Sem vložte zadání Vaší práce.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Detekce anomálií v provozu IoT sítí

Bc. Dominik Soukup

Katedra počítačových systémů
Vedoucí práce: Tomáš Čejka

27. března 2018

Poděkování

Doplňte, máte-li komu a za co děkovat. V opačném případě úplně odstraňte tento příkaz.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické 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. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 27. března 2018 .....

Odkaz na tuto práci

Soukup, Dominik. *Detekce anomálií v provozu IoT sítí.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

V několika větách shrňte obsah a přínos této práce v češtině. Po přečtení abstraktu by se čtenář měl mít čtenář dost informací pro rozhodnutí, zda chce Vaši práci číst.

Klíčová slova Nahraďte seznamem klíčových slov v češtině oddělených čárkou.

Abstract

Sem doplňte ekvivalent abstraktu Vaší práce v angličtině.

Keywords Nahraďte seznamem klíčových slov v angličtině oddělených čárkou.

Obsah

Seznam obrázků

Úvod

Koncept internetu existuje již několik desítek let a pro spoustu lidí se stal nedílnou součástí pracovního i osobního života. V poslední době je možné sledovat stále rostoucí počet zařízení, která jsou do něj zapojena. Tento trend by měl pokračovat i do budoucnosti, a dokonce v ještě větším měřítku. Odhadem je přes 30 miliard připojených zařízení do roku 2020 [1]. Důvodem zrychleného růstu je expanze síťového připojení na veškeré elektronické zařízení a senzory, které umožní vzdálené ovládání a monitorování. Pro označení tohoto trendu se používá termín Internet věcí (Internet of Things, IoT).

Cílem IoT je usnadnit, zlepšit a ušetřit lidskou činnost napříč všemi odvětvími. Uplatnění se nachází zejména ve výrobních podnicích, dopravě nebo běžných domácnostech. Příklad konkrétního nasazení popisuje článek [2], jehož cílem je měření kvality spánku a odhalení případných poruch. Monitorovací systém lze dále propojit například s ovládáním místnosti, které bude reagovat na aktuální úroveň spánku úpravou světel, oken nebo vzduchu.

Dále je upraven model komunikace, který již nevyžaduje zasílání zpráv centrálnímu serveru (north-south), ale podporuje přímou komunikaci mezi připojenými uzly (east-west). Oblast IoT nezahrnuje jen malá a nevýkonná zařízení, ale jeho součástí jsou i výkonná datová centra a pokročilé algoritmy, které vyhodnocují získané informace.

Hlavní hrozbou Internetu věcí je bezpečnost. Dochází zde k přenosu citlivých dat, která slouží k automatizovanému řízení dalších systémů, monitorování prostředí a zabezpečovacím účelům. Zároveň s masivním rozšířením nově připojených zařízení roste riziko vzniku nových útoků a možnosti způsobení větších škod. Příkladem bezpečnostního incidentu je útok na distribuční síť elektrického produ na Ukrajině, který měl dopad na 225 000 zákazníků [3].

Pro potlačení možného vzniku hrozeb musí být součástí každné dnešní IoT sítě sada procesů, které umožní důvěryhodné získávání validních dat, vzdálenou správu a možnost detekce anomálií jako v běžných IP (Internet Protocol) sítích.

Kapitola 1

Cíl práce

Cílem magisterské práce je analyzovat množinu aktuálně používaných protokolů pro komunikaci v IoT sítích a identifikovat jejich bezpečnostní zranitelnosti. Při analýze bude věnována pozornost zejména bezdrátovým senzorovým protokolům. Na základě získaných znalostí bude navržen, implementován a otestován algoritmus pro monitorování a automatickou detekci anomálního provozu v IoT sítích. Algoritmus bude možné spustit v prostředí nově vznikající opensource brány BeeeOn, čímž dojde k rozšíření dostupných bezpečnostních funkcí.

Kapitola 2

Analýza

Kapitola se zabývá analýzou celkové architektury IoT sítí a způsoby pro její zabezpečení. Postupně je prozkoumán komunikační model, možné bezpečnostní hrozby a existující řešení pro obranu. Na základě analýzy jsou uvedeny funkční a nefunkční požadavky, které jsou kladeny na výsledný program. V závěru jsou vybrány konkrétní technologie pro realizaci.

2.1 Architektura IoT sítí

V blízké době se očekává stále větší nárůst zařízení, která jsou připojena k internetu. Dle odhadů by jejich počet měl v roce 2020 překročit 30 miliard [1]. Pro takové množství připojení už není možné, aby každé zařízení komunikovalo přímo se vzdáleným datovým centrem, protože nároky na potřebnou šířku pásma by byly obrovské [4]. Dalším problémem je často velmi omezený výkon připojených prvků, který je nezbytný pro použití bezpečnostních funkcí umožňujících kompletně zabezpezpečenou komunikaci.

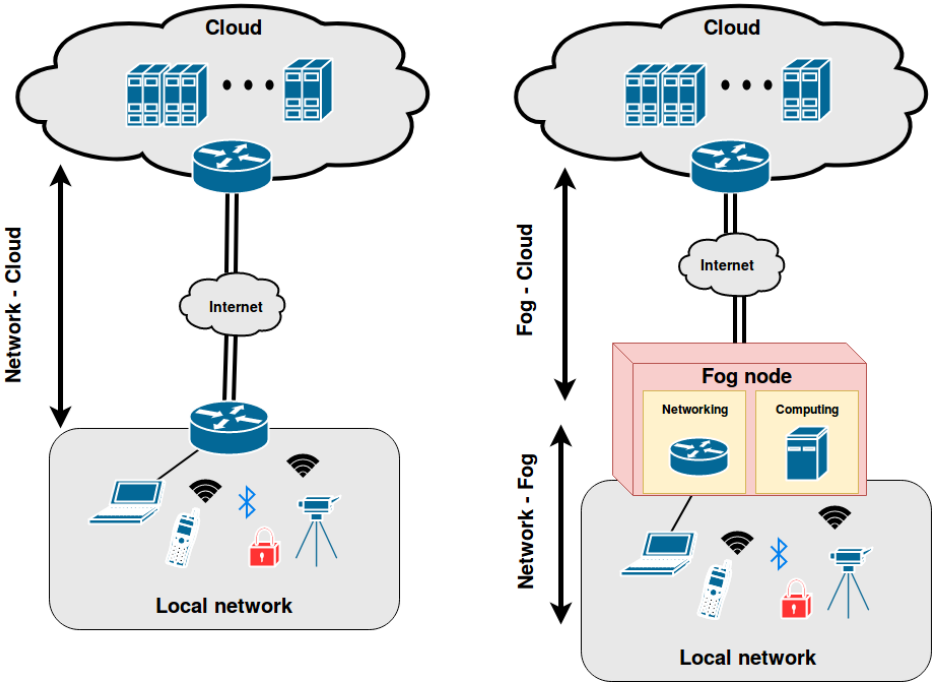
Řešení těchto problémů je do probíhající komunikace přidat několik podvrstev, které umožní přesunout výpočetní výkon blíže ke koncovým zařízením, a tím celý proces zpracování dat provést efektivněji.

V následujících podkapitolách bude popsán nový model komunikace, který přináší IoT sítě.

2.1.1 Fog computing

Fog computing je rozšíření cloud computingu, které spočívá v přesunutí výpočetního výkonu blíže k okraji sítě. Rozšíření je umožněno pomocí přidání síťových zařízení, které kromě běžných funkcionalit poskytují i výpočetní výkon pro běh externích programů. Programy je často možné nasadit pomocí kontejnerů nebo samostatných virtuálních strojů, což velmi usnadňuje jejich distribuci [4].

Porovnání klasické a fog architektury se nachází na obrázku 2.1. V reálném nasazení může být použito i více Fog vrstev, kde každá provádí určitý stupeň předzpracování a řízení dat. Zavedením principů fog computingu vznikají pro



Obrázek 2.1: Porovnání klasické a Fog architektury

síť následující výhody:

- **Zlepšení bezpečnosti**

  Síťové prvky jsou trvale napájené a připojené k internetu. Podporují pokročilé bezpečnostní funkce, a proto je možné například vytvářet šifrované tunelové spojení pro bezpečný přenos dat [4].

- **Nižší nároky na šířku pásma a latenci**

  Odeslaná data z koncových zařízení jsou zpracovávána a filtrována na okraji sítě. Tím je možné rychleji reagovat na přijaté zprávy a snížit nároky na latency a šířku pásma. Zároveň krátkodobá data mohou být uložené ve Fog vrstvě a centrální datové cetrum může být využito pro dlouhodobé údaje, které se zpracovávají pokročilými algoritmy pro analýzu dat [4].

- **Jednotná správa**

Při správě sítě už se nemusí přistupovat přímo na koncové prvky, které často komunikují různými protokoly, ale stačí pouze řídit síťová zařízení v jednotlivých Fog vrstvách, které odstiňují různorodost protokolů a nabízí standardizovaný přístup. Díky této abstrakci je zároveň zjednodušeno zpracování získaných dat a je umožněno přímé zasílání zpráv mezi koncovými prvky, které používají odlišné komunikační protokoly [4].

2.1.2 IoT brána

IoT brána je síťové zařízení, které je umístěno velmi blízko koncových zařízení a představuje vstup do Fog vrstvy. Jejím hlavním cílem je získávat data z připojených zařízení a poskytovat je vyšším vrstvám. Pokud je brána reprezentována výkonnějším síťovým prvkem, tak v něm zároveň může probíhat i základní zpracování dat.

Pro IoT sítě je typické, že obsahují velké množství koncových prvků komunikujících různorodými způsoby. Zejména senzory používají protokoly, které nepodporují IP spojení. Důvodem použití této komunikace je často velký důraz na nízkou spotřebu a specifické požadavky na způsob zasílání zpráv. Příkladem protokolů pro senzorové sítě je například: Z-Wave, Bluetooth a Zigbee. Jejich detailní popis se nachází v kapitole 2.2 Používané komunikační protokoly. Tato různorodost vyžaduje, aby brána obsahovala dodatečná rozhraní, které umožní připojení nejrůznějších bezdrátových i drátových koncových prvků.

V současné době existuje mnoho různých bran jejichž parametry se liší dle způsobu nasazení a provozních nároků. Velkým problémem v této oblasti je malý důraz na bezpečnostní funkce, které umožní vzdálené řízení brány, kontrolu provozu a aktualizace programového vybavení. Z těchto důvodů vznikl opensource projekt BeeeOn [5] jehož cílem je vytvořit softwarou IoT bránu, kterou bude možné spustit na různých hardwarových platformách. BeeeOn brána je navržena modulárně tak, aby byla schopna zpracovávat více rozdílných senzorových protokolů, a tím bude možné provozovat jedno univerzální zařízení namísto několika proprietárních. Zároveň je kladen důraz na bezpečnost, a proto veškeré údaje, které je možné získat o provozu, jsou poskytovány pro analýzu. Nad těmito údaji je postaven detekční algoritmus, který je výsledkem této diplomové práce.

2.1.3 Komunikační model a jeho hrozby

Při použití principů popsaných v předchzích kapitolách lze model komunikace rozdělit do následujících vrstev [6]:

- **Senzorová vrstva**

  Senzorová vrstva obsahuje veškerá koncová zařízení, které získávají informace ze svého okolí nebo vykonávají potřebnou službu [7]. Tato zařízení

jsou připojena kabelově nebo bezdrátově k IoT bráně. K jedné bráně může být připojeno několik prvků, které komunikují odlišnými způsoby. Princip komunikace a možnosti topologie se odlišují na základě použité technologie.

Velkým nebezpečím této vrstvy jsou zejména bezdrátové protokoly, protože při nepoužití zabezpečení může snadno dojít k odposlouchávání nebo úpravám provozu [6]. Dále se zde mohou vyskytovat zařízení, které jsou označeny jako zabezpečené, ale díky starší verzi komunikačního protokolu používají zastaralé bezpečnostní funkce nebo obsahují implementační chyby. Tento případ je velmi nebezpečný, protože vyvolává falešný pocit bezpečí. Útoky se také mohou zaměřovat na prvky s bateriovými zdroji, které mohou být nadměrnou komunikací účelově vybíjeny.

- **Síťová vrstva**

  Po zpracování senzorových dat na bráně je nutné získané informace odeslat dalším službám. K tomuto účelu slouží síťová vrstva. Cílem této vrstvy je také umožnit vzdálenou správu brány [7]. Pro výběr konkrétního protokolu je nutné znát rozhraní aplikační vrstvy. Ve většině případů je spojení vytvořeno pomocí protokolu HTTPS (Hypertext Transfer Protocol Secure) nebo technologie VPN (Virtual Private Network). Nad tímto spojením je postavena další služba pro výměnu zpráv. Příkladem může být: MQTT (Message Queuing Telemetry Transport), COAP (Constrained Application Protocol) nebo AMQP (Advanced Message Queuing Protocol). Tyto protokoly jsou podrobněji posány v kapitole 2.2 Používané komunikační protokoly.

  Bezpečnostní hrozby síťové vrstvy jsou stejné jako v klasických sítích. Je potřeba dodržet principy důvěry, integrity a dostupnosti. Tímto přístupem je možné předejít útokům jako: DDoS (Distributed Denial Of Service), MITM (Man In The Middle) a podvržení informací. Zároveň je nutné nezapomenout, že se zde většinou vyskytuje M2M (Machine To Machine) komunikace a je důležité použít vhodná komunikační rozhraní [6]. Častým případem bývá zastaralá verze firmware, jehož napadení může vést k nestandartnímu chování nebo dokonce ke kompletnímu převzetí kontroly.

- **Aplikační vrstva**

  Aplikační vrstva se stará o ukládání dlouhodobých dat a jejich finální zpracování. Zároveň zobrazuje uživateli zpracované informace a umožňuje provádět konfiguraci celé sítě. Z důvodu její možné rozsáhlosti je kritické, aby správa topologie podporovala automatizaci. [7].

  Tato vrstva je umístěna vetšinou v datovém centru a umožňuje vzdálený přístup. Její bezpečnostní problémy lze přirovnat k problémům cloud computingu. Dle množství požadovaných funkcí může být různě složitá a

s rostoucí složitostí se také liší nároky na úroveň zabezpečení. Příkladem možných útoků může být: Buffer Overflow, SQL Injection nebo DDoS.

2.2 Používané komunikační protokoly

Jedním z hlavích cílů IoT je možnost vzájemného propojení různých komunikačních protokolů, které umožní automatizovanou výměnu zpráv mezi všemi dostupnými zařízeními. Tímto způsobem je následně možné zefektivňovat a usnaďnoat lidskou práci.

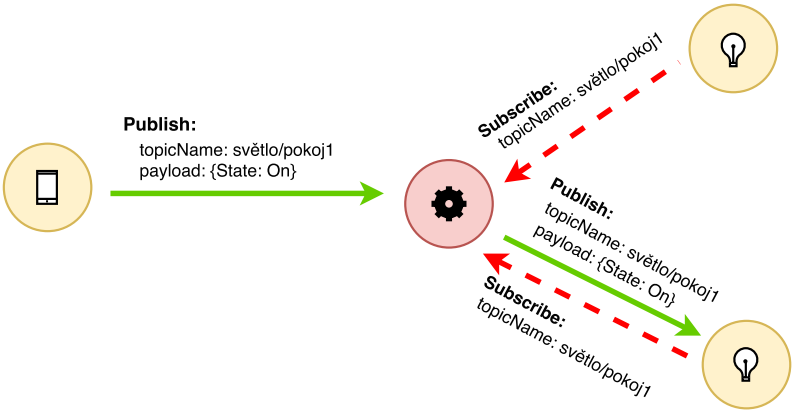
V následujích podkapitolách bude popsána množina aktuálně často používaných protokolů včetně jejich bezpečnostních funkcí.

2.2.1 MQTT

MQTT je otevřený síťový komunikační protokol typu pulish/subcribe, který byl navržen v roce 1999. Již od návrhu byl zaměřen na nízkou náročnost komunikace a jednoduchost implementace. Díky těmto vlastnostem je velmi vhodný pro IoT a M2M systémy. [8]

2.2.1.1 Způsob komunikace

Protokol MQTT je postaven nad transportní vrstvou TCP (Transmission Control Protocol) a využívá model pulish/subcribe, který vychází z tradičního způsobu zasílání zpráv typu klient-server. Roli serveru zde plní speciální uzel, který se nazývá broker. Broker je známý všem ostatním klientům, kteří mohou zasílat zprávy pomocí operace *publish* nebo se přihlásit o příjem zpráv díky operaci *subscribe*. Na základě provedených operací broker přijímá zprávy a rozhoduje o jejich přeposlání. Způsob odeslání zprávy závisí na obsažených metadatech.



Obrázek 2.2: MQTT architektura

Nejčastěji o směru odeslání rozhoduje předmět (topic) zprávy. Předmět je tvořen jednoduchým UTF-8 řetězcem s hierarchickou strukturou, ve které jsou jednolivé vrstvy odděleny dopředným lomítkem (např. /domov/přízemí/světloŠatna).

V předmětu zprávy mohou být některé vrstvy nahrazeny zástupnými symboly + a #. Symbol + dokáže nahradit pouze jednu úroveň předmětu libovolným řetězcem a symbol # umožňuje zastoupit více úrovní. Díky těmto symbolům mohou klienti odeslat nebo přijímat zprávy z více témat.

Další užitečnou položkou protokolu MQTT je QoS (Quality of Service), která může nabývat hodnot 0, 1 nebo 2.

- **QoS 0**

  Veškeré zprávy jsou odesílány bez potvrzení a žádným způsobem není zvýšena úroveň spolehlivosti, která je shodná se spolehlivostí protokolu TCP.

- **QoS 1**

  Pomocí potvrzování zajišťuje, že každá zpráva bude příjemci doručena alespoň jednou.

- **QoS 2**

  Umožňuje, aby každá zpráva byla spolehlivě doručena právě jednou.

Hodnota QoS se nastavuje vždy mezi dvěma uzly při navazování spojení. Z pohledu brokeru se může stát, že přijatá a odeslaná zpráva mají jiné QoS. Úrovně 1 a 2 dále umožňují perzistetní ukládání zpráv v případě, že příjemce je nedostupný. Zároveň platí, že s vyšší úrovní roste i režie komunikace. [9]

2.2.1.2 Bezpečnost

Zabezpečení protokolu MQTT je možné rozdělit do následujících vrstev:

- **Síťová vrstva**

  Veškerá komunikace je postavena nad TCP/IP, a proto lze probíhající komunikaci zapouzdřit pomocí VPN připojení jako v běžných počítačových sítích. Kvůli větším nárokům na výkon je toto řešení vhodnější pro výkonější zařízení jako jsou například IoT brány, které mohou s brokerem navázat site-to-site spojení. [10]

- **Transportní vrstva**

  Na této úrovni se využívá šifrování provozu pomocí protokolu TLS (Transport Layer Security). Omezením této metody jsou požadavky na výkon, které mohou být poměrně vysoké pokud nastává časté navazování spojení. [10]

- **Aplikační vrstva**

  Samotný protokol MQTT nedefinuje žádné šifrovací mechanismy na aplikační úrovni. Zabezpeční dat zde musí zajistit uživatel ještě před jejich

zapouzdřením do MQTT zprávy. Ovšem tímto způsobem je možné šifrovat jen tělo zprávy a hlavička zůstává nezměněná.

Pro autentizaci je možné využít ověření pomocí jména a hesla nebo x.509 certifikátu. Jméno a heslo je přenášeno nešifrovaně, a proto je vhodné tuto metodu doplnit se zabezpečením síťové nebo transportní vrstvy. Autentizaci pomocí certifikátů je možné využít jen v případě použití TLS. Tato metoda je vhodnější pokud všechna zařízení jsou po jednotnou správou a je možné automatizovat distribuci klientských certifikátů.

Dále je na straně brokeru možné definovat pravidla pro autorizaci. Tyto pravidla přiřazují klientů oprávnění pro provedení operací publish a subscribe nad příslušnými tématy. [10]

2.2.2 COAP

CoAP (Constrained Application Protocol) je otevřený přenosový protokol určený pro komunikaci síťových zařízení s velmi omezeným výkonem. Návrh vychází z RESTful (Representational State Transfer) principů, tudíž jeho použití je velmi vhodné v prostředích s již existujícím webových rozhraním, do kterého se snadno integruje. [11]

2.2.2.1 Způsob komunikace

Komunikace je postavena nad protokolem UDP (User Datagram Protocol) a vychází z modelu request/response, který se využívá u protokolu HTTP (Hypertext Transfer Protocol). Oproti protokolu HTTP je zde omezen počet možných operací a komunikace probíhá asynchronně. Dle důležitosti odesílané zprávy je možné určit, zda se má zasílat potvrzení či nikoli. Protože veškerá komunikace probíhá nad protokolem UDP, tak přenášené zprávy obsahují položku Message ID, která dokáže ošetřit duplikaci přijatých dat.

Pro zajištění integrace s běžnými webovými službami a zachování nízkých nároků se v CoAP topologii velmi často vyskytují proxy servery. Tyto servery mohou plnit funkci běžných reverzních a dopředných proxy, mapování mezi CoAP a HTTP protokolem nebo vyvažování zátěže.

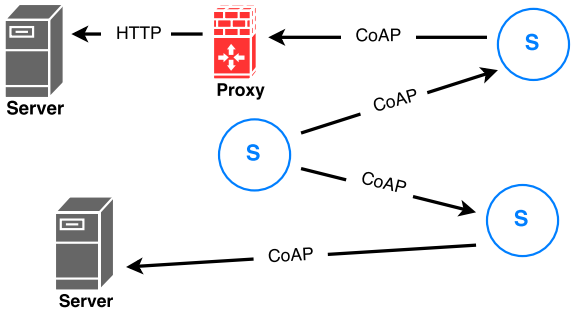
Ve specfikaci protokolu CoAP jsou definovány následující operace:

- **GET**

  Metoda *GET* vrací aktuální stav požadovaného zdroje, který je identifikován pomocí URI (Uniform resource identifier). Tato operace je vždy bezpečná a idempotentní.

- **POST**

  POST požadavek obsahuje ve svém těle novou reprezentaci cílového zdroje a požaduje jeho zpracování. Funkce, která novou reprezentaci přijímá je definovaná na cílovém uzlu s příslušným URI. Výsledkem je



Obrázek 2.3: COAP architektura

vytvoření nového zdroje nebo aktualizace původního. Tato metoda není z pohledu zpracování bezpečná ani idempotentní.

- **PUT**

  Tato metoda specifikuje nový stav cílového zdroje, který se buď vytvoří, nebo v případě jeho existence aktualizuje. Provedení operace není bezpečné, ale je idempotentní.

- **DELETE**

  Operace *DELETE* požaduje smazání zdroje s příslušným URI. Tato metoda není bezpečná, ale je idempotentní.

Vyhledání potřebného zdroje je v topologii protokolu CoAP možné provést pomocí znalosti jeho URI nebo odesláním multicastového požadavku na definovanou skupinu uzlů. Možnost vyhledávání cílových zdrojů je velmi důležitá zejména v M2M prostředích. [11]

2.2.2.2 Bezpečnost

Samotný protokol nijak nedefinuje možnost autentizace a autorizace. V případě potřeby je nutné tyto mechanismy implementovat v aplikačním kódu.

Pro zajištění šifrování provozu nabízí CoAP následující režimy:

- **NoSec**

  Tento mód neobsahuje žádnou úroveň zabezpeční a veškeré zprávy jsou zasílány v otevřené podobě.

- **PreSharedKey**

  V tomto případě se náváže spojení pomocí protokolu DTLS (Datagram Transport Layer Security), které využívá symetrické šifrování. Předsdílený klíč musí být známý všem uzlům před zahájením komunikace.

- **RawPublicKey**

  Tento režim také navuzuje zabezpečené spojení pomocí protokolu DTLS, ale místo symetrické šifry se používá asymetrická.

- **Certificate**

  Mód *Certificate* rozšiřuje *RawPublicKey* o přidání certifikátu.

Podobně jako u protokolu MQTT je i zde možné ochránit provoz na síťové vrstvě pomocí VPN. Při nasazení libovolného režimu zabezpeční je nutné počítat s většími nároky na výkon, které musí klient splňovat pro navazování a udržování spojení. [11]

2.2.3 Z-Wave

Z-Wave je bezdrátový komunikační protokol určený pro senzorové sítě, který vysílá v subgigahercových pásmech ISM (Industrial, Scientific and Medical). Veškeré komunikační prvky jsou certifikovány aliancí Z-Wave, která zároveň poskytuje technickou dokumentaci a licence pro vývoj. V současné době existují certifikace Z-Wave a Z-Wave Plus. Z-Wave Plus zařízení obsahují nový chipset, který vylepšuje komunikační parametry sítě a zároveň je zpětně kompatibilní se staršími modely. [12] Otevřená implementace celého protokolu se nazývá OpenZWave [13], která je vyvíjena komunitou. [14]

2.2.3.1 Způsob komunikace

V Z-Wave síti se může maximálně vyskytovat 232 uzlů, mezi kterými se vždy nachází jeden označený jako kontroler. Pro přidání libovolného zařízení do sítě musí být nejprve provedo příme spárování s kontrolerem. Během párovacího procesu nový prvek získá vlastní 8 bitový identifikátor (*Node ID*) a unikátní 32 bitový identifikátor sítě (*Home ID*), který má již od výroby kontroler uložen v nepřepisovatelné paměti. Následné zasílání zpráv už nemusí probíhat přímo mezi senzorem a kontrolerem, ale zprávy mohou být přeposílány sousední prvky, které jsou trvale napájené, čímž se velmi zvyšuje možná rozloha sítě. Pro odebrání libolného zařízení je opět nutné zajisiti přimé spojení s kontrolerem a spustit odstraňovací proces. [14]

V rámci specifikace Z-Wave [15] je definován mechanismus pro určení dostupných příkazů. Každý senzor obsahuje svou definici tříd funkcionalit (*Command Class*), které obsahují dostupné příkazy a formát odpovědi. Tyto definice jsou předány kontroleru během procesu párování. Příkladem může být třída *Binary Switch*, která obsahuje příkazy:

- *SET* - odesílá kontroler pro nastavení hodnoty
- *GET* - odesílá kontroler pro získání hodnoty
- *REPORT* - odesílá senzor jako odpověď na dotaz *GET*

Při posílání zpráv je vždy vyžadováno potvrzení. V případě neobdržení potvrzení se vysílání opakuje. Po třetím neúspěšném pokusu je požadavek zahozen.

2.2.3.2 Bezpečnost

Původní verze protokolu Z-Wave umožňovala volitelně využívat šifrování pomocí 128 bitového AES (Advanced Encryption Standard). Výměna symetrického klíče probíhá během počátečního párování, kde je vygenerovaný klíč šifrován pomocí výchozího klíče, který je uložen ve firmwaru. Vzhledem k tomuto postupu je dobré provádět počáteční párování na bezpečném místě, kde nemůže dojít k odposlechu. [16]

V roce 2016 Z-Wave aliance vydala nový S2 (Security 2) framework, který vylepšuje bezpečnostní funkcionality a od roku 2017 je jeho použití povinné pro všechny nově certifikovaná zarízení.

S2 framework umožňuje zařízení rozdělit do následujících skupin s rozdílnými šifrovacími klíči:

- **Access Control** - nejdůvěryhodnější třída, která obsahuje bezpečnostní prvky jako jsou například zámky, které zároveň podporují autentizaci
- **Authenticated** - skupina určená pro běžné senzory, které podporují autentizaci
- **Unauthenticated** - třída pro ostatní prvky, které nepodporují autentizaci.

Autentizace probíhá pomocí PIN (Personal Identification Number) nebo QR (Quick Response) kódu. Výměna klíče je založena na algortmu ECDH (Elliptic-curve Diffie–Hellman), který zajišťuje dostatečnou úroveň bezpečnosti během párovacího procesu. [14]

2.2.4 BLE (Bluetooth Low Energy)

BLE je bezdrátový prtokol určený pro senzorové sítě s důrazem na nízkou spotřebu. Byl představen v roce 2010 jako součást specifikace Bluetooth 4.0 a je nekompatibilní s původními verzemi. Za vývoj a údržbu protokolu je odpovědná skupina Bluetooth SIG (Special Interest Group). V současné době je nejnovější verzí Bluetooth 5.

2.2.4.1 Způsob komunikace

BLE vysílá v bezlicenčním ISM pásmu na frekvencích od 2.4 GHz do 2.4835 GHz. Specifikace protokolu vychází z Bluetooth, ale zejména díky změnám parametrů v rádiové vrstvě jsou navzájem nekompatibilní. Do verze 4 umožňuje navázat pro koncová zařízení pouze jedno spojení a vytváří tak hvězdicovou

topologii s jedním centrálním prvkem. Od verze 5 je možné využívat více připojení a vytvořit tak flexibilnější mesh topologii.

Před zahájením komunikace mezi centrálním prvkem a senzory je nutné nejprve prvést párování, které probíhá v následujících krocích:

- **Vysílání žádostí**

  Při spuštění párování začně koncový prvek na kanálech určených pro propagaci všesměrově vysílat žádosti o připojení, které obsahují název zařízení, jméno výrobce a podporované služby.

- **Přijímání žádostí**

  Pokud je centrální prvek přepnutý do párovacího režimu, tak naslouchá příchozím požadavkům, které zobrazuje uživateli. Po výběru správného zařízení se ukončí mód naslouchání a začne se navazovat spojení.

- **Inicializace připojení**

  Během této fáze se obě strany domlouvají na parametrech komunikace.

- **Komunikace**

  Po úspěšném navázání spojení je možná na základě dostupných služeb provádět zasílání zpráv.

2.2.4.2 Bezpečnost

BLE umožňuje volitelně používat šifrování pomocí AES, jehož šifrovací klíč je vytvořen během párování v části inicializace připojení. Velmi důležité je zabezpečit výměnu sdíleného klíče, která až do verze 4.1 není bezpečná, protože neumožňuje ochranu před odposloucháváním. Vylepšení přichází až od verze 4.2, ve které lze využít ECDH.

Vzhledem k různorodosti možných typů zařízení definuje BLE následující kategorie určující způsob výměny klíče:

- **Just Works**

  Nejjednodušší třída, která provádí výměnu automaticky a zároveň má nejmenší nároky na připojované zařízení, které nemusí podporovat autentizaci. Díky chybějící autentizaci je tato metoda zranitelná vůči MITM (Man in the middle) útokům.

- **Out of Band**

  V této kategorii jsou veškeré klíče vyměňovány odlišným komunikačním kanálem např. přes NFC (Near Field Communication). Celková bezpečnost této metody závisí na důvěryhodnoti použitého kanálu.

- **Passkey**

  Tato metoda vylepšuje *Just Works* o autentizaci, která spočívá v uživatelském zadání šesti místného kódu. Pro ochranu před odposlouchávání musí být použit algoritmus ECDH, který je dostupný až od verze 4.2.

- **Numeric Comparison**

  Využití tohoto způsobu párování je možné pouze od verze 4.2. Dochází zde k rozšíření metody *Just Works* o jeden kontrolní krok, který slouží jako ochrana před MITM útokem. Po výměně klíčů každé zařízení vygeneruje šestimístný číselný kód, který následně zobrazí uživateli a čeká na jeho potvrzení.

2.3 Bezpečnostní slabiny

Na základě provedené analýzy aktuálně používaných protokolů budou v této kapitole popsány jejich bezpečnotní slabiny, které budou později využity při návrhu detekčních algoritmů. Jelikož je tato práce zaměřena na senzorové protokoly nekumunikující přes IP, budou v této kapitole popsány slabiny protokolů Z-Wave a BLE.

2.3.1 Z-Wave

Hlavním problém jsou zařízení certifikovaná před březnem 2017, jelikož nepodporují S2 framework. Tyto prvky nemusí při své komunikaci využívat žádnou formu šifrování a síť se tak stává velmi zranitelnou. V minulosti již bylo provedeno několik útoků, které pomocí projektu Scapy-Radio projektu [17] nebo knihovny OpenZWave byly schnopny ovládat jednotlivé senzory. Při využití šifrování je velmi zranitelná doba během párovacího procesu, jelikož je při výměně šifrovacích klíčů použit výchozí klíč uložený ve firmware zařizení. Zároveň se u jednoho typu zámku podařilo objevit implementační chybu [16], které umožňovala vnutit nový šifrovací klíč a převzít tak kontrolu nad senzorem.

Při využití S2 frameworku dosud nebyly objeveny žádné zranitelnosti, ovšem tento framework je poměrně nový a většina používaných zařízení byla certifikována ještě před jeho zavedením.

2.3.2 BLE

Velkou hrozbou jsou zařízení s verzí 4.1 nebo nižší, protože nepodporují žádnou ochranu před odposlouchávání a MITM útokům v době párování. Od verze 4.2 je již pro výměnu klíčů použit ECDH algoritmus, který zabraňuje možnému odposlouchávání, ale v případě použití párovací metody, které nepodporuje autentizaci není zajištěna ochrana před MITM útoky. [14]

Dalším problémem jsou samotní výrobci, kteří často nevyužívají bezpečnostní funkce protokolu [18] nebo implementují vlastní způsoby zabezpečení na aplikační úrovni. Tímto dochází k nezabepečení fáze párování a často se vyskytují i implementační chyby, které přinášení další zranitelnosti [19] a umožňují útočníkovi získat kontrolu nad celým provozem. [14]

Nevýhodou je velmi dlouhá a komplikovaná specifikace protokolu, která vede k implementačním chybám samotného BLE [20]. Tím vzniká nebezpečí, že i u výrobce, který využívá všech bezpečnostní funcionalit, se mohou vyskytovat zranitelnosti díky chybné implementaci komunikačních vrstev protokolu. [14]

2.4 Možnosti detekce

V běžných IP sítích se pro detekci hrozeb nejčastěji používají IDS (Intrusion Detection System) a IPS (Intrustion Prevention System) systémy. Tyto služby rozšiřují koncept klasického firewallu, který blokuje nebo povoluje síťový provoz na základě statických pravidel, o podrobnější sledování datových toků, které jsou zablokovány na základě nestandartního chování.

IDS/IPS může být reprezentováno samostatným hardwarovým zařízením nebo softwarovým programem, který může být dále rozšířen o monitorovací sondy, které se starají pouze o sběr dat a jejich odesílání pro následnou analýzu. Zároveň se může lišit i umístění v síti. Detekční systémy lze provozovat před hraničním směrovačem a detekovat tak kompletní příchozí a odchozí data nebo za hraničním směrovačem, což umožní vyhodnocovat jen vyfiltrovaný provoz. Případně je možné nasadit IDS/IPS přímo na koncové stanice, kde kromě síťových dat lze získávat i informace o běhu systému. Poslední způsob poskytuje nejvíce detailní možnost analýzy, protože se provádí na míste vzniku komunikace a případný útok je možné zastavit již při jeho vzniku a zabránit případnému rozšíření. Nevýhodou takového nasazení je ztráta celkového pohledu na síť. Při zavedení IDS/IPS systému je z těchto důvodů dobré kombinovat způsoby nasazení a umožnit vzájemné sdílení detekovaných incidentů.

Při zaměření na zpracovávání datových toků lze dekteční systémy rozdělit do dvou základních kategorií:

- **Detekce anomálií**

  Tato metoda je založena na statistickém modelování. Nejprve se vytvoří profil běžného chování sítě, který se následně porovnává s aktuálním provozem. Dle použité metody se může profil běžného provozu průběžně aktualizovat. Pokud měřené hodnoty síťového provozu překročí hodnotu stanoveného profilu nad definovaný limit, tak je detekován incident. Výhodou metody anomálií je její uplatnění i na dosud neznámé útoky. Tento princip zároveň způsobuje větší míru falešných poplachů, a proto je nutné jejich pečlivější ověření. Příkladem detekčních metod je: strojové učení, časové řady nebo stavové automaty [21].

- **Detekce signatur**

  V případě použití této metody je využíváno signatur (profilů) předem známých útoků. Výhodou je, že díky popsaným signaturám je tato metoda poměrně přesná a detekuje málo falešných incidentů. Naopak nevýhodou je, že není možné detekovat nové druhy útoků, pro které není známý profil. Problémem také je, že signatury musí být uloženy na nějakém perzistetním úložišti, které je dostupné lokálně nebo vzdáleně [21].

IoT sítě k běžnému IP provozu přidávají senzorové protokoly, jejichž chování je také dobré monitorovat, protože jsou připojeny do počítačové sítě a

mohou být zneužity při útocích. Bohužel detekční metody nejsou pro tyto sítě moc rozšířené, a tím dochází ke zvýšení rizika a dopadu možných útoků. Pro určení incidentů lze využít stejných principů jako v IP sítích, ale liší se způsob získávání dat pro analýzu. Pro sběr informací lze využít následující přístupy:

- **Testbed**

  Tato metoda spočívá ve vytvoření specializovaného prostředí, ve kterém se nachází pouze testované a měřící zařízení. Cílem je ověřit, že nově připojovaný senzor splňuje veškeré bezpečnostní požadavky a neobsahuje žádné známe zranitelnosti. Měřící prvky reprezentují nástroje, které jsou schopné odposlouchávat komunikaci a využívájí se také například při automatizovaných penetračních testech.

- **Externí sonda**

  Funkce sondy je stejná jako v IP sítích. Jedná se o samostatné zařízení umístěné v sítí, které umožňuje sledovat probíhající komunikaci a odesílat získatné údaje ke zpracování. Výhodou je velké množství, které lze získat, ale zároveň komplikací je šifrovaný provoz a často i cena kvalitní sondy. Dalším využitím může být honeypot, kdy se sonda tváří jako zranitelný prvek a reportuje veškeré pokusy o útok.

- **Provozní statistiky**

  Posledním způsobem je sběr dat z příslušných rozhraní na IoT bráně. Tento postup nevyžaduje použití žádného dalšího zařízení, ale potřebuje, aby brána umožňovala získávat tyto statistiky. Statistiky nejsou tak podrobné jako u externí sondy, ale výhodou je, že získávání aktuálních dat o provozu je poměrně nenáročné.

Každá z předchozích metod používá jiný styl sběru dat. Při reálném použití je dobré vyhodnotit bezpečnostní rizika a hrozby, podle kterých lze jednotlivé metody vhodně kombinovat.

2.5 Existující řešení

V současnosti veškerá dostupná řešení se zaměřují na detekci útoků v IP protokolech. Mezi současnými IDS/IPS existují signatury pouze pro SCADA (Supervisory Control And Data Acquisition) protokoly. Router Turris Omnia umožňuje nasadit systém Suricata, pro který nabízí rozšíření PaKon [22]. Toto rozšíření zpracovává data ze Suricaty, které následně ukládá v přehledné formě. Díky tomu užívatel získá podrovný přehled o stavu provozu. Získaná data může dále využívat například k vylepšení stávajících bezpečnostních pravidel. Nevýhodou totoho řešení jsou vyšší hardwarové požadavky, a proto ho nelze provozovat na branách s omezenými prostředky. Další komplikací může být centralizovaná architektura a zameření pouze na IP provoz.

Na základě provedené analýzy nebylo nalezeno žadné řešení, které umožňuje vyhodnocovat provoz aktuálně používaných IoT protoklů s ohledem na možné omezení hardwarových prostředků.

2.6 Analýza požadavků

Podstatnou částí návrhu výsledného řešení je přesné určení požadavků, které se dělí na funkční a nefunkční. Funkční požadavky specifikují funkcionality kladené přímo na vznikající program, zatímco nefunkční spíše určují omezení vlastností systému a architekturu návrhu. V následující kapitole budou popsány nároky na vznikají detekční nástroj, které vycházejí z obsahu zadání této práce a provedené analýzy.

2.6.1 Funkční požadavky

Kapitola popisuje funkční požadavky, které jsou od detekčního nástroje očekávány.

- **Sběr informací o provozu**

  Program bude umožňovat sběr dat na IoT bráně o aktuálním provozu z dostupných komunikačních rozhraní. Získané informace bude možné pomocí NEMEA [23] (Network Measurements Analysis) frameworku předávat do dalších modulů.

- **Detekce anomálií**

  Vytvořený detekční algorimus bude schopen na základě získaných dat odhalit definované anomálie, které reprezentují neočekávané změny v síti.

- **Konfigurace způsobu detekce**

  Detekční modul bude umožňovat nastavení parametrů pro jednotlivé zpracovávané položky pomocí konfiguračního souboru. Tyto parametry budou následně sloužit jako vstup pro vytvořený detektor.

- **Zpracování získaných informací**

  Kromě analýzy dat a hlášení nalezených incidentů bude také možné zpracovaná data pravidelně exportovat do dalších rozšiřujících modulů.

2.6.2 Nefunkční požadavky

Kapitola obsahuje nefunkční požadavky, které jsou kladené na výsledný nástroj.

- **Rozšiřitelnost**

  Architektura celého řešení bude navržena tak, aby bylo možné rozšíření o nové způsoby detekce anomálií a další typy provozních dat.

- **Flexibilita nasazení**

  Návrh způsobu detekce umožní fyzicky oddělit komponetu zajišťující sběr dat a komponetu vyhodnocující provoz. Díky tomu bude možné provádět analýzu i na IoT branách s velmi omezeným prostředky, protože tyto brány budou sloužit jako sondy, které budou posílat data do externího detektoru s dostatečnými výkonem k provedení analýzy.

- **Operační systém**

  Výsledné řešení bude implementováno a otestováno na operačním systému Ubuntu 16.04, na kterém bude nasazena IoT brána BeeeOn a framework NEMEA. Zároveň budou při návrhu a implementaci vybírány takové technologie, aby bylo možné vytvořený program spustit pod distribuci OpenWrt, které je velmi rozšířena mezi síťovými prvky.

2.7 Zvolené řešení

Obsahem této kapitoli je popis zvoleného řešení, které bylo na základě provedé analýzy určeno pro realizaci výsledného nástroje.

Vytvořený detekční algorimus bude zaměřen na senzorové IoT protkoly, protože v současné době neexistuje řešení, které by to umožňovalo. Pro analýzu budou použity protokoly Z-Wave a BLE, které jsou v dnešních sítích velmi rozšířené. Algoritmus bude umístěň přímo na IoT bráně, která je ideálním místem, protože se náchází velmi blízko koncových zařízení a má dostatečný výkon k vyhodnocování provozu. Konkrétně bude použita brána BeeeOn, protože v současné době jako jediná umožňuje sběr provozních dat o senzorových protokolech.

Pro účely detekce budou sbírány jen informace dostupné z lokálních rozhraní brány. Toto řešení je z hlediska reálného nasazení nejpravděpodobnější. Zároveň bude umožňěno rozšíření i pro další způsoby získávání dat.

Vyhodnocování dat bude probíhat metodou detekce anomálií, která je vhodná pro statistickou povahu dat , má menší nároky na množství dostupných prostředků a umožňuje rozpoznat i neznámé útoky.

Jako programovací jazyk bude použit C++, protože podporuje objektový přístup a úspornou implementaci na paměť i procesorový čas. Zároveň tento jazyk je použit i v bráně BeeeOn, tudíž bude usnadněna integrace. Pro zajištění flexibility a možnosti dalšího rozšiřování bude využito systému NEMEA, který také podporuje jazyk C++.

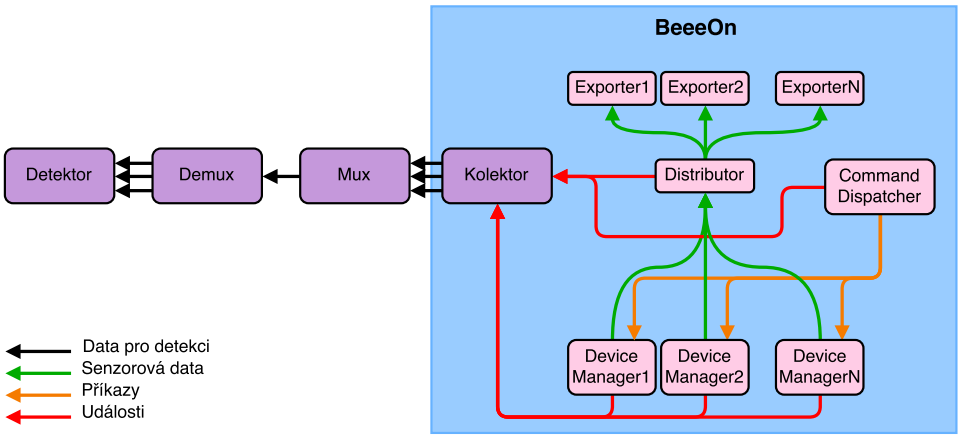
Kapitola 3

Návrh

Kapitola se zabývá návrhem způsobu detekce anomálií v IoT sítích, které s pomocí NEMEA systému rozšiřuje bránu BeeeOn. Nejprve je popsána celková architektura a možné způsoby nasazení. Dále jsou popsány jednotlivé komponenty detekčního systému. V závěru jsou identifikovány možné scénáře útoků.

3.1 Architektura

Cílem této práce je vytvoření programu, který bude schopen detekovat anomální provoz v IoT sítích. Vzhledem k hardwarovým omezením, které můžou na IoT branách nastat, je architektura navržena tak, aby měla co nejmenší nároky na dostupné prostředky. Schéma nasazení detekčního systému se nachází na obrázku 3.1



Obrázek 3.1: Architektura detekčního systému

Implementace BeeeOn brány obsahuje pro zpracování senzorových dat následující komponenty:

- **DeviceManager**: komponenta definovaná pro každý senzorový protokol, která implementuje veškerou komunikaci a zpracování dat
- **Distributor**: přijímá data od *DeviceManageru*, která nasledně předává příslušnému *Exporteru*
- **Exporter**: implementuje protokol, kterým jsou data odesílány z brány
- **CommandDispatcher**: komponenta, která přijímá uživatelské příkazy a distribuuje je cílovým komponentám

Každá z komponent v BeeeOn bráně navíc využívá návrhový vzor *Observer*, pomocí kterého jsou definovány události poskytující informace o každé komponentě. Tímto způsobem bude vytvořený kolektor zíkávat data o provozu, která převede do fomátu UniRec (Unified Record) a pomocí systému NEMEA je odešle k analýze.

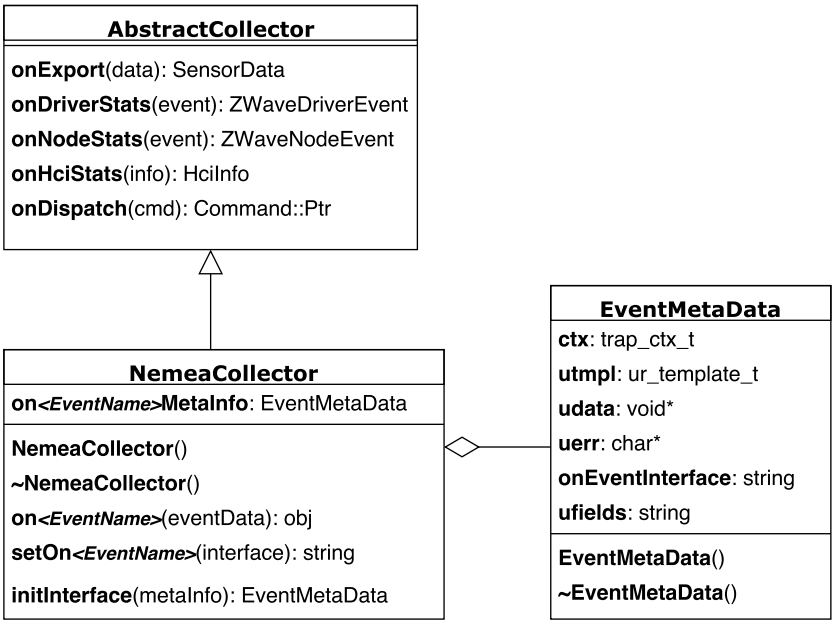
Exportované informace o provozu bude přijímat detektor, který následně provede jejich zpracování a vyhodnocení stavu. Všechny vytvořené komponety vytvořené pro detekci budou používat rozhraní NEMEA. Díky tomu bude možné komponenty flexibilně provozovat na jednom nebo více různých zařízeních. Oddělené nasazení je důležité zejména pro brány s omezenými prostředky, které mohou provádět pouze export dat a o vyhodnocení se bude starat odlišné zařízení s dostatečným výkonem. V případě provozování více bran lze brány používat jako exportéry a veškerá získaná data zpracovávat centrálně.

Kolektor pro každou událost vytvoří samostnatné výstupní rozhraní. Protože událostí může být velké množství, tak bude vytvořen modul *Mux*, který všechny výstupní rozhraní spojí do jednoho. Pro zpětné rozdělení na jednotlivá rozhraní budou dloužit modul *Demux*. Tyto moduly budou velmi užitečné zejména v případě kdy kolektor a detektor budou na rozdílných síťových prvcích, protože údaje o provozu budou přenášeny přes počítačovou síť a bude vyžadováno zabezpeční všech odchozích rozhraní. S využítím *Mux* a *Demux* modulů bude stačit zabezpečit pouze jedno sjednocené rozhraní.

Výhodou návrhu řešení je flexibilita a modularita. Každá komponenta vždy samostatně pokrývá pouze jednu část detekčního systému a se díky NEMEA se každá z nich může nacházet na různých síťových prvcích. Zároveň v případě potřeby lze vytvořit další moduly, které budou rozšiřovat stávající funcionalitu. Podrobnější popis návrhu nově vytvořených komponent detekčního systému se nachází v následujících kapitolách.

3.2 Kolektor

Úkolem kolektoru bude sběr dostupných dat a jejich odeslání k následné analýze. Informace o provozu budou sbírány z jednotlivých komponent BeeeOn brány, které jsou zpřístupňovány pomocí návrhového vzoru *Observer*. Z tohoto důvodu bude kolektor vždy přímou součástí BeeOn brány. Návrh struktury kolektoru je na obrázku 3.2



Obrázek 3.2: Návrh kolektoru provozních dat

AbstractCollector, je třída vytvořená v projektu BeeeOn, které shromažďuje všechny dostupné události. V současné době jsou k dispozici následující události:

- **onExport**: reprezentuje hodnoty dat přicházejících ze senzorů, která jsou odeslána vždy po jejich příjmu
- **onDriverStats**: v pravidelných intervalech generuje statistiky o Z-Wave sítí dostupné na komunikačních rozhraní
- **onNodeStats**: periodicky získává informace o jednotlivých prvcích Z-Wava sítě, které jsou dostupné pomocí OpenZWave knihovny
- **onHciStats**: v definovaných intervalech získává z komunikačního rozhraní pro BLE statistiky o provozu sítě
- **onDispatch**: umožňuje získávat zadané uživatelské příkazy

Vytvořený kolektor bude reprezentován třídou NemeaCollector, která bude potomkem třídy *AbstractCollector* a bude implementovat všechny její členské funce pro zachytávání událostí. Pro každou členskou funkci bude vytvořen setter, který při spuštění brány provede počáteční nastavení parametrů pro obsluhu dané události. Vstupním parametrem bude řetězec určující název výstupního rozhraní, který bude specifikovaný v konfiguračním souboru BeeeOn brány. Po nastavení počátečních parametrů jako je název výstupního rozhraní a seznam UniRec polí se zavolá členská funkce initInterface, jejímž cílem bude inicializace veškerých struktur vyžadovaných NEMEA frameworkem. Jako vstupní parametr bude přijímán objekt *EventMetaData* udržující veškeré informace potřebné k odesílání získaných dat.

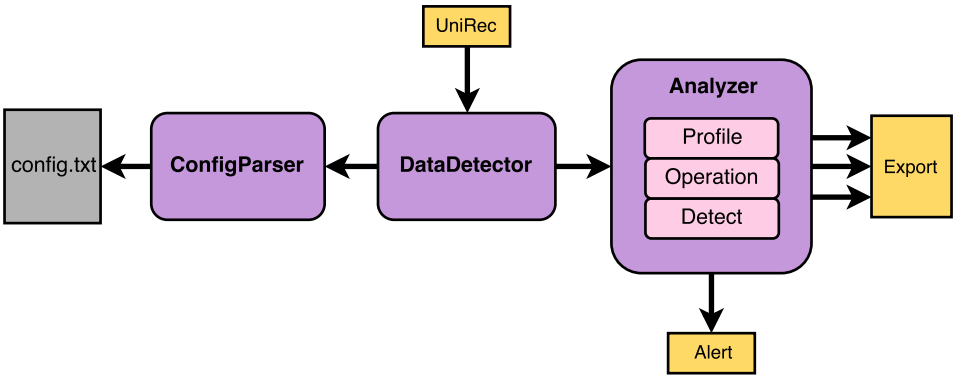
Instace objektu *EventMetaData* bude vytvořena jako členská proměnná třídy *NemeaCollector* pro každou členskou funkci události. Díky tomu bude možné každou událost zpracovávat nezávisle dle definovaných parametrů.

Definované konstruktory a destuktory budou sloužit jen pro vytvoření a destrukci potřebných struktur. Tímto bude zajištěň programovací idiom RAII (resource acquisition is initialization).

Veškeré odesílané údaje budou rozšířeny o časovou značku a číselný identifikátor, aby mohl detektor přesně určit původ případné anomálie.

3.3 Detektor

Detektor bude hlavní komponentou celého řešení, protože bude vyhodnocovat získaná dat. Vzhledem k zadaným požadavkům musí být návrh proveden tak, aby bylo umožněno budoucí rozšiřování. Návrh architektury se nachází na obrázku 3.3



Obrázek 3.3: Architektura detektoru

Komponenta *DataDetector* bude mít jedno vstupní rozhraní, kterým bude přijímat data z kolektoru. V případě potřeby zpracování více událostí z kolek-

toru lze využít již existující modul z NEMEA repozitáře, který dokáže sloučit více vstupů do jednoho konzolidovaného výstupu.

Dále se v rámci třídy *DataDetector* bude volat *ConfigParser*, který bude mít na starosti zpracování konfiguračního souboru. Konfigurační soubor bude obsahovat seznam parametrů pro každé UniRec pole, na základě kterých bude vytvořena instance třídy *Analyzer* pro analýzu dat.

Po provedení inicializace všech modulů bude DataDetector pouze přijímat data z kolektoru a předávat je ke zpracování komponentě *Analyzer*. Tato komponenta bude obsahovat tři hlavní části:

- **Profile**

  Bude obsahovat členské funkce, které se budou starat o připravení výchozího profilu sítě. Vytvořený profil bude složen z: průměru, klouzavého průměru, rozptylu a mediánu. Tyto hodnoty budou následně použity při porovnávání s aktuálním provozem v rámci detekčních metod. V případě potřeby bude možné profil rozšířit o další hodnoty.

- **Detect**

  Bude sdružovat všechny členské funkce určené pro detekci anomálií v získaných datech. Každá členská funkce bude reprezentovat jedno kritérium a bude možné budoucí rozšíření o další potřebné metody.

- **Operation**

  Cílém této části bude udržování veškerých použitých struktur. Bude se jednat o struktury pro konfiguraci, zpracovávaná data a odesílání získaných výsledků. Odesílat lze informaci o nalezené hrozbě, pro kterou se používá jedno společné rozhraní, nebo lze pravidelně exportovat definované části aktuálního profilu sítě pro účelý dalšího zpracování.

Díky konfiguraci organizované po jednotlivých UniRec polích je možné nastavit chování výše popsaných částí nezávisle na sobě pro každé pole.

3.3.1 Konfigurace modulu

Jedním z funkčních požadků je možnost konfigurace způsobu detekce, která se tak bude moci přizpůsobit cílovým podmínkám. Každá instance detektoru bude mít jeden konfigurační souboru, který bude popisovat jednotlivá UniRec pole, které se budou analyzovat.

Konfigurace bude v textové podobě ve formátu klíč a seznam hodnot. Textová podoba je velmi vhodná pro fázi ladění a vývoje nástroje, protože jsou možné manuální úpravy. Cílem ovšem je generovat konfiguraci pomocí externího nástroje, který by mohl být součástí řídící vrstvy IoT brány. Generátor nastavení bude velmi vhodný pro koncové uživatele, protože si budou moci přehledně vybrat své parametry bez znalosti všech závislotí, které textová

konfigurace ani modul pro její zpracování nebude hlídat. Externí nástroj pro definování nastavení analýzy nebude součástí vytvořeného řešení.

Klíčem bude vždy název UniRec pole, které se bude nacházet v příchozích datech. Tento klíč bude oddělen dvojtečnou, za kterou bude sekvence parametrů, které budou odděleny středníkem . Každý klíč se bude nacházet na samostatném řádku. Řádky prázdné a začínající znakem # budou považovány za komentář.

Definované parametry budou rozděleny do následujících hlavních skupin:

- **General**

  Tato skupina bude obsahovat obecné volby pro definování časové řady. Pomocí číselných parametrů bude možné určit velikost řady, délku učící fáze, během které je vytvořen profil sítě a její délka musí být stejná nebo delší než velikost časové řady, a rozsah ignorovací části, která je vhodná k odfiltrování počáteční výměny zprav při navazování spojení.

  Dalším parametrem bude způsob ukládání dat do časové řady. K dispozici budou tři režimy. Prvním z nich bude *simple*, který přijatou hodnotu pouze uloží. Dalším bude *delta*, který bude vkládat pouze rozdíly dvou po sobě jdoucích hodnot. Toto je vhodné zejména u časových značek nebo počtu odeslaných zpráv, protože bude vhodnější jednotlivé přírůstky než pouhé absolutní hodnoty. Poslední možností bude *average*, kdy nová příchozí hodnota bude odečtena od aktuálního průměru a výsledek bude uložen.

  Poslední možností bude číselná hodnota určující rozmezí mezi pravidelnými kontrolami přijetí dat a číselný parametr definující periodu pro exportování vybraných parametrů.

- **Profile**

  Zápis této skupiny je identifikovaný klíčovým slovem *profile*, který má uvnitř kulatých závorek vyjmenované jednotlivé položky, které budou součástí vytvořeného profilu. Ve vytvořeném řešení budou k dispozici následující volby: *average* (průměr), *variance* (rozptyl), *median* (medián) a *cum_average* (klouzavý průměr).

- **Profile Values**

  Pro každou položku definovanou v rámci profilu musí být uveden seznam jejich detekčních parametrů. Zápis bude probíhat pomocí klíčového slova z profilu, které bude mít jednotlivé hodnoty specifikované uvnitř kulatých závorek. Zde bude možné určit minimální a maximálni soft a hard limitých, kterých daná položka aktuálního profilu může dosahovat. Pro soft limit bude možné určit povolenou délku, po kterou může být soft limit překročen (grace period).

Dalšími parametry budou číselné hodnoty určující hranice pro minimální a maximální velikost změny sledované položky.

Zadání veškerých hodnot nebude nutné a tudíž bude záležet na uživateli, které způsoby detekce bude chtít využít. V případě nedefinování parametru je zapsán znak -.

- **Export**

  Poslední nastavitelnou možností bude export vybraných dat z profilu. Zápis bude probíhat pomocí klíčového slova *export*, který v kulatých závorkách bude mít seznam položek z profilu. Zároveň v případě využití této možnosti bude nutné ve skupině *general* určit časovou periodu odesílání. Názvy výstupních trap rozhraní budou odvozené od klíčů, které se nacházejí před dvojtečkou v konfiguraci.

Příklad záznamu s konfigurací je na obrázku 3.4. Nastavení formou souboru bude jedninou volbou pro přizpůsobení detekce. Dále při spuštění programu bude vyžadován jeden přepínač *-i*, který je vyžadován knihovnou trap a bude definovat název vstupního rozhraní pro příchozí data a název výstupního rozhraní pro detekované incidenty.

```
TIME: 5;12;3;delta;20;30; profile(average,variance,); average(0,6,-1,10,5,2,0.2,0,0);
    variance(-,-,-,-,-,0.5,5,0,0); export(variance,);
```

Obrázek 3.4: Popis konfiguračních parametrů pro jeden záznam

3.3.2 Použité struktury

Velmi důležitou částí návrhu bude určení formátu použitých struktur, které budou urdžovat veškeré údaje nutné pro analýzu. V rámci dektektoru budou potřeba udržovat informace pro následující položky:

- **Konfigurační parametry**

  Tato struktura bude hierarchicky ukládat veškeré hodnoty z konfiguračního souboru a také údaje nutné k provedení analýzy. V první úrovni se budou nacházet názvy jednotlivých UniRec polí, které budou fungovat jako klíče. V druhé vrstvě se bude nacházet vždy stejná sekvence podklíčů, které budou určovat typ uchovávaných dat. Budou zde uloženy jednotlivé skupiny z konfiguračního souboru 3.3.1, které budou označeny stejnojmenými identifikátory. Hodnotou těchto skupin budou specifikované parametry detekce dle zapsané konfigurace. Dále zde budou podklíče s názvem *metaProfile* a *metaData*, které budou mít stejné hodnoty. Tyto hodnoty budou obsahovat vypočtené údaje z příslušné časové

řady a zároveň budou umožňovat ukládání dat, která budou nutná k provedení výpočtu a vkládání nových prvků. Rozdíl mezi *metaProfile* a *metaData* bude typ profilu, pro který budou určeny. Podklíč *metaProfile* bude uchovávat údaje pro základní profil, který bude vytvořen během učící fáze a bude sloužit pro určení anomálie. Zatímco hodnoty v *metaData* budou vztaženy k aktuálnímu profilu sítě, který se mění s každým nový prvkem v časové řadě.

- **Senzorová data**

  Cílem této struktury bude hierarchicky ukládat přicházející data z kolektoru do časových řad. V první úrovni se jako v předchozím případě budou nacházet názvy jednotlivých UniRec polí. Protože jedno UniRec pole může reprezentovat data z více různých senzorů nebo bran, tak se na další úrovni nachází identifikátor zdroje dat. Předpokládá se, že každá použitá brána a senzor má v rámci jedné instance detektoru unikatní identifikátory. Na třetí úrovni se již vyskytuje definovaná časová, která ukládá data podle zadané konfigurace.

- **Nastavení pro export**

3.3.3 Způsob analýzy

popis analýzy stavový diagram zpracování soft hard limit, data change, ... časové řady

3.4 Multiplexor a demultiplexor

3.5 Scénáře útoků

Odvození na základě experimentů a analýzy

Kapitola 4

Realizace

Kapitola 5

Testování

Závěr

Literatura

[1] Statista: Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025. Statista. [online], 2016 [cit. 2018-01-14]. Dostupné z: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] Milici, S.; Lázaro, A.; Villarino, R.; aj.: Wireless Wearable Magnetometer-Based Sensor for Sleep Quality Monitoring. *IEEE Sensors Journal*, 2018: s. 2145–2152.

[3] Whitehead, D. E.; Owens, K.; Gammel, D.; aj.: Ukraine cyber-induced power outage: Analysis and practical mitigation strategies. In *2017 70th Annual Conference for Protective Relay Engineers (CPRE)*, 2017, s. 1–8.

[4] Statista: Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. Cisco. [online], 2015 [cit. 2018-01-15]. Dostupné z: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[5] BeeeOn: Main Page. beeeon.org. [online], 2017 [cit. 2018-02-06]. Dostupné z: https://beeeon.org

[6] Zhao, K.; Ge, L.: A Survey on the Internet of Things Security. In *2013 Ninth International Conference on Computational Intelligence and Security*, 2013, s. 663–667.

[7] Pacheco, J.; Hariri, S.: IoT Security Framework for Smart Cyber Infrastructures. In *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, s. 242–247.

[8] OASIS: MQTT Version 3.1.1 Plus Errata 01. OASIS. [online], 2015 [cit. 2018-03-08]. Dostupné z: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html

[9] dc square: MQTT Essentials Wrap-Up. HiveMQ. [online], 2018 [cit. 2018-03-08]. Dostupné z: https://www.hivemq.com/blog/mqtt-essentials-wrap-up

[10] dc square: MQTT Security Fundamentals. HiveMQ. [online], 2018 [cit. 2018-03-08]. Dostupné z: https://www.hivemq.com/mqtt-security-fundamentals/

[11] Shelby, Z.; Hartke, K.; Bormann, C.: The Constrained Application Protocol (CoAP). RFC7252. [online], 2014 [cit. 2018-03-08]. Dostupné z: https://tools.ietf.org/html/rfc7252

[12] Alliance, Z.-W.: Z-Wave Plus Certification. Z-Wave Alliance. [online], 2018 [cit. 2018-03-17]. Dostupné z: https://z-wavealliance.org/z-wave_plus_certification/

[13] OpenZWave: OpenZWave. OpenZWave. [online], 2018 [cit. 2018-03-17]. Dostupné z: http://www.openzwave.net/

[14] Krejčí, R.; Hujňák, O.; Švepeš, M.: Security survey of the IoT wireless protocols. In *2017 25th Telecommunication Forum (TELFOR)*, 2017.

[15] Designs, S.: Introduction to Z-Wave Specifications. ZWave. [online], 2016 [cit. 2018-03-18]. Dostupné z: http://zwavepublic.com/specifications

[16] Fouladi, B.; Ghanoun, S.: Security Evaluation of the Z-Wave Wireless Protocol. In *BlackHat Conference, Las Vegas, NV, USA*, 2013 [cit. 2018-03-18].

[17] Fouladi, B.; Ghanoun, S.: EZ-Wave. Github. [online], 2016 [cit. 2018-03-18]. Dostupné z: https://github.com/cureHsu/EZ-Wave

[18] Rose, A.; Ramsey, B.: Picking Bluetooth Low Energy Locks from a Quarter Mile Away. In *DEF CON, Las Vegas, NV, USA*, 2016 [cit. 2018-03-18].

[19] Jasek, S.: Gattacking Bluetooth Smart Devices. SecuRing. [online], 2016 [cit. 2018-03-18]. Dostupné z: http://gattack.io/whitepaper.pdf

[20] Seri, B.; Vishnepolsky, G.: BlueBorne. Armis. [online], 2017 [cit. 2018-03-18]. Dostupné z: http://go.armis.com/hubfs/BlueBorne%20Technical%20White%20Paper-1.pdf

[21] TechTarget: Do you need an IDS or IPS, or both? TechTarget. [online], 2009 [cit. 2018-03-19]. Dostupné z: http://searchsecurity.techtarget.com/Do-you-need-an-IDS-or-IPS-or-both

[22] CZ.NIC: PaKon - sledování sítě (Parental Control). CZ.NIC. [online], 2018 [cit. 2018-03-20]. Dostupné z: `https://doc.turris.cz/doc/cs/howto/pakon`

[23] Cejka, T.; Bartos, V.; Svepes, M.; aj.: NEMEA: A framework for network traffic analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*, 2016, doi:10.1109/CNSM.2016.7818417.

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS