo Z-Wave siete (Z-Wave Stats Creator) prijíma rámce z modulu pre odchytávanie Z-Wave rámcov a ukladá si z nich potrebné štatistiky. V zadanom intervale reportuje štatistiky na výstup. Má jedno vstupné rozhranie, ktorým prijíma rámce a 2 výstupné rozhrania, cez ktoré posiela štatistiky pre celú sieť alebo konkrétne uzly spolu s časovou značkou. Pomocou vstupných parametrov je možné špecifikovať HomeID siete, ktorej rámce má spracovávať a interval reportovania štatistík. V tabuľke 6.7 je zobrazená šablóna výstupného rozhrania pre štatistiky o sieti a v tabuľke 6.8 šablóna výstupného rozhrania pre štatistiky o konkrétnych uzloch. Všetky jednotlivé štatistiky sa ukladajú a posielajú ako typ double z dôvodu kompatibility s existujúcim detektorom WSN Anomaly.

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp vygenerovania štatistík.
uint64	DEV_ADDR	Home ID siete.
double	CORRUPTED_C	Celkový počet poškodených rámcov.
double	CORRUPTED_CH1_C	Počet poškodených rámcov na kanáli 1.
double	CORUPPTED_CH2_C	Počet poškodených rámcov na kanáli 2.
double	CORRUPTED_CH3_C	Počet poškodených rámcov na kanáli 3.
double	TOTAL_OK_C	Celkový počet validných rámcov.
double	ROUTED_C	Celkový počet smerovaných rámcov.
double	ROUTED_ACK_C	Počet smerovaných ack rámcov.
double	ROUTED_NACK_C	Počet smerovaných nack rámcov.
double	ROUTED_APP_C	Počet smerovaných aplikačných rámcov.
double	SINGLECAST_C	Počet singlecast rámcov.
double	ACK_C	Počet ack rámcov.
double	MULTICAST_C	Počet multicast rámcov.
double	BROADCAST_C	Počet broadcast rámcov.
double	NET_MANAG_C	Celkový počet správ na správu siete.

Tabuľka 6.7: Šablóna prvého výstupného rozhrania pre štatistiky o sieti Modulu pre vytváranie štatistík zo Z-Wave siete

Dátový typ	Názov položky	Popis
time	TIMESTAMP	Timestamp vygenerovania štatistík.
uint64	DEV_ADDR	Node ID uzla.
double	SRC_NACK_C	Počet smer. nack rámcov (uzol je zdroj).
double	DST_NACK_C	Počet smer. nack rámcov (uzol je cieľ).
double	F_HOP_NACK_C	Počet smer. nack rámcov (failed hop).
double	SRC_TOTAL_C	Celkový počet rámcov (zdroj).
double	DST_TOTAL_C	Celkový počet rámcov (cieľ).
double	SRC_SINGL_C	Počet singlecast rámcov (zdroj).
double	DST_SINGL_C	Počet singlecast rámcov (cieľ).
double	SRC_ACK_C	Počet ack rámcov (zdroj).
double	DST_ACK_C	Počet ack rámcov (cieľ).
double	SRC_MULTICAST_C	Počet multicast rámcov (zdroj).
double	DST_BROADCAST_C	Počet broadcast rámcov (zdroj).
double	SRC_TOTAL_LM_T	Timestamp poslednej správy (zdroj).
double	DST_TOTAL_LM_T	Timestamp poslednej správy (cieľ).
double	SRC_SINGL_LM_T	Timestamp poslednej singl. správy (zdroj).
double	DST_SINGL_LM_T	Timestamp poslednej singl. správy (cieľ).

Tabuľka 6.8: Šablóna druhého výstupného rozhrania pre štatistiky o konkrétnom uzle Modulu pre vytváranie štatistík zo Z-Wave siete

Kapitola 7

Realizácia detekčného systému

Obsahom kapitoly je popis realizácie najzaujímavejších častí vytvoreného riešenia. Pre zaistenie efektivity a možnosti nasadenia na domácich smerovačoch so systémom OpenWRT, prípadne inej hardvérovej platforme s procesorom ARM boli všetky moduly detekčného systému implementované v jazyku C++. V tomto jazyku je zároveň napísaná aj BeeeOn brána a všetky ostatné moduly zabezpečenej brány SIoT. Kvôli potrebe nasadenia na OpenWRT bol dôraz kladený na použitie čo najmenšieho počtu knižníc. Taktiež bol dôraz kladený na rýchlosť behu jednotlivých modulov. Je potrebné, aby program spotrebúval čo najmenej výpočetných prostriedkov.

Všetky moduly využívajú NEMEA framework na komunikáciu a štandardné knižnice jazyka C a C++. HCI Collector a BLE Pairing Detector využívajú knižnicu BlueZ[1] pre získavanie sieťového toku z HCI a analýzu jednotlivých paketov. Pri Z-Wave nie je využívaná žiadna externá knižnica, ale Z-Wave SDR Sniffer vyžaduje nainštalovaný nástroj rtl-sdr[2].

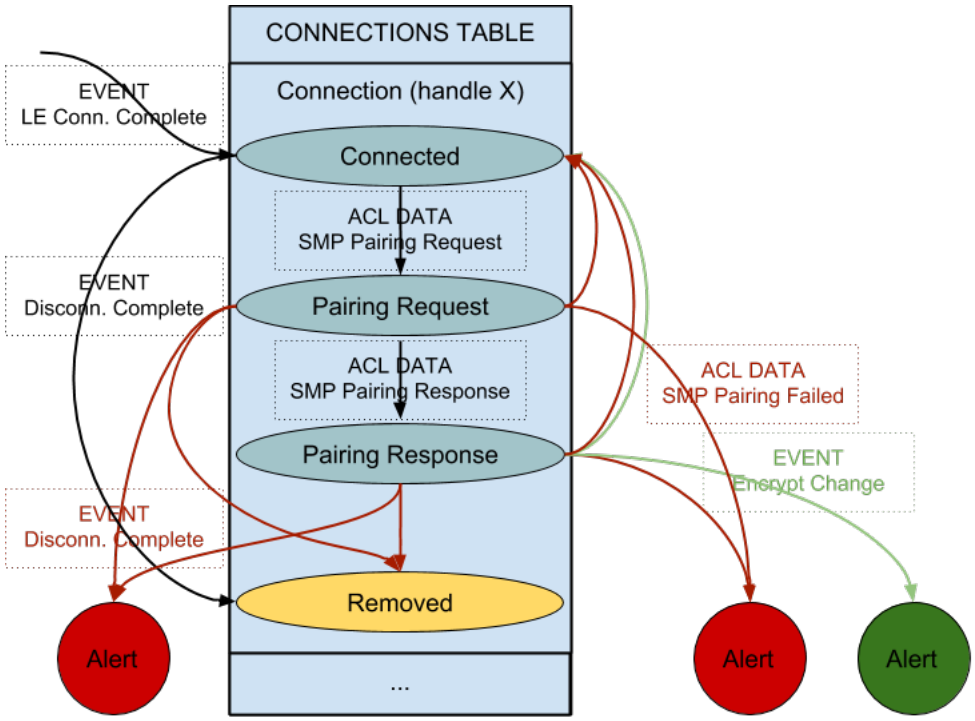
Celý vytvorený detekčný systém je umiestnený v repozitári NEMEA-SIoT[3]. Každý modul sa nachádza vo svojej zložke a obsahuje zdrojové kódy, Makefile súbory a README s popisom daného modulu. Popísaná je úloha modulu, jeho závislosti, vstupné parametre a šablóny UniRec rozhraní. Z-Wave moduly zdieľajú kód pre spracovanie rámcov v zložke zwave. Spoločný preklad je zaistený pomocou vytvorených Makefile súborov a taktiež je možné preložiť len vybrané moduly vo svojich zložkách.

OpenWRT balíky pre Turris-Omnia všetkých modulov detekčného systému spolu s BeeeOn bránou a jej závislosťami je možné nájsť v repozitári Nemea-OpenWRT[4]. Balíky sú vytvárané automaticky pomocou nástroja Travis CI [5] zo zdrojových kódov master vetvy repozitára NEMEA-SIoT. Pre vytvorenie nových balíkov je potrebné aktualizovať Makefile súbor v devel vetve repozitára Nemea-OpenWRT[6]. Konkrétne je potrebné aktualizovať hash posledného commitu master vetvy repozitára NEMEA-SIoT, z ktorého sa majú balíky vytvoriť.

[1] http://www.bluez.org/
[2] https://osmocom.org/projects/rtl-sdr/wiki
[3] https://github.com/CESNET/NEMEA-SIoT
[4] https://github.com/CESNET/Nemea-OpenWRT/tree/turris-packages-devel/turris-omnia/packages/siot
[5] https://travis-ci.org/
[6] https://github.com/CESNET/Nemea-OpenWRT/blob/devel/net/nemea-siot/Makefile

7.1 Detekcia párovacieho procesu BLE

Detektor BLE párovania obsahuje tabuľku aktívnych spojení s ich stavom a ďalšími informáciami ako napr. zvolená párovacia metóda atď. Samotný algoritmus detekcie je zobrazený na obrázku 7.1 a popísaný ďalej. Je potrebné si uvedomiť, že jednotlivé kroky nenasledujú priamo za sebou, ale sú spracovávané pakety pre rôzne spojenia, dôležité sú len niektoré a ostatné sú zahadzované. Spracovávané sú teda len eventy (o pripojení, odpojení a začatí šifrovania) a dátové správy s SMP protokolom (párovacia požiadavka, párovacia odpoveď a zlyhanie párovania).



Obr. 7.1: Detekcia párovacieho procesu

Postupne sa čítajú typy paketov a zahadzujú sa až kým sa nenarazí na event LE Meta (LE Connection Complete). Z tohto paketu sa vyextrahuje položka BD_ADRR predstavujúca adresu pripojeného zariadenia a položka Connection Handle identifikujúca toto spojenie. Do tabuľky pripojených zariadení sa vloží nový záznam so stavom Connected. Vždy keď sa odchytí event Disconnect Complete ja záznam vymazaný z tabuľky pripojených zariadení, ale musí sa skontrolovať stav zariadenia, pretože ak bolo započaté párovanie je potrebné generovať alert o neúspešnom párovaní.

Ďalším paketom, ktorý je potrebný pre detekciu je dátový paket nesúci SMP protokol, konkrétne správu Pairing Request, po ktorej prijatí si aktualizujeme stav spojenia v tabuľke na započaté párovanie. Taktiež si ku spojeniu musíme uložiť párovacie informácie, na základe ktorých je po prijatí odpovede odvodená párovacia metóda. Nasleduje správa Pairing Reponse a výmena párovacích informácií, na základe ktorých sa určí párovacia metóda. Ďalším paketom, ktorý nás zaujíma je SMP paket Pairing Failed, po ktorom sa generuje alert o neúspešnom párovaní a aktualizuje sa stav spojenia na Connected, pretože spojenie nemusí byť ukončené a proces sa môže opakovať. Okrem neúspešného párovania očakávame tiež úspešné párovanie v podaní eventu Encryption Change a na základe jeho statusu ge-

nerujeme alert o úspešnom alebo neúspešnom párovaní. V prípade úspešného párovania si musíme perzistentne uložiť adresu spárovaného zariadenia a taktiež vždy pri generovaní alertu vyhľadať, či zariadenie už bolo spárované. Ukladanie spárovaných zariadení je realizované vytvorením prázdneho súboru pomenovaného podľa adresy zariadenia v zložke definovanej pomocou vstupného parametra detektora.

7.2 Integrácia nástroja rtl-zwave

Dôvody prečo som pre odchytávanie a dekódovanie Z-Wave rámcov zo signálu vybral rtl-sdr v kombinácií s rtl-zwave [5] som už popísal v 5.2. Hlavným dôvodom bola schopnosť dekódovania rámcov všetkých prenosových kanálov a možnosť jednoduchej integrácie do projektu bez ďalších závislostí. Nástroj rtl-zwave pri testovaní dokonca dosahoval aj najlepšie výsledky. Nástroj pracuje tak, že sa spustí rtl-sdr a jeho výstup sa presmeruje na štandardný vstup rtl-zwave. Ten dokáže dekódovať rámce do podoby sekvencie bajtov a posielať ich na výstup.

Vytvoril som triedu ZWaveRtlSdr, ktorá obaľuje túto funkcionalitu. Pri inštanciácii triedy sa pomocou volania popen spustí rtl-sdr so zadanými parametrami predanými do konštruktora a do členskej premennej sa uloží ukazovateľ na štandardný výstup tohto podprocesu. Ukazovateľ sa ukladá ako std::unique_ptr a pri deštrukcii inštancie sa teda automaticky uzavrie. Špecifikovať je možné frekvenciu, na ktorej sa bude odchytávať komunikácia, index pripojeného SDR donglu a gain (dB).

Trieda obsahuje niekoľko privátnych metód reprezentujúcich demodulátor a filtre rtl-zwave a jedinú verejnú metódu run, ktorá predstavuje celú logiku dekódovania Z-Wave rámca zo signálu získaného pomocou rtl-sdr. Po zavolaní metódy run sa v cykle číta z uloženého štandardného výstupu podprocesu rtl-sdr a s využitím filtrov, demodulátora a stavového automatu je získaný rámec. Rámec je získaný ako sekvencia bajtov s jeho veľkosťou a značkou určujúcou, či bolo použité Manchester kódovanie, ktoré je použité na kanále 1.

V momente získania rámca je zavolaná std::function, ktorá bola metóde predaná ako vstupný parameter. Funkcia sa zavolá s parametrami ukazovateľa na začiatok rámca, veľkosťou a boolean hodnotou, označujúcou či bolo použité kódovanie Manchester na základe čoho je odlíšené prijatie rámca na kanáloch 1 a 2.

Pre získavanie rámcov zo všetkých prenosových kanálov sú v module Z-Wave SDR Sniffer použité 2 inštancie triedy ZWaveRtlSdr. Prvá pre odchytávanie rámcov na frekvencii 868,4 MHz a teda kanálu 1 a 2 a druhá pre frekvenciu 869,85 MHz a kanál 3. Získavanie rámcov prebieha v 2 vláknach, ktoré synchronizovane posielajú získané rámce výstupným UniRec rozhraním.

7.3 Spracovanie Z-Wave rámcov

Z-Wave rámce je potrebné spracovávať vo viacerých moduloch detekčného systému a tak bolo potrebné vytvoriť vhodný spôsob spracovania tak, aby sa neduplikoval kód a získanie potrebných položiek rámca bolo čo najjednoduchšie. Samotné parsovanie jednotlivých položiek transportnej hlavičky funguje jednoduchým namapovaním sekvencie bajtov rámca, v podobe ukazovateľa na pamäť, na štruktúru s odpovedajúcimi položkami. Aplikačný payload rámca by mal nasledovať po transportnej hlavičke. V rámci môže byť však prítomná sieťová hlavička, ktorá má premenlivú veľkosť. Nie je teda možné získanie všetkých potrebných položiek rámca prostým namapovaním štruktúry na pamäť, ale je k tomu potrebná

ďalšia logika závislá na typoch rámca, prítomnosti sieťovej hlavičky a type aplikačného payloadu. Odchytené rámce môžu byť prijaté čiastočne alebo byť poškodené a tak je potrebné pre každý rámec overiť jeho veľkosť a kontrolný súčet. Spôsob počítania a veľkosť kontrolného súčtu v rámci sa navyše líši podľa prenosového kanála, z ktorého rámec pochádza.

Z tohto dôvodu som vytvoril triedu FrameWrapper, ktorá poskytuje metódy na získanie potrebných informácií z rámca. Inštancia triedy sa vytvára nad pamäťou, kde je umiestnený rámec. Predáva sa ukazovateľ na začiatok rámca, veľkosť pamäte a kanál, na ktorom bol rámec odchytený z dôvodu rôznorodosti počítania kontrolného súčtu. Trieda neslúži na uloženie rámca. Je to len obálka nad ním, poskytujúca metódy na overenie jeho validity a jednotné získanie potrebných položiek z pamäte, kde je rámec reálne uložený v jednotlivých moduloch detekčného systému. Pri spracovávaní prijatých rámcov zo vstupného rozhrania konrétneho NEMEA modulu sa teda zbytočne nekopíruje pamäť.

Rámce sa neparsujú priamo v module Z-Wave SDR Sniffer, pretože napr. pre počítanie štatistík sa využívajú aj poškodené rámce a tie nie je možné parsovať. Z dôvodu voliteľnej sieťovej hlavičky a rôznych typov rámcov by bolo navyše parsovanie na výstup zbytočne komplikované a neprinieslo by veľa výhod. Stále by pre prístup k jednotlivých položkám bolo potrebné overovať iné položky a zbytočne by sa posielali prázdne alebo nulové položky ak by nebola prítomná sieťová hlavička, atď.

```
Channel 3, frame corrupted: bad size
e1 94 22 49 02 41 05 0f

Channel 2, frame corrupted: bad checksum
e1 94 22 49 01 05 03 16 ff 20 00 fa 40 00 00 00 00 04 88 04 02 dc 00

Channel 2, frame OK
e1 94 22 49 04 81 03 14 01 10 20 02 03 31 05 03 0a 00 03 7d 00
============================================================
HomeID: e1 94 22 49
Source NodeID: 4
Destination NodeID: 1
------------------------------------------------------------
Header Type: 1
Speed: 0
Low Power: 0
Ack/Req: 0
Routed: 1
Sequence Number: 3
Beam Control: 0
------------------------------------------------------------
NETWORK HEADER:
SR Type: 0
Failed Hop: 1
SR Length: 2
Hop Index: 0
Hops: 2 3
------------------------------------------------------------
PAYLOAD:  31 05 03 0a 00 03
CommandClass: Sensor Multilevel
Command: 5
============================================================
```

Obr. 7.2: Príklad výpisu Z-Wave rámcov pomocou triedy FrameWrapper

Trieda poskytuje metódy na overenie validity rámca, získanie všetkých potrebných položiek rámca a prevod niektorých položiek do human-readable formy kvôli analýze. Pre

analýzu umožňuje aj jednoduchý výpis celého rámca a jeho jednotlivých častí. Na obrázku 7.2 je možné vidieť výpis 2 poškodených a jedného validného rámca.

7.4 Z-Wave detektor

Ako už bolo spomenuté v návrhu tak detektor má 2 vstupy. Jedno na prijímanie zachytených rámcov a druhé na prijímanie udalostí z brány o inicializácii kontroléra a spárovaní nových zariadení. Brána pri inicializácií kontroléra vždy pošle všetky spárované zariadenia, takže si ich nie je potrebné perzistentne ukladať kvôli reštartu detekčného systému. Z dôvodu dvoch vstupov nie je teda možné zasielať hlásenie o detekovanom útoku hneď pri spracovaní daného rámca. Je potrebné počkať, či nepríde v povolenom časovom limite udalosť o spárovaní. Generovanie hlásenia o útoku pre každý rámec by ani nebolo vhodné, pretože z povahy komunikácie sa môžu správy zasielať viac krát po sebe alebo napr. pri smerovaní bude zachytených niekoľko správ týkajúcich sa jednej požiadavky. Hlásenia o útokoch sa preto generujú v definovaných intervaloch.

Vstupnými parametrami detektora je možné špecifikovať interval, v ktorom sú generované alerty o detekovaných útokoch (východzia hodnota je 10 sekúnd). Ďalej je možné špecifikovať dobu na synchronizáciu medzi jednotlivými vstupmi. Inak povedané dobu, po ktorú nebude potenciálny útok vyhodnotený, aby sa počkalo na prípadnú oneskorenú udalosť o spárovaní uzlu, na základe ktorého bol potenciálny útok poznačený (východzia hodnota sú 3 sekundy). Posledným parametrom je doba, ktorá musí uplynúť od času spárovania nejakého uzlu, aby bol vyhodnotený potenciálny útok skenovania siete za reálny a nešlo o bežnú komunikáciu pri párovaní uzla (východzia hodnota je 5 sekúnd).

Detektor si v pamäti pre každý možný uzol v sieti uchováva určité informácie. Pre každé NodeID si drží v pamäti, či je uzol spárovaný spolu s časovou značkou spárovania a potom informácie o každom podporovanom útoku, ktorý môže detekovať na tento uzol. Pre každý podporovaný útok si uchováva značku o prítomnosti útoku, časovú značku začiatku útoku označujúcu kedy bol odchytený prvý rámec týkajúci sa daného útoku a časovú značku prijatia tohto rámca v detektore, práve kvôli synchronizácii s možnou udalosťou o spárovaní daného uzlu. Okrem týchto spoločných informácií sú pre každý typ útoku uchovávané špecifické informácie. Pre skenovanie siete sú to značky využitia jednotlivých detekovaných správ pri útoku. Pri modifikácii NL a SR Cache sú to nespárované uzly, ktoré obsahovali tieto správy. Pri MITM sú to zas smerovacie cesty, v ktorých bol nájdený neznámy podozrivý MITM uzol.

Detektor spracováva rámce a hľadá v nich útok tak ako bolo popísané v návrhu. V prípade detekovania niektorého potenciálneho útoku pre daný uzol nastaví značku prítomnosti útoku a uloží si potrebné informácie. Detektor spracováva rámce len tej sieti, ktorej HomeID dostane ako udalosť pri inicializácii kontroléra. Ostatné rámce zahadzuje. Pracuje v troch vláknach kde dve slúžia na príjem udalostí a rámcov a ich spracovanie. Tretie hlavné vlákno je zodpovedné za generovanie alertov.

Algoritmus generovania alertov sa spustí až po prijatí udalosti o inicializácií kontroléra z brány a prvom prijatom rámci. Dovtedy vlákno aktívne čaká. Po inicializácii a prijatí prvého rámca sa spúšťa hlavný cyklus, kde sa vlákno uspáva a prebúdza v intervale generovania alertov. Po prebudení prejde v hlavnom cykle všetky uzly a zisťuje prítomnosť jednotlivých útokov. Pri všetkých útokoch zistí, či bol potenciálny útok prijatý dávnejšie ako je doba potrebná na synchronizáciu so spárovanými uzlami z brány a ak nebol tak ho ignoruje a bude vyhodnotený v ďalšej iterácií hlavného cyklu (po ďalšom prebudení).

Pri skenovaní siete kontroluje, či je rozdiel časovej značky štartu útoku a spárovania daného uzlu väčší ako doba kedy sa nemajú hlásiť alerty, keďže ide o správy vymieňané pri párovaní. Ak áno tak generuje alert, ktorý posiela na výstup. Pri MITM skontroluje, či uzol nebol spárovaný a ak nie generuje alert. Pri modifikácii NL a SR Cache zas skontroluje, či uzly posielané v týchto správach neboli spárované a ak aspoň jeden nebol generuje alert. Po vygenerovaní alertu o konkrétnom útoku alebo jeho zamietnutí sa uzlu resetuje informácia o danom útoku.

Kapitola 8

Experimenty, testovanie a overenie detekčného systému

Táto kapitola je venovaná popisu experimentov, testovania a overenia detekčného systému vytvoreného v rámci diplomovej práce. Najprv sú predstavené experimenty a ako som testoval a overoval riešenie popri vývoji. V ďalšej časti je popísaný princíp automatických testov a jednotlivé testované scenáre. Nasleduje popis testovania na smerovači Turris so systémom OpenWRT. Poslednou časťou je overenie systému v reálnej prevádzke.

8.1 Experimenty

Experimenty prebiehali na notebooku s operačným systémom Arch Linux, na ktorom zároveň prebiehal vývoj detekčného systému. Počítač je vybavený procesorom Intel Core i3-2310M 2.1GHz a 8 GB pamäte RAM. Zároveň bolo riešenie priebežne overované aj na Raspberry Pi s procesorom ARM, ktorého hardvérové prostriedky približne odpovedajú domácim smerovačom so systémom OpenWRT.

Pre testovanie presnosti detekcie párovania BLE bolo použitých niekoľko BLE zariadení napr. BeeWi Door Sensor[1] alebo fitness náramok Xiaomi Mi Band 2[2] s využitím Bluetooth 4.0 kontroléra ASUS USB-BT400[3]. Zariadenia som niekoľko krát spároval s kontrolérom pomocou nástroja bluetoothctl[4]. Detektor dokázal odhaliť všetky párovacie procesy, či už prvé alebo opakované. Neúspešné párovanie sa mi však nepodarilo nasimulovať a taktiež som bol s dostupnými zariadeniami schopný overiť len verziu párovania Legacy Pairing a metódu Just Works. Zistenie zvolenej verzie a metódy párovania je však realizované výberom z tabuľky na základe vymieňaných párovacích informácií presne podľa Bluetooth špecifikácie a tak by malo byť funkčné zistenie aj inej párovacej metódy.

Pre experimenty so Z-Wave som využíval senzorické zariadenia so Z-Wave kontrolérom popísané v 5.1. Kontrolér bol pripojený k počítaču, na ktorom bola nasadená IoT brána BeeeOn a jednotlivé moduly detekčného systému. Prijímanie Z-Wave komunikácie všetkých prenosových kanálov bolo realizované dvoma pripojenými SDR prijímačmi RTL2832U. Prijímač je možné vidieť na obrázku 5.6.

[1] http://www.bee-wi.com/wp-content/uploads/2017/10/BSD00_FR_EN_User_Manual.pdf
[2] https://www.mi.com/global/miband2/
[3] https://www.asus.com/Networking/USBBT400/
[4] https://www.archlinux.org/packages/?name=bluez-utils

Vytvoril som niekoľko skriptov pre simulovanie jednotlivých útokov. Na odosielanie dát som použil zariadenie HackRF One zobrazené na obrázku 3.8. Jednotlivé skripty sa líšia vlastne len obsahom odosielaných dát pomocou tohto SDR zariadenia. K samotnému odosielaniu dát bol použitý existujúci nástroj hackrf_transfer[5]. Ten umožňuje vysielanie rádiového signálu zo vstupného súboru na zadanej frekvencii, prípadne periodické odosielanie. Pre vytvorenie vstupného súboru s dátami k odoslaniu bol použitý nástroj Waving-Z[6], konkrétne jeho časť wave-out, ktorá dokáže modulovať a kódovať Z-Wave rámce v hexadecimálnej podobe. Takto boli dáta simulujúce konkrétny útok uložené do súboru a následne vyslané pomocou hackrf_transfer.

Ako prvé som ladil a testoval zber sieťovej komunikácie pomocou modulov Z-Wave SDR Sniffer, Z-Wave Collector a vytváranie štatistík modulom Z-Wave Statistics Creator. Odhalil som niekoľko chýb pri spracovávaní Z-Wave rámcov, ktoré som následne opravil. Vykonával som rôzne experimenty, kde som zariadenia pároval a odpároval s kontrolérom, resetoval ich, skúšal s nimi komunikovať pomocou HackRF One, nechal ich určitý čas komunikovať bez môjho pričinenia atď. Na základe odchytenej komunikácie som si overil správnosť vykonanej analýzy sieťovej prevádzky týchto zariadení a presnosť zachytávania sieťovej prevádzky. Presnosť odchytávania rámcov závisí na polohe brány s pripojenými SDR donglami a polohe útočníka. Pri priamej viditeľnosti medzi nimi som zistil úspešnosť zachytenia nepoškodeného rámca približne 85 %. Testoval som to zasielaním požiadaviek senzoru, ktorý na požiadavky odpovedal alebo stláčaním tlačidla na senzore, kedy do siete posiela NodeInfo rámec.

Ďalším krokom bolo ladenie a testovanie modulu Z-Wave Detector. Pomocou vytvorených skriptov som simuloval jednotlivé detekované útoky a sledoval reakcie detektora. Odhalil som niekoľko implementačných chýb, ktoré som postupne opravil. Odhalenie útokov je závislé na presnosti odchytávania rámcov. V prípade, že sa nestalo, že by potrebné rámce pre detekciu útoku neboli zachytené, dokázal detektor odhaliť všetky simulované útoky. Ako už bolo popísané, skenovanie siete, modifikácia zoznamu susedov a MITM uzol v komunikácii generujú v sieťovej prevádzke niekoľko rámcov a pre detekciu útoku stačí zachytiť jeden z nich. Neodhalenie útoku sa preto objavilo len v ojedinelých prípadoch. Detekcia modifikácie SR a Backbone cache však vyžaduje zachytenie jedného konkrétneho rámca a tak je viac závislá na presnosti odchytávania rámcov.

Zachytené dáta pri spomínaných experimentoch som si pomocou NEMEA modulu Logger ukladal do súborov a následne som ich použil v automatických testoch, ktoré sú popísané v ďalšej sekcii.

Poslednou časťou experimentov bolo overenie detekcie navrhnutých scenárov anomálií s využitím existujúceho detektora WSN Anomaly a štatistík vytváraných pomocou modulu Z-Wave Statistics Creator. Presnosť detekcií je závislá hlavne na konfigurácií detektora (použité položky profilu, dĺžka časovej rady, atď.) a ďalej taktiež na presnosti odchytávania rámcov. Jednotlivé scenáre som overil experimentálne na krátkej časovej rade a pri testovaní som sa zameral hlavne na možnosť detekcie navrhnutých scenárov a nie presnosť detektora WSN Anomaly.

- **Kvalita komunikačných kanálov** – Cieľom tohto scenára bolo overenie nárastu počtu poškodených rámcov v prípade rušenia siete útočníkom. K dispozícií som však nemal žiadnu rušičku signálu a tak som tento útok simuloval odpojením anténok od SDR prijímačov, kedy bolo zachytených omnoho viac poškodených rámcov. Sledované boli položky CORRUPTED_C, CORRUPTED_CH1_C1, CORRUPTED_CH1_C2

[5] https://github.com/mossmann/hackrf/blob/master/host/hackrf-tools/src/hackrf_transfer.c
[6] https://github.com/baol/waving-z

a CORRUPTED_CH1_C3 štatistík siete. Výsledok experimentu bol pozitívny a simulované rušenie bolo hlásené ako incident.

- **Kvalita smerovania** – Scenár mal overiť nárast počtu smerovaných NACK správ v prípade, že útočník napadol sieť a vnútil uzlom nesprávne smerovacie cesty. Útok som simuloval zaslaním nesprávnych smerovacích ciest senzoru v sieti. Použitie týchto nesprávnych ciest som pre test vynútil odpojením senzora, cez ktorý viedla správna smerovacia cesta. Následne napadnutý senzor skúšal využívať falošné smerovacie cesty a boli generované NACK správy. Sledovaná bola položka ROUTED_NACK_C štatistík siete a SRC_NACK_C štatistík pre konkrétny uzol. V tomto prípade bol vždy hlásený incident, pretože v bežnej komunikácií sa smerovacie NACK správy vôbec nevyskytovali.

- **Skenovanie a zásahy do topológie siete** – Cieľom scenára bola detekcia neoprávnených zásahov do topológie siete, kedy sa útočník nesnaží vstúpiť do komunikácie ako falošný uzol a tak nie je odhaliteľný pomocou modulu Z-Wave Detector. Útok som simuloval zasielaním viacerých koordinačných správ pre manažment smerovania jednotlivým senzorom. Sledovaná bola položka NET_MANAG_C štatistík siete a útok bol detekovaný.

- **Periodicita komunikácie od sieťových prvkov** – Scenár mal overiť porušenie periodicity zasielania správ od senzora a to konkrétne výpadok komunikácie od senzoru, čo môže značiť stratu konektivity senzora alebo jeho odcudzenie. Keďže použité senzory posielajú namerané dáta každú hodinu, pre experimenty to nebolo vhodné. Periodicitu zasielania dát som simuloval využítím senzoru otvorenia dverí a jeho aktivovaním v pravidelných intervaloch. Interval som postupne zväčšoval. Sledované boli položky SRC_TOTAL_C a SRC_TOTAL_LM_T štatistík pre konkrétny uzol. Test dopadol úspešne, pretože detektor ohlásil porušenie periodicity zvyšovania hodnôt týchto položiek.

- **Množstvo prenášaných správ** – Cieľom scenára bolo overenie nárastu počtu správ počas generovaného DoS útoku na senzor s cieľom vybiť jeho batériu. Sledovaná bola položka TOTAL_OK_C štatistík siete a DST_SINGL_C štatistík pre konkrétny uzol. Tento útok bol detektorom taktiež odhalený.

8.2 Automatické testy

Pre automatické testovanie vznikol v rámci projektu SIoT skript Autotest.py s pomocnými skriptmi. S využitím nástroja Travis sa automaticky pri každej zmene v repozitári preložia všetky moduly a spustia automatické testy, ktoré sa vyhodnotia. Cieľom automatického testovania je možnosť niekoľko násobného overenia funkčnosti jednotlivých modulov zo zadefinovaných testovacích scenárov. Pri úpravách a rozširovaní daných modulov je potom jednoduché znovu overiť ich funkčnosť a zistiť, či sa do kódu nezaniesla nejaká chyba.

Skript pre automatické testovanie prehľadá adresárovú štruktúru repozitára a otestuje každý vytvorený modul, ktorý to podporuje. Skript vyhodnotí modul ako testovateľný ak obsahuje podadresár tests. Na testovanie využíva NEMEA moduly Logreplay a Logger. Logreplay umožňuje prehrať uložený vstup vo formáte CSV na vstupné rozhranie testovaného modulu. Logger umožňuje z výstupného rozhrania testovaného modulu uložiť výstup vo formáte CSV. Pôvodne skript pracoval tak, že jednotlivé testovacie vstupy a očakávané

výstupy boli uložené v zložke tests ako 2 súbory s rovnakým názvom líšiace sa len v prípone a to .csv pre vstup a .out pre očakávaný výstup. Pre každý takýto testovací pár sa potom spustí inštancia testovaného modulu, pomocou Logreplay sa prehrá testovací vstup a výstup sa uloží do súboru pomenovaného ako aktuálny test s príponou .realout s využitím modulu Logger. Následne sa porovná uložený výstup s očakávaným výstupom a ak sa zhodujú, test je vyhodnotený ako úspešný. Testovací skript umožňuje špecifikovať s akými parametrami má byť spustený testovaný modul a vykonanie shell príkazov pred a po skončení každého testu. Tieto nastavenia je potrebné mať uložené v súboru autotest.json v zložke modulu.

Toto riešenie však bolo nevyhovujúce pre moduly, ktoré majú viacero vstupov (Z-Wave Detector) alebo viacero výstupov (Z-Wave Stats Creator). Testovacie skripty som preto upravil, aby podporovali aj takéto moduly. Jednotlivé testovacie vstupy a vzorové výstupy sú vo formáte testname.X.csv a testname.Y.out, kde rovnako prípona .csv označuje vstupný súbor a prípona .out vzorový výstupný súbor. X a Y sú čísla od 0 do 9 a označujú na koľké vstupné rozhranie má byť prehratý testovací vstup, resp. z koľkého výstupného rozhrania má byť uložený výstup vo formáte testname.Y.realout pre porovnanie. Spúšťa sa teda niekoľko inštancií modulov Logreplay a Logger.

8.2.1 Detektor párovania BLE

Vstupné datasety obsahujú párovanie fitness náramku Xiaomi Mi Band2 s BLE kontrolérom ASUS USB-BT400 podporujúceho BLE verzie 4.0, ktoré bolo zachytené pomocou modulu HCI Collector. Pomocou vstupného parametru detektora je špecifikovaná zložka, do ktorej sa uloží zariadenie pri spárovaní. Pri každom teste sa táto zložka najprv vytvorí a po teste sa celá zmaže. Testovanými sú nasledovné scenáre. Ich bližší popis je možné nájsť v prílohe B.

- Úspešné spárovanie zariadenia
- Opakované úspešné spárovanie zariadenia

8.2.2 Z-Wave kolektor

Pre Z-Wave Collector som vytvoril niekoľko testov na overenie správneho parsovania a prevodu niektorých položiek do human-readable formy. Všetky testovacie rámce som zachytil pomocou Z-Wave SDR Sniffer a následne som vybral určité zaujímavé rámce pre otestovanie konkrétnych položiek rámca. Ide o jednoduché testy, ktoré nevyžadujú dodatočný popis, preto uvediem len ich výpis.

- 86 validných rámcov zo 122 zachytených
- Ack rámec nasledujúci po singlecast request rámci
- Ack rámec, veľkosť 13, sekvenčné číslo 9
- Broadcast rámec, System CC, Node Info
- Poškodený rámec, zlý checksum, kanál 2
- Poškodený rámec, zlý checksum, kanál 3
- Poškodený rámec, zlá veľkosť
- Home-id 3452127626, dev-addr 2, dst-id 1

- Smerovaný rámec, Switch Binary CC
- Singlecast request rámec, Battery CC

8.2.3 Z-Wave detektor

Všetky testovacie vstupné datasety obsahujú UniRec záznamy simulujúce podporované Z-Wave útoky a to skenovanie siete a útoky na smerovanie. Keďže detektor má 2 vstupné rozhrania, každý test je zložený z dvoch vstupov. Prvý vstup obsahuje rámce simulujúce útok a druhý vstup obsahuje eventy z brány. Z dôvodu časovej závislosti detektora som zachytenú sieťovú prevádzku pomocou Z-Wave SDR Sniffer a eventy pomocou NEMEA Collectora na bráne minimalizoval a upravil pre jednotlivé testy. Taktiež sa mi niektoré čiastkové stavy nepodarilo reálne nasimulovať pri experimentoch a tak som dané rámce na vstupe upravil tak, ako by sa mohli vyskytnúť v reálnej sieťovej prevádzke, aby bolo možné otestovať aj tieto stavy. Všetky vstupy rámcov obsahujú prvý nevalidný pomocný rámec pre nastavenie časovania modulu logreplay, ktorým sú prehrávané na vstupné rozhranie detektora. Druhý rámec nasleduje až po 2 sekundách, aby mohol byť detektor správne inicializovaný z druhého vstupného rozhrania, ktorým sú mu predané eventy o inicializácií kontroléra z čoho získa HomeID siete, aby rámce nezahadzoval a spárovaní senzorického zariadenia. Detektor je spúšťaný pre všetky testy so vstupnými parametrami:

- interval posielania hlásení o útokoch = 5 sekúnd
- časové okno pre synchronizáciu medzi rámcami a eventami = 1 sekunda
- časové okno kedy nehlásiť útok pri párovaní = 1 sekunda

Testovanými sú nasledovné scenáre. Ich bližší popis je možné nájsť v prílohe B.

- Skenovanie siete – vyžiadanie všetkých podporovaných informácií
- Skenovanie siete – uzol posiela všetky podporované informácie
- Skenovanie siete – uzol posiela informácie v čase párovania
- Modifikácia zoznamu susedov – požiadanie uzla o vykonanie Do NL Test na neznámy uzol
- Modifikácia zoznamu susedov – uzol posiela NL Test neznámemu uzlu
- Modifikácia zoznamu susedov – nereportovanie falošného poplachu
- Modifikácia zoznamu susedov – nereportovanie možného útoku po dodatočnom vyradení
- Modifikácia SR Cache – SR Cache Assignment
- Modifikácia SR Cache – Backbone Cache Assignment
- MITM uzol
- MITM uzol – nereportovanie falošného poplachu

8.2.4 Modul pre vytváranie štatistík zo Z-Wave siete

Na automatické testovanie modulu pre vytváranie štatistík zo Z-Wave siete som vytvoril jeden test na overenie správneho vytvárania štatistík. Test využíva 10 vybraných rámcov zachytených pomocou Z-Wave SDR Sniffer a jeho cieľom je overiť, že sa štatistiky počítajú a posielajú na výstup správne. Vybrané rámce pokrývajú všetky individuálne štatistiky pre sieť a taktiež jednotlivé uzly a bolo jednoducho manuálne spočítateľné, ako má vyzerať očakávaný výstup. Týchto 10 rámcov som duplikoval na 500. Vstupnými parametrami v autotest.json je nastavené HomeID siete 0xE1453473, interval generovania štatistík 5 sekúnd a je špecifikovaný parameter testing, kedy sa štatistiky vytvárajú s nulovým timestampom, aby bolo možné automatické testovanie.

8.2.5 Vyhodnotenie automatických testov

Všetky vytvorené testy boli niekoľko krát opakovane vyhodnotené a vždy dopadli úspešne. Ako už bolo spomenuté, automatické testovanie sa spúšťa pri každej zmene v repozitári NEMEA-SIoT a tak je možné vytvorené riešenie ďalej upravovať a rozširovať bez toho, aby bolo potrebné neustále pracne overovať funkčnosť.

8.3 Testovanie na smerovači so systémom OpenWRT

Všetky vytvorené moduly som otestoval aj na smerovači Turris Omnia so systémom OpenWRT. Cieľom testovania bolo overenie, že vytvorené balíky ide bez problémov nainštalovať, spustiť a celý detekčný systém funguje rovnako, ako na notebooku s Arch linuxom a Raspberry Pi so sytémom Raspbian, kde som systém vyvíjal a priebežne testoval. Pri testovaní som odhalil 2 chyby, ktoré sa v mojom vývojovom prostredí neprejavili. Prvá sa týkala modulu Z-Wave SDR Sniffer, kde som neinicializoval niektoré premenné na nulu a na systéme OpenWRT tak odchytávanie Z-Wave rámcov vôbec nefungovalo.

Druhú chybu som odhalil v module Z-Wave Detector a prejavovala sa pádom programu na Bus Error, v momente prijatia eventu z BeeeOn brány. Jej odhalenie mi zabralo pomerne dosť veľa času. Bol potrebný preklad modulu s debug symbolmi a priame testovanie binárky vyextrahovanej pred vytvorením inštalačného balíku, pretože pri vytváraní balíku sa odstraňujú debug symboly. Následne som debuggovaním našiel riadok kódu, na ktorom program padal. Išlo o získanie hodnoty typu double položky EVENT_TYPE z prijatej UniRec správy pomocou makra ur_get NEMEA frameworku, pretypovanie na uint8_t a priradenie do premennej typu uint8_t. Chybu som najprv odstránil priradením získanej hodnoty z makra do premennej typu double a až následným pretypovaním. Na vyextrahovanej binárke preloženej bez optimalizácií sa zdalo, že je problém odstránený. Po vytvorení balíku, kedy sú zapnuté optimalizácie sa však chyba prejavila rovnako. Vyskúšal som preklad a spustenie na lokálnom počítači s využitím ASan[7] (Address Sanitizer – detektor chýb pamäte). ASan označil riadky kódu, kde detektor na OpenWRT končil pádom programu na Bus Error. Chybou by malo byť nesprávne zarovnanie pamäte v interných štruktúrach NEMEA frameworku, čo môže spôsobovať práve Bus Error na architektúre ARM. Toto chovanie na OpenWRT som nahlásil autorovi frameworku a vytvoril nové hlásenie o chybe[8] v repozitári s popisom problému a postupom ako problém reprodukovať. Do odstránenia

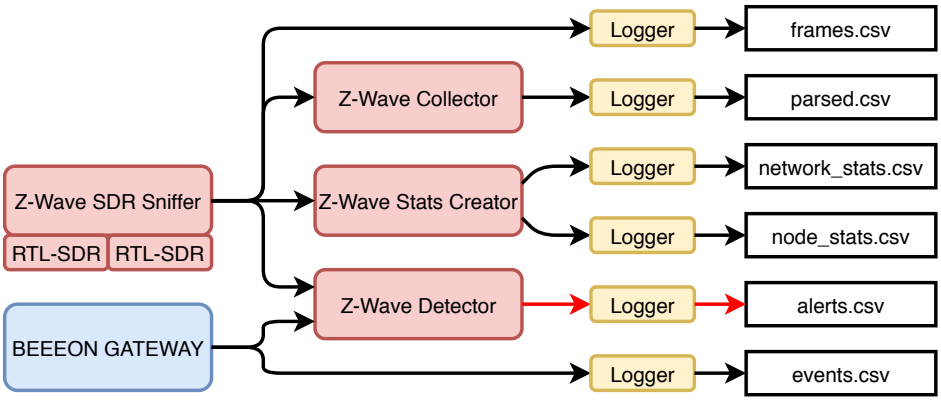
[7] https://github.com/google/sanitizers/wiki/AddressSanitizer
[8] https://github.com/CESNET/Nemea-Framework/issues/146

tohto problému som chybu dočasne opravil vynútením nepoužitia optimalizácií prekladačom na funkciu prijímania eventov z BeeeOn brány.

Okrem týchto dvoch chýb sa už pri testovaní detekčného systému na smerovači Turris Omnia, nevyskytli žiadne problémy. Úspešne som otestoval niekoľko detekcií útokov a pre detekciu skenovania siete sme v rámci projektu SIoT natočili video-demonštráciu, ktorá bude použitá pri obhajobe finálnych výstupov projektu.

8.4 Overenie systému v reálnej prevádzke

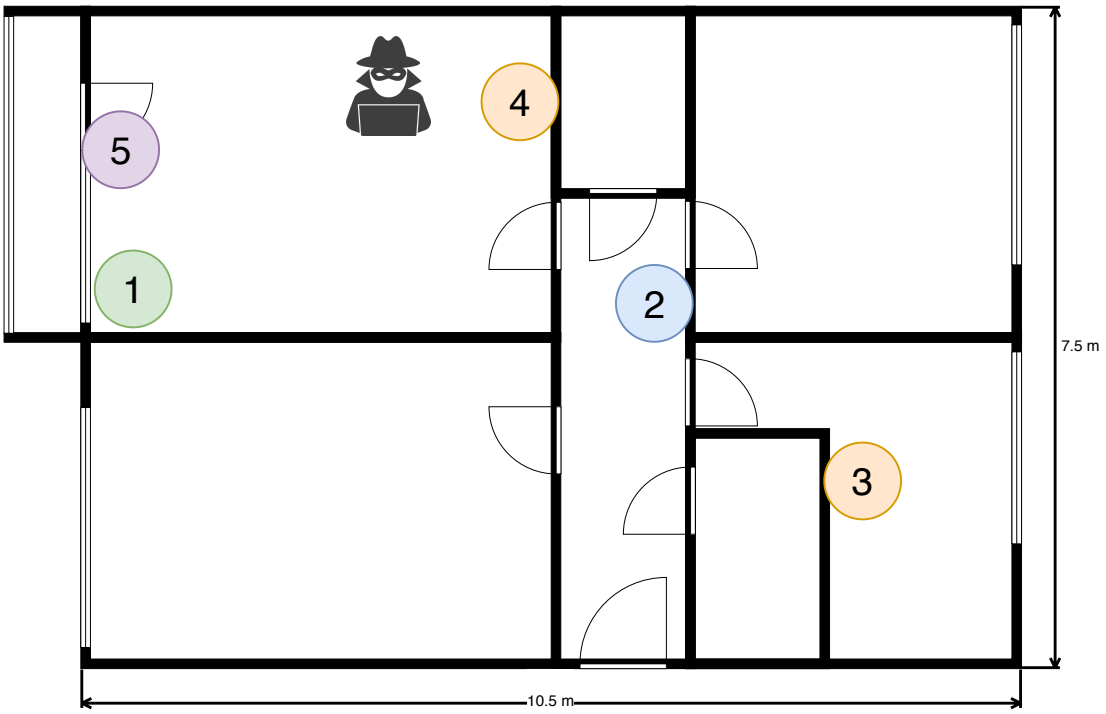
Poslednou časťou testovania bolo overenie systému v reálnej prevádzke. Cieľom testovania bolo overenie, že systém dokáže bežať dlhší čas bez toho, aby niektorý modul skončil s chybou a nebudú hlásené falošné poplachy. Zároveň dokáže odhaliť reálne prevádzané útoky pomocou HackRF One a pripravených skriptov.



Obr. 8.1: Zapojenie testovaných modulov

Testovanými boli vytvorené Z-Wave moduly avšak bez napojenia na detektor WSN Anomaly. Jednotlivé navrhnuté scenáre anomálii z vytváraných štatistických dát boli overené experimentálne a detektor WSN Anomaly nebol predmetom tohto testovania. Zapojenie testovaných modulov je možné vidieť na obrázku 8.1. UniRec výstupy všetkých modulov detekčného systému tohto reálneho nasadenia boli pomocou NEMEA modulu Logger uložené do odpovedajúcich súborov vo formáte CSV (Comma Separated Values). Výstupy sú uložené spolu s vytvorenými skriptami pre simuláciu útokov na priloženom CD. Priložený je taktiež štandardný výstup modulu Z-Wave SDR Sniffer so zapnutým parsovaním a logovaním odchytených rámcov.

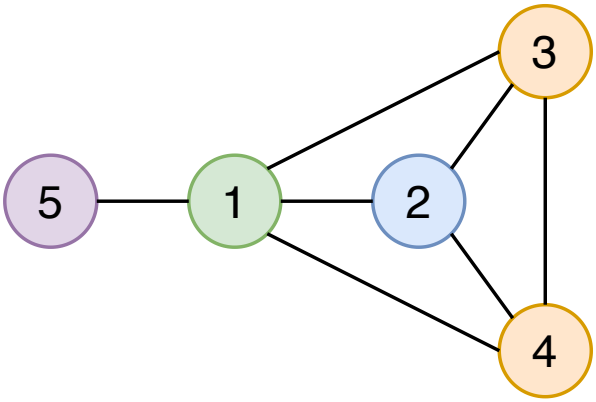
Ako prvé bolo potrebné pripraviť testovacie prostredie. BeeeOn bránu ako aj celý detekčný systém som nasadil na notebook s operačným systémom Arch Linux, ktorý som používal na vývoj a experimentovanie. Jednotlivé senzorické zariadenia som postupne spároval s bránou a rozmiestnil ich v byte tak, ako je možné vidieť na obrázku 8.2. Použil som batériovo napájaný senzor otvorenia dverí, inteligentnú zásuvku a 2 multisenzory trvalo napájané zo siete. K nasadenej IoT bráne bol pripojený Z-Wave kontrolér a 2 SDR dongly pre získavanie sieťovej prevádzky. Použité zariadenia sú popísané v 5.1.



Obr. 8.2: Pôdorys bytu s vyznačenými senzormi, bránou s pripojeným Z-Wave kontrolérom a SDR donglami, a útočníkom (1 – brána s pripojeným kontrolérom a SDR donglami, 2 – chytrá zásuvka, 3 – multisenzor, 4 – multisenzor, 5 – senzor otvorenia dverí)

Senzorické zariadenia mali však dosah v celom byte a potreboval som docieliť, aby sa určitá komunikácia smerovala. Senzor č. 3 som teda umiestnil do kovovej krabice, aby som mu zhoršil signál. Následne bolo potrebné odpojiť a zapojiť použitý Z-Wave kontrolér, aby sa reštartoval a vytvoril topológiu smerovania v sieti, ktorú je možné vidieť na obrázku 8.3. Topológiu som zistil analýzou zachytenej komunikácie v sieti. Uzly 2 a 5 komunikujú s kontrolérom výlučne priamo, uzol 4 väčšinou tiež, ale zachytených bolo aj pár rámcov kedy bola komunikácia smerovaná cez uzol 2 alebo uzly 2 a 3. Komunikácia medzi uzlom 3 a kontrolérom je prevažne smerovaná cez uzol 2, prípadne uzly 3 a 2.

Systém som nechal bežať po dobu dvoch dní kedy senzory pravidelne posielali namerané dáta, detektor otvorenia dverí reagoval pri otvorení dverí a chytrú zásuvku som občas zapol a vypol buď tlačítkom na nej alebo cez užívateľské rozhranie BeeeOn brány tak, aby sa generovala bežná komunikácia v sieti.



Obr. 8.3: Topológia vytvorenej siete (1 – kontrolér, 2 – chytrá zásuvka, 3 – multisenzor, 4 – multisenzor, 5 – senzor otvorenia dverí)

Na záver som pomocou HackRF One a pripravených skriptov zoskenoval sieť a vykonal útoky na smerovanie. 10 krát od každého senzoru som si vyžiadal výrobcu zariadenia a jeho typ, verziu softvéru, podporované triedy príkazov a základný prevádzkový stav. Všetky senzory okrem senzora otvorenia dverí na požiadavky odpovedali. Senzor 5 neodpovedal, pretože je napájaný batériovo a tak je väčšinu času uspatý. Následne som senzoru 4 zaslal požiadavku o vykonanie Do NL Test na neznámy uzol 8 s cieľom napadnúť jeho zoznam susedných uzlov. Senzor reagoval a vykonal NL Test na tento uzol. Tento útok som vykonal taktiež 10 krát ako aj posledný útok a to modifikáciu Backbone cache senzora 4. Zaslal som mu teda falošnú cestu ku kontroléru s podvrhnutým uzlom 8. Senzor však stále na posielanie využíval správnu smerovaciu trasu. Zrejme z dôvodu, že fungovala a tak neskúšal ďalšiu. Po odpojení senzorov, cez ktoré sa smerovala komunikácia zas skúsil priamu správu kontroléru bez smerovania a kontrolér mu odpovedal, takže moje riešenie rušenia signálu pomocou umiestnenia do kovovej krabice bolo nedostatočné. Pri reálnom nasadení sa mi nepodarilo senzor donútiť, aby používal smerovaciu cestu cez môj falošný uzol a tak nebolo možné otestovať detekciu MITM uzlu v komunikácií. Túto detekciu som však úspešne otestoval inými experimentami a automatickými testami.

Typ rámcov	**Rola uzla**	**Počet rámcov pre daný uzol**				
		1	**2**	**3**	**4**	**5**
Validný rámec	zdroj	3427	161	448	1041	1139
	cieľ	2651	501	803	1025	1167
Singlecast rámec	zdroj	1723	60	373	757	631
	cieľ	1789	173	739	281	539
Broadcast rámec	zdroj	3	0	2	40	1
Ack rámec	zdroj	1701	101	73	244	507
	cieľ	862	328	64	744	628
Smerovaný NACK rámec	zdroj	8	0	5	2	0
	cieľ	7	0	6	2	0
	failed hop	0	13	2	0	0

Tabuľka 8.1: Štatistiky pre jednotlivé uzly v sieti vytvorené modulom Z-Wave Statistics Creator

Typ rámcov	Počet
Validný rámec	6216
Poškodený rámec	490
Poškodený rámec na kanále 1	10
Poškodený rámec na kanále 2	248
Poškodený rámec na kanále 3	232
Smerovaný rámec	1014
Smerovaný ACK rámec	574
Smerovaný NACK rámec	15
Smerovaný aplikačný rámec	425
Singlecast rámec	3544
Ack rámec	2626
Multicast rámec	0
Broadcast rámec	46
Rámec na správu siete	78

Tabuľka 8.2: Štatistiky pre celú sieť vytvorené modulom Z-Wave Statistics Creator

V tabuľke 8.1 je možné vidieť vytvorené štatistiky pre jednotlivé uzly v sieti a v tabuľke 6.7 štatistiky pre celú sieť. Štatistiky boli vytvárané každú minútu nasadenia. Z pomeru validných a poškodených rámcov je vidieť, že bolo zachytených približne 93 % validných rámcov, čo si dovolím tvrdiť je skvelý výsledok. Treba do toho zarátať však, že niektoré rámce nemuseli byť vôbec zachytené. Dá sa to vidieť v pomere smerovaných aplikačných rámcov a smerovaných ACK/NACK rámcov, kde aplikačných rámcov je 73 %. Je to spôsobené tým, že bolo zachytených menej správ od vzdialeného izolovaného senzora 3. SDR prijímače boli umiestené pri IoT bráne, ktorá bola schválne umiestená na okraji bytu kvôli docieleniu smerovania. Pri centrálnom umiestnení by tak boli výsledky odchytávania rámcov lepšie. Porovnaním vytvorených štatistík modulom Z-Wave Statistics Creator s manuálnym spočítaním štatistík nad rámcami na výstupe modulu Z-Wave Collector som overil správne generovanie štatistík.

Detekčný systém dokázal bežať po celú dobu nasadenia a za túto dobu nebol hlásený žiaden falošný poplach. Zároveň boli všetky prevedené útoky zachytené a hlásené detektorom. Pri jednom skenovaní senzoru 2 nebola zachytená požiadavka s odpoveďou na základný prevádzkový stav. Útok bol preto detektorom hlásený bez tejto informácie. Detekcia skenovania siete a útokov na smerovanie je pomerne exaktná a závisí hlavne na úspešnosti odchytávania Z-Wave rámcov. Overenie detekčného systému v reálnej prevádzke dopadlo úspešne.

Kapitola 9

Záver

Cieľom práce bola analýza komunikačných protokolov IoT, ich zraniteľností, návrh a následná implementácia detektora anomálií tak, aby bolo možné prevádzkovať tento detektor na domácich smerovačoch so systémom OpenWRT alebo na vhodnej vstavanej hardvérovej platforme s procesorom ARM. Všetky body zadania boli splnené.

Práca sa venovala zraniteľnostiam komunikačných protokolov Bluetooth Low Energy a Z-Wave. U Bluetooth Low Energy som sa venoval detekcií potencionálneho útoku na párovací proces, ktorý je najzraniteľnejšou časťou tohto protokolu. Pri Z-Wave protokole, ktorý tvoril hlavnú časť tejto práce, som sa venoval získavaniu sieťovej komunikácie pomocou SDR a detekcii útokov nad touto komunikáciou.

Teoretická časť práce spočívala v zoznámení sa s architektúrou IoT sietí a s komunikačnými protokolmi IoT. Následne bolo potrebné naštudovať princípy, analyzovať charakteristiku sieťovej prevádzky a identifikovať zraniteľnosti protokolov Bluetooth Low Energy a Z-Wave. Okrem toho som sa zoznámil s princípmi detekcie anomálií v IoT sieťach a detekčným systémom NEMEA umožňujúcim jednoduché vytvorenie modulárneho riešenia. Ďalej som sa zoznámil s existujúcim detektorom anomálií nad štatistickými IoT dátami vyvinutého v rámci projektu SIoT.

Praktická časť práce sa venovala hlbšej analýze sieťovej prevádzky vybraných senzorických zariadení Z-Wave, návrhu a vytvoreniu detekčného systému. Prvá časť sa venovala detekcií potenciálneho útoku na párovanie u Bluetooth Low Energy. Na základe naštudovaných informácií som navrhol a implementoval detektor schopný odhaliť opakované párovanie zariadení predstavujúce potenciálny útok. Druhá časť sa venovala detekcií anomálií v sieti Z-Wave. S využitím nadobudnutých znalostí z literatúry a skúmaním komunikácie v sieti som navrhol a implementoval detekčný systém schopný z komunikácie odchytenej pomocou SDR detekovať skenovanie siete a niekoľko útokov na smerovanie. Ďalej som rozšíril existujúci detektor WSN Anomaly o podrobnejšie štatistiky so širším pohľadom na sieť. Pôvodné riešenie malo k dispozícií len Z-Wave štatistiky s obmedzeným pohľadom na sieť získané zo Z-Wave kontroléra.

Veľký dôraz bol kladený na testovanie vytvoreného detekčného systému. Prvým krokom bolo overovanie systému pomocou rôznych experimentov a simulovaných útokov. V tomto kroku bolo objavených niekoľko implementačných chýb, ktoré boli následne opravené. Druhým krokom bolo vytvorenie automatických testov pre opakované overovanie funkčnosti jednotlivých modulov. Testy boli vytvorené zo zachytených dát pri experimentoch a pokrývajú veľké množstvo scenárov. Ďalším krokom bolo otestovanie systému na smerovači Turris Omnia so systémom OpenWRT. Pri testovaní som objavil niekoľko chýb, ktoré som odstránil. Navyše som objavil chybu v NEMEA frameworku, ktorá sa prejavila na archi-

tektúre ARM. Chybu som nahlásil autorom frameworku. Simuloval som niekoľko scenárov útokov, ktoré detekčný systém dokázal odhaliť. Tak som overil, že vytvorený systém pracuje správne aj na smerovači so systémom OpenWRT. Poslednou časťou testovania bolo overenie systému v reálnej prevádzke. Cieľom bolo overiť, že nasadený detekčný systém dokáže bežať dlhšiu dobu bez toho, aby niektorý modul skončil s chybou. Zároveň dokáže odhaliť všetky simulované útoky. Testovanie dopadlo úspešne.

Detekčný systém bol vytvorený tak, aby spĺňal všetky špecifikované požiadavky. Systém je možné nasadiť na domácom smerovači so systémom OpenWRT alebo na inej vhodnej platforme s procesorom ARM, kde bude bežať aj IoT brána. Modulárnosť detekčného systému poskytuje flexibilitu nasadenia a umožňuje jednoduchú rozšíriteľnosť o ďalšie detekčné moduly. Otestovaním bolo zistené, že detekčný systém je funkčný a schopný odhaliť podporované útoky a anomálie.

Výsledky práce sa snažia riešiť aktuálne problémy v oblasti bezpečnosti IoT. Hlavným problémom je to, že IoT siete obsahujú často nezabezpečené a ľahko napadnuteľné prvky. Vytvorené riešenie sa snaží poskytnúť dodatočnú ochranu týmto prvkom v podobe včasnej detekcie anomálií a odhalenia útočníka. Výsledky práce boli využité v rámci projektu SIoT.

Vytvorený detekčný systém bude možné do budúcna rozširovať o ďalšie detekčné metódy. Zaistený je zber dát pomocou SDR priamo zo Z-Wave siete a tak bude po identifikovaní ďalších prípadných zraniteľností jednoduché pridať do systému ďalší detekčný modul. BeeeOn brána poskytuje len obmedzenú funkcionalitu, čo sa týka Z-Wave siete. Bolo by preto vhodné túto funkcionalitu rozšíriť a následne skúsiť identifikovať ďalšie možnosti detekcie anomálií nad touto funkcionalitou.

Literatúra

[1] Al Sarawi, S., Anbar, M., Alieyan, K. a Alzubaidi, M. Internet of Things (IoT) communication protocols: Review. In: *2017 8th International Conference on Information Technology (ICIT)*. 2017, s. 685–690. DOI: 10.1109/ICITECH.2017.8079928.

[2] Badenhop, C. W., Graham, S. R., Ramsey, B. W., Mullins, B. E. a Mailloux, L. O. The Z-Wave routing protocol and its security implications. *Computers & Security*. Elsevier Ltd. 2017, zv. 68, s. 112–129. DOI: 10.1016/j.cose.2017.04.004. ISSN 0167-4048.

[3] Cejka, T., Bartos, V., Svepes, M., Rosa, Z. a Kubatova, H. NEMEA: A Framework for Network Traffic Analysis. In: *2016 12th International Conference on Network and Service Management (CNSM)*. IEEE, 2016, s. 195–201. DOI: 10.1109/CNSM.2016.7818417.

[4] Cisco. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are* [online]. 2015 [cit. 2020-05-28]. Dostupné z: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.

[5] Esbensen, A. *Rtl-zwave* [online]. [cit. 2020-05-28]. Dostupné z: https://github.com/andersesbensen/rtl-zwave.

[6] Hall, J. L. *A Practical Wireless Exploitation Framework for Z-Wave Networks*. 2016. Diplomová práca. Air Force Institute of Technology, Department of Electrical and Computer Engineering.

[7] Hall, J. a Ramsey, B. *EZ–Wave: Tools for Evaluating and Exploiting Z-Wave Networks using Software-Defined Radios* [online]. [cit. 2020-05-28]. Dostupné z: https://github.com/cureHsu/EZ-Wave.

[8] ITU-T. *Recommendation G.9959: Short range narrowband digital radiocommunication transceivers – PHY, MAC, SAR and LLC layer specifications*. International Telecommunication Union, 2015.

[9] Krejčí, R., Hujňák, O. a Švepeš, M. Security survey of the IoT wireless protocols. In: *2017 25th Telecommunication Forum (TELFOR)*. IEEE, 2017, 2017-, s. 1–4. DOI: 10.1109/TELFOR.2017.8249286.

[10] Paetz, C. *Z-Wave Essentials*. 4. vyd. Zwickau: CreateSpace Independent Publishing Platform, 2017. ISBN 978-1545394540.

[11] Ryan, M. *Bluetooth: With Low Energy Comes Low Security* [online]. [cit. 2020-05-28]. Dostupné z: https://www.usenix.org/system/files/conference/woot13/woot13-ryan.pdf.

[12] Slawomir, J. *Gattacking Bluetooth Smart Devices* [online]. [cit. 2020-05-28]. Dostupné z: https://gattack.io/whitepaper.pdf.

[13] Soukup, D. *Detekce anomálií v provozu IoT sítí*. Praha, 2018. Diplomová práca. České vysoké učení technické v Praze, Fakulta informačních technologií.

[14] Statista. *Internet of Things – number of connected devices worldwide 2015-2025* [online]. Statista Research Department, 17. novembra 2016 [cit. 2020-05-28]. Dostupné z: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[15] Tierney, A. *Z-Shave. Expoiting Z-Wave downgrade attacks* [online]. Pen Test Partners, 23. mája 2018 [cit. 2020-05-28]. Dostupné z: https://www.pentestpartners.com/security-blog/z-shave-exploiting-z-wave-downgrade-attacks//.

[16] Townsend, K., Davidson, R. a Cufí, C. *Getting Started with Bluetooth Low Energy*. O'Reilly, 2014. ISBN 9781491949511.

[17] Z-Wave Alliance. *D-Link Z-Wave Motion Sensor – DCH-Z120* [online]. 2014 [cit. 2020-05-28]. Dostupné z: https://products.z-wavealliance.org/products/1152?selectedFrequencyId=1.

[18] Z-Wave Alliance. *AeoTec Z-Stick Gen5 – ZW090-C* [online]. 2015 [cit. 2020-05-28]. Dostupné z: https://products.z-wavealliance.org/products/1355?selectedFrequencyId=1.

[19] Z-Wave Alliance. *AeoTec MultiSensor 6 – ZW100-C* [online]. 2017 [cit. 2020-05-28]. Dostupné z: https://products.z-wavealliance.org/products/2714?selectedFrequencyId=1.

[20] Z-Wave Alliance. *Climax Door Contact – DC-23ZW* [online]. 2017 [cit. 2020-05-28]. Dostupné z: https://products.z-wavealliance.org/products/2542?selectedFrequencyId=1.

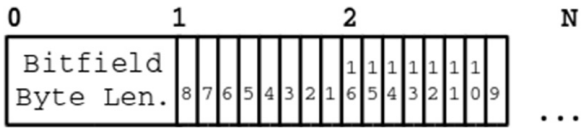
[21] Z-Wave Alliance. *Fibaro Wall Plug E/F - FGWPE/F-102* [online]. 2018 [cit. 2020-05-28]. Dostupné z: https://products.z-wavealliance.org/products/2818?selectedFrequencyId=1.

Príloha A

Dátové štruktúry používané pri manažmente smerovania Z-Wave siete

NL primitív

Prvá dátová štruktúra je tzv. NL primitív [2]. Používa na prenos stavu topológie v koordinačných (network management) správach. Prvý bajt NL primitívu označuje dĺžku zvyšných bajtov. Nasleduje bitové pole s premenlivou dĺžkou. Štruktúra bitového poľa je konzistentná so záznamom v tabuľke susednosti, kde každý bit odpovedá konkrétnemu ID uzla. Presné priradenie je $NodeID = 8i + j + 1;\ i = 0, ..., n - 1;\ j = 0, ..., 7$, kde i je bajtový posun od prvého bajtu bitového poľa, n je dĺžka bitového poľa v bajtoch, a pre každý bajt je j bitové posunutie. Bity v každom bajte sú zapísané ako big-endian a j je index od najmenej po najviac dôležitý bit. Toto je zobrazené na obrázku A.1, aby sa ukázalo, že ID uzlov sa pri čítaní bitového poľa zľava doprava monotónne nezvyšujú. Command bajt správy určuje implikáciu bitovej hodnoty na každej pozícii ID uzla. Napríklad, keď uzol poskytuje svoj lokálny NL kontroléru, bit nastavený na 1 indikuje, že odosielateľ susedí s node ID odpovedajúcim tej bitovej pozícii.

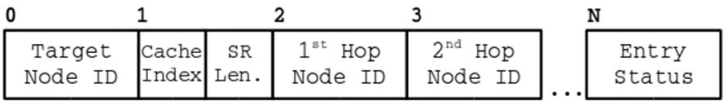


Obr. A.1: Dátová štruktúra NL obsahujúca prvých 16 node ID [2]

Záznam v SR cache

Druhá dátová štruktúra je záznam v SR cache [2]. Je určený na výmenu smerovacích ciest a jeho formát je zobrazený na obrázku A.2. Prvý bajt označuje cieľový uzol cesty. Druhý bajt je zložený z 2 častí. Prvá časť uchováva index záznamu cache, na základe čoho sa záznam ukladá do cache regiónu asociovaného s cieľovým uzlom. Druhá časť označuje dĺžku trasy SR a za ním nasleduje zoznam vnútorných skokov (uzlov) danej smerovacej cesty. Posledný bajt má vplyv na výber trasy, ale podľa [2] sa ho zatiaľ nepodarilo úplne preskúmať a je potrebné celkové preskúmanie firmvéru zariadení pomocou reverzného inžinierstva. Objavené boli len

2 hodnoty 0x10 a 0x08. V prípade 0x10 je táto SR použitá zainteresovaným uzlom. V prípade 0x08 niektoré uzly úplne odmietajú použiť túto trasu a iné ju používajú s nízkou prioritou až po vyčerpaní všetkých s hodnotou 0x10.



Obr. A.2: Dátová štruktúra záznam v SR cache [2]

Príloha B

Scenáre automatických testov

Detektor párovania BLE

Úspešné spárovanie zariadenia

- **Use case:** Zariadenie sa s kontrolérom úspešne spáruje prvý raz.
- **Vstup:** zachytená komunikácia párovania zariadenia d7:e7:e7:6b:d6:3f s kontrolérom 5c:f3:70:87:46:2b.
- **Očakávaný výstup:** Hlásenie o úspešnom párovaní zariadenia pomocou Legacy Pairing s využitím metódy Just Works.

Opakované úspešné spárovanie zariadenia

- **Use case:** Zariadenie sa s kontrolérom úspešne spáruje 2 krát po sebe.
- **Vstup:** zachytená komunikácia párovania zariadenia d7:e7:e7:6b:d6:3f s kontrolérom 5c:f3:70:87:46:2b a následné opakované párovanie.
- **Očakávaný výstup:** Hlásenie o úspešnom párovaní zariadenia pomocou Legacy Pairing s využitím metódy Just Works a ďalšie rovnaké hlásenie s rozdielom, že je to opakované párovanie.

Z-Wave detektor

Skenovanie siete – vyžiadanie všetkých podporovaných informácií

- **Use case:** útočník predstiera, že je kontrolér a požiada uzol 2 o informácie o ňom (výrobca a typ zariadenia, verzia softvéru, podporované triedy príkazov, atď.). Tieto správy sa typicky vyskytujú v komunikácii len pri párovaní, čo však nie je tento prípad a tak by mal byť hlásený alert. Mal by byť hlásený len jeden alert z dôvodu intervalu posielania hlásení o útokoch.
- **Vstup:**
  - Rámce: niekoľko požiadaviek na spomínané informácie o uzle 2.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2.

- **Očakávaný výstup:** hlásenie o skenovaní uzlu 2 so všetkými triedami príkazov, ktoré boli skenované.

Skenovanie siete – uzol posiela všetky podporované informácie

- **Use case:** útočník predstiera, že je kontrolér a požiada uzol 2 o informácie o ňom. Túto komunikáciu sa však nepodarilo zachytiť a zachytené sú len odpovede uzla 2 (výrobca a typ zariadenia, verzia softvéru, podporované triedy príkazov, atď.). Tieto správy sa typicky vyskytujú v komunikácii len pri párovaní, čo však nie je tento prípad a tak by mal byť hlásený alert. Mal by byť hlásený len jeden alert z dôvodu intervalu posielania hlásení o útokoch.
- **Vstup:**
  - Rámce: niekoľko odpovedí spomínaných informácií o uzle 2.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2.
- **Očakávaný výstup:** hlásenie o skenovaní uzlu 2 so všetkými triedami príkazov, ktoré boli skenované.

Skenovanie siete – uzol posiela informácie v čase párovania

- **Use case:** Požiadavka na informácie od uzlu 2 môže znamenať skenovanie útočníkom, ale v časovom okne párovania je uzol 2 reálne spárovaný, takže tieto informácie si reálne vypýtal kontrolér. Nemal by byť preto hlásený žiaden alert.
- **Vstup:**
  - Rámce: Požiadavka na výrobcu zariadenia a typ zariadenia od uzlu 2.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2 v časovom okne párovania.
- **Očakávaný výstup:** žiadne hlásenie o útoku.

Modifikácia zoznamu susedov – požiadanie uzla o vykonanie Do NL Test na neznámy uzol

- **Use case:** Útočník predstiera, že je kontrolér a zasiela uzlu 2 správu Do NL Test, s jeho NodeID 5 s cieľom votrieť sa do NL uzlu 2 a postupne sa tak pokúsiť napadnúť tabuľku susednosti kontroléra.
- **Vstup:**
  - Rámce: správa Do NL Test pre uzol 2 s neznámym NodeID 5.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2.
- **Očakávaný výstup:** hlásenie o modifikácii NL uzlu 2 s neznámym uzlom 5.

Modifikácia zoznamu susedov – uzol posiela NL Test neznámemu uzlu

- **Use case:** Uzol posiela správu NL Test neznámemu uzlu 5 po tom, čo mu útočník poslal požiadavku Do NL Test s jeho NodeID 5. Kontrolér si môže potom vyžiadať NL od uzlu 2 a tak útočník môže napadnúť tabuľku susednosti kontroléra.
- **Vstup:**
  - Rámce: správa NL Test od uzlu 2 neznámemu uzlu 5.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2.
- **Očakávaný výstup:** hlásenie o modifikácií NL uzlu 2 s neznámym uzlom 5.

Modifikácia zoznamu susedov – uzol posiela NL Report s neznámym uzlom

- **Use case:** Uzol posiela správu NL Report s neznámym susedom 5 kontroléru po tom, čo mu útočník poslal požiadavku Do NL Test a on vykonal NL Test na útočníka. Niektoré kontroléry môžu akceptovať túto správu a tak útočník napadne tabuľku susednosti kontroléra.
- **Vstup:**
  - Rámce: správa NL Report od uzlu 2 kontroléru s uzlom útočníka 5.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2.
- **Očakávaný výstup:** hlásenie o modifikácií NL uzlu 2 s neznámym uzlom 5.

Modifikácia zoznamu susedov – nereportovanie falošného poplachu

- **Use case:** Typická komunikácia požiadania uzlu 2 o vykonanie testu, či je uzol 5 jeho sused. Uzol 5 je spárovaný, takže by nemal byť hlásený žiaden alert.
- **Vstup:**
  - Rámce: Do NL Test, NL Test, NL Report správy s potvrdeniami medzi kontrolérom a uzlom 2 s otestovaním a pridaním uzlu 5 do NL uzlu 2.
  - Eventy: inicializácia kontroléra, spárovanie uzlov 2 a 5.
- **Očakávaný výstup:** žiadne hlásenie o útoku.

Modifikácia zoznamu susedov – nereportovanie možného útoku po dodatočnom vyradení

- **Use case:** Typická komunikácia požiadania uzlu 2 o vykonanie testu, či je uzol 5 jeho sused. V čase spracovania rámcov, detektor nepozná uzol 5, takže si poznačí podozrenie z útoku. Uzol 5 je však spárovaný v čase do hlásenia útoku a tak je hlásenie zahodené.
- **Vstup:**
  - Rámce: Do NL Test, NL Test, NL Report správy s potvrdeniami medzi kontrolérom a uzlom 2 s otestovaním a pridaním uzlu 5 do NL uzlu 2.
  - Eventy: inicializácia kontroléra, spárovanie uzlu 2 a opozdene uzlu 5.
- **Očakávaný výstup:** žiadne hlásenie o útoku.

Modifikácia SR Cache – SR Cache Assignment

- **Use case:** Útočník sa vydáva za kontrolér a posiela správu SR Cache Assignment uzlu 2 s falošnou cestou cez jeho uzol 5. Komunikácia by potom mohla byť smerovaná cez uzol útočníka vystupujúceho ako MITM a ten by ju tak mohol upravovať alebo zahadzovať.
- **Vstup:**
  - Rámce: správa SR Cache Assignment pre uzol 2 s cestou obsahujúcou neznámy uzol 5.
  - Eventy: inicializácia kontroléra, spárovanie uzlov 2 a 3.
- **Očakávaný výstup:** hlásenie o modifikácii SR Cache uzlu 2 s neznámym uzlom 5.

Modifikácia SR Cache – Backbone Cache Assignment

- **Use case:** Útočník sa vydáva za kontrolér a posiela správu Backbone Cache Assignment uzlu 2 s falošnou cestou cez jeho uzol 5. Komunikácia by potom mohla byť smerovaná cez uzol útočníka vystupujúceho ako MITM a ten by ju tak mohol upravovať alebo zahadzovať.
- **Vstup:**
  - Rámce: správa Backbone Cache Assignment pre uzol 2 s cestou obsahujúcou neznámy uzol 5.
  - Eventy: inicializácia kontroléra, spárovanie uzlov 2 a 3.
- **Očakávaný výstup:** hlásenie o modifikácii SR Cache uzlu 2 s neznámym uzlom 5.

MITM uzol

- **Use case:** Útočníkovi sa podarilo napadnúť smerovanie sieti a vystupuje v smerovanej komunikácii ako MITM uzol 5.
- **Vstup:**
  - Rámce: smerovaná komunikácia $3 \rightarrow 2 \rightarrow 5 \rightarrow 1$.
  - Eventy: inicializácia kontroléra, spárovanie uzlov 2 a 3.
- **Očakávaný výstup:** hlásenie o MITM uzle 5 a napadnutej smerovacej ceste $3 \rightarrow 2 \rightarrow 5 \rightarrow 1$.

MITM uzol – nereportovanie falošného poplachu

- **Use case:** Overuje, že nie je hlásený žiaden alert, ak sú všetky uzly smerovacej cesty spárované.
- **Vstup:**
  - Rámce: smerovaná komunikácia $3 \rightarrow 2 \rightarrow 5 \rightarrow 1$.
  - Eventy: inicializácia kontroléra, spárovanie uzlov 2,3 a 5.
- **Očakávaný výstup:** žiadne hlásenie o útoku.