FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza protokolu ZigBee v IoT zařízeních
Student:	Tomáš Rosenbaum
Vedoucí:	Ing. Jiří Dostál, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

ZigBee je jeden z nejrozšířenějších standardů pro bezdrátovou komunikaci IoT (Internet of Things) zařízení. Protože se jedná o relativně levná zařízení, nemusí být jejich implementace nejbezpečnější. Zařízení mohou být náchylná např. na odposlouchávání citlivých dat (https://en.wikipedia.org/wiki/Sniffing_attack), narušení fungování sítě (https://en.wikipedia.org/wiki/Denial-of-service_attack) či přímo ovládání sítě (https://en.wikipedia.org/wiki/Replay_attack).

1. Seznamte se s protokolem ZigBee a zaměřte se na jeho bezpečnostní rozšíření.
2. Sestavte testovací síť ze zařízení implementující protokol ZigBee.
3. Vytvořte aplikaci zjednodušující bezpečnostní analýzu ZigBee sítě.
4. Pomocí vytvořené aplikace proveďte bezpečnostní analýzu zařízení na vámi vytvořené testovací síti.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 13. února 2023 v Praze.

Bakalářská práce

BEZPEČNOSTNÍ ANALÝZA PROTOKOLU ZIGBEE V IOT ZAŘÍZENÍCH

Tomáš Rosenbaum

Fakulta informačních technologií
Katedra počítačových systémů
Vedoucí: Ing. Jiří Dostál, Ph.D.
11. května 2023

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

V první řadě bych chtěl poděkovat Ing. Jiřímu Dostálovi, Ph.D. za vedení této práce a jeho pomoc při její tvorbě, zejména za nasměrování tématu a zapůjčení potřebného technického vybavení. Také bych rád poděkoval mé rodině a přátelům za podporu, ochotu při potřebných konzultacích a korekturu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 Tomáš Rosenbaum

Abstrakt

Tato bakalářská práce se zabývá bezpečnostní analýzou protokolu ZigBee a protokolu IEEE 802.15.4, na kterém je protokol ZigBee vystavěný. Vysvětluje základní fungování obou protokolů, detailněji popisuje jejich bezpečnostní rozšíření a jejich slabá místa. Věnuje se také popisu vytvoření testovací sítě pomocí koordinátoru CC2652P, chytré žárovky, vypínače a aplikace Home Assistant. Výsledkem práce je aplikace umožňující snadnou bezpečnostní analýzu zařízení a sítí, kterou je možné jednoduchým způsobem rozšiřovat. Aplikace je psaná v jazyce Python s využitím knihovny Scapy. Využívá koordinátor CC2531 k odposlouchávání komunikace. Navržená aplikace je použita pro analýzu vytvořené testovací sítě, jejíž bezpečnost je v práci vyhodnocována.

Klíčová slova IoT, ZigBee, IEEE 802.15.4, bezpečnostní analýza, CC2531, Scapy

Abstract

This bachelor thesis deals with the security analysis of the ZigBee protocol and the IEEE 802.15.4 protocol on which the ZigBee protocol is built. It explains the basic functioning of both protocols and their security extensions and weaknesses in detail. It also discusses how to create a test network using the CC2652P coordinator, smart bulb, switch and Home Assistant application. The work results in an application that allows easy security analysis of devices and networks, which can be extended simply. The application is written in Python using the Scapy library. It uses the CC2531 coordinator to eavesdrop on communications. The proposed application is used to analyze the created test network whose security is evaluated in the work.

Keywords IoT, ZigBee, IEEE 802.15.4, security analysis, CC2531, Scapy

Seznam zkratek

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
ABCs	abstract base classes
AES	advanced encryption standard
APDU	application support sub-layer protocol data unit
APL	application layer
APS	application support sub-layer
ASN	absolute slot number
CAP	contention access period
CCM	counter mode encryption and cipher block chaining message authentication code
CCM*	extension of counter mode encryption and cipher block chaining message authentication code
CFP	contention-free period
CLI	command line interface
CSMA-CA	carrier sense multiple access with collision avoidance
DSME	deterministic and synchronous multichannel extension
ECDHE	elliptic curve Diffie-Hellman ephemeral
EUI-64	64-bit extended unique identifier
FFD	full-function device
GTS	guaranteed timeslot
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IPv6	internet protocol version 6
MAC	medium access control
MFR	MAC footer
MHR	MAC header
MIC	message integrity code
NPDU	network layer protocol data unit
NWK	network layer
PAN	personal area network
PCAP	packet capture
PHR	PHY header
PHY	physical layer
PSDU	PHY service data unit
RFD	reduced-function device
SHR	synchronization header
SPEKE	simple password exponential key exchange
TMCTP	TVWS multichannel cluster tree PAN
TSCH	timeslotted channel hopping
TVWS	television white space
ZDO	ZigBee device object

Kapitola 1
Úvod

S rostoucím zájmem o zapojení chytrých zařízení do každodenního života se zvětšuje i zranitelnost každého z nás vůči kybernetickým útokům. Internet věcí (IoT) dělá z každodenních spotřebičů chytrá zařízení, která mezi sebou umí komunikovat. Tato zařízení jsou navíc čím dál tím dostupnější. Připojení těchto zařízení do Internetu věcí s sebou ovšem přináší rizika kybernetických útoků. Vzniká tedy potřeba zaručit, že tato chytrá zařízení jsou bezpečná a odolná před potenciálními útočníky.

V prostoru IoT zařízení se vyskytuje mnoho různých komunikačních protokolů. Mezi ně patří například WiFi, Z-Wave, LoRaWAN či ZigBee. Tato práce se zabývá posledním z uvedených protokolů, spravovaným Connectivity Standards Alliance, protokolem ZigBee. Hlavními důvody jsou jeho otevřenost a rozšířenost. Tyto vlastnosti sice splňují i protokoly WiFi, nicméně na rozdíl od nich je protokol ZigBee přímo navržený pro IoT zařízení a navíc není tak probádaný. Kromě specifikace protokolu je také důležitá jeho implementace v konkrétních zařízeních, kterou se tento text také zabývá.

Cílem této práce je nejprve provést bezpečnostní analýzu protokolu jako takového a následně vybraných zařízení využívajících tento protokol za účelem zjištění, zda jsou konkrétní zařízení implementující protokol ZigBee bezpečná pro použití širokou veřejností. Výstupem bude aplikace, která provede bezpečnostní analýzu ZigBee zařízení. Protože není možné otestovat všechna zařízení, bude aplikace primárně určena pro lidi v oboru kybernetické bezpečnosti, kterým by měla usnadnit analýzu dalších zařízení. Zároveň by měla být aplikace dostatečně snadná na použití běžnými uživateli, kteří si chtějí ověřit bezpečnost jejich domácí sítě.

Teoretická část této práce se bude zaměřovat na bezpečnostní analýzu protokolu ZigBee, resp. protokolu IEEE 802.15.4, na kterém je tento protokol vystaven. Budou zde popsány jednotlivé vrstvy tohoto protokolu a bude zde vysvětleno, jak na nich probíhá komunikace. Bude též objasněno, jak je u jednotlivých vrstev řešena bezpečnost a následně budou diskutována potenciálně zranitelná místa.

Součástí teoretické části bude i popis aktuálních řešení pro sestavení domácí sítě spolu s popisem existujících aplikací pro jejich bezpečnostní analýzu.

Samotné sestavení malé testovací sítě zařízení, která využívají tento protokol, bude obsahem praktické části práce. Účelem je napodobit průměrnou domácí síť chytrých zařízení. Bude zde popsán postup sestavení sítě a nastavení jednoduché automatizace.

Výstupem praktické části bude návrh a vytvoření aplikace pro bezpečnostní analýzu, která bude ověřovat dříve diskutované potenciální zranitelnosti. Tato aplikace bude jednoduchým způsobem rozšiřitelná o kontrolu dalších potenciálních zranitelností. Z důvodu lepší dostupnosti a jednodušší rozšiřitelnosti bude využívat aplikace snadno dostupný ZigBee koordinátor CC2531, který po nahrání speciálního firmwaru umožňuje odposlouchávat komunikaci tohoto protokolu, již bude navržená aplikace dále analyzovat. Toto řešení sice neumožňuje přímo komunikovat s dal-

šími zařízeními, nicméně je mnohonásobně levnější a jednodušší, než řešení, která to umožňují (např. softwarově definované rádio).

Nakonec bude vytvořená aplikace použita pro analýzu sestavené testovací sítě a bude vyhodnocena bezpečnost jednotlivých zařízení. Závěrem budou diskutovány způsoby zlepšení bezpečnosti této sítě.

Kapitola 2
Cíle

Cílem práce je provést bezpečnostní analýzu protokolů ZigBee a IEEE 802.15.4 a následně jejich implementací ve vybraných zařízeních.

Teoretická část by měla vysvětlit základní fungování těchto protokolů a dále podrobněji popsat jejich bezpečnostní funkce. Měla by též představit a popsat jejich slabá místa.

Dalším cílem teoretické části je představit aktuální technická řešení pro vytvoření a správu domácích ZigBee sítí a pro jejich analýzu.

V praktické části by práce měla navázat na analýzu z části teoretické a využít diskutované technologie k vytvoření testované sítě. Ta by měla reprezentovat průměrnou domácí síť s nastaveným ovládáním a automatizací.

Hlavním cílem praktické části je navrhnout a vytvořit aplikaci pro testování ZigBee sítí, která využije poznatků z analýzy slabých míst a aktuálních řešení. Důležitým aspektem této sítě by měla být její snadná rozšířitelnost.

Závěrečným cílem je provedení analýzy vytvořené sítě pomocí výsledné aplikace. Pomocí té by měla být vyhodnocena bezpečnost této sítě a měly by být navrženy možné úpravy pro zvýšení bezpečnosti. Takto upravená síť by měla být znovu zanalyzována aplikací, aby bylo demonstrováno zlepšení bezpečnosti.

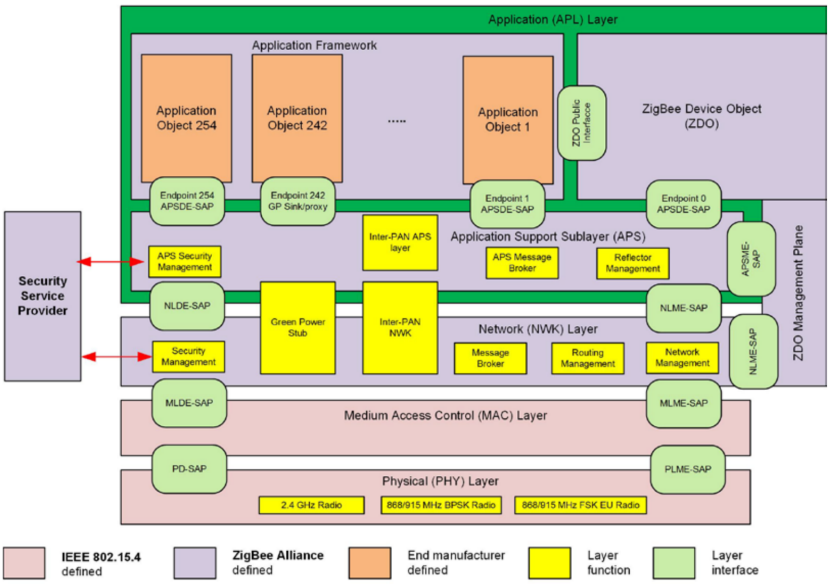
Kapitola 3
Úvod do protokolu

Tato kapitola představuje protokol ZigBee. Jsou zde popsány jeho jednotlivé vrstvy, struktury jejich rámců a způsob komunikace.

První verze protokolu ZigBee byla schválena již v prosinci 2004. Od té doby prošel protokol mnoha revizemi. Poslední (23.) revize byla schválena v lednu 2023 a zveřejněna v březnu téhož roku [1]. Protokol byl vybudován aliancí ZigBee Alliance, která se později přejmenovala na Connectivity Standard Alliance [2].

Hlavními cíli protokolu je cenová dostupnost a nízká spotřeba energie zařízení, která ho implementují. Proto je také vystavěn na protokolu *IEEE Standard for Low-Rate Wireless Networks* (IEEE 802.15.4).

Protokol tvoří 4 vrstvy: fyzická (PHY), *Medium Access Control* (MAC), síťová (NWK) a aplikační (APL). Aplikační vrstva dále obsahuje vrstvy *Application Support Sublayer* (APS), *ZigBee Device Object* (ZDO) a *Application Framework*. Vrstvy PHY a MAC jsou definovány ve výše zmíněném standardu IEEE, zbylé potom v samotném ZigBee. [1]



Obrázek 3.1 Vrstvy protokolu ZigBee [1]

3.1 Protokol IEEE 802.15.4

Tento standard definuje dva typy zařízení: *full-function device* (FFD) a *reduced-function device* (RFD). FFD jsou zařízení, která jsou schopna fungovat jako koordinátor sítě, naopak RFD jsou zařízení, která toho schopna nejsou. RFD jsou tedy jednoduchá zařízení, která nepotřebují posílat velké objemy dat a která jsou v jeden okamžik spojena pouze s jedním FFD.

Každé zařízení má svou 64bitovou rozšířenou adresu, kterou je *64-bit extended unique identifier* (EUI-64) definovaná v [3]. Při připojení do sítě může být zařízení přidělena další kratší 16bitová adresa.

Při vytvoření sítě je vybrán unikátní identifikátor sítě – *personal area network ID* (PAN ID). Ten umožní zařízením v rámci této sítě využívat výše zmíněné kratší adresy. [4]

3.1.1 Fyzická vrstva

Fyzická vrstva definuje, jak jsou data mezi zařízeními přenášena. Bezdrátově na jedné z následujících frekvencích: 868 MHz, 915 MHz, nebo 2,4 GHz. 868 MHz je frekvence používaná v Evropě, zatímco frekvence 915 MHz se používá např. v USA, či Austrálii. 2,4 MHz se používá celosvětově. [1]

Pro zařízení využívající frekvenci 868 MHz je dostupný pouze jeden kanál (0), zařízení využívající frekvenci 915 MHz mají k dispozici kanálů 10 (1–10) a zařízení, která používají frekvenci 2,4 GHz, mají k dispozici 16 (11–26) různých komunikačních kanálů. Od čísla kanálu je odvozena přesná hodnota frekvence, na které zařízení komunikují. [4]

3.1.2 Vrstva MAC

Hlavní funkcí vrstvy MAC je řídit přístup ke komunikačnímu médiu. K tomu mohou sloužit tzv. *superframes* („superrámce“), které definují, kdy a kdo může vysílat. Na začátku každého superrámce vyšle koordinátor *beacon* („maják“), který slouží k synchronizaci zařízení, identifikaci sítě a popisu superrámce. Superrámce mohou mít aktivní a neaktivní část, během které může koordinátor přejít do režimu nízké spotřeby. Aktivní část se poté dělí na dvě části: *contention access period* (CAP) a *contention-free period* (CFP). Přesný typ superrámců určuje koordinátor.

CAP je úsek, v němž jakékoli zařízení může vysílat za použití mechanismu *carrier sense multiple access with collision avoidance* (CSMA-CA), nebo ALOHA mechanismu [4]. Tyto mechanismy zajišťují, že v jeden okamžik vysílá pouze jedno zařízení. Mechanismus ALOHA nejdříve začne vysílat data a následně zjišťuje, zda se přenos povedl. Naproti tomu CSMA-CA nejdříve čeká, dokud není médium volné, a až poté posílá data. [5]

CFP je úsek, který se dále dělí na maximálně 7 *guaranteed timeslots* (GTS). GTS je úsek speciálně vyhrazený pro konkrétní aplikaci. CFP vždy následuje za CAP.

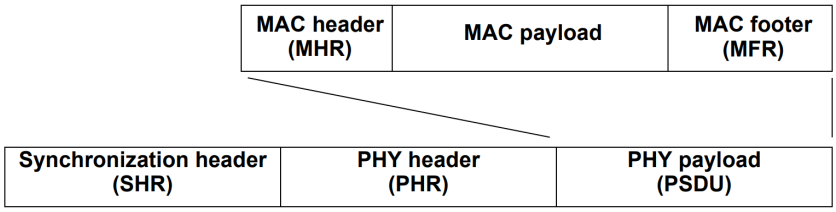
Kromě jednoduchých superrámců existují ještě metody *deterministic and synchronous multichannel extension* (DSME), *timeslotted channel hopping* (TSCH) nebo *TVWS multichannel cluster tree PAN* (TMCTP), které koncept superrámců buď vylepšují nebo nahrazují. Koordinátor se také může rozhodnout superrámce vůbec nepoužít a tedy nevysílat žádné beacons.

Pokud chce zařízení poslat data koordinátorovi, počká na beacon a ve vhodný okamžik data pošle. Jestliže koordinátor nepoužívá beacons, vyšle zařízení data rovnou.

Pokud chce koordinátor poslat data zařízení, signalizuje tuto informaci v beaconu. Zařízení poté zašle žádost o data a koordinátor je pošle. V případě komunikace bez beacons vyčká koordinátor na žádost o data a následně, pokud nějaká má, je odešle. [4]

3.1.3 Rámec IEEE 802.15.4

Rámec představuje strukturu, ve které jsou data přenášena. Struktura rámce definovaná standardem je k vidění na obrázku 3.2, na kterém je znázorněno rozdělení vrstvy MAC (na hlavičku, data a patičku) a její zakomponování do fyzické vrstvy, která se dělí na synchronizační hlavičku, hlavičku fyzické vrstvy a data.



Obrázek 3.2 Struktura rámce IEEE 802.15.4 [4]

Struktura synchronizační hlavičky a hlavičky PHY záleží na konkrétní vrstvě PHY a je mimo rozsah této práce.

Hlavička, data a patička MAC dohromady tvoří MAC rámec. Jeho struktura je znázorněna v 3.3. [4]

Octets: 1/2	0/1	0/2	0/2/8	0/2	0/2/8	variable	variable		variable	2/4
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Auxiliary Security Header	IE		Frame Payload	FCS
		Addressing fields					Header IEs	Payload IEs		
MHR							MAC Payload			MFR

Obrázek 3.3 Struktura MAC rámce [4]

3.2 Protokol ZigBee

Samotný protokol ZigBee je na vybudován na protokolu IEEE 802.15.4. Nad jeho rámec přidává síťovou a aplikační vrstvu.

3.2.1 Síťová vrstva

Síťová vrstva podporuje dva typy topologií: star („hvězda“) a mesh („síť“). Oba typy topologií jsou řízeny tzv. koordinátorem. Jeho hlavním úkolem je vytvoření a udržování sítě. Ve hvězdicové topologii komunikují všechna zařízení přímo s koordinátorem. V síťové topologii je možné rozšířit rozsah sítě pomocí routerů („směrovačů“) [1]. Koncová zařízení potom komunikují pouze s jedním z routerů. Routery také přebírají část povinností koordinátora. Koordinátor i routery jsou zařízení typu FFD, koncová zařízení většinou typu RFD. [6]

Rámec této vrstvy se skládá z hlavičky a dat a tvoří tzv. rámec *network layer protocol data unit* NPDU, který je k vidění na obrázku 3.4.

ZigBee zařízení musí být schopna připojit se k síti, odpojit se od sítě a obnovit připojení k síti. Koordinátor a routery musí zařízením dovolit připojit se k síti a odpojit se od sítě. Koordinátory navíc musí být schopny vytvořit novou síť. Routery a koncová zařízení musí být schopna přenositelnosti v rámci sítě. [1]

Octets: 2	2	2	1	1	0/8	0/8		Variable	Variable
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address		Source route subframe	Frame payload
NWK Header									Payload

Obrázek 3.4 Struktura rámce NPDU [1]

3.2.2 Aplikační vrstva

Jak již bylo zmíněno, vrstva APL obsahuje vrstvy Application Support Sublayer, Application Framework a ZigBee Device Object.

Application Support Sublayer tvoří rozhraní mezi síťovou vrstvou a ZDO a aplikacemi v Application Framework. Rámec této vrstvy se skládá z hlavičky a dat a tvoří tzv. rámec *application support sub-layer protocol data unit* (APDU), jehož struktura je znázorněna na obrázku 3.5.

Octets: 1	0/1	0/2	0/2	0/2	0/1	1	0/ Variable	Variable
Frame control	Destination endpoint	Group address	Cluster identifier	Profile identifier	Source endpoint	APS counter	Extended header	Frame payload
	Addressing fields							
APS header								APS payload

Obrázek 3.5 Struktura rámce APDU [1]

Application Framework je prostředí, ve kterém se nachází aplikace definované výrobcem. Těchto aplikací může být až 254. Příkladem takové aplikace je Home Automation.

ZigBee Device Object tvoří rozhraní mezi Application Support Sublayer a aplikacemi v Application Framework. Také je zodpovědný za inicializaci APS a síťové vrstvy a poskytovatele bezpečnostních služeb. [1]

Kapitola 4
Bezpečnost

Tato kapitola se zabývá bezpečnostními schopnostmi protokolů ZigBee a IEEE 802.15.4, resp. jejich vrstev. Definuje jejich bezpečnostní vlastnosti a popisuje fungování klíčových částí. Dále kapitola obsahuje definici základních typů útoků relevantních pro tento protokol a analyzuje jeho slabá místa.

4.1 Bezpečnostní vlastnosti protokolu IEEE 802.15.4

Zařízení používající tento protokol vůbec nemusí využít jím nabízené bezpečnostní funkce. Pokud zařízení bezpečnost implementuje, stará se o ni MAC vrstva. Ta zajišťuje následující vlastnosti:

důvěrnost dat – k datům se dostanou jen oprávněné osoby/zařízení

pravost dat – data, která obdržím, nikdo po cestě neupravil

ochrana proti přehrání – není možné znovu odeslat stejná data (není dostupná v TSCH módu)

Zda rámec je, či není zabezpečen, indikuje bit *Security Enabled*, který se nachází ve *Frame Control* části (vyzobrazena na obrázku 4.1) MAC hlavičky.

Bits: 0–2	3	4	5	6	7	8	9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Sequence Number Suppression	IE Present	Destination Addressing Mode	Frame Version	Source Addressing Mode

Obrázek 4.1 Struktura Frame Control [4]

V případě, že je bit *Security Enabled* roven 1, je součástí MAC hlavičky i bezpečnostní hlavička (*Auxiliary Security Header* – znázorněna na obrázku 4.2).

Hodnota *Security Level* nacházející se v části *Security Control* (vyobrazena na obrázku 4.3) bezpečnostní MAC hlavičky určuje číslem bezpečnostní úrovně míru zabezpečení rámce.

K dispozici je 8 (resp. 7[1]) různých bezpečnostních úrovní. Vysvětlení jednotlivých hodnot se nachází v tabulce 4.1.

[1] Hodnota 4 je zastaralá a neměla by se používat.

Octets: 1	0/4	0/1/5/9
Security Control	Frame Counter	Key Identifier

Obrázek 4.2 Struktura bezpečnostní hlavičky MAC [4]

Bit: 0–2	3–4	5	6	7
Security Level	Key Identifier Mode	Frame Counter Suppression	ASN in Nonce	Reserved

Obrázek 4.3 Struktura Security Control [4]

Tabulka 4.1 Význam bezpečnostních úrovní MAC vrstvy [4]

bezpečnostní úroveň	šifrování	integrita
0	ne	ne
1	ne	ano (MIC-32)
2	ne	ano (MIC-64)
3	ne	ano (MIC-128)
4	—	—
5	ano	ano (MIC-32)
6	ano	ano (MIC-64)
7	ano	ano (MIC-128)

Šifrování zajišťuje důvěrnost dat. Pro šifrování byla zvolena 128bitová šifra *Advanced Encryption Standard* (AES-128) v operačním módu CCM*.

AES-128 je rozšířená bloková symetrická šifra s velikostí bloku i klíče 128 bitů. Předání/inicializaci klíčů přenechává tento standard vyšším vrstvám.

Operační mód určuje, jak je daná bloková šifra použita. Mód CCM* je rozšířená verze módu CCM, která navíc umožňuje šifrování bez kontroly integrity.

Integrita (pravost dat) je též zajištěna pomocí operačního módu CCM*, který kromě šifrování dat vytvoří i tzv. *Message Integrity Code* (MIC, „kód integrity zprávy“). Ten slouží k ověření, že přijatá data jsou těmi samými, která byla odeslána. Tento standard využívá MIC délek 32, 64 a 128 bitů.

Ochrana proti přehrání znamená, že narušitel nemůže zachycená data poslat znovu a tím ovládat zařízení. Toho je dosaženo tak, že se se zprávou zašifruje i tzv. *nonce* (number used once), který v sobě obsahuje i čítač rámců. To znamená, že dvě stejné zprávy poslané po sobě budou mít rozdílnou hodnotu čítače – tedy i rozdílnou hodnotu nonce – a tedy se zašifrují na dva různé šifrové texty.

V módu TSCH je místo čítače rámců použito pořadové číslo časového úseku od počátku sítě / uměle nastaveného počátku. Během jednoho časového úseku se tedy dvě stejné zprávy zašifrují na dva stejné šifrové texty. Proto v tomto módu není zaručena ochrana proti přehrání. [4]

4.2 Bezpečnostní vlastnosti protokolu ZigBee

Protokol ZigBee kvůli ceně zařízení, která ho implementují, nemůže předpokládat, že jednotlivé aplikace na jednom zařízení jsou řádně odděleny (např. firewallem). Také nemůže předpokládat kryptografické oddělení mezi aplikacemi a vrstvami. To vede k tzv. otevřenému modelu důvěry (open trust model) – vrstvy protokolu a všechny aplikace si musí navzájem důvěřovat.

Bezpečnost protokolu byla vystavěna s následujícími rozhodnutími:

- Vrstva, ze které rámec pochází, je zodpovědná za jeho zabezpečení.
- Pokud je žádána ochrana dat, je využita bezpečnost síťové vrstvy pro všechny rámce. Výjimkou jsou zprávy mezi routerem a nově připojeným zařízením – ty jsou nezabezpečené, dokud zařízení neobdrží klíč sítě.
- Kvůli otevřenému modelu důvěry mohou různé vrstvy používat stejný klíč. Toto zmenší náklady na uskladnění klíčů.
- Je zajištěno, že pouze zdroj a cíl zprávy jsou schopni číst zprávy chráněné společným klíčem.
- Bezpečnostní úroveň všech zařízení v jedné síti je stejná. Pokud aplikace vyžaduje vyšší úroveň bezpečnosti, může si ji zařídit na vyšší vrstvě.

Protokol definuje dva typy zabezpečených sítí: centralizované a distribuované. Centralizované mají jedno trust center, které spolu s koordinátorem udržuje síť. V distribuovaných sítích může jakýkoli router plnit funkci trust centra, tedy může např. autorizovat nové zařízení.

Podobně jako na MAC vrstvě jsou i na NWK a APS vrstvách definovány bezpečnostní úrovně, jejichž význam je popsán v tabulce 4.2. Na rozdíl od MAC vrstvy je zde možnost využít i 4. úroveň, která umožňuje pouze šifrování bez kontroly integrity. [1]

Tabulka 4.2 Význam bezpečnostních úrovní vrstev NWK a APS [4]

bezpečnostní úroveň	šifrování	integrita
0	ne	ne
1	ne	ano (MIC-32)
2	ne	ano (MIC-64)
3	ne	ano (MIC-128)
4	ano	ne
5	ano	ano (MIC-32)
6	ano	ano (MIC-64)
7	ano	ano (MIC-128)

4.2.1 Trust center

Trust center („centrum důvěry“) je zařízení, kterému důvěřují ostatní zařízení na síti. Jeho úkoly jsou distribuce klíčů a nastavení, udržování a aktualizace bezpečnostní politiky. V centralizované síti je vždy právě jedno trust center, většinou se jedná o koordinátora. V distribuované síti se mohou všechny routery chovat jako trust centra. Ty na rozdíl od trust centra v centralizované síti mohou přenášet pouze síťový klíč, nikoli trust center link klíč. [1]

4.2.2 Klíče

Klíče jsou základem bezpečnosti každého komunikačního protokolu. Umožňují totiž šifrování dat a tím dávají možnost zaručit důvěrnost a integritu dat. Při správě klíčů je důležité, aby daný klíč měla k dispozici jen legitimní zařízení. Jakmile se totiž cizí subjekt dostane k tajným klíčům, je schopný celou komunikaci dešifrovat a tím veškerá bezpečnost zaniká.

ZigBee specifikuje dva druhy 128bitových klíčů: [1]

link klíče se používají při unicastové komunikaci na aplikační vrstvě (klíč je sdílen mezi 2 zařízeními)

síťové klíče se používají při broadcastové komunikaci a jakékoli komunikaci na síťové vrstvě (klíč je sdílen mezi všemi zařízeními v síti)

4.2.2.1 Link klíče

Link klíče může používat jen aplikační vrstva a jeden takovýto klíč je unikátní pro dvojici zařízení. ZigBee využívá 5 typů link klíčů:

centralizovaný globální trust center link klíč se používá pro připojení k centralizovaným zabezpečeným sítím[2]

distribuovaný globální link klíč se používá pro připojení k distribuovaným zabezpečeným sítím

link klíč instalačního kódu je klíč odvozen z instalačního kódu připojovaného zařízení

aplikační link klíč se používá mezi dvěma zařízeními (kde ani jedno není trust center) pro šifrování na aplikační vrstvě

trust center link klíč je klíč použitý pro komunikaci mezi trust centrem a zařízením

Pro připojení nového zařízení do sítě je ve většině případů potřeba, aby zařízení a koordinátor měli nastavený společný link klíč. Tyto link klíče se dle způsobu získání dělí na statické a dynamické. Statickým klíčem může být buď centralizovaný globální trust center link klíč, nebo se může jednat o klíče, jejichž hodnota je odvozená z instalačních kódů. Využití známé výchozí hodnoty sice zaručí snadné připojení jakéhokoli zařízení, ale v okamžiku posílání síťového klíče dojde ke zranitelnému momentu. Dynamické klíče jsou nastaveny pomocí jednoho ze dvou mechanismů: *Elliptic Curve Diffie-Hellman Ephemeral* (ECDHE) nebo *Simple Password Exponential Key Exchange* (SPEKE). Výhoda těchto mechanismů je, že umožňují bezpečné připojení zařízení, která nepodporují instalační kódy. Navíc se při tomto způsobu výsledný klíč nepřenáší mezi zařízeními.

Z link klíče se dále pro konkrétní účely odvozují další klíče: [1]

data klíč je stejný jako původní link klíč a používá se k ochraně běžné komunikace

key-transport klíč je odvozený klíč, který se používá pro ochranu přenosu síťového klíče

key-load klíč je odvozený klíč, který se používá pro ochranu přenosu link klíče

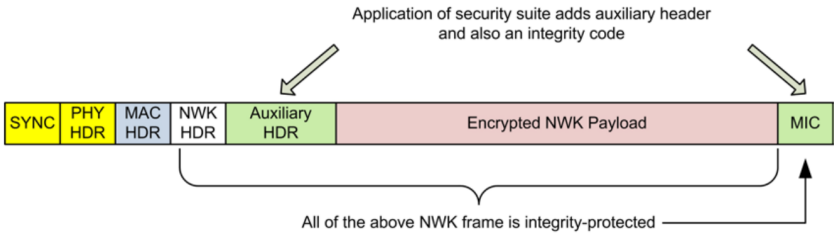
4.2.2.2 Síťové klíče

Síťový klíč můžou používat vrstvy NWK a APS. Jedná se o klíč sdílený všemi zařízeními v síti. Ta si mohou pamatovat více síťových klíčů, každý se svým identifikátorem, ovšem vždy jen jeden z nich je aktivně využíván k šifrování. Pokud se koordinátor rozhodne pro změnu klíče, přestanou zařízení využívat aktuální klíč a přejdou na nový klíč určený koordinátorem. [1]

4.2.3 Zabezpečení síťové vrstvy

Při šifrování na síťové vrstvě se do ZigBee rámce za síťovou hlavičku přidá pomocná bezpečnostní hlavička a dojde k zašifrování dat. Síťová hlavička, bezpečnostní hlavička a zašifrovaná data jsou využita k výpočtu MIC, který se připojí na konec rámce. Výsledná struktura je znázorněna na obrázku 4.4. [1]

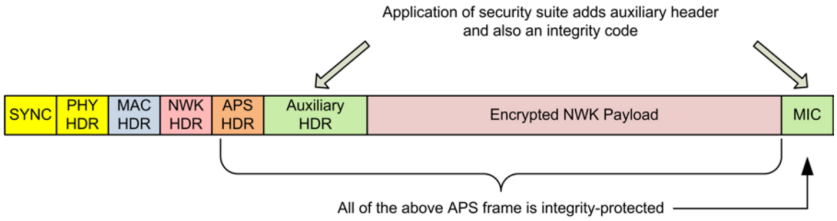
[2] Výchozí hodnota tohoto klíče je `5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39` (ZigBeeAlliance09).



Obrázek 4.4 Struktura ZigBee rámce zašifrovaného na síťové vrstvě [1]

4.2.4 Zabezpečení aplikační vrstvy

Při šifrování na aplikační vrstvě se pomocná bezpečnostní hlavička přidá až za APS hlavičku. Pro výpočet MIC se na rozdíl od síťové vrstvy využije APS hlavička místo hlavičky síťové. Struktura takto zabezpečeného rámce je vidět na obrázku 4.5.



Obrázek 4.5 Struktura ZigBee rámce zašifrovaného na aplikační vrstvě [1]

Aplikační vrstva nabízí několik bezpečnostních služeb, většinou určených ke správě klíčů: [1]

transport key („přenos klíče“) slouží k přenosu link nebo síťového klíče mezi zařízeními

update device („aktualizace zařízení“) slouží k informování trust centra routerem o změnách zařízení v síti

remove device („odstranění zařízení“) slouží k informování routeru trust centrem o odstranění zařízení ze sítě

request key („žádost o klíč“) slouží k žádosti o zaslání aplikačního, či trust center link klíče

switch key („změna klíče“) slouží k informování zařízení trust centrem, že by měl změnit aktivní síťový klíč

verify key („ověření klíče“) slouží k ověření, že zařízení i trust center pracují se stejným link klíčem

confirm key („potvrzení klíče“) slouží k potvrzení výše zmíněného ověření

4.2.4.1 Transport key

Tato služba umožňující přenos klíče je důležitým bezpečnostním aspektem protokolu. Služba může být využita jak během připojování nového zařízení pro přenos síťového klíče, tak i kdykoli jindy během komunikace pro přenos nového síťového, či link klíče.

Při přenosu prvního síťového klíče je buď možné zprávu nezabezpečit, nebo ji zabezpečit link klíčem.

Nový klíč je možné rozeslat buď broadcastem či unicastem. Při broadcastu je zpráva zašifrována aktuálním síťovým klíčem, zatímco při unicastu link klíčem. Unicastový způsob je

bezpečnější, protože je tím zaručeno, že se o novém síťovém klíči dozví jen zamýšlená zařízení.[3] Protože je síťový klíč vysílán častěji a slabším mechanismem než link klíče, je v případě použití broadcastového způsobu větší šance na dešifrování nového klíče útočníkem. [1]

4.2.4.2 Switch key

Protokol umožňuje příkazem switch key měnit síťové klíče. Příkaz switch key v sobě obsahuje pouze identifikátor klíče, na který by měla zařízení přejít. Je proto potřeba nový klíč dříve distribuovat pomocí transport key.

Při změně klíče se také obnoví čítač síťových rámců, který může nabývat maximálně hodnoty $2^{32} - 1$. Ke změně by mělo dojít: [1]

- vždy, když čítač rámců některého ze zařízení přesáhne hodnotu 2^{31}
- alespoň jednou za rok
- nejvýše jednou za 30 dní
- když hodnota čítače odchozích rámců trust centra dosáhne hodnoty 2^{30} (pokud trust center nemá hodiny reálného času)

4.2.5 Připojení do sítě

Proces připojování zařízení do sítě se napříč revizemi mění a vylepšuje. Protože ale není možné zaručit, že používaná zařízení jsou v souladu s nejnovější (23.) revizí, která byla vydána v březnu 2023, je potřeba uvést i starší způsoby připojování.

Do 20. revize protokolu, vypadal postup připojení následovně:

1. zařízení se připojí k routeru na MAC vrstvě
2. router informuje trust center o novém zařízení (update device)
3. trust center přes router pošle zařízení síťový klíč (transport key)
4. zařízení vyšle broadcastem oznámení o svých schopnostech (ZDO vrstva)

Ve 21. (srpen 2015) a 22. (březen 2017) revizi přibyla nutnost obdržet po připojení k síti nový trust center link klíč. Kroky 1–4 zůstaly stejné, jako tomu bylo do 20. revize, a byly přidány následující:

5. zařízení požádá trust center o informace o sobě (ZDO vrstva)
6. trust center vyšle informace o sobě (ZDO vrstva)
7. zařízení zašle žádost o trust center link key (request key)
8. trust center zašle trust center link key (transport key)
9. zařízení zašle zprávu o potvrzení klíče (verify key)
10. trust center potvrdí správnost klíče (confirm key)

Kroky 6 a 7 slouží k určení, zda trust center odpovídá alespoň 21. revizi.

Zároveň ale přibyla možnost rychlého připojení do distribuované sítě. Tato možnost vypadá stejně jako připojení do 20. revize s tím, že router je zároveň trust center. Nedochází zde tedy k aktualizaci link klíče.

[3] Tento mechanismus by měl být použit i pro odebráni zařízení ze sítě.

Ve 23. revizi přibyla možnost vyjednávání o dynamickém klíči. Tento mechanismus využívá buď algoritmus ECDHE nebo SPEKE. Postup připojení je tedy následující:

1. zařízení se připojí k routeru na síťové vrstvě
2. router informuje trust center o novém zařízení (update device)
3. trust center vyšle žádost zařízení o aktualizaci klíče (ZDO vrstva)
4. zařízení potvrdí žádost a začne vyjednávání o klíči (ZDO vrstva)
5. trust center vyšle odpověď na vyjednávání o klíči (ZDO vrstva)
6. zařízení zašle zprávu o potvrzení klíče (verify key)
7. trust center potvrdí správnost klíče (confirm key)
8. trust center pošle zařízení síťový klíč (transport key)
9. zařízení vyšle broadcastem oznámení o svých schopnostech (ZDO vrstva)
10. zařízení zažádá trust center o autentizační token (ZDO vrstva)
11. trust center zašle zařízení autentizační token (ZDO vrstva)

V krocích 3–5 dochází k vyjednávání pomocí ECDHE či SPEKE. Kroky 10 a 11 slouží k aktualizaci link klíče. Tu není nutné provést ihned po připojení.

Pokud připojení probíhá přes router, který nepodporuje revizi 23, dojde nejprve k přenosu síťového klíče a až následně dojde k vyjednávání o klíči (mezi zařízením a trust centrem na přímo). [1]

4.3 Typy útoků

Aby bylo možné zkoumat bezpečnost protokolu, je nejdříve potřeba definovat několik různých typů útoků.

4.3.1 Odposlouchávání sítě

Jednou z běžných zranitelností bezdrátových protokolů je odposlouchávání sítě. Útočník může snadno zachytávat komunikaci probíhající mezi zařízeními na určité frekvenci [7]. Pokud komunikace není chráněna šifrováním, odhaluje útočníkovi citlivé informace. Ten může zjistit např. která zařízení člověk používá, či hodnoty jednotlivých čidel.

4.3.2 Útok přehráním

Replay attack neboli útok přehráním je útok, který útočníkovi umožní ovládat a narušovat síť. Útočník při tomto útoku vyšle dříve odposlechnuté zprávy, čímž se vydává za legitimní zařízení. V případě špatné bezpečnosti na tyto zprávy cílová zařízení reagují stejně, jako by zprávy skutečně vyslalo legitimní zařízení. To může útočníkovi umožnit ovládat síť a to i v případě, kdy není schopen komunikaci dešifrovat. [8]

4.3.3 Rušení sítě

Rušení sítě je primitivní útok, proti kterému se těžko brání. Jedná se o typ *Denial of Service* (DoS, „odepření služby“) útoku, kdy útočník různými způsoby může zamezit legitimním zařízením, aby spolu komunikovala. [9]

4.4 Slabá místa

Žádný bezdrátový protokol není bez chyb, zvlášť co se bezpečnosti týká. Jedním z důvodů je, že při zvyšování bezpečnosti dochází ke snižování praktické použitelnosti. Důležitým faktorem je též cena, protože bezpečnější řešení jsou obecně dražší. Při tvorbě protokolu musí tedy dojít k jistým kompromisům.

4.4.1 Slabá místa protokolu IEEE 802.15.4

Pro analýzu slabých míst je potřeba začít u nižších vrstev, které mají možnost ochránit vrstvy vyšší.

4.4.1.1 Ochrana proti přehrání

V módu TSCH sice není 100% ochrana proti přehrání [4], nicméně tento nedostatek nahrazuje specifikace ZigBee, která již čítač rámců používá vždy. [1]

4.4.1.2 Volitelná bezpečnost

IEEE standard definuje, že zabezpečení MAC vrstvy je volitelné [4] a protokol ZigBee tento nedostatek nijak nedoplňuje. MAC vrstvu je dobré zabezpečit z důvodu skrytí ZigBee komunikace. Když odposlouchávající útočník vidí, že se jedná o ZigBee komunikaci, může rovnou provádět útoky cílené na ZigBee. Také může například provádět enumeraci zařízení a zjistit například, které zařízení je koordinátor. Dalším důvodem pro zabezpečení MAC vrstvy je tzv. „cibulový přístup“, který říká, že je nejlepší zabezpečit co nejvíce vrstev, aby se potenciálnímu útočníkovi snížily šance na úspěch o dešifrování celé komunikace. [10]

4.4.1.3 Znovupoužití nonce

Nonce je číslo, které se používá při šifrování a které by při každém šifrování mělo být jiné. Jeho účel je takový, že pokud dvě stejné zprávy zašifrujeme stejným klíčem, ale použijeme různý nonce, bude výsledný šifrový text odlišný. Pokud ovšem použijeme jeden nonce více než jednou, je pro útočníka možné získat informace o původních zprávách, ne-li původní zprávy jako takové. [11]

Pokud MAC vrstva běží v TSCH módu, nepoužívá čítač rámců, ale čítač časových úseků (ASN). Nonce pro šifrování se skládá ze zdrojové adresy a právě ASN. Znamená to tedy, že všechny zprávy poslané jedním zařízením v jednom časovém úseku používají stejný nonce. Toto vede k výše popsané zranitelnosti. [1]

4.4.1.4 Rušení sítě

Jako každý bezdrátový protokol, je i tento náchylný na rušení.

Jedním ze způsobů je rušení rádiem, kdy útočník vysíláním zvýší množství šumu na komunikačním médiu. To vyústí ve špatnou kvalitu přenosu mezi zařízeními.

Jiná verze tohoto útoku útočí na algoritmus CSMA-CA použitý MAC vrstvou. Většina zařízení (v závislosti na použitém přístupovém algoritmu) vyšle zprávu pouze pokud nikdo jiný na komunikačním médiu aktuálně nevysílá. Pokud tedy útočník začne posílat velké množství zpráv, zařízení si mezou sebou nikdy nevymění zprávy a dojde tak k DoS. [12]

4.4.2 Slabá místa protokolu ZigBee

Jak je vidět, protokol ZigBee je schopen nahradit některé nedostatky protokolu IEEE 802.15.4. Přesto sám trpí vlastními nedostatky.

4.4.2.1 Hardwarová bezpečnost

Jak autoři protokolu sami připouštějí, pokud má člověk fyzický přístup k zařízení, nelze zaručit bezpečnost vůči neoprávněné manipulaci.

„Vzhledem k nízké ceně ad hoc síťových zařízení nelze obecně předpokládat dostupnost hardwaru odolného proti neoprávněné manipulaci. Proto fyzický přístup k zařízení MŮŽE umožnit přístup k tajnému klíčovému materiálu a dalším privilegovaným informacím, stejně jako přístup k bezpečnostnímu softwaru a hardwaru.“ [1, přeloženo autorem]

4.4.2.2 Otevřený model důvěry

Otevřený model důvěry, popsaný v kapitole 4.2 znamená, že v případě kompromitace jedné části zařízení – například konkrétní aplikace – je potenciálně možné získat přístup ke klíčovému materiálu a jiným tajným informacím.

4.4.2.3 Změna síťového klíče

Změna klíče pomocí switch key je důležitým bezpečnostním mechanismem. Když čítač některého ze zařízení dosáhne maximální hodnoty $(2^{32} - 1)$, přestane zařízení přijímat a posílat zprávy, a tím se stane nefunkční. Tomu dokáže tento mechanismus předcházet.

Nejdůležitějším pravidlem pro změnu klíče je ovšem to, že by se měl klíč měnit alespoň jednou za rok. V případě kompromitace jednoho klíče totiž nedojde ke kompromitaci zpráv zašifrovaných jiným klíčem.

Pravidlo o změně klíče nejvýše jednou za 30 dní je ovšem neméně důležité. V případě časté změny klíče se narušuje funkčnost sítě. Zařízení, která během změny klíče byla v režimu nízké spotřeby, změnu klíče nezaznamenají, a tak po probuzení nejsou schopny rovnou komunikovat. Dalším aspektem je, že častá změna znamená velké množství klíčů, které musejí být rozeslány. Kromě hardwarové náročnosti na správu klíčů se tím i zvyšuje riziko zachycení nějakého klíče útočníkem.

Z důvodu hladkého přechodu na nový klíč se zařízením dovoluje po změně klíče přijímat zprávy zašifrované starým klíčem. Toto potenciálně umožní zařízení, které má starý klíč, ale neobdrželo nový, rozesílat zprávy a ovládat síť. Doporučuje se tedy změnu klíče provést dvojím odesláním příkazu switch key, čímž dojde ke zneplatnění starého klíče.

Z těchto důvodů je důležité dodržovat stanovené zásady o změně klíče.

4.4.2.4 Připojení do sítě

Při připojování nového zařízení do sítě musí zařízení obdržet zprávou transport key síťový klíč. Největší bezpečnostní slabinou je povolení posílání nezašifrovaného klíče. Útočník si poté může snadno síťový klíč přečíst a celá síť je kompromitována. Alternativou k tomuto způsobu je zašifrování síťového klíče link klíčem.

V případě, že zařízení pro přenos využijí dobře známý link klíč, nedojde sice k přímému odhalení hodnoty síťového klíče, ovšem útočník si jeho hodnotu snadno dešifruje.

Zařízení by pro bezpečné předání síťového klíče měla využít buď link klíč odvozený z instalačního kódu, či link klíč vygenerovaný jednou z dynamických metod. Problém protokolu je, že nelze vynutit použití pouze těchto dvou způsobů, protože pokud je trust center nastavené tak, že vyžaduje přednastavený klíč, neumožní novým zařízením využít bezpečné dynamické vyjednávání o klíči. Vynucení jednoho z těchto mechanismů je tedy na implementaci vyšších vrstev.

4.4.2.5 Známé klíče

Jak již bylo zmíněno, klíče jsou základním kamenem bezpečnosti. Je proto naprosto nutné, aby byly uchovány v tajnosti. V případě tajného klíče by pro dešifrování komunikace musel útoč-

ník vyzkoušet všechny možné klíče[4]. Pro usnadnění útoku může ovšem útočník vyzkoušet, zda komunikace k šifrování nepoužívá dobře známé klíče.

Prvním takovým klíčem je výchozí link klíč pro připojení zařízení do sítě. Byť se jedná o klíč definovaný ve standardu, jeho použitím je celá síť vystavena nebezpečí.

Druhým dobře známým klíčem je klíč `01030507090B0D0F00020406080A0C0D`, který je ve spoustě zařízení přednastavený jako výchozí síťový klíč. Použitím tohoto klíče se provinily i výrobky od značky Homey. Tato zranitelnost byla zaevidována do seznamu *Common Vulnerabilities and Exposures* (CVE)[5].

Za zmínku též stojí síťový klíč `07030507090B0D0F00020406080B0C0D`, který je využit stránkou ZigBee2MQTT v návodu na změnu klíče [13]. Na stránce je sice tučným písmem napsáno, že uživatelé nemají používat přesně tento klíč, ale s dodržováním této rady nelze 100% počítat.

[4]Těch je při použití 128bitového klíče $2^{128} \approx 3{,}4 * 10^{38}$.
[5]CVE-2020-28952

Kapitola 5
Aktuální řešení

Tato kapitola popisuje aktuální dostupná řešení pro tvorbu domácích sítí a jejich analýzu a diskutuje jejich pro a proti.

5.1 Domácí síť

Systémů, pomocí kterých je možné ovládat chytrá zařízení, je spoustu. Ty kromě jednoduchého ovládání umožňují i nastavení různých automatizací (např. rozsvícení světla při západu slunce).

Většina výrobců chytrých ZigBee zařízení mají i vlastní tzv. huby. Ty v síti plní roli koordinátora, který většinou funguje tak, že data ze zařízení posílá na servery společnosti. Uživatel potom může zařízení ovládat přes mobilní aplikaci, která se servery komunikuje. Toto je pro spoustu lidí nejjednodušší řešení, které rychle umožní uživateli ovládat svá zařízení i na dálku. Jednou z výhod je též častá podpora propojení s Google Home či Home od Applu. Nevýhoda je ovšem uzavřenost systému – nelze zaručit kompatibilitu s různými zařízeními. Dále se uživatel vystavuje potenciálním zranitelnostem při ovládání přes cizí server. [14]

Alternativou jsou různé nástroje, které si musí uživatel zprovoznit sám. Nejrozšířenějším takovýmto nástrojem je *Home Assistant*, existuje ale spoustu dalších alternativ, jako např. *openHAB*, či *ioBroker*. Tato řešení jsou pracnější, ale dávají uživateli plnou kontrolu nad jeho zařízeními.

Home Assistant je možné získat ve 4 variantách: *OS*, *container*, *core* a *supervised*. Všechny funkcionality, které Home Assistant nabízí jsou dostupné pouze ve variantě OS, která se nejčastěji instaluje na Raspberry Pi. To potom tvoří jádro chytré domácnosti. [15]

Home Assistant umožňuje ovládání a nejrůznější automatizace zařízení využívajících nejrůznější protokoly. Pro připojení ZigBee zařízení do Home Assistant existují dva způsoby: *ZigBee Home Automation* (ZHA) a *ZigBee2MQTT*. Není jednoznačné, který ze způsobů je lepší. Z diskuzních fór většinou vyplývá, že ZHA je jednodušší na zprovoznění (je zabudované přímo v Home Assistant), ale ZigBee2MQTT podporuje více zařízení a lépe se používá [16]. Zatímco ZHA umožňuje přímou komunikaci s Home Assistant, ZigBee2MQTT funguje tak, že ZigBee komunikaci koordinátora překládá do MQTT[1] a posílá ji na MQTT broker („zprostředkovatele“). Home Assistant si poté pomocí jeho MQTT podpory odebírá zprávy z MQTT brokeru. [17]

5.2 Analýza sítě

Bezpečností protokolu ZigBee se zabývalo již více lidí a tak vznikly i nástroje pro analýzu sítí a zařízení využívajících tento protokol.

[1] MQTT je podobně jako ZigBee také komunikační protokol pro IoT.

5.2.1 KillerBee

Nejkomplexnějším řešením dostupným pro bezpečnostní analýzu ZigBee je projekt KillerBee vyvinutý River Loop Security.

Jedná se o framework pro psaní nástrojů na analýzu či útoky na ZigBee zařízení. Poskytuje tedy různé funkce načítání, dešifrování, úpravu a posílání ZigBee rámců, kterých následně mohou konečné nástroje využívat. Součástí frameworku je i několik připravených nástrojů, kterých může uživatel využít.

KillerBee také poskytuje podporu pro více typů zařízení, ze kterých ovšem značná část buď nepodporuje plný rozsah možností frameworku, nebo jsou obtížné k sehnání.

Dalším problémem je složitost – uživatel si musí být schopen vybrat, který jednoúčelový nástroj použije. To znamená, že uživatel musí projít dostupné nástroje a porozumět tomu, které potřebuje. Mezi nabízenými nástroji však chybí testovací nástroj, který by vyzkoušel více/všechny zranitelnosti najednou. Takový nástroj by si musel uživatel napsat sám, což snižuje jeho použitelnost širší veřejností.

Projekt KillerBee také není tak často aktualizován – nejnovější vydání vyšlo v červenci 2021. Toto je problém zejména v tak rozrůstajícím odvětví jako je Internet věcí. [18]

5.2.2 SecBee

SecBee je nástroj založený na KillerBee vyvinut firmou Cognosec pro analýzu pomocí softwarově definovaného rádia (SDR). Ke komunikaci s SDR využívá projekt Scapy-radio a GNU Radio block. Projekt byl ovšem naposledy aktualizován v roce 2016 a ukončen byl v roce 2018. Navíc stránka projektu nepopisuje, jak by se měl nástroj používat. [19]

5.2.3 Zigtoucher

Za zmínku též stojí aplikace *Zigtoucher* vyvinutá Jakubem Šatopletem v rámci jeho bakalářské práce *Bezpečnostní analýza protokolu ZigBee Touchlink*. Tato aplikace umožňuje uživatelům provádět útoky na zařízení využívající ZigBee Touchlink. Zigtoucher je z velké části napsán v jazyce Python s využitím knihovny Scapy. Pro odposlouchávání a komunikaci se zařízeními využívá aplikace SDR. [20]

Aplikace je ovšem omezená na protokol ZigBee Touchlink – funkce protokolu ZigBee, která umožňuje zařízením komunikovat na přímo mezi sebou, aniž by byly ve stejné síti. [21]

Kapitola 6
Sestavení sítě

Tato kapitola se zabývá postupem sestavení jednoduché testovací sítě. Popisuje jednotlivá zařízení, využité technologie a postup vytvoření sítě. Nakonec vysvětluje postup nastavení ovládání sítě a její automatizace.

6.1 Zařízení

Hlavní částí každé sítě je koordinátor. Pro tuto práci byl využit koordinátor s čipem CC2652P od Texas Instruments (obrázek 6.1). Jedná se o čip s podporou různých protokolů: ZigBee, Bluetooth 5.2 Low Energy, IEEE 802.15.4, 6LoWPAN aj. Senzitivita přijímače pro IEEE 802.15.4 (2,4 GHz) je −100 dBm a výstupní výkon až 20 dBm. [22]

Dalším využitým zařízením je žárovka Philips hue white ambiance (obrázek 6.2). Ta nabízí nastavení jasu a teploty barvy v rozsahu 2200–6500 K [23]. Z důvodu absence adaptéru na patici E14 bylo potřeba pořídit jednoduchou redukci na E27.

Posledním zařízením je vypínač Immax NEO Smart (s jedním tlačítkem, obrázek 6.3), který dává možnost zachytit stisknutí, dvojité stisknutí a držení tlačítka. [24]

6.2 Ovládací systém

Testovací síť v této práci využívá Home Assistant ve verzi container, která běží v docker kontejneru. Pro propojení ZigBee zařízení s Home Assistant byla z důvodu větší online podpory

Obrázek 6.1 Koordinátor CC2652P

Obrázek 6.2 Žárovka Philips hue white ambiance

Obrázek 6.3 Vypínač Immax NEO Smart

zvolena aplikace ZigBee2MQTT a open-sourcový MQTT broker Mosquitto.

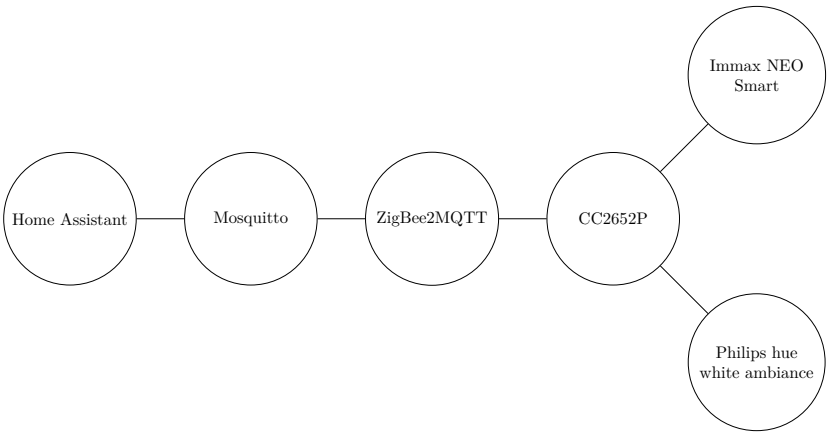
ZigBee2MQTT i MQTT broker existují ve formě add-onů („rozšíření“) do Home Assistant, která se do něj jednoduše připojí. Add-ony ovšem nejsou v container verzi Home Assistant k dispozici, a tak bylo také využito jejich docker container verze. Po propojení všech docker containerů bylo ještě využito Home Assistant iframů k přidání ZigBee2MQTT do navigační lišty Home Assistant. Soubor docker compose s nastavením containerů je v příloze A.1.

6.3 Připojení zařízení

Do čipu CC2652P byl nejprve nahrán pomocí FLASH-PROGRAMMER-2 [25] firmware [26], který z čipu dělá ZigBee koordinátora.

Po zapnutí aplikace ZigBee2MQTT je možné připojený koordinátor ovládat, měnit nastavení sítě, spravovat zařízení aj. Po povolení připojování zařízení zbývalo jen spárovat spínač a žárovku.

Každé koncové zařízení má svůj způsob, kterým se dostane do párovacího módu. V případě spínače to bylo dlouhé podržení tlačítka, u žárovky se párovací mód spustil pomocí mobilní aplikace. Po spuštění párovacího módu se zařízení sama připojila ke koordinátorovi, zobrazila se v ZigBee2MQTT a následně i v Home Assistant.

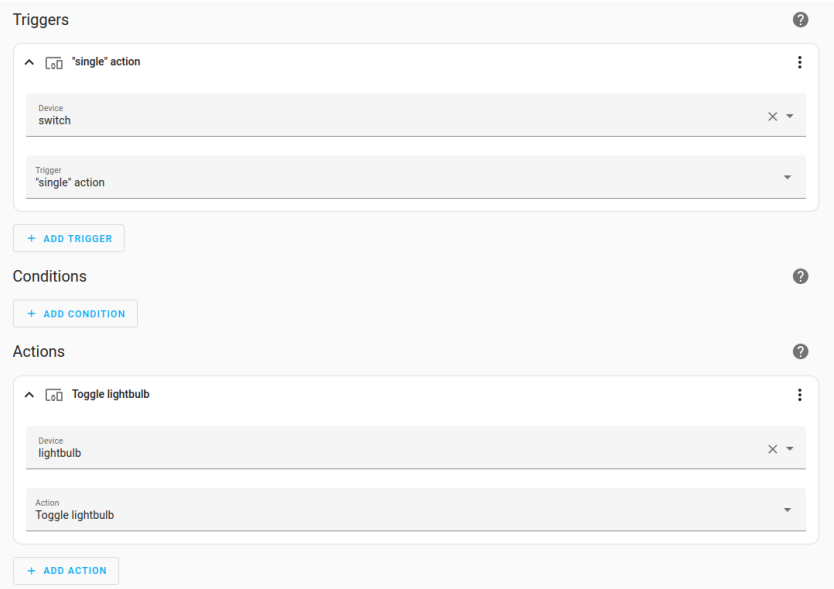


Obrázek 6.4 Schéma propojení zařízení a ovládacího systému

6.4 Ovládání a automatizace

Posledním krokem ke zprovoznění testovací sítě je nastavení ovládání a automatizace zařízení v Home Assistant. Ten umožňuje vytváření automatizací pomocí grafického i textového rozhraní[1]. Grafické prostředí je jednoduché na použití i pro nezkušeného uživatele, naopak zkušenější uživatelé uvítají větší volnost textového prostředí. Prvním nastavením bylo zapínání/vypínání žárovky při stisku spínače, na kterém jsou na obrázku 6.5 a výpisu kódu 6.1 demonstrována jednotlivá rozhraní. Dále bylo podobným způsobem nastaveno zvýšení jasu žárovky při dvojím stisknutí spínače a snížení jasu žárovky při podržení spínače.

Nakonec byla vytvořena automatizace, která rozsvítí žárovku při západu slunce.

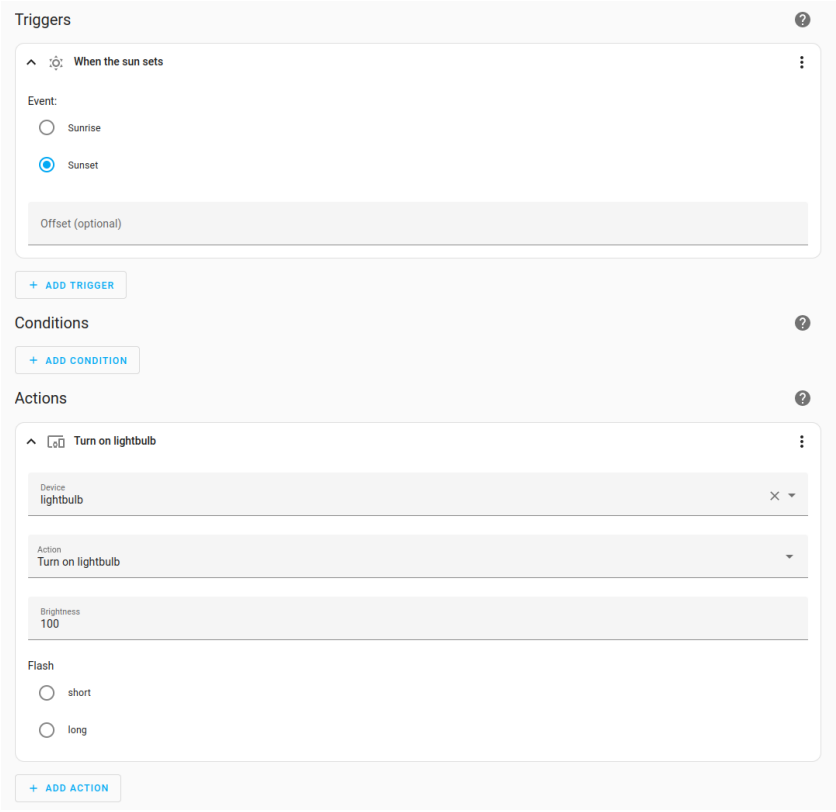


Obrázek 6.5 Nastavení zapínání a vypínání žárovky v grafickém rozhraní

Výpis kódu 6.1 Nastavení zapínání a vypínání žárovky v jazyce YAML

```
alias: light toggle
description: ""
trigger:
  - platform: device
    domain: mqtt
    device_id: 4aca62dd912c40c60f77df4c305d40c4
    type: action
    subtype: single
    discovery_id: 0x60a423fffef8a1ce action_single
condition: []
action:
  - type: toggle
    device_id: de779ec64480670029b1090ab3908ed7
    entity_id: light.lightbulb
    domain: light
mode: single
```

[1] Textové rozhraní je psáno v YAML.



Obrázek 6.6 Nastavení automatického zapnutí žárovky při západu slunce

Kapitola 7

Aplikace pro bezpečnostní analýzu

V této kapitole se nachází definice cílů aplikace spolu se zvolenými technologiemi. Je zde popsán postup zprovoznění odposlouchávání pomocí koordinátoru CC2531, návrh aplikace a nakonec jsou popsány jednotlivé části výsledné aplikace.

7.1 Cíle aplikace

Hlavním cílem aplikace, vyvinuté v rámci této práce, je aplikace umožňující bezpečnostní analýzu. Nicméně důležité jsou i další cíle aplikace:

Prvním takovým je **jednoduchost**. Je žádoucí, aby základní použití aplikace bylo co nejjednodušší, aby i běžný uživatel měl možnost zvýšení bezpečnosti své sítě.

Dalším důležitým cílem je **rozšiřitelnost**. Aplikaci by mělo být snadné rozšiřovat jak o kontroly nových zranitelností, tak i o podporu dalších zařízení potřebných pro analýzu.

Posledním stanoveným cílem je **cena** – hardware potřebný k provedení analýzy by měl být co nejlépe cenově dostupný. Cenová dostupnost spolu s jednoduchostí použití by měla zvýšit dostupnost aplikace pro běžné koncové uživatele.

7.2 Technické parametry

Vzhledem ke stanoveným cílům byla provedena následující rozhodnutí. Bude se jednat o aplikaci pro příkazovou řádku napsanou v jazyce Python, který je rozšířený a jednoduchý. Pro rozhraní příkazové řádky bude využita snadno použitelná knihovna Click [27]. Python pomocí knihovny Scapy [28] navíc umožňuje jednoduchou manipulaci s rámci protokolu ZigBee / IEEE 802.15.4.

S ohledem na cenu bylo zvoleno použití snadno dostupného koordinátoru CC2531 (obrázek 7.1), které umožňuje pouze odposlouchávání komunikace, nikoli její vysílání. K odposlouchávání je využito firmwaru obsaženém v nástroji PACKET-SNIFFER [29] od Texas Instruments a projektu whsniff [30].

Aplikaci by mělo být možné spustit na distribuci Ubuntu 22.04 LTS operačního systému GNU/Linux. To by mělo zajistit kompatibilitu i s dalšími aktuálními distribucemi.

7.3 Postup vývoje

Ze všeho nejdříve bylo potřeba aplikaci vymyslet název, kterým se stal *ZigCheck*.

Prvním úkolem opravdového vývoje bylo zprovoznit odposlouchávání sítě pomocí koordinátoru CC2531. K tomu bylo využito návodu na stránkách ZigBee2MQTT [31]. Nejdříve byl stažen firmware pro koordinátor obsažený v nástroji PACKET-SNIFFER [29]. Ten následně bylo nutné nahrát na CC2531. K tomu opět posloužil návod od ZigBee2MQTT dostupný z [32], resp. [33].

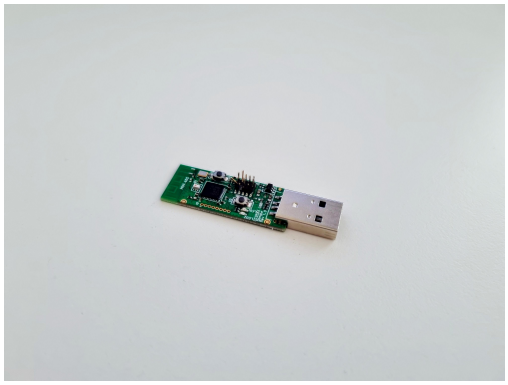
Způsobů nahrání zmíněného firmwaru je více. Nejjednodušší způsob je pomocí CC debuggeru a kabelu k CC2531 [32]. Nicméně vzhledem k dostupnosti mikrokontroleru Arduino Uno bylo využito alternativního způsobu využívajícího právě tento mikrokontroler a nástroj CCLoader [34]. [33]

CCLoader byl tedy nejprve stažen a následně zkompilován. Jeho firmware pro Arduino do něj byl nahrán pomocí Arudino IDE [35]. Firmware určený k nahrání na CC2531 byl převeden z hex formátu na formát binární.

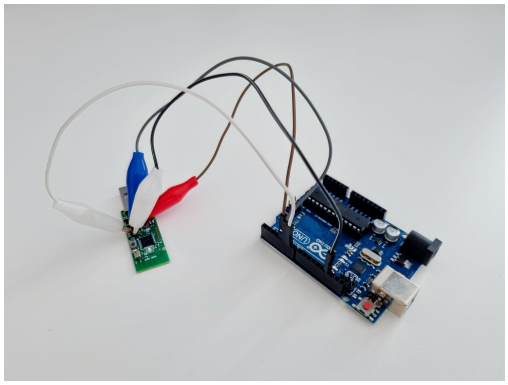
Pomocí vodičů bylo Arduino Uno propojeno s CC2531 způsobem popsaným v tabulce 7.1 a vyobrazeným na obrázku 7.2.

Tabulka 7.1 Propojení mikrokontroleru Arduino Uno a čipu CC2531

Arduino pin	CC2531 pin
GND	GND (1)
D4	RESETn (7)
D5	Debug Clock (3)
D6	Debug Data (4)



Obrázek 7.1 Koordinátor CC2531



Obrázek 7.2 Propojení mikrokontroleru Arduino Uno a čipu CC2531

Následně bylo potřeba zapojit do počítače Arduino a CC2531 a spustit CCLoader, který do CC2531 nahrál žádoucí firmware.

Whsniff byl nainstalován pomocí návodu na stránce nástroje. Po spuštění vypisuje nástroj zachycené rámce ve formátu PCAP na standardní výstup. Výstup nástroje je tedy například možné přesměrovat a uložit do souboru či přesměrovat do nástroje Wireshark – nástroje pro zkoumání a analýzu síťových packetů. Aplikace vzniklá v této práci využije rámce zachycené nástrojem whsniff, které bude dále analyzovat.

7.3.1 Návrh aplikace

Jádrem aplikace je třída `TestSuite`, která reprezentuje testovací prostředí. Jejím úkolem je spravovat všechny testy a předávat jim rámce zachycené zařízením.

Testy představují třídy implementující abstraktní třídu `Tester`, která definuje rozhraní, které se očekává od jednotlivých testů. Testy fungují tak, že postupně přijímají zachycené rámce, testují

je a nakonec vypíší zprávu s výsledkem testu.

Mezi testy je zahrnuta i pseudo-testovací třída, která při výpisu hlášení vypíše obsah jednotlivých přijatých rámců.

Zařízení jsou v aplikaci reprezentována třídami, které implementují abstraktní třídu `Device`. Tato abstraktní třída definuje, jaké funkcionality se očekávají od všech zařízení. Tím ulehčí přidávání podpory pro nová zařízení. Účelem tříd reprezentujících zařízení je číst rámce zachycené zařízením fyzickým a předávat je třídě `TestSuite`.

V rámci této práce aplikace podporuje jen koordinátor CC2531. Pro přístupnost aplikace navíc bylo přidáno tzv. offline zařízení – třída, která implementuje třídu `Device` a umožňuje číst data ze souborů typu PCAP.

Vytvořeny také byly 2 pomocné třídy: `Decrypter` a `ConfigReader`. První z těchto tříd slouží k dešifrování předaného rámce předaným klíčem. Druhá ze tříd slouží k načtení konfigurace z předaného souboru a její případné úpravě.

Abstraktní třídy umožňují definovat abstraktní metody, které zaručí správné fungování aplikace a snadné začlenění nových zařízení/testů do aplikace. Abstraktní třídy v této aplikaci jsou vytvořeny pomocí Python modulu `abc` (Abstract base Classes).

Aplikace se spouští zavoláním funkce `scan` ze souboru main.py, která se stará o zpracování přepínačů z příkazové řádky, načtení konfigurace a spuštění třídy `TestSuite`.

Adresářová struktura projektu vypadá následovně:

```
main.py ..... bod spuštění aplikace
testsuite.py ..... soubor obsahující definici třídy TestSuite
config.yml ..... konfigurační soubor
testers\ ..... složka obsahující jednotlivé testy
    tester.py ..... soubor obsahující definici abstraktní třídy Tester
devices\ ..... složka obsahující jednotlivé zařízení
    device.py ..... soubor obsahující definici abstraktní třídy Device
tools\ ..... složka obsahující podpůrné třídy
```

7.3.2 Konfigurace

Pro provedení analýzy je nutné nastavení různých hodnot. Jednou z nich jsou například testy, které se mají provést, které zařízení se má využít k odposlechu nebo jak dlouho má testování probíhat. Protože je žádoucí, aby si tyto hodnoty mohl upravovat uživatel bez zásahu do zdrojového kódu, je potřeba tyto hodnoty vyčlenit do konfiguračního souboru, který bude obsahovat hodnoty pro konkrétní klíče. Pro konfigurační soubory je využíváno více typů souborů, avšak nejrozšířenějšími jsou JSON a YAML. Práce z důvodu lepší přehlednosti pro nezkušené uživatele a snadnosti použití v jazyku Python využívá YAML.

Výpis kódu 7.1 Ukázkový konfigurační soubor v jazyce YAML

```
tests:
  - MAC_SECURITY
  - NONCE_REUSE
  - TEST_KEY
  - SWITCH_KEY
  - TRANSPORT_KEY
print: off
device: CC2531
stop_after: 60
channel: 11
pan_id: 6754
devices:
  Offline:
    file: capture.pcap
testers:
```

```
  MAC_SECURITY:
    status_threshold: 0.5
  PRINTER:
    print_ieee: off
```

Na výpisu kódu 7.1 je ukázka konfiguračního souboru v jazyce YAML. Tento konfigurační soubor definuje například následující klíče a jejich hodnoty: hodnotou klíče `tests` je seznam obsahující `MAC_SECURITY`, `NONCE_REUSE`, `TEST_KEY`, `SWITCH_KEY` a `TRANSPORT_KEY`; klíč `print` má hodnotu `off` a klíč `devices` obsahuje klíč `Offline`, který obsahuje klíč `file`, jehož hodnota je `capture.pcap`.

Aplikace počítá s konfiguračním souborem `config.yml`, který se nachází ve stejné složce jako program. Protože může uživatel potřebovat přepínat mezi více konfiguracemi a tedy i více konfiguračními soubory, podporuje aplikace možnost načtení jiného konfiguračního souboru.

Občas je aplikaci potřeba spustit jednorázově s upravenou konfigurací. V takovém případě by bylo zbytečné upravovat konfigurační soubor, který by po jednom spuštění bylo potřeba vrátit do původního stavu. Aplikace tedy umožňuje pomocí parametrů na příkazové řádce jednorázově konfigurační hodnoty upravit.

7.3.3 Command line interface

Pro vytvoření *command line interface* (CLI, „rozhraní příkazové řádky“) byla využita Python knihovna Click. Knihovna Click umožňuje jednoduše zpracovávat přepínače z příkazové řádky, kontrolovat jejich hodnoty nebo například automaticky generovat nápovědu.

Cílem rozhraní je umožnit uživateli jednoduchým způsobem jednorázově upravit konfiguraci aplikace. Hodnoty, které je možné přímo změnit, jsou konfigurační soubor, seznam testů, zapnutí/vypnutí výpisu rámců, zařízení pro odposlouchávání, PAN ID, doba trvání a kanál. Pokud uživatel potřebuje upravit jinou z možností, může využít přepínače `--set`, který mu umožní změnit jakoukoli hodnotu konfigurace.

Výpis kódu 7.2 Ukázka spuštění aplikace s přepínači

```
zigcheck --tests MAC_SECURITY,NONCE_REUSE \
         --set devices:Offline:file capture.pcap
```

Na příkazu kódu 7.2 je vidět příkaz spuštění aplikace s úpravou výchozí konfigurace. Na příkladu je množina testů nastavena na kontrolu šifrování na MAC vrstvě a kontrolu použití ASN v nonce a pomocí přepínače `--set` je nastavena hodnota klíče file pro pseudo-zařízení Offline na `capture.pcap`.

7.3.4 TestSuite

Jak již bylo řečeno, tato třída reprezentuje testovací prostředí.

Třída obsahuje definici řetězců odpovídajících jednotlivým testům a názvu jejich příslušných tříd. Ty jsou navíc v konkrétním pořadí, aby například došlo k dešifrování rámce dříve, než k dalším testům. V konstruktoru třída projde všechny předané názvy testů, zkontroluje, zda jsou validní, a vytvoří instance odpovídajících tříd. K těmto třídám je případně (dle vstupního parametru `printer`) přidána pseudo-testovací třída `Printer` vypisující zachycené rámce.

Dle předaného názvu zařízení k odposlouchávání je také vytvořena instance stejnojmenné třídy.

`TestSuite` má též 3 další metody: `start`, `stop` a `handle_packet`. Metoda `start` spustí odposlouchávání zařízení, `stop` odposlouchávání vypne a vypíše hlášení o jednotlivých testech. Pomocí metody `handle_packet` přijímá `TestSuite` rámce zachycené zařízením a následně je posílá jednotlivým testům.

Protože odposlouchávající zařízení může zachytit komunikaci cizích sítí v blízkosti, umožňuje aplikace specifikováním konkrétního PAN ID omezit testování pouze na zamýšlenou síť. O toto se stará třída `TestSuite` v metodě `handle_packet`.

7.3.5 Testy

Testy jsou nejdůležitější částí aplikace. Každý test je reprezentován vlastní třídou, která se stará o testování rámců a zpracování výsledků.

7.3.5.1 Tester

`Tester` je abstraktní třída, jejíž abstraktní metody by měly implementovat všechny její podtřídy. Abstraktní metody definované touto třídou jsou `__init__`, `recv` a `print_report`.

`__init__` přijímá jako parametr aktuální konfiguraci, `recv` slouží k přijetí a zpracování nových rámců a `print_report` je použito pro výpis hlášení o výsledku testu.

Kromě abstraktních metod ale také třída poskytuje funkci pro vypsání výsledku testu jednotným formátem. Metoda při výpisu barevně indikuje výsledek testu (dobrý/neutrální/špatný), který je doplněn o podrobnější zprávu.

7.3.5.2 MacSecurity

Tento test kontroluje použití šifrování na MAC vrstvě. Pro zapnutí tohoto testu je v konfiguraci potřeba uvést `MAC_SECURITY`.

Po přijetí rámce se třída podívá na hodnotu *security* bitu obsaženého ve *frame control* části MAC hlavičky. Protože knihovna Scapy při indikaci zabezpečeného rámce nezobrazuje bezpečnostní hlavičku, přidává v tomto případě třída bezpečnostní hlavičku sama. Následně je z bezpečnostní hlavičky přečtena bezpečnostní úroveň rámce. Třída poté řadí pakety do 3 skupin: nezabezpečené (*security* bit je nastaven na 0, či bezpečnostní úroveň je 0), zabezpečené pouze kontrolou integrity (bezpečnostní úroveň je v rozsahu 1–4) a zašifrované (bezpečnostní úroveň je v rozsahu 5–8).

Při vyhodnocování výsledku zkoumá třída poměr počtu zašifrovaných rámců k počtu všech rámců. Vyhodnocení výsledku je závislé na parametru `STATUS_THRESHOLD`, který je ve výchozím nastavení roven 0,5. Tuto hodnotu je možno pomocí konfiguračního souboru upravit. Pokud je tedy poměr zašifrovaných rámců větší než 0,5, je výsledek označen jako „dobrý“. Pokud je poměr zašifrovaných rámců a rámců pouze s kontrolou integrity větší než 0,5, je výsledek označen jako „neutrální“. V jiném případě je výsledek označen jako „špatný“. Podrobnější zpráva o výsledku testu obsahuje hodnoty všech 3 počítadel.

7.3.5.3 NonceReuse

Třída `NonceReuse` testuje, zda při šifrování na MAC vrstvě může dojít ke znovupoužití nonce a tím i tedy, zda je komunikace plně chráněna před útoky přehrání. V konfiguraci lze tento test zapnout pomocí `NONCE_REUSE`.

V případě, že třída obdrží rámec zašifrovaný na MAC vrstvě, podívá se do jeho bezpečnostní hlavičky do části *Security Control* na bit *ASN in Nonce*, který indikuje, že nonce obsahuje ASN místo čítače.

Pokud během testování nalezne třída nějaký rámec s bitem *ASN in Nonce* rovným 1, je výsledek označen jako „špatný“. Pokud nedojde k zachycení žádného zabezpečeného rámce, indikuje výsledek jako „neutrální“, protože test nebyl průkazný. Nakonec, pokud dojde k odposlechu zabezpečených rámců, ale žádný z nich nepoužil nonce s ASN, je výsledek označen jako „dobrý“.

7.3.5.4 KeyTester

Tento test slouží k testování použití dobře známých klíčů. V konfiguraci tomuto testu odpovídá `TEST_KEY`.

Třída aktuálně testuje tyto klíče: `01030507090B0D0F00020406080A0C0D` (síťový klíč přednastavený velkým množstvím aplikací), `07030507090B0D0F00020406080B0C0D` (síťový klíč z návodu od ZigBee2MQTT) a `5A6967426565416C6C69616E63653039` (protokolem definovaný výchozí link klíč určený pro přenos síťového klíče).

Pokud třída nezachytila žádné zašifrované zprávy, označí výsledek jako „neutrální“; pokud zachytila nějaké rámce využívající tyto známé klíče, označí výsledek jako „špatný“ a vypíše, které klíče byly použity. V jiném případě je výsledek označen jako „dobrý“.

7.3.5.5 TransportKey

Tato třída testuje správné použití příkazu *transport key*. Pro aktivaci tohoto testu je třeba v konfiguraci testů uvést `TRANSPORT_KEY`.

Cílem testu je zjistit, zda v případě přenosu klíče pomocí *transport key* dochází k zašifrování známým link klíčem, či zda zařízení využívají např. předinstalované kódy. Také je kontrolováno, zda je příkaz poslán jako unicast, nebo broadcast.

Výsledek je „dobrý“ v případě, že příkaz *transport key* byl zachycen, ale nepodařilo se ho dešifrovat pomocí známého link klíče. Pokud je klíč přenesen bez šifrování, pomocí známého klíče, nebo není poslán unicastem, je výsledek označen jako „špatný“; pokud nedojde k zachycení tohoto příkazu, je výsledek testu „neutrální“.

7.3.5.6 SwitchKey

`SwitchKey` je test kontrolující správné použití služby *switch key* umožňující změnu síťového klíče. Správné využití této služby je popsáno v kapitole 4.2.4.2. Test lze zapnout uvedením `SWITCH_KEY` v konfiguraci.

Mezi situacemi, ve kterých by mělo dojít ke změně klíče, je i nutnost měnit klíč alespoň jednou ročně. To je časově velmi náročný test, ovšem v definici testu je také obsažen pro jeho kompletnost.

Další situací, kterou je potřeba zmínit, je změna klíče „nejvýše jednou za 30 dní“. Test zde povoluje provedení dvou změn klíče, aby bylo možné provést změnu klíče dvakrát po sobě, a tím zajistit 100% zneplatnění starého klíče.

Výsledek je tedy „špatný“, pokud dojde k porušení nějaké ze zásad, s výjimkou porušení pravidla o změně klíče při překročení hodnoty 2^{30} čítačem. V takovém případě je výsledek označen jako „neutrální“, protože se toto pravidlo týká pouze *trust center*, která nemají hodiny reálného času. Pokud nedošlo k porušení žádného z pravidel, je stav nastaven buď jako „dobrý“ nebo „neutrální“, podle toho, zda k nějaké změně během testu vůbec došlo.

7.3.6 Zařízení

Pro získání rámců k testování využívá aplikace zařízení. Každé z nich je opět reprezentováno vlastní třídou.

7.3.6.1 Device

Podobně jako pro testy, je i pro zařízení definována jednoduchá abstraktní třída definující rozhraní, které se od nich očekává. `Device` definuje 3 metody: `__init__`, `start` a `stop`.

Metoda `__init__` (konstruktor) přijímá jako parametr `handler` – funkce, která zpracovává jednotlivé rámce – a `config` – třída reprezentující aktuální konfiguraci. Metoda `start` slouží ke spuštění odposlouchávání v novém vlákně. Třída poté při každém zachyceném rámci zavolá

`handler`, kterému zachycený rámec předá jako parametr. Metoda `stop` ukončí vytvořené vlákno a tím i odposlouchávání.

7.3.6.2 CC2531

Třída `CC2531` umožňuje aplikaci odposlouchávat ZigBee komunikaci pomocí čipu CC2531 a aplikace *whsniff*.

Proces odposlouchávání funguje následovně:

1. třída si z konfigurace aplikace načte požadovaný kanál, na kterém má odposlouchávat
2. třída spustí aplikaci *whsniff* a předá jí číslo kanálu jako parametr
3. pomocí třídy `AsyncSniffer` knihovny *Scapy* je výstup z aplikace *whsniff* čten a interpretován
4. `AsyncSniffer` předá interpretované rámce funkci `handler`, která se stará o jejich další zpracování

Při pokynu pro zastavené odposlouchávání je nejprve ukončen program *whsniff* a následně i třída `AsyncSniffer`.

Třída naráží na problém nedostatečných práv při spouštění programu whsniff – programu je zamítnut přístup k USB, na kterém se CC2531 nachází. Uživatel buď může celou aplikaci spustit s administrátorskými privilegii, ovšem tím se vystavuje nebezpečí v případě zranitelnosti v části aplikace. Je tedy doporučené, aby uživatel upravil vlastníka daného USB na aktuální uživatele. Při prvotním pokusu o spuštění vypíše aplikace příkaz, kterým toho uživatel může dosáhnout.

7.3.6.3 Offline

Toto pseudo-zařízení umožňuje analýzu dříve zachycené komunikace uložené v souboru PCAP.

Nejprve dojde k načtení názvu souboru z konfigurace a následně k přečtení souboru samotného pomocí knihovny *Scapy*. Při spuštění odposlouchávání dojde k vytvoření nového vlákna, ve kterém aplikace načte jednotlivé rámce ze souboru a postupně je předá funkci `handler`. Při ukončení odposlouchávání dojde k ukončení vlákna.

7.3.7 Pomocné třídy

Pro správné fungování aplikace byly též vytvořeny podpůrné třídy, které jsou všem třídám k dispozici.

7.3.7.1 ConfigReader

Třída `ConfigReader` byla vytvořena pro snadné načítání, úpravu a předávání konfigurace mezi jednotlivými třídami.

Třída nejdříve načte konfigurační soubor předaný v konstruktoru. Pomocí metody `get` je možné obdržet hodnotu pro jakýkoli klíč v konfiguraci. Naopak metoda `set` umožňuje změnu, či přidání nového klíče a jeho hodnoty.

7.3.7.2 Decrypter

`Decrypter` je pomocná třída umožňující dešifrování ZigBee rámců, a to jak na síťové, tak i aplikační vrstvě.

Hlavní metodou této třídy je metoda `decrypt`, kterou využívá například třída `KeyTester`. Postup dešifrování je převzat ze standardu ZigBee[1], který následuje krok po kroku. Třída se vždy snaží dešifrovat bezpečnostní hlavičku na co nejnižší vrstvě. To znamená, že v případě

šifrování na síťové i aplikační vrstvě je pro dešifrování aplikační vrstvy nutné třídě předat rámec bezpečnostní hlavičky síťové vrstvy. Navíc, protože druhá bezpečnostní hlavička často neobsahuje informaci o zdroji rámce[1], je potřeba třídě zdrojovou adresu rámce specifikovat.

7.4 Používání aplikace

Bez dalších úprav by bylo potřeba aplikaci spouštět příkazem `python3 main.py scan`, což je zdlouhavé a špatně zapamatovatelné. Do projektu byl tedy přidán soubor `setup.py`, který využívá knihovnu `setuptools` a který nastavuje informace a vstupní bod do výsledné aplikace. Přidán byl i soubor `requirements.txt`, který definuje, které balíčky aplikace vyžaduje pro její správné fungování.

Po vytvoření těchto souborů je možné aplikaci nainstalovat příkazem `pip install .`[2] Výsledkem je aplikace spustitelná jen příkazem `zigcheck`.

Aplikaci je doporučeno používat ve virtuálním prostředí Python venv, které umožňuje oddělení balíčků nainstalovaných v rámci tohoto projektu od projektů ostatních.

[1] Informace potřebná k jeho dešifrování.
[2] Pip je aplikace pro správu balíčků.

Kapitola 8
Analýza testovací sítě

Tato kapitola demonstruje použití vyvinuté aplikace na testovací síti. Diskutuje výsledek testu a možná bezpečnostní zlepšení. Po úpravě sítě je její analýza zopakována pro ověření zlepšení.

Pro analýzu sítě byla vytvořená síť, obsahující koordinátor CC2652P, žárovku Philips a vypínač Immax, otestována jako celek. Pomocí koordinátoru CC2531 upraveného pro odposlouchávání ZigBee sítí byla zachytávána veškerá komunikace mezi zařízeními na testovací síti.

Pro provedení bezpečnostní analýzy byla aplikace nakonfigurována, aby provedla všechny testy, aby testování probíhalo 5 minut a aby byla odposlouchávána komunikace na kanálu 11 sítě s PAN ID 6754. Informace o kanálu a PAN ID byly vyčteny z konfigurace ZigBee2MQTT.

Po spuštění aplikace začalo 5minutové testování sítě. Pro co nejkompletnější výsledek bylo potřeba provést co nejvíce operací se zařízeními. Všechna zařízení byla nejprve odpojena ze sítě a následně byla znovu připojena. Nakonec byla vyzkoušena základní funkcionalita (zapínání/vypínání žárovky atd.).

Výstup aplikace je možné najít v příloze B.1. V žádném z testů testovaná síť neobstála, testy `NONCE_REUSE` a `SWITCH_KEY` byly neprůkazné a ostatní dopadly negativně. V tabulce 8.1 jsou k vidění výsledky jednotlivých testů pro jednotlivá zařízení. Položky označené „—“ značí, že se daný test zařízení netýká.

Tabulka 8.1 Výsledek testů pro jednotlivá zařízení

Označení testu	CC2652P	Žárovka Philips	Vypínač Immax
MAC SECURITY	špatný	špatný	špatný
NONCE REUSE	neprůkazný	—	—
KEY TESTER	špatný	špatný	špatný
TRANSPORT KEY	špatný	špatný	špatný
SWITCH KEY	neprůkazný	—	—

Žádné ze zařízení během testu nešifrovalo svou komunikaci na MAC vrstvě. Test NONCE REUSE, který testuje, zda komunikace neprobíhá v módu TSCH, dopadl neprůkazně pro koordinátor. Výsledek se týká pouze koordinátoru, neboť on je zodpovědný za výběr komunikačního módu. Koordinátor se provinil využíváním dobře známého síťového klíče a všechna zařízení k připojení k síti využila předdefinovaný link klíč. Výsledek v testech KEY TESTER a TRANSPORT KEY je proto tedy špatný. Test SWITCH KEY je opět relevantní pouze pro koordinátor, protože je jediný, kdo tento příkaz může a má využívat. Během testu nedošlo k vyslání toho příkazu a zároveň nenastala žádná z podmínek definovaných v kapitole 4.2.4.2. Výsledek testu byl tedy neprůkazný.

8.1 Oprava chyb

Nastavení chování žárovky a vypínače není výrobci umožněno, a tak je potřeba chyby opravit pomocí úpravy nastavení koordinátoru.

Během testu nebyl žádný ze všech 629 zachycených rámců zabezpečený na MAC vrstvě. V nastavení koordinátoru bohužel nebyl nalezen žádný způsob jak ochranu vynutit.

Výsledek testu zabezpečení MAC vrstvy se dále projevil na průkaznosti dalšího testu – bez zabezpečených rámců není test schopen určit, zda je komunikace náchylná na znovupoužití nonce. Na produktové stránce koordinátoru[22] se informace o použití TSCH módu nenachází.

Síť se nejvíce provinila v testu použití dobře známých klíčů. Síť totiž pro připojování využívá známý link klíč a výchozí hodnotu síťového klíče. ZigBee2MQTT umožňuje výchozí klíč změnit úpravou konfiguračního souboru. Uživatel buď může zadat klíč vlastní, nebo si ho může nechat vygenerovat. Po úpravě konfigurace, aby byl klíč vygenerován, byl ZigBee2MQTT restartován a nový klíč byl vygenerován. Po změně klíče navíc bylo potřeba znovu spárovat zařízení. Použití výchozího klíče není možné obejít, protože ZigBee2MQTT nepodporuje instalační kódy. Všechna zařízení v testovací síti jsou navíc příliš stará na to, aby podporovala dynamické vyjednávání o klíči.

V testu TRANSPORT KEY byl problém opět s použitím výchozího link klíče – s tím, jak již bylo zmíněno, nelze nic dělat.

Neprůkaznost testu SWITCH KEY se může zdát neproblémová, ale není tomu tak. Na začátku testu byla zařízení odpojena od sítě. Pro větší bezpečnost měl ale koordinátor v tento moment změnit (ideálně dvakrát po sobě) síťový klíč. Nastavení této funkcionality v nastavení ZigBee2MQTT nebylo nalezeno.

Po opakovaném provedení analýzy, jejíž výsledek je dostupný v příloze B.2, je vidět určité zlepšení ve výsledcích. Jednak se již v seznamu použitých známých klíčů nenachází dobře známý klíč `01030507090b0d0f00020406080a0c0d` a jednak test zachytil pouze 2 použití transport key s výchozím link klíčem namísto 4. Stalo se tak, protože druhá dvojice transport key přenášející nový link klíč byla zašifrována i na síťové vrstvě novým síťovým klíčem. Došlo tak k jednoznačnému zlepšení bezpečnosti sítě.

Kapitola 9
Diskuse

Výsledná aplikace umožňuje snadnou analýzu ZigBee sítí, kterou lze využít ke zvýšení jejich bezpečnosti. Tím se aplikace liší od rozšířené aplikace KillerBee, která vyžaduje značné znalosti pro dosažení kompletní analýzy. Na rozdíl od aplikací SecBee a Zigtoucher aplikace nevyžaduje cenově náročné SDR, čímž se stává dostupnější pro běžné uživatele.

Slabší částí aplikace jsou závěrečná hlášení – ta by v ideálním případě měla obsahovat více informací pro snadnější pochopení uživatelem. Aplikace také v aktuální fázi nedešifruje komunikaci mimo rozsah jednotlivých testů. To znamená, že v případě komunikace zašifrované neznámým klíčem není aplikace schopná přesvědčivě provádět ostatní testy.

Aplikaci je třeba dále rozvíjet a zlepšovat. Nejdříve je potřeba do aplikace přidat zmíněnou podporu dešifrování, aby výsledná analýza byla co nejúplnější. Aktuálně chybí dešifrování MAC vrstvy, možnost zadání síťového klíče a zachytávání klíčů předaných mezi zařízeními. V druhé fázi je potřeba přidat více informací do závěrečných hlášení a nakonec přidat pokyny pro uživatele během skenování, aby testy dosáhly co nejlepších výsledků. Průběžně je samozřejmě žádoucí aplikaci rozšiřovat o nové testy a zařízení.

Zajímavým rozšířením do budoucna by mohlo být zkoumání jednotlivých ovládacích systémů a jejich schopnost konfigurovat bezpečnost sítí a zařízení.

Kapitola 10
Závěr

Cílem práce bylo provést bezpečnostní analýzu protokolu ZigBee a zařízení, která ho využívají. Pro provedení analýzy zařízení bylo třeba navrhnout a vytvořit aplikaci, která umožní jejich jednoduchou analýzu. Posledním cílem bylo sestavit testovací síť ze ZigBee zařízení a provést její analýzu vytvořenou aplikací.

Na začátku práce bylo čtenáři přiblíženo základní fungování protokolu ZigBee a protokolu IEEE 802.15.4, které bylo následně doplněno o popis jejich bezpečnostních funkcionalit. Dále byla diskutována slabá místa protokolu a jeho potenciálních implementací.

Ke konci teoretické části byla představena aktuální technologická řešení pro tvorbu a analýzu domácích sítí.

V praktické části byla popsána použitá zařízení a s využitím několika dříve popsaných technologií byla vytvořena testovací síť. Pomocí aplikace Home Assistant bylo nastaveno ovládání a automatizace sítě.

Pro zadanou aplikaci byly stanoveny cíle a určeny její technické parametry. Bylo zprovozněno odposlouchávání komunikace pomocí koordinátoru CC2531. Samotná aplikace byla následně navržena a její jednotlivé komponenty byly řádně popsány, stejně jako její konfigurace a uživatelské rozhraní.

Závěrem praktické části byla vytvořená aplikace použita na testovací síť, jejíž bezpečnost byla vyhodnocena. V návaznosti na výsledek testu byla diskutována a provedena možná vylepšení testovací sítě, načež byla síť znovu otestována. Při opakovaném testu dosáhla aplikace lepších výsledků.

Vytvořená aplikace ZigCheck splnila cíle, které byly zadány, avšak je možné její rozšíření o podporu nových zařízení či kontroly dalších zranitelností, kterými se tato práce nezabývá či které vzniknou v budoucnu.

Příloha A

Konfigurační soubor aplikace Docker

Výpis kódu A.1 Konfigurační soubor aplikace Docker `docker-compose.yml`

```
services:
  homeassistant:
    container_name: homeassistant
    image: "ghcr.io/home-assistant/home-assistant:stable"
    volumes:
      - /home/tomas/docker/homeassistant:/config
      - /etc/localtime:/etc/localtime:ro
    restart: unless-stopped
    privileged: true
    network_mode: host
    devices:
      - /dev/serial/by-id/usb-1a86_USB_Serial-if00-port0:/dev/ttyUSB0
  mosquitto:
    image: eclipse-mosquitto
    container_name: mosquitto
    restart: unless-stopped
    volumes:
      - /home/tomas/docker/mosquitto:/mosquitto
      - /home/tomas/docker/mosquitto/config:/mosquitto/config
      - /home/tomas/docker/mosquitto/data:/mosquitto/data
      - /home/tomas/docker/mosquitto/log:/mosquitto/log
    ports:
      - 1883:1883
      - 9001:9001
  zigbee2mqtt:
    container_name: zigbee2mqtt
    image: koenkk/zigbee2mqtt
    restart: unless-stopped
    volumes:
      - /home/tomas/docker/zigbee2mqtt/data:/app/data
      - /run/udev:/run/udev:ro
    ports:
      - 8080:8080
    devices:
      - /dev/serial/by-id/usb-1a86_USB_Serial-if00-port0:/dev/ttyACM0
```

Příloha B

Výstup aplikace ZigCheck

Výpis kódu B.1 Výstup aplikace ZigCheck po analýze testovací sítě

```
////////////////////////////////////////////////

  _______          _____ _                 _
 |___  (_)       / ____| |               | |
    / / _  __ _| |    | |__   ___  ___| | __
   / / | |/ _` | |    | '_ \ / _ \/ __| |/ /
  / /__| | (_| | |____| | | |  __/ (__|   <
 /_____|_|\__, |\_____|_| |_|\___|\___|_|\_\
           __/ |
          |___/

////////////////////////////////////////////////

Testing in progress. Use Ctrl-C to stop.
[####################################]  100%

Stopping...

MAC SECURITY:
[X] Encrypted: 0        Integrity only: 0       Unsecured: 629

--------------------------------------------------------------------------

NONCE REUSE:
[-] No encrypted data was sent to determine whether TSCH mode was used.

--------------------------------------------------------------------------

KEY TESTER:
[X] The following well-known keys were used:
        5a:69:67:42:65:65:41:6c:6c:69:61:6e:63:65:30:39
        01:03:05:07:09:0b:0d:0f:00:02:04:06:08:0a:0c:0d

--------------------------------------------------------------------------

TRANSPORT KEY:
[X] Transport key mechanism used with a well-known link key.
        Transport key mechanism used with a well-known link key.
```

```
        Transport key mechanism used with a well-known link key.
        Transport key mechanism used with a well-known link key.

--------------------------------------------------------------------------

SWITCH KEY:
[-] No switch key command was sent to determine its correct use.

--------------------------------------------------------------------------
```

Výpis kódu B.2 Výstup aplikace ZigCheck po bezpečnostních úpravách

```
////////////////////////////////////////////////

  _______         _____ _                 _
 |___  (_)       / ____| |               | |
    / / _  __ _| |    | |__   ___  ___| | __
   / / | |/ _` | |    | '_ \ / _ \/ __| |/ /
  / /__| | (_| | |____| | | |  __/ (__|   <
 /_____|_|\__, |\_____|_| |_|\___|\___|_|\_\
           __/ |
          |___/

////////////////////////////////////////////////

Testing in progress. Use Ctrl-C to stop.
[#####################################]  100%

Stopping...

MAC SECURITY:
[X] Encrypted: 0        Integrity only: 0       Unsecured: 697

--------------------------------------------------------------------------

NONCE REUSE:
[-] No encrypted data was sent to determine whether TSCH mode was used.

--------------------------------------------------------------------------

KEY TESTER:
[X] The following well-known keys were used:
        5a:69:67:42:65:65:41:6c:6c:69:61:6e:63:65:30:39

--------------------------------------------------------------------------

TRANSPORT KEY:
[X] Transport key mechanism used with a well-known link key.
        Transport key mechanism used with a well-known link key.

--------------------------------------------------------------------------

SWITCH KEY:
[-] No switch key command was sent to determine its correct use.

--------------------------------------------------------------------------
```

Příloha C

Uživatelská příručka

C.1 Instalace

Program ZigCheck je primárně určen pro distribuci Ubuntu 22.04 LTS operačního systému GNU/Linux, kterou je doporučeno pro spuštění programu použít. Pro nainstalování jsou potřeba zdrojové kódy, které jsou dostupné na přiloženém médiu. V případě zájmu je možné zdrojové kódy stáhnout pomocí nástroje git.

Po obdržení zdrojových kódů je potřeba přejít do složky `zigcheck` (složka obsahující soubor `install.sh`) a nastavit soubor `install.sh` jako spustitelný. Nakonec zbývá jen instalační skript `install.sh` spustit. Přesné příkazy potřebné k instalaci lze najít ve výpisu kódu C.1.

Výpis kódu C.1 Sada příkazů pro instalaci aplikace ZigCheck

```
git clone https://github.com/tomasrosenbaum/zigcheck.git
cd zigcheck
chmod +x install.sh
./install.sh
```

C.2 Použití aplikace

Aplikace je nainstalovaná do Python virtuálního prostředí, které je potřeba spustit příkazem `source venv/bin/activate`. Po aktivaci se stane aplikace ZigCheck dostupná.

Nyní je možné aplikaci spustit jen zavoláním příkazu `zigcheck`. Pro zobrazení všech dostupných přepínačů lze využít přepínač `--help`, jehož výstup se nachází na výpisu kódu C.2.

Virtuální prostředí lze deaktivovat pomocí příkazu `deactivate`.

C.2.1 Konfigurace

Pro správné fungování aplikace je potřeba nastavit správné hodnoty do souboru `config.yml`. Těmi stěžejními jsou nastavení PAN ID skenované sítě a kanál, na kterém má skenování probíhat.

Uživatel samozřejmě může měnit jakékoli hodnoty konfigurace. Popis významu jednotlivých hodnot je uveden v konfiguračním souboru.

Konfiguraci je také možné jednorázově upravit využitím přepínačů vysvětlených v nápovědě.

Výpis kódu C.2 Nápověda pro použití aplikace ZigCheck

```
Usage: zigcheck [OPTIONS]

Options:
  -v, --version                  Print version and exit.
  -c, --configfile FILENAME      Configuration file.
  -t, --tests TEXT               Comma-separated names of tests to run.
  --print / --no-print           Enable/disable printing of captured
                                 packets.
  -d, --device [CC2531|Offline]  Device used for capturing traffic.
  --no-pan-id                    Don't filter packets by PAN ID.
  --pan-id INTEGER               PAN ID to filter packets.
  --stop-after INTEGER           How long the scan should take in seconds.
                                 Non-positive value means infinite.
  --channel INTEGER RANGE        On which channel the scan should be
                                 performed.  [11<=x<=26]
  --set TEXT...                  Set any value of the configuration.
                                 Multiple set options allowed.
                                 Usage: --set <colon-separated key> <value>
                                 (e.g. --set devices:Offline:file
                                 capture.pcap)
  --help                         Show this message and exit.
```

Bibliografie

1. CONNECTIVITY STANDARDS ALLIANCE. *Zigbee Specification: Revision 23* [online]. Connectivity Standards Alliance, 2023 [cit. 2023-04-14]. Dostupné z: https://csa-iot.org/developer-resource/specifications-download-request/.
2. *What is the ZigBee Alliance?* [Online]. digicert, [b.r.] [cit. 2023-04-14]. Dostupné z: https://www.digicert.com/faq/industry-standards-for-security-and-trust/what-is-the-zigbee-alliance.
3. IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*. 2014, s. 1–74. Dostupné z DOI: 10.1109/IEEESTD.2014.6847097.
4. IEEE Standard for Low-Rate Wireless Networks. *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*. 2020, s. 1–800. Dostupné z DOI: 10.1109/IEEESTD.2020.9144691.
5. INDIKA. *Difference Between CSMA and ALOHA* [online]. Difference Between, 2011 [cit. 2023-04-15]. Dostupné z: https://www.differencebetween.com/difference-between-csma-and-vs-aloha/.
6. DATTATRAY. *IoT Security – Part 5 (ZigBee Protocol – 101)* [online]. Payatu, 2020 [cit. 2023-04-16]. Dostupné z: https://payatu.com/masterclass/iot-security-part-5-zigbee-protocol-101/.
7. *What Are Sniffing Attacks, and How Can They Be Prevented?* [Online]. EC-Council, 2023 [cit. 2023-05-08]. Dostupné z: https://www.eccouncil.org/cybersecurity-exchange/ethical-hacking/what-are-sniffing-attacks/.
8. DATTA, Subham. *What Are Replay Attacks?* [Online]. Baeldung, 2023 [cit. 2023-05-08]. Dostupné z: https://www.baeldung.com/cs/replay-attacks.
9. ADEJUMOLA, Richard. *What Is Signal Jamming and What Can You Do About It?* [Online]. Make Use Of, 2022 [cit. 2023-05-08]. Dostupné z: https://www.makeuseof.com/what-is-signal-jamming/.
10. *Cyber Security: The Onion Approach* [online]. JFG, Inc., [b.r.] [cit. 2023-04-20]. Dostupné z: https://www.jfg-nc.com/cyber-security-the-onion-approach/.
11. LUNDKVIST, Christian. *Nonce reuse in encryption - what's the worst that can happen?* [Online]. GitHub, 2021 [cit. 2023-04-20]. Dostupné z: https://www.github.com/christianlundkvist/blog/blob/6a33c65138d2dd5df3aea68971fab5fc61e0e7bd/2021_01_25_nonce_reuse_in_encryption/nonce_reuse_in_encryption.md.
12. DATTATRAY. *Zigbee Security 101 (Architecture And Security Issues)* [online]. Payatu, 2023 [cit. 2023-04-25]. Dostupné z: https://payatu.com/blog/zigbee-security-101/.

13. *Secure your Zigbee network* [online]. Zigbee2MQTT, 2023 [cit. 2023-05-03]. Dostupné z: https://www.zigbee2mqtt.io/advanced/zigbee/03_secure_network.html.

14. HALL, Chris. *What is Zigbee and why is it important for your smart home?* [Online]. Pocket-lint, 2023 [cit. 2023-05-08]. Dostupné z: https://www.pocket-lint.com/what-is-zigbee-and-why-is-it-important-for-your-smart-home/.

15. *Installation* [online]. Home Assistant, 2023 [cit. 2023-05-08]. Dostupné z: https://www.home-assistant.io/installation/.

16. JERRM. *ZHA Vs Zigbee2Mqtt* [online]. Home Assistant, 2022 [cit. 2023-04-21]. Dostupné z: https://community.home-assistant.io/t/zha-vs-zigbee2mqtt/433161/3.

17. COPE, Steve. *Using Zigbee2MQTT- A Beginners Guide* [online]. Steve's Smart Home Guide, 2023 [cit. 2023-05-08]. Dostupné z: https://stevessmarthomeguide.com/using-zigbee2mqtt-beginners-guide/.

18. RIVER LOOP SECURITY. *killerbee* [online]. GitHub, 2022 [cit. 2023-04-23]. Dostupné z: https://github.com/riverloopsec/killerbee.

19. COGNOSEC GMBH. *SecBee* [online]. GitHub, 2016 [cit. 2023-05-04]. Dostupné z: https://github.com/Cognosec/SecBee.

20. ŠATOPLET, Jakub. *Bezpečnostní analýza protokolu ZigBee Touchlink*. Praha, 2022. Dostupné také z: http://hdl.handle.net/10467/101663. Bakalářská práce. České vysoké učení technické v Praze.

21. *Touchlink* [online]. Zigbee2MQTT, 2023 [cit. 2023-05-08]. Dostupné z: https://www.zigbee2mqtt.io/guide/usage/touchlink.html.

22. *CC2652P* [online]. Texas Instruments, 2023 [cit. 2023-04-20]. Dostupné z: https://www.ti.com/product/CC2652P.

23. *A60 – chytrá žárovka s paticí E27 – 1100* [online]. Philips, 2023 [cit. 2023-05-08]. Dostupné z: https://www.philips-hue.com/cs-cz/p/hue-white-ambiance-a60-%5C%E2%5C%80%5C%93-chytra-zarovka-s-patici-e27-%5C%E2%5C%80%5C%93-1100/8719514291119.

24. *Immax NEO Smart vypínač 1-tlačítkový Zigbee 3.0* [online]. Immax, 2023 [cit. 2023-05-08]. Dostupné z: https://www.immax.cz/immax-neo-smart-vypinac-1-tlacitkovy-zigbee-3-0-p11820/.

25. *FLASH-PROGRAMMER-2* [soft.]. Texas Instruments, 2020. Ver. 1.8.2 [cit. 2023-04-27]. Dostupné z: https://www.ti.com/tool/FLASH-PROGRAMMER.

26. KANTERS, Koen. *Z-Stack-firmware* [online]. GitHub, 2023 [cit. 2023-04-21]. Dostupné z: https://github.com/Koenkk/Z-Stack-firmware.

27. PALLETS. *click* [online]. GitHub, 2023 [cit. 2023-04-27]. Dostupné z: https://github.com/pallets/click/.

28. SECDEV. *scapy* [online]. GitHub, 2023 [cit. 2023-04-25]. Dostupné z: https://github.com/secdev/scapy.

29. *PACKET-SNIFFER* [soft.]. Texas Instruments, 2014. Ver. 2.18 [cit. 2023-04-25]. Dostupné z: https://www.ti.com/tool/PACKET-SNIFFER.

30. ALEMASOV, Vladimir. *whsniff* [online]. GitHub, 2021 [cit. 2023-04-25]. Dostupné z: www.github.com/homewsn/whsniff.

31. *Sniff Zigbee traffic* [online]. Zigbee2MQTT, 2023 [cit. 2023-04-25]. Dostupné z: https://www.zigbee2mqtt.io/advanced/zigbee/04_sniff_zigbee_traffic.html.

32. *Flashing the CC2531 USB stick* [online]. Zigbee2MQTT, 2023 [cit. 2023-04-25]. Dostupné z: www.zigbee2mqtt.io/guide/adapters/flashing/flashing_the_cc2531.html.

33. *Alternative flashing methods* [online]. Zigbee2MQTT, 2023 [cit. 2023-04-25]. Dostupné z: www.zigbee2mqtt.io/guide/adapters/flashing/alternative_flashing_methods.html.

34. REDBEARLAB. *CCLoader* [online]. GitHub, 2023 [cit. 2023-04-25]. Dostupné z: https://github.com/RedBearLab/CCLoader.

35. *Arduino IDE* [soft.]. Arduino, 2023. Ver. 2.1.0 [cit. 2023-04-25]. Dostupné z: https://www.arduino.cc/en/software.

Obsah přiloženého média

- `license.txt` ... licenční ujednání
- `install.sh` ... instalační skript
- `config.yml` ... konfigurační soubor
- `requirements.txt` ... seznam potřebných balíčků pro `pip`
- `setup.py` ... popis aplikace pro `pip`
- `docs\` ... složka s dokumentací zdrojového kódu
  - `index.html` ... úvodní stránka dokumentace
- `whsniff\` ... složka obsahující podpůrnou aplikaci whsniff
  - `LICENSE` ... licenční ujednání aplikace
  - `Makefile` ... Makefile pro kompilaci aplikace
  - `ReadMe.md` ... návod na použití aplikace
  - `src\` ... složka se zdrojovým kódem aplikace
    - `whsniff.c` ... zdrojový kód aplikace
- `zigcheck\` ... složka obsahující zdrojové kódy aplikace zigcheck
- `thesis\` ... složka obsahující tuto bakalářskou práci
  - `rosento2-zigbee-analysis.pdf` ... text práce ve formátu PDF
  - `src\` ... složka obsahující zdrojový kód práce ve formátu LaTeX