VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Adam Detko

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

KYBERNETICKÁ BEZPEČNOST V SYSTÉMECH TYPU PUBLISH-SUBSCRIBE
CYBER-SECURITY IN SYSTEMS OF PUBLISH-SUBSCRIBE TYPE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Adam Detko

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jiří Pokorný

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Adam Detko ***ID:*** 186044
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Kybernetická bezpečnost v systémech typu publish-subscribe

POKYNY PRO VYPRACOVÁNÍ:

Bude provedena analýza a srovnání dostupných implementací z pohledu bezpečnosti, funkcionalit a další (mj. budou zahrnuty implementace OMG-DDS Security, Vortex OpenSplice DDS a OpenDDS). Dále se student zaměří na OpenDDS. Přehledně budou zpracovány možnosti konfigurace bezpečnostních prvků a navrhnuty komplexní komunikační scénáře pro služby publish-subscribe. Následně budou navrženy metody pro ověření bezpečnosti této implementace, společně s návrhem scénářů simulujících bezpečnostní incidenty. Výsledkem studentovi práce budou implementované komunikační scénáře (regulérní provoz), společně s neregulérním provozem (bezpečnostní incidenty), což vyústí ve vytvoření obecného nástroje pro ověření bezpečnosti a funkčnosti systémů typu DDS.

DOPORUČENÁ LITERATURA:

[1] SCHLESSELMAN, Joseph M.; PARDO-CASTELLOTE, Gerardo; FARABAUGH, Bert. OMG data-distribution service (DDS): architectural update. In: Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE. IEEE, 2004. p. 961-967.

[2] CALVO, Isidro, et al. Towards a OMG DDS communication backbone for factory automation applications. In: Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on. IEEE, 2011. p. 1-4.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Jiří Pokorný

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto práca sa zaoberá službou distribúcie dát a je rozčlenená na tri časti. Prvá časť práce sa zameriava na teoretické poznatky o službe distribúcie dát. Postupne je v tejto časti predstavený model využívaný v službe distribúcie dát, kvalita služby a protokol RTPS, ktorý bol špeciálne vyvinutý pre potreby služby distribúcie dát. Najobsiahlejšia časť úvodnej teoretickej časti sa zaoberá bezpečnosťou služby distribúcie dát. Na záver prvej časti práce sú predstavené vybrané implementácie služby distribúcie dát. Druhá časť práce sa zaoberá praktickým využitím služby distribúcie dát, pričom pre praktickú časť je zvolená implementácia OpenDDS. Táto časť sa spočiatku zaoberá návrhom základných komunikačných scenárov, za ktorým nasleduje návrh komplexných komunikačných scenárov. V ďalších kapitolách v rámci praktickej časti je práca zameraná na overenie bezpečnosti využívanej implementácie služby distribúcie dát, návrh bezpečnostných incidentov a ich následnú simuláciu. Náplňou záverečnej časti práce je predstavenie nástroja navrhnutého pre túto prácu, ktorý slúži k realizácii vybraných typov útokov.

KĽÚČOVÉ SLOVÁ

DDS, RTPS, QoS, publikovanie, odber, bezpečnosť, incident, OpenDDS

ABSTRACT

This thesis deals with data distribution service and is divided into three parts. The first part of this thesis focuses on theoretical knowledge of data distribution service. This section gradually introduces the model used in data distribution service, quality of service and the RTPS protocol, which was specially developed for the needs of data distribution service. The most comprehensive part of the theoretical part deals with security in data distribution service. At the end of the first part of this thesis, selected impelementations of data distribution service are introduced. The second part of this thesis deals with practical use of data distribution service, while OpenDDS implementation is chosen for the practical part. This part initially deals with the design of basic communication scenarios, followed by the design of complex communication scenarios. In the next chapters, within the practical part, the thesis is focused on verifyng security of the data distribution service selected implementation, design of security incidents and their subsequent simulation. Content of the final part of this thesis is the introduction of a tool designed for this thesis, which is used to implement selected types of attacks.

KEYWORDS

DDS, RTPS, QoS, publish, subscribe, security, incident, OpenDDS

DETKO, Adam. *Kybernetická bezpečnost v systémech typu publish-subscribe*. Brno, Rok, 86 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Jiří Pokorný

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Kybernetická bezpečnost v systémech typu publish-subscribe“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno . . . .

podpis autora

POĎAKOVANIE

Ďakujem vedúcemu práce pánovi Ing. Jiřímu Pokornému za vedenie práce. Rovnako ďakujem pánovi Ing. Radkovi Fujdiakovi, Ph.D. za jeho čas venovaný počas celého akademického roku a jeho odborné rady počas konzultácií.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Táto diplomová práca sa zaoberá službou distribúcie dát, ktorá využíva model publikovanie-odber. Jedná sa o štandard, ktorý je využívaný systémami v reálnom čase [1]. Bližšie je táto práca zameraná na možnosti modelu publikovanie-odber a následný návrh komunikácie využívajúcej spomínaný model. Navrhnuté sú dva typy komunikačných scenárov a síce základné a komplexné komunikačné scenáre. Okrem návrhu komunikácie rieši táto práca aj simuláciu navrhnutej komunikácie. V neposlednom rade sa táto práca zaoberá aj návrhom bezpečnostných incidentov a ich následnou simuláciou.

V úvodnej časti tejto práce, ktorá je zameraná na teoretické poznatky o službe distribúcie dát, sa v jej prvej kapitole nachádza všeobecný popis služby distribúcie dát a definícia rozdielu medzi dátovo a objektovo zameraným komunikačným modelom. V druhej kapitole je uvedený dátovo zameraný model publikovanie-odber s definíciou entít, ktoré sú v ňom definované. Tretia kapitola sa zameriava na kvalitu služby, pričom dôraz v rámci tejto kapitoly je kladený na model kvality služby požiadavka verzus ponuka, ktorý je využívaný v službe distribúcie dát. V rámci štvrtej kapitoly je predstavený protokol RTPS. Piata kapitola sa zaoberá bezpečnosťou v službe distribúcie dát. Jedná sa o najobsiahlejšiu kapitolu v časti zameranej na teoretické poznatky, ktorá postupne predstavuje architektúru zabezpečenia v službe distribúcie dát, bezpečnostný model a bezpečnostné hrozby, ktoré sa v službe distribúcie dát môžu vyskytnúť. Šiesta kapitola teoretickej časti je zameraná na analýzu vybraných implementácií služby distribúcie dát. Konkrétne sú analyzované implementácie služby distribúcie dát Vortex OpenSplice, OpenDDS a Connext DDS. V rámci tejto kapitoly sa nachádza zhrnutie bezpečnosti vybraných implementácii služby distribúcie dát.

V praktickej časti tejto diplomovej práce je v jej úvodných dvoch kapitolách uvedený hardware, spolu so schémou zapojenia použitých zariadení v práci, a následne postup kompilácie vybranej implementácie služby distribúcie dát na zariadeniach využitých v tejto práci. Ďalšia kapitola praktickej časti rieši návrh základných komunikačných scenárov a popis jednotlivých navrhnutých komunikačných scenárov. Štvrtá kapitola časti zameranej na praktické využitie služby distribúcie dát predstavuje možný spôsob konfigurácie správania komunikácie, spôsob spúšťania procesov jednotlivých entít a simuláciu navrhnutých základných komunikačných scenárov. V ďalších dvoch kapitolách praktickej časti sa nachádza návrh komplexných komunikačných scenárov a ich následná simulácia. Siedma kapitola praktickej časti obsahuje overenie bezpečnosti vybranej implementácie služby distribúcie dát. Posledné kapitoly praktickej časti sú zamerané na návrh bezpečnostných incidentov a ich simuláciu.

Záverečná časť práce predstavuje nástroj pre realizáciu vybraných typov útokov.

1 Služba Distribúcie Dát

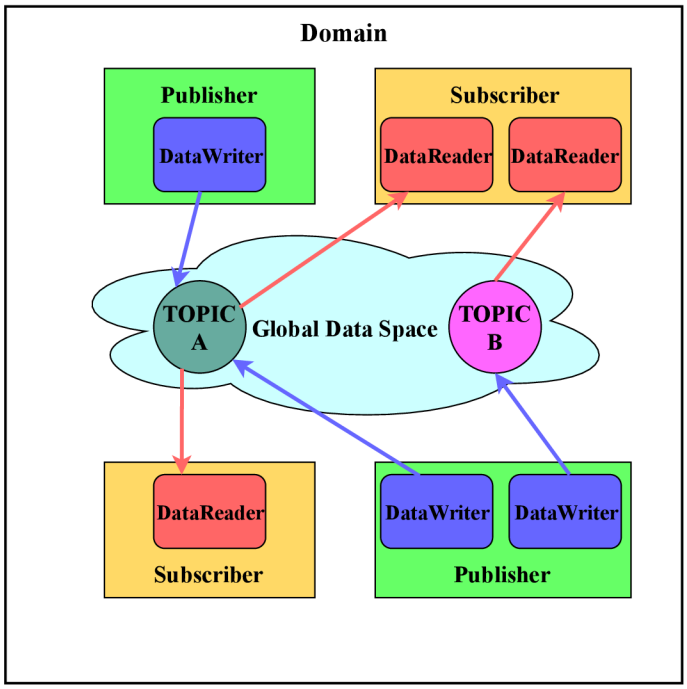
Data Distribution Service (DDS) je protokol medzivrstvy a API štandard pre prenos dát s využitím modelu publikovanie-odber (publish-subscribe) zo skupiny Object Managment Group (OMG). Medzi najvýznamnejšie výhody DDS patrí dátová konektivita s nízkou latenciou, vysoká spoľahlivosť a škálovateľná architektúra, čo je výhodné pre požiadavky aplikácií internetu vecí (IoT) [1, 2].
Distribuované aplikácie v reálnom čase sú niekedy viac zamerané na dáta ako na služby, čo znamená, že primárnym cieľom pre účastníkov v distribuovanom systéme je skôr šírenie aplikačných dát než prístup k zdieľaným službám. Dodávatelia a/alebo spotrebitelia aplikačných dát nemusia byť v čase návrhu známy a môžu sa meniť počas doby vykonávania aplikácie [3]. Prevláda teda názor, že v prípade distribuovaných aplikácií v reálnom čase zameraných na dáta je účinnejšia realizácia prostredníctvom komunikačného modelu publikovanie-odber ako využitie modelu žiadosť-odpoveď (request-response). DDS pre systémy v reálnom čase rieši požiadavky na výkonnosť a požiadavky v reálnom čase na distribuované aplikácie zamerané na dáta. Pre efektívne šírenie dát je využívaný Data-Centric Publish-Subscribe (DCPS) model. Tento model vychádza z konceptu Global Data Space (GDS), ktorý je prístupný všetkým zainteresovaným aplikáciám, ktoré môžu do tohto dátového priestoru dáta pridávať alebo ich z neho získavať [4]. Nad DCPS môže fungovať vyššia voliteľná vrstva Data local reconstruction layer (DLRL), ktorej účelom je poskytnúť priamejší prístup k výmene dát, čo umožňuje ľahšiu integráciu do aplikačnej vrstvy [5].

1.1 Dátovo a objektovo zameraný komunikačný model

Pre pochopenie potreby štandardu DDS je nutné definovať základné rozdiely medzi dátovo zameraným (Data-Centric) a objektovo zameraným (Object-Centric) komunikačným modelom. V prípade DDS sa jedná o kontrastný model k už existujúcemu známemu modelu CORBA. U modelu CORBA sa dáta oznamujú nepriamo prostredníctvom argumentov vo vyvolaní metódy alebo prostredníctvom ich návratových hodnôt. U DDS sú prístupy k vyvolávaniu metód na vzdialených objektoch prostredníctvom rozhrania definovaného v jazyku popisujúceho rozhranie (IDL).
V mnohých aplikáciách v reálnom čase je komunikačný model často modelovaný ako čisto dátovo orientovaná výmena, kde aplikácie publikujú dáta, ktoré sú k dispozícií vzdialeným aplikáciám, zainteresovaným o dané dáta. Vzhľadom k tomu je záujem o čo najefektívnejšiu distribúciu dát s minimálnymi režijnými nákladmi [1].

1.2 Model Data-Centric Publish-Subscribe

Model Data-Centric Publish-Subscribe (DCPS) je súčasťou štandardu OMG DDS. DCPS rozširuje model publikovanie-odber tak, aby riešil špecifické potreby aplikácií v reálnom čase a poskytuje niekoľko mechanizmov, ktoré vývojárom aplikácií umožňujú kontrolu funkčnosti komunikácie [6]. Pri realizácií sú využité objekty *Publisher* a *DataWriter* na strane odosielateľa a objekty *Subscriber* a *DataReader* na strane prijímateľa [5]. DCPS pozostáva z jednej alebo viacerých dátových domén, z ktorej každá obsahuje skupinu účastníkov (publishers a subscribers), ktorí komunikujú prostredníctvom DDS. Každý účastník patrí do domény a každý proces má jedného účastníka domény pre každú dátovú doménu, ktorej je členom. V rámci akejkoľvek dátovej domény sú dáta identifikovateľné podľa témy (Topic), ktorá je segmentom domény špecifickým pre daný typ, ktorý umožňuje účastníkom komunikácie (publishers a subscribers) jednoznačne sa odkazovať na dáta [3]. Jednotlivé entity modelu DCPS a ich interakcie sú zobrazené na obrázku 1.1 [7] a sú popísané v ďalšej časti textu.



Obr. 1.1: Príklad diagramu objektov modelu DCPS.

Publisher. *Publisher* reprezentuje DCPS objekt, ktorý je zodpovedný za distribúciu dát a ich šírenie všetkým relevantným odberateľom (objekt typu *Subscriber)* dát [8]. Objekt typu *Publisher* môže publikovať dáta rôznych dátových typov [5].

DataWriter. Objekt typu *DataWriter* musí aplikácia použiť na to, aby objektu

typu *Publisher* oznámila existenciu a hodnotu dát daného typu. Rovnako to platí aj pre oznámenie zmeny dát daného typu [5]. Po oznámení hodnôt je zodpovednosťou objektu typu *Publisher* rozhodnúť, kedy je vhodné vykonať distribúciu dát. Objekt typu *Publisher* urobí rozhodnutie na základe vlastnej kvality služby (QoS) alebo kvality služby pripojenej k zodpovedajúcemu objektu typu *DataWriter* [5]. Každý objekt typu *DataWriter* je viazaný na konkrétnu tému (*Topic*) [8].

Subscriber. *Subscriber* je DCPS objekt zodpovedný za prijímanie publikovaných dát a ich sprístupnenie prijímajúcej aplikácií, vzhľadom na QoS objektu typu *Subscriber* [9]. *Subscriber* môže prijímať dáta rôznych dátových typov [5].

DataReader. Objekt typu *DataReader* je použitý pre prístup k prijatým dátam a ich následnému predaniu konkrétnemu objektu typu *Subscriber* [8]. Objekty typu *Subscriber* a *DataReader* sú medzi sebou asociované, čo znamená, že každý objekt typu *DataReader* je priradený k jednému objektu typu *Subscriber*. Rovnako ako objekt typu *DataWriter*, aj každý objekt typu *DataReader* je viazaný na tému [5].

Topic. Téma (*Topic*) asociuje jedinečný názov v systéme, dátový typ a kvalitu služby súvisiacu so samotnými dátami. Objekt typu *Topic* je základným prostriedkom interakcie medzi aplikáciami publikujúcimi dáta a aplikáciami odoberajúcimi dáta. Pri publikovaní dát, proces publikovania vždy špecifikuje tému [8]. Pri odoberaní dát sa objekt typu *Subscriber* dotazuje na dáta prostredníctvom témy [9]. Hodnota dát asociovaných s objektom typu *Topic* sa môže časom zmeniť a preto sa rôzne hodnoty objektu typu *Topic*, ktoré sú distribuované medzi aplikáciami, nazývajú vzorky (*samples*). Pre unikátnu identifikáciu inštancií (*instances*) rovnakej témy sa využíva kľúč (*key*). V prípade, že objekt typu *Topic* neobsahuje kľúč, existuje iba jedna inštancia danej témy [10].

Domain. Základnou rozdeľovacou jednotkou v rámci modelu DCPS je doména (*Domain*). Každá z ďalších entít patriacich do konkrétnej domény sa môže integrovať iba s entitami patriacimi do rovnakej domény. Doména je teda určitá distribuovaná koncepcia spájajúca všetky aplikácie, ktoré sú schopné vzájomne komunikovať [8].

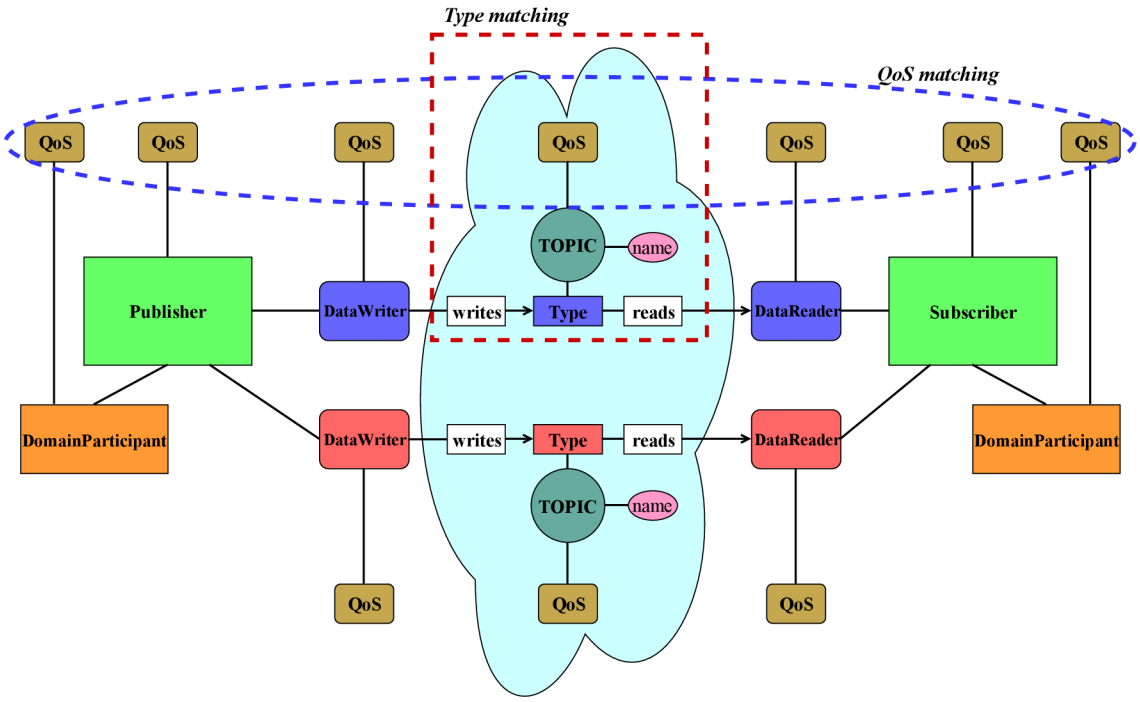
DomainParticipant. *DomainParticipant* predstavuje lokálne členstvo aplikácie v rámci domény [5]. Jedná sa v podstate o vstupný bod pre aplikáciu v konkrétnej doméne. *DomainParticipant* vlastní objekt typu *Publisher*, *Subscriber* a *Topic*. Jeho dôležitou funkciou je izolovanie množiny aplikácií, ktoré zdieľajú rovnakú fyzickú sieť. To znamená, že viaceré aplikácie bežiace na rovnakej skupine hostiteľov, ale na rôznych doménach sú vzájomne úplne izolované [11].

Global Data Space. Global Data Space (GDS) je dátový priestor, ktorý je prístupný všetkým zainteresovaným aplikáciám a umožňuje im zdieľať dáta s úplnou kontrolou spoľahlivosti a načasovania [12]. Aplikácie, ktoré chcú do tohto priestoru dáta pridávať deklarujú záujem stať sa objektami typu *Publisher* a naopak aplikácie, ktoré chcú z GDS dáta odoberať deklarujú záujem stať sa objektami typu *Subscriber*. Pri publikovaní dát do GDS, middleware propaguje informáciu o publikovaní dát všetkým zainteresovaným objektom typu *Subscriber* [5].

1.3 Kvalita služby v službe distribúcie dát

Kvalita služby (QoS) v DDS je súbor konfigurovateľných parametrov, ktoré riadia správanie systému DDS. Medzi takéto parametre patria napríklad odolnosť voči chybám, spoľahlivosť komunikácie a ďalšie [13]. Vo všeobecnosti sa QoS skladá z niekoľkých politík QoS (QoS policies), ktorými je možné riadiť entity modelu DCPS. Každá politika kvality služby je nezávislým popisom, ktorý asociuje názov s hodnotou [5]. Popis kvality služby prostredníctvom zoznamu nezávislých QoS politík vedie k väčšej flexibilite. Tým, že sa DDS spolieha na veľmi bohatý súbor politík QoS, môže okrem iného kontrolovať a obmedzovať šírku pásma siete a pamäť, rovnako aj mnoho nefunkčných vlastností objektov typu Topic ako sú spoľahlivosť alebo včasnosť [14].

DDS využíva prístup porovnania QoS typu požiadavka verzus ponuka (request versus offer), ktorý je znázornený na obrázku 1.2 [15]. Objekt typu *DataReader* môže komunikovať s objektom typu *DataWriter* iba vtedy, ak kvalita služby, ktorú požaduje objekt typu *DataReader* pre danú tému nie je striktnejšia ako kvalita služby, s ktorou sú dáta produkované objektom typu *DataWriter*. V prípade splnenia podmienky, kedy objekt typu *DataReader* požaduje menej striktnú kvalitu služby, komunikácia prebieha s kvalitou služby, ktorú požadoval objekt typu *DataReader*. Pri odoberaní dát, o ktoré má záujem objekt typu Subscriber sa porovnávajú tieto dáta s typom a názvom témy (type matching), rovnako aj s kvalitou služby ponúkanou a žiadanou objektami typu DataWriter a DataReader (QoS matching). Tento porovnávací mechanizmus zabezpečuje, že dátové typy sú zachované od začiatku do konca komunikácie kvôli zhode s typom témy a rovnako, že zachované sú aj invarianty QoS [15]. Najdôležitejšie politiky kvality služby (QoS policies) v službe distribúcie dát sú uvedené v [8] a [15].



Obr. 1.2: QoS model požiadavka verzus ponuka.

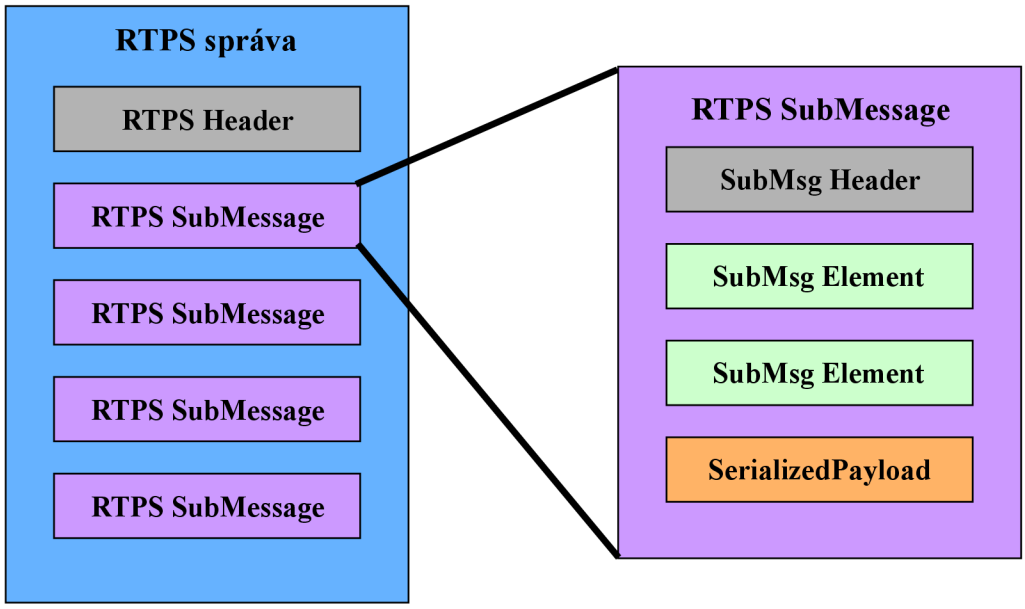
1.4 Real-Time Publish Subscribe (RTPS)

Protokol RTPS je protokol pre best effort a spoľahlivú komunikáciu typu publikovanie-odber cez nespoľahlivé transportné protokoly ako je UDP a môže fungovať cez multicast aj unicast [19]. RTPS bol špeciálne vyvinutý pre podporu jedinečných požiadaviek systémov služby distribúcie dáta a štandardizovaný bol organizáciou OMG ako interoperabilný protokol pre implementáciu DDS. Hlavnými funkciami protokolu RTPS sú [20]:

- **výkonnostné nastavenia a nastavenia kvality služby**, ktoré umožňujú best effort a spoľahlivú komunikáciu typu publikovanie-odber pre aplikácie v reálnom čase pomocou štandardných IP sietí,
- **odolnosť voči chybám**,
- **možnosť rozšírenia a vylepšenia protokolu** o nové služby bez narušenia kompatibility a interoperability,
- **konektivita typu plug-and-play** pre nové aplikácie a služby, ktorá umožňuje automatické objavenie v sieti a možnosť pripojenia alebo odpojenia sa kedykoľvek od danej siete bez následnej opätovnej konfigurácie,
- **konfigurovateľnosť** umožňujúca vyváženie požiadaviek na spoľahlivosť a včasnosť pre dodanie dát,

- **modularita**,
- **škálovateľnosť**, ktorá umožňuje systémom prispôsobenie pre veľmi veľké siete,
- **typová bezpečnosť**, ktorá slúži k predídeniu chýb pri programovaní aplikácií, ktoré by mohli ohroziť prevádzku vzdialených uzlov.

V bezpečnom systéme, v ktorom je zohľadnená efektivita a latencia správ je potrebné presne definovať, čo je potrebné zabezpečiť, keďže niektoré aplikácie vyžadujú dôvernosť iba u obsahu dát a napríklad informácia o objavení správy alebo spoľahlivosti je prípustná, ak je chránená pred modifikáciou. Pre zmenu iné aplikácie môžu chcieť takýto typ informácie zachovať dôverný. Na obrázku 1.3 [18] je objasnená štruktúra RTPS správy a RTPS podsprávy (Submessage). RTPS podspráv môže RTPS správa obsahovať hneď niekoľko [18].



Obr. 1.3: Štruktúra RTPS správy.

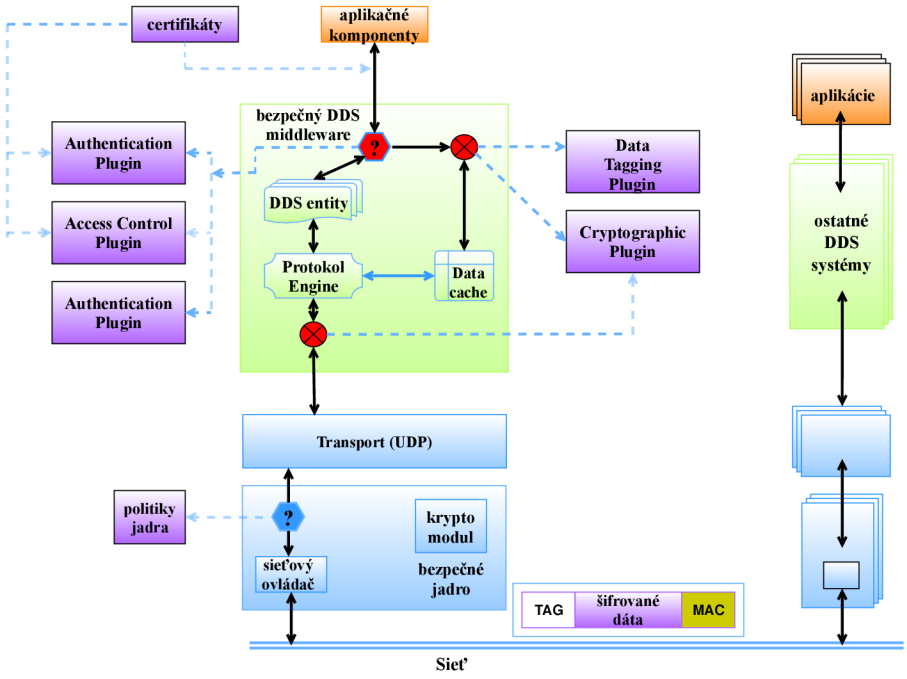
Štruktúra RTPS správy sa skladá zo záhlavia RTPS s nemennou veľkosťou (RTPS Header), za ktorým nasleduje variabilné množstvo RTPS podspráv (SubMessage). Každá RTPS podspráva sa následne skladá zo záhlavia (SubMsg Header) a variabilného množstva prvkov podsprávy (SubMsgElement). Prvkov RTPS podsprávy (SubMsgElement) existuje niekoľko druhov, ktoré obsahujú informácie ako sú poradové číslo správy, jedinečné identifikátory pre objekty typu *DataReader* a *DataWriter*, zdrojové časové razítka, kvalita služby a ďalšie. Špeciálnym prvkom RTPS podsprávy je SerializedPayload, ktorý prenáša dáta odoslané aplikáciou služby distribúcie dát [18, 20].

1.5 Zabezpečenie služby distribúcie dát

Nakoľko sa stále viac systémov internetu vecí (IoT) buduje s využitím služby distribúcie dát je nevyhnutná potreba zabezpečenia výmeny dát, najmä v prípade ak sa jedná o citlivé dáta [16]. Avšak štandard OMG DDS sám o sebe nešpecifikuje bezpečnosť komunikácie. Štandard DDS Security rieši problém absencie interoperabilného mechanizmu viacerých výrobcov, ktorý zabezpečuje bezpečnú komunikáciu DDS aplikácií. Interoperabilita viacerých výrobcov DDS je možná prostredníctvom špecifikácie OMG DDS–RTPS (Real-Time Publish Subscribe), ktorá definuje štandardizovaný mechanizmus zisťovania a drôtový protokol pre službu distribúcie dát [17]. DDS Security štandard definuje bezpečnostný model a Service Plugin Interface (SPI) architektúru pre kompatibilné implementácie DDS. Rovnako tento štandard definuje aj súbor piatich zabudovaných implementácií týchto Service Plugin Interfaces (SPIs). Používanie zmienených SPIs umožňuje používateľom DDS prispôsobiť správanie a technológie, ktoré jednotlivé DDS implementácie používajú pre [18]:

- autentifikáciu (Authentication Service Plugin),
- kontrolu prístupu (Access Control Service Plugin),
- kryptografiu (Cryptographic Service Plugin),
- logovanie dát (Logging Service Plugin),
- tagovanie dát (Data Tagging Service Plugin).

Architektúra zabezpečenia v DDS je zobrazená na obrázku 1.4 [18].



Obr. 1.4: Architektúra zabezpečenia v DDS.

Štandard DDS Security využíva priemyselne štandardné kryptografické algoritmy a techniky na ochranu DDS aplikácií pred konkrétnymi hrozbami. Medzi takéto hrozby patria napríklad neautorizované publikovanie a odoberanie dát alebo neoprávnený prístup k údajom. Z architektonického hľadiska sa jedná o podobnosť s Hypertext Transfer Protocol Secure (HTTPS) s rozdielom, že DDS-RTPS + DDS Security funguje cez protokol UDP a podporuje multicast, vrátane šifrovania [17].

1.5.1 Zabezpečenie služby distribúcie dát – bezpečnostný model

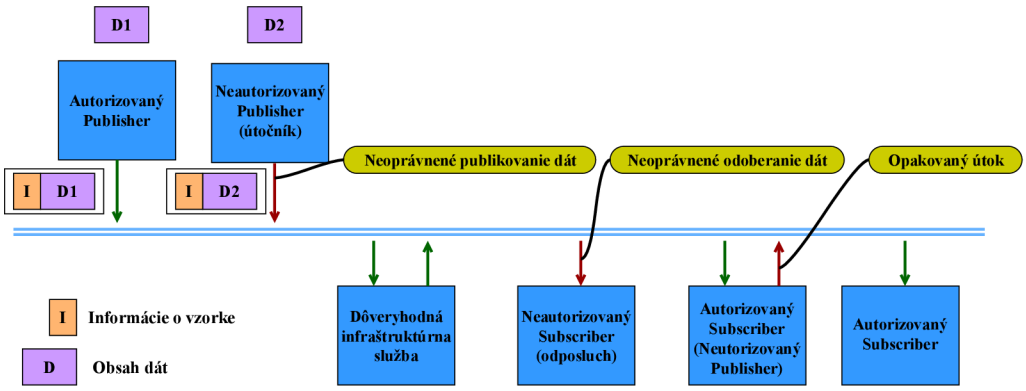
Bezpečnostné princípy, zabezpečované entity modelu DCPS a operácie na týchto entitách, ktoré majú byť obmedzené definuje bezpečnostný model pre DDS. Predmetom zabezpečenia je konkrétna doména a v rámci danej domény schopnosť prístupu k dátam, či už sa jedná o ich čítanie alebo zapisovanie. Zabezpečením DDS sa rozumie poskytnutie [18]:

- dôvernosti vzoriek dát,
- integrity vzoriek dát a správ, ktoré ich obsahujú,
- autentifikácie objektov typu *DataReader* a *DataWriter*,
- autorizácie objektov typu *DataReader* a *DataWriter*,
- autentifikácie pôvodu správy,
- autentifikácie pôvodu dát,
- nepopierateľnosti dát (voliteľne).

Pre zabezpečenie bezpečného prístupu do globálneho dátového priestoru (GDS) je nutné DDS aplikácie autentifikovať, aby bolo možné overiť totožnosť aplikácie. Následne po autentifikácií je ako ďalší krok v poradí rozhodnutie o kontrole prístupu, kedy je určené či má aplikácia práva k vykonaniu konkrétnych akcií ako sú napríklad pripojenie sa k danej DDS doméne, definovanie novej témy, prípadne čítanie alebo zápis dát v rámci objektu typu *Topic*. Kontrola prístupu je podporená pomocou kryptografických techník, kvôli zachovaniu dôvernosti a integrity dát. To si vyžaduje infraštruktúru na správu a distribúciu potrebných kryptografických kľúčov [18].

1.5.2 Zabezpečenie služby distribúcie dát – bezpečnostné hrozby

Pre lepšie porozumenie funkcií jednotlivých Service Plugin Interfaces je nutné popísať najrelevantnejšie typy útokov, ktoré ovplyvňujú DDS aplikácie využívajúce DDS-RTPS. Obrázok 1.5 [18] popisuje časť bezpečnostných hrozieb, ktoré sa môžu vyskytnúť v prípade neexistencie DDS Security. Jedná sa o komunikačný scenár, ktorý pozostáva zo 6 komunikačných entít pripojených k rovnakej sieti.



Obr. 1.5: Komunikačný scenár popisujúci bezpečnostné hrozby.

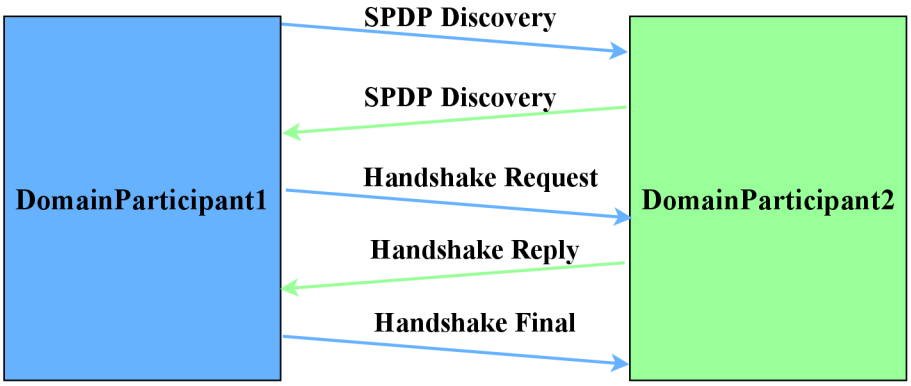
Popis jednotlivých typov bezpečnostných hrozieb v prípade neexistencie DDS Security podľa [16]:

- **neautorizované publikovanie dát** - Neautorizovaný objekt typu *Publisher* je pripojený k danej sieti a je schopný vložiť pakety s akýmkoľvek dátovým obsahom, hlavičkami a cieľovými adresami, ktoré si praje.
- **neautorizované odoberanie dát** - Neautorizovaný objekt typu *Subscriber* je pripojený k danej sieti a dokáže pozorovať publikované pakety. V prípade, ak autorizované objekty typu *Publisher* a *Subscriber* medzi sebou komunikujú cez multicast, stačí sa neautorizovanému objektu typu *Subscriber* prihlásiť k rovnakej multicast adrese.
- **manipulácia a opakovaný útok** - Autorizovaný objekt typu *Subscriber*, ktorý je zároveň neautorizovaným *Publisherom* dát môže pomocou zdieľaného tajného kľúča vytvárať správy v sieti a predstierať, že správa prišla od autorizovaného objektu typu *Publisher*, nakoľko má tajný kľúč, ktorý sa používa na výpočet Hash-based Message Authentication Code (HMAC). Z toho vyplýva, že tiež môže vytvoriť platný HMAC pre novú správu.
- Jedným z typov útokov by mohol byť aj **neoprávnený prístup k dátam v podaní infraštruktúrnej služby**, ktorá môže dáta ukladať a posielať ich ďalej, ale nemá oprávnenie čítať ich obsah. Infraštruktúrna služba môže vykonať takýto typ útoku nakoľko môže vidieť hlavičky a informácie o daných vzorkách ako sú poradové čísla, hash kľúča a podobne.

1.5.3 Autentifikačný doplnok

Autentifikačný doplnok (Authentication Service Plugin) definuje typy a operácie, ktoré sú potrebné pre podporu autentifikácie objektov typu *DomainParticipant* [16]. V zabezpečenom DDS systéme sa od každého účastníka komunikácie vyžaduje jeho

autentifikácia, aby sa predišlo publikovaniu dát neoverenými účastníkmi. Z toho vyplýva, že v prípade chránenej DDS domény môže objekt typu *DomainParticipant* komunikovať iba s rovnako overeným objektom typu *DomainParticipant*. Autentifikácia je implementovaná pomocou dôveryhodnej certifikačnej autority, ktorá vykonáva vzájomnú autentifikáciu medzi účastníkmi komunikácie pomocou algoritmu RSA alebo ECDSA a vytvára zdieľané tajomstvo použitím Diffie-Hellman (DH) algoritmu alebo pomocou algoritmu Elliptic Curve Diffie-Hellman (ECDH) [18]. Zdieľané tajomstvo je často odvodené z úspešnej výmeny autentifikačných správ a môže byť použité na výmenu kryptografických údajov o šifrovaní a autentifikácií správ. Na obrázku 1.6 [17] je zobrazený postup výmeny správ pri autentifikácií, ktorý pozostáva z objavenia účastníkov komunikácie a z takzvaného 3-way handshake. Objavenie objektov typu *DomainParticipant* je vykonané pomocou protokolu Simple Participant Discovery Protocol (SPDP) a okrem výmeny troch handshake správ medzi komunikujúcimi objektami typu *DomainParticipant* dôjde aj k výmene certifikátov a dokumentov o povolení, k overeniu digitálnych podpisov a k dohode o tajnom kľúči pomocou Diffie-Hellman algoritmu.



Obr. 1.6: Postup výmeny správ pri autentifikácií účastníkov komunikácie.

1.5.4 Doplnok pre riadenie prístupu

Po autentifikácií objektu typu *DomainParticipant* je potrebné overenie a nastavenie jeho práv. Doplnok pre riadenie prístupu (Access Control Service Plugin) vykonáva kontrolu riadenia a kontrolu prístupu. Kontrola riadenia je proces konfigurácie objektov typu *DomainParticipant*, *Topic*, *DataWriter* a *DataReader* vytvorených v rámci danej DDS domény, aby vykonávali zabezpečenie pre správne prípady. Kontrola prístupu je proces zabezpečenia lokálne vytvorených a vzdialene objavených objektov typu *DomainParticipant*, ktorým je umožnené komunikovať podľa ich možností [16]. Oprávnenia alebo prístupové práva sú často opísané pomocou matice riadenia

prístupu, kde riadky predstavujú subjekty ako sú napríklad používatelia a stĺpce sú objekty, pričom bunka definuje prístupové práva, ktoré má daný subjekt k objektom. Kontrola riadenia (*governance*) aj kontrola prístupu (*permissions*) sú spracované ako XML dokumenty. Tretím dokumentom je autorizačný dokument (*permissions_ca*) podpísaný zdieľanou certifikačnou autoritou (CA), ktorá môže byť už existujúca a totožná s tou, ktorá je použitá pri autentifikácií alebo môže byť vytvorená úplne nová [18].

1.5.5 Kryptografický doplnok

Kryptografický doplnok (Cryptographic Service Plugin) definuje typy a operácie potrebné pre podporu šifrovania, autentifikačných kódov správ a výmeny kľúčov pre objekty typu *DomainParticipant*, *DataReader* a *DataWriter*. Jedná sa o proces zabezpečenia proti manipulácií alebo odpočúvaniu komunikácie dvoch overených účastníkov komunikácie treťou stranou. DDS aplikácie môžu mať špecifické kryptografické knižnice, ktoré využívajú na šifrovanie, ale aj špecifické požiadavky týkajúce sa algoritmov na overovanie správ a ich podpisovanie. Okrem toho môžu žiadať, aby sa niektoré z uvedených funkcií vykonávali iba pre dané témy, čím je umožnená flexibilita jednotlivých scenárov a konfigurácie. Kryptografický doplnok poskytuje šifrovanie pomocou Advanced Encryption Standard (AES) v režime počítadla Galois (AES-GCM), pričom podporuje dve možnosti veľkosti AES kľúčov. Jeden variant je kľúč o veľkosti 128 bitov a druhý variant je kľúč o veľkosti 256 bitov [16, 18].

Operácia šifrovania autentifikovaná štandardom AES-GCM je transformáciou, ktorá berie do úvahy štyri vstupy a symbolicky vytvára dva výstupy [18]:

$$C, T = AES{-}GCM(K, P, AAD, IV). \tag{1.1}$$

Vstupy:

- **K** – 128 alebo 256 bitový kľúč, ktorý je použitý s blokovou šifrou AES.
- **P** – text, ktorý je predmetom šifrovania a autentifikácie. V prípade, ak sú autentifikované iba dáta je tento vstup prázdny.
- **AAD** – dodatočné autentifikované dáta. Jedná sa o dáta, mimo textu ktoré sú iba autentifikované a teda nie šifrované.
- **IV** – inicializačný vektor, 96 bitové číslo, ktoré sa nesmie opakovať pre rovnaký kľúč.

Výstupy:

- **C** – šifrovanie textu **“P”**.
- **T** – autentifikačná značka, jedná sa o autentifikačný kód správy (MAC), ktorý poskytuje autentifikáciu pre **“C”** a **“AAD”**.

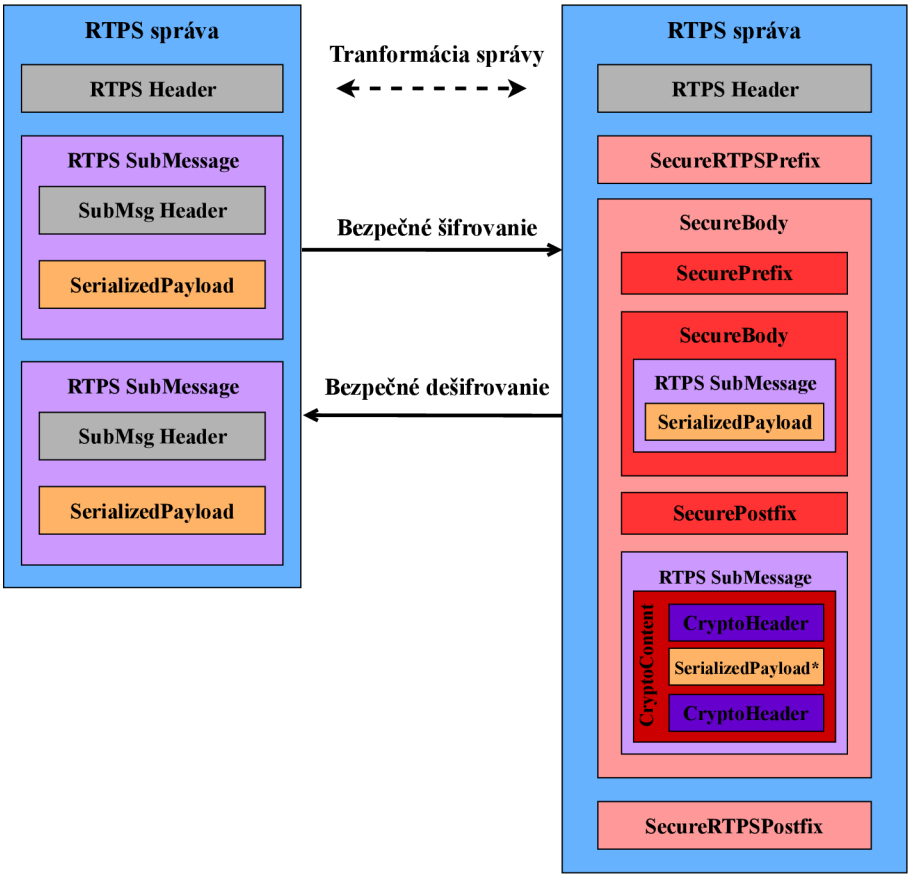
Vplyv na RTPS protokol

Kryptografický doplnok má vplyv aj na protokol RTPS. Rozhranie CryptoTransform zoskupuje operácie týkajúce sa šifrovania a dešifrovania, výpočtov, overovania digescie správ (hash) a autentifikačných kódov správ (MAC). Tieto operácie v rozhraní CryptoTransform sú definované veľmi všeobecne, kedy ako vstup berú bajtové pole na transformáciu a ako výstup produkujú transformované pole bajtov. Služba distribúcie dát je zodpovedná iba za volanie príslušných operácií vo vhodnom čase, keď generuje a spracováva RTPS správy, nahrádza vstupné bajty transformovanými bajtami, ktoré sú produkované operáciami rozhrania CryptoTransform a pokračuje generovaním, prípadne odoslaním alebo spracovaním RTPS správy štandardne, ale s nahradenými bajtmi [18]. Na obrázku 1.7 [17] je zobrazená transformácia RTPS správy.
Štandard DDS Security definuje nové RTPS podsprávy pre zabezpečenú správu RTPS. Popis jednotlivých RTPS podspráv a ich účel podľa [18]:

- **SecureRTPSPrefix** – záhlavie zabezpečenej RTPS správy. Obsah tejto podsprávy je zabezpečený šifrovaním, autentifikáciou správy a prípadne aj digitálnymi podpismi.
- **SecureBody** – používa sa na zapuzdrenie jednej alebo viacerých štandardných RTPS podspráv. Jej obsah je zabezpečený šifrovaním, autentifikáciou správy a prípadne aj digitálnymi podpismi.
- **SecurePrefix** – záhlavie podsprávy SecureBody.
- **SecurePostfix** – zápätie, ktoré sa používa na autentifikáciu podsprávy RTPS, ktorá jej predchádza.
- **SecureRTPSPostfix** – zápätie zabezpečenej RTPS správy, ktoré sa používa na autentifikáciu podspráv RTPS.

Ako je uvedené v kapitole 1.4 každá podspráva RTPS sa skladá z variabilného množstva prvkov podsprávy (SubMsgElement). Štandard DDS Security definuje okrem podspráv RTPS pre zabezpečenú správu RTPS aj prvky podsprávy. Popis jednotlivých prvkov RTPS podspráv a ich účel podľa [18]:

- **CryptoContent** – prvok podsprávy RTPS, ktorý sa používa na zapuzdrenie špeciálneho prvku podsprávy RTPS SerializedPayload, podsprávy RTPS alebo celej správy RTPS.
- **CryptoHeader** – prvok podsprávy RTPS, ktorý slúži ako záhlavie pre CryptoContent.
- **CryptoFooter** – prvok podsprávy RTPS, ktorý slúži ako zápätie pre CryptoContent.



Obr. 1.7: Transformácia RTPS správy.

1.5.6 Doplnok pre logovanie dát

Tento doplnok definuje typy a operácie potrebné pre podporu zaznamenávania bezpečnostných udalostí pre objekt typu *DomainParticipant*. Z toho vyplýva, že poskytuje možnosť zaznamenávať všetky bezpečnostné udalosti, ale aj očakávané správanie a všetky chyby alebo narušenia zabezpečenia. Cieľom doplnku pre logovanie dát (Logging Service Plugin) je vytvoriť bezpečnostné záznamy, ktoré je možné použiť na podporu zabezpečenia. Funguje tak, že každému dátovému záznamu pridelí identifikátor, ktorý jedinečne špecifikuje daného účastníka v rámci domény. Okrem toho pridá do každého záznamu aj časové razítko. Pre zber dát existujú dve možnosti, pričom prvou je zaznamenanie všetkých udalostí do lokálneho súboru a druhou je bezpečné distribuovanie záznamov cez službu distribúcie dát [18].

1.5.7 Doplnok pre tagovanie dát

Doplnok pre tagovanie dát (Data Tagging Service Plugin) definuje možnosť pridania bezpečnostnej značky k dátam. Najčastejšie využitie je pre určenie stupňa utajenia

dát, vrátane informácií o ich uvoľniteľnosti. Pre DDS môže byť tento doplnok použitý napríklad pre riadenie prístupu na základe značky alebo uprednostnenie správy. Existujú štyri prístupy k označovaniu dát. Jedným z nich je takzvané značenie *DataWriter*, ktoré DDS Security podporuje, kedy údaje doručené od konkrétneho objektu typu *DataWriter* majú značku *DataWriter*. Pre službu distribúcie dát je to mimoriadne výhodné nakoľko všetky metadáta pre všetky vzorky od objektu typu *DataWriter* sú rovnaké a teda stačí, že značka bude vymenená iba raz a síce pri objavení objektu typu *DataWriter* a nemusí byť odosielaná s každou vzorkou. Výhodou je podpora štandardných prípadov použitia, kedy aplikácia alebo objekt typu *DomainParticipant* zapisuje dáta do objektu typu *Topic* so spoločnou sadou značiek. V prípade, kedy aplikácia vytvára rôzne klasifikované dáta je vytvorených viacero objektov typu *DataWriter* s rôznymi značkami [18].

1.6 Analýza implementácií služby distribúcie dát

Služba distribúcie dát kompatibilná so špecifikáciou OMG-DDS je implementovaná viacerými spoločnosťami. Kompatibilita so špecifikáciou OMG-DDS je poväčšine zabezpečená základnými funkcionalitami, samotnou distribúciou dát alebo kvalitou služby. Viaceré implementácie neimplementujú časť zabezpečenia. Kritérií pre rozdelenie DDS implementácie existuje niekoľko. Jednou z možností je delenie DDS implementácií na tie, ktoré zabezpečenie implementujú a DDS implementácie, ktoré zabezpečenie neimplementujú. Ďalším kritériom rozdelenia môže byť typ licencie DDS implementácií. Zatiaľ čo niektoré implementácie majú platenú licenciu, z čoho väčšinou vyplýva aj väčšia využiteľnosť a väčšie možnosti konfigurácie danej DDS implementácie, prípadne implementované zabezpečenie, existujú aj implementácie služby distribúcie dát, ktoré majú voľnú licenciu (Open Source). V tomto texte sú analyzované tri vybrané DDS implementácie.

OpenDDS

OpenDDS je DDS implementácia s voľnou licenciou a je vyvíjaná spoločnosťou Object Computing, Inc. (OCI). Jedná sa o C++ implementáciu, ktorá ale podporuje aj JAVA aplikácie prostredníctvom JNI väzieb. Pre prenos dát sú využívané transportné protokoly UDP/IP, TCP/IP, RTPS/UDP a podporovaný je multicast aj unicast. Okrem protokolu RTPS je v OpenDDS možné pre objavovanie participujúcich objektov komunikácie využiť proces DCPS Information Repository (DCPSInfoRepo), ktorý je možné použiť pre objavovanie objektov typu Publisher a Subscriber [21]. Implementácia OpenDDS je aplikovateľná na rôznych operačných systémoch ako sú Windows, Linux, MacOSX, Solaris alebo Raspbian. Zatiaľ čo donedávna sa

jednalo o DDS implementáciu nepodporujúcu zabezpečenie, vo verzii OpenDDS 3.13 sa objavila beta implementácia OMG DDS Security. Aktuálna verzia je OpenDDS 3.14 (máj 2020)[22].

Vortex OpenSplice

Táto DDS implementácia je šírená pod platenou licenciou a je vyvíjaná spoločnosťou ADLINK Technology. Vortex OpenSplice je implementovaná vo viacerých programovacích jazykoch vrátane C, C++, C#, JAVA, JavaScript, Ruby a iných [23]. Táto implementácia môže byť nasadená v dvoch architektúrach. Prvým prípadom je keď sa DDS aplikácie a správa Vortex OpenSplice nachádzajú v rámci jedného operačného systému. To je výhodné v prípadoch, ak je druhý variant, ktorým je zdieľaná pamäť nedostupná alebo nežiadúca. Pri zdieľanej pamäti sú dáta fyzicky prítomné iba raz na ľubovoľnom počítači, ale inteligentná správa stále poskytuje každému účastníkovi vlastný pohľad na dané dáta. Vortex OpenSplice plne podporuje interoperabilný protokol RTPS a transportné protokoly UDP aj TCP. Funguje na viacerých operačných systémoch vrátane Windowsu, Linuxu alebo Solarisu. Jedná sa o DDS implementáciu podporujúcu zabezpečenie v súlade so špecifikáciou OMG DDS Security. V DDS implementácií Vortex OpenSplice je bezpečnostný model tvorený tromi doplnkami, definovanými v OMG DDS Security, pre [24]:

- autentifikáciu,
- kontrolu prístupu,
- kryptografiu.

Vortex OpenSplice poskytuje rôzne nástroje pre konfiguráciu (Configurator), kontrolu nad nasadeným DDS systémom (Tuner), automatizované testovanie a ladenie systémov založených na službe distribúcie dát (Tester) a ďalšie služby [25]. Aktuálna verzia je Vortex OpenSplice 6.10.3 (november 2019).

Connext DDS

Connext DDS je DDS implementácia s platenou licenciou, ktorá je vyvíjaná spoločnosťou Real-Time Innovations (RTI). Rovnako ako Vortex OpenSplice aj Connext DDS je implementovaná vo viacerých programovacích jazykoch vrátane C, C++, C#, JAVA a experimentálne aj v jazykoch Python alebo JavaScript. Taktiež podporuje interoperabilný protokol RTPS a transportné protokoly TCP aj UDP, rovnako aj unicast a multicast [26]. Táto DDS implementácia je rovnako ako predchádzajúce aplikovateľná pre rôzne operačné systémy vrátane Windowsu, Linuxu, macOS a ďalších. Connext DDS podporuje zabezpečenie v súlade s OMG DDS Security a rovnako ako Vortex OpenSplice je bezpečnostný model tvorený doplnkami pre autentifikáciu, kontrolu prístupu a kryptografiu. Okrem týchto troch doplnkov podporuje aj

doplnok pre záznam dát [27]. Aktuálna verzia je Connext DDS 6.0 (november 2019).

Zhrnutie bezpečnostných možností vybraných DDS implementácií

Nakoľko v súčasnosti dve z troch vybraných implementácií podporujú zabezpečenie v súlade so špecifikáciou OMG DDS Security, ich možností sa z hľadiska zabezpečenia veľmi nelíšia. Výnimkou je DDS implementácia OpenDDS, ktorá momentálne implementuje iba beta verziu OMG DDS Security. Pri autentifikácii je možnosťou u Vortex OpenSplice a RTI Connext algoritmus RSA s dĺžkou kľúču 2048 bitov. Vortex OpenSplice okrem RSA môže k autentifikácii využívať aj eliptické krivky s dĺžkou kľúču 256 bitov a pre výmenu kľúčov algoritmus Diffie-Hellman. U RTI Connext DDS je pre autentifikáciu okrem RSA možné použiť eliptické krivky využívajúce variantu s algoritmom digitálneho podpisu (ECDSA) a pri výmene zdieľaného tajomstva je možné použiť algoritmus Diffie-Hellman, prípadne aj jeho verziu s využitím eliptických kriviek (ECDH). OpenDDS podporuje pre autentifikáciu algoritmus RSA s dĺžkou kľúču 2048 bitov alebo eliptické krivky s dĺžkou kľúču 256 bitov. Autentifikáciu pôvodu správy momentálne neimplementuje.
Autentifikačný formát u DDS implementácii OpenDDS a Vortex OpenSplice je rovnaký, pričom pre 2048-bitové RSA využíva štandard X.509, ktorý definuje formát certifikátov s verejným kľúčom a krivku prime256v1 pre eliptickú krivku s dĺžkou kľúču 256 bitov. RTI Connext DDS využíva ako autentifikačný formát štandard X.509.
U RTI Connext DDS je pre šifrovanie podpora algoritmu AES s dĺžkou kľúčov 128 bitov a 256 bitov. AES využívajúci Galoise Counter Mode (GCM) slúži k šifrovaniu a k overeniu správ slúži AES využívajúci variantu GCM Galoise Message Authentication Code (GMAC). Vortex OpenSplice podporuje u algoritmu AES okrem 128 bitových a 256 bitových kľučov aj kľúče s dĺžkou 192 bitov. Okrem AES podporuje pre šifrovanie aj algoritmus RSA v rovnakých variantách kľúčov ako u AES a algoritmus Blowfish. OpenDDS momentálne pre šifrovanie podporuje algoritmus AES s dĺžkou kľúču 256 bitov. U OpenDDS nie je podporované šifrovanie celej správy, ale iba šifrovanie podspráv a užitočného zaťaženia (payload). Tabuľka 1.1 zhŕňa možnosti bezpečnosti vybraných implementácií [22, 24, 27, 30].

	Vortex OpenSplice	RTI Connext DDS	OpenDDS
Autentifikácia	2048-bit RSA 256-bit EC DH	2048-bit RSA ECDSA DH ECDH	2048-bit RSA 256-bit EC
Autentifikačný formát	X.509 PKI prime256v1	X.509 PKI	X.509 PKI prime256v1
Šifrovanie	AES128, AES192, AES256, Blowfish, RSA128, RSA192, RSA256	AES128-GCM AES256-GCM AES128-GMAC AES256-GMAC	AES256-GCM iba podspráva a payload

Tab. 1.1: Zhrnutie bezpečnostných možností DDS implementácií.

1.7 Zhrnutie služby distribúcie dát

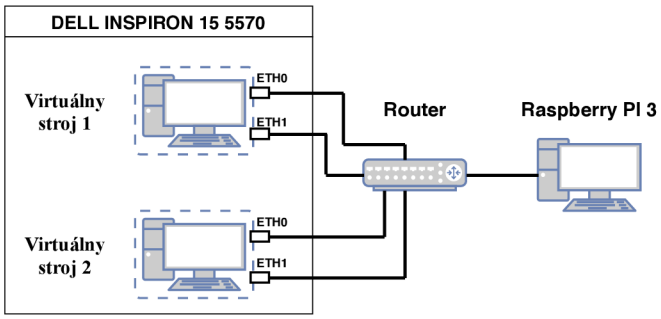
V tejto časti práce je predstavená teoretická časť o službe distribúcie dát. Následne je postupne zamerané na jednotlivé súčasti služby distribúcie dát ako je dátovo zameraný komunikačný model DCPS, kvalita služby v DDS a v neposlednom rade sa táto časť práce venuje možnostiam zabezpečenia služby distribúcie dát. V rámci časti zabezpečenia je predstavený aj štandardizovaný protokol RTPS. V závere tejto časti sú analyzované vybrané implementácie služby distribúcie dát vrátane zhrnutia ich bezpečnostných možností. Ďalšia časť práce je zameraná na praktické využitie služby distribúcie dát a pre túto praktickú časť je zvolená DDS implementácia OpenDDS.

2 DDS implementácia OpenDDS

Táto časť práce sa zaoberá návrhom a simuláciou základných a komplexných komunikačných scenárov, overením bezpečnosti použitej DDS implementácie OpenDDS a návrhom a simuláciou bezpečnostných incidentov.

2.1 Hardware

Pre túto prácu je použitý notebook Dell Inspiron 15 5570 a jednodoskový počítač Raspberry Pi 3 model B+. Na zariadení Dell Inspiron 15 5570, s veľkosťou operačnej pamäte RAM 8GB, pracujú dva virtuálne stroje, pričom samotná virtualizácia prebieha cez softvér VMware Workstation 15 Player. V oboch virtuálnych strojoch je nasadený operačný systém Ubuntu 18.04.3 LTS, ktorému je v oboch prípadoch priradená operačná pamäť RAM 2,7GB. Jednodoskový počítač Raspberry Pi 3 model B+ pracuje s operačným systémom Raspbian Buster a veľkosť jeho operačnej pamäte RAM je 1GB. Obrázok 2.1 zobrazuje schému zapojenia použitých zariadení, ktorá je využitá pre základné komunikačné scenáre a je podľa potreby upravovaná pre jednotlivé komunikačné scenáre.



Obr. 2.1: Schéma zapojenia použitých zariadení - základné komunikačné scenáre.

2.2 Kompilácia OpenDDS na použitých zariadeniach

Nakoľko je DDS implementácia OpenDDS nasadená na zariadeniach, na ktorých pracujú operačné systémy založené na linuxovej distribúcii Debian, je nutné pre úspešnú kompiláciu tejto DDS implementácie na daných zariadeniach vopred nainštalovať definovaný softvér podľa [28]. Existujú dve možnosti kompilácie vzhľadom na časť zabezpečenia, pričom v jednom prípade je OpenDDS bez možnosti využívať časť zabezpečenia (security) a v druhom prípade, ktorý je použitý pre túto prácu je možnosť využívať časť zabezpečenia. V prípade kompilácie s časťou zabezpečenia je

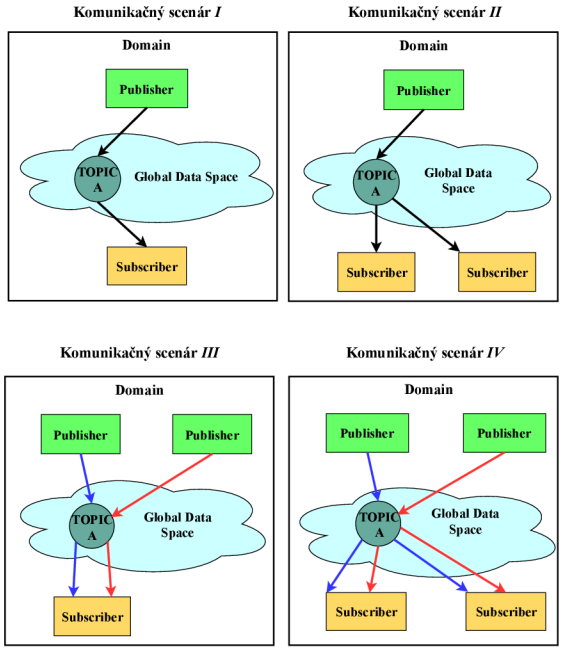
nutné nainštalovať na cieľové zariadenie ďalší softvér podľa [22]. Kompilácia pre dva virtuálne stroje je totožná, avšak pre jednodoskový počítač Raspberry Pi je odlišná. V prípade kompilácie OpenDDS na dvoch použitých virtuálnych strojoch s operačným systémom Ubuntu je nainštalovaný nasledujúci softvér:

- GCC 8.3.0,
- GNU Make 4.1,
- Perl v5.26.1,
- OpenSSL 1.1.1d (v prípade kompilácie s časťou zabezpečenia),
- Xerces-C++ v3.2.2 (v prípade kompilácie s časťou zabezpečenia),
- CMake v3.10.2 (v prípade kompilácie s časťou zabezpečenia).

Kompilácia pre zariadenie Raspberry Pi 3 B+ je odlišná a jej postup je popísaný v [29]. Použitý softvér ostáva rovnaký ako v predchádzajúcom prípade, ale v tomto prípade je využitá metóda krížovej kompilácie, čo znamená že kompilácia OpenDDS je vykonaná na zariadení s operačným systémom Ubuntu a zariadenie s operačným systémom Raspbian je cieľom kompilácie.

2.3 Návrh základných komunikačných scenárov

Táto kapitola sa zaoberá návrhom komunikačných scenárov, ktoré sú predmetom simulovania komunikácie medzi jednotlivými participujúcimi entitami modelu DCPS. Schémy navrhnutých komunikačných scenárov sa nachádzajú na obrázku 2.2.



Obr. 2.2: Schémy navrhnutých komunikačných scenárov.

Komunikačný scenár *I*. Tento komunikačný scenár rieši komunikáciu medzi jedným objektom typu *Publisher* a jedným objektom typu *Subscriber*. Objekt typu *Publisher* má priradený jeden objekt typu *DataWriter*, objekt typu *Subscriber* má priradený jeden objekt typu *DataReader* a definovaný je jeden objekt typu *Topic*. Schéma komunikačného scenára *I* je zobrazená na obrázku 2.2 v ľavom hornom rohu.

Komunikačný scenár *II*. Komunikačný scenár *II* definuje komunikáciu medzi jedným objektom typu *Publisher* a dvomi objektami typu *Subscriber*. Objektu typu *Publisher* je priradený jeden objekt typu *DataWriter*, každému objektu typu *Subscriber* je priradený jeden objekt typu *DataReader* a definovaný je jeden objekt typu *Topic*. Oba objekty typu *Subscriber* odoberajú dáta od objektu typu *Publisher*. Schéma komunikačného scenára *II* je zobrazená na obrázku 2.2 v pravom hornom rohu.

Komunikačný scenár *III*. Komunikačný scenár *III* sa zaoberá komunikáciou medzi dvomi objektami typu *Publisher* a jedným objektom typu *Subscriber*. Každý objekt typu *Publisher* má priradené dva objekty typu *DataWriter*, objekt typu *Subscriber* má priradené dva objekty typu *DataReader* a definovaný je jeden objekt typu *Topic*. Schéma komunikačného scenára *III* je zobrazená na obrázku 2.2 v ľavom dolnom rohu.

Komunikačný scenár *IV*. Tento komunikačný scenár sa zaoberá komunikáciou medzi dvomi objektami typu *Publisher* a dvomi objektami typu *Subscriber*. Každý objekt typu *Publisher* má priradené dva objekty typu *DataWriter*, každý objekt typu *Subscriber* má priradené dva objekty typu *DataReader* a definovaný je jeden objekt typu *Topic*. Oba objekty typu *Publisher* publikujú dáta pre každý z objektov typu *Subscriber*. Schéma komunikačného scenára *IV* je zobrazená na obrázku 2.2 v pravom dolnom rohu.

2.4 Simulácia základných komunikačných scenárov

V tejto kapitole je na začiatku ukážka jedného z konfiguračných súborov, pomocou ktorých je nastavené správanie komunikácie, následne definícia spôsobu spúšťania jednotlivých procesov a na záver sú spracované simulácie jednotlivých navrhnutých komunikačných scenárov.

2.4.1 Konfiguračný súbor

DDS implementácia OpenDDS obsahuje základný konfiguračný rámec pre konfiguráciu nastavení súvisiacich s konkrétnymi entitami komunikácie alebo s globálnymi nastaveniami. Konfiguračný súbor má svoju danú štruktúru a výsledný súbor je

v tvare *názov_súboru.ini*. Konfiguráciu v OpenDDS možno rozdeliť do troch hlavných častí [8]:

- konfigurácia správania jednotlivých entít modelu DCPS na globálnej úrovni (**common**),
- konfigurácia správania jednotlivých DCPS entít pri vzájomnom objavovaní sa (**discovery**),
- konfigurácia týkajúca sa prenosu dát (**transport**).

V rámci konfigurácie správania jednotlivých entít modelu DCPS na globálnej úrovni je možné napríklad nastaviť povolenie zabezpečenia DDS systému, konfigurácia správania jednotlivých DCPS entít pri vzájomnom objavovaní sa umožňuje nastaviť rozšírené možnosti pri vzájomnom objavovaní sa ako je napríklad počet sekúnd, ktoré proces čaká medzi oznámeniami participujúcich entít alebo čas, do ktorého je možné považovať participujúcu entitu za živú. Konfigurácia týkajúca sa prenosu dát umožňuje definovať aký transportný protokol je využitý pre prenos dát alebo spoľahlivosť prenosu a mnoho ďalších. Ukážka konfiguračného súboru sa nachádza na výpise 2.1.

Výpis 2.1: Ukážka konfiguračného súboru.

```
[common]
DCPSGlobalTransportConfig=$file
DCPSSecurity=0

[domain/4]
DiscoveryConfig=uni_rtps

[rtps_discovery/uni_rtps]
SedpMulticast=0
ResendPeriod=10
SpdpSendAddrs=192.168.100.66:8404,192.168.100.69:8412

[transport/the_rtps_transport]
transport_type=rtps_udp
use_multicast=0
local_address=192.168.100.65:8402
```

Význam jednotlivých nakonfigurovaných nastavení a zoznam všetkých nastaviteľných parametrov v OpenDDS s podrobným popisom je možné nájsť v [8].

2.4.2 Spôsob spúšťania procesov

Pre distribúcie operačného systému Linux sa procesy objektu typu *Publisher* a objektu typu *Subscriber* v DDS implementácií OpenDDS spúšťajú pomocou príkazového riadku v okne terminálu pomocou nasledujúceho príkazu:

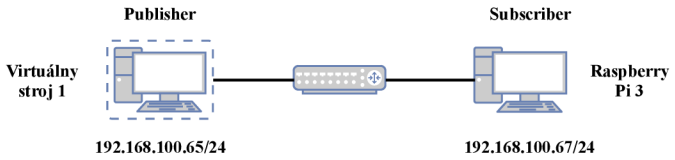
./názov_spúšťanej_entity* -DCPSConfigFile (definuje použitie konfiguračného súboru) *názov_konfiguračného_súboru.ini

V prípade spustenia procesu objektu typu *Publisher* je príkaz nasledovný:
./publisher -DCPSConfigFile *názov_konfiguračného_súboru.ini*

2.4.3 Simulácia komunikačného scenára *I*

Pre simulovanie komunikačného scenára *I*, kedy komunikácia prebieha medzi jedným objektom typu *Publisher* a jedným objektom typu *Subscriber* je využitá aplikácia Messenger, ktorá je implementovaná v DDS implementácii OpenDDS. Jedná sa o aplikáciu, kedy objekt typu *Publisher* odošle 40 správ a vo výpise objektu typu *Subscriber* je následne možné vidieť odosielateľa správy, predmet správy, text správy a poradové číslo správy. Počet odoslaných správ sa nastavuje v prípade aplikácie Messenger v hlavičkovom súbore *Args.h*, kde je okrem iného možné nastaviť napríklad aj spoľahlivosť prenosu dát. Zvyšné uvedené parametre o správe sa nastavujú v zdrojovom súbore *Writer.cpp* objektu typu *DataWriter*. Tento komunikačný scenár je simulovaný vo dvoch variantách.

Prvou variantou je komunikačný scenár bez časti zabezpečenia a komunikácia prebieha medzi jedným z virtuálnych strojov (objekt typu *Publisher*) a jednodoskovým počítačom Raspberry Pi 3 (objekt typu *Subscriber*). Na obrázku 2.3 je zobrazená komunikačná topológia s priradenými IP adresami jednotlivých zariadení.



Obr. 2.3: Komunikačný scenár *I* - komunikačná topológia.

Nasledujúci výpis 2.2 definuje spustenie procesu objektu typu *Subscriber* na zariadení Raspberry Pi 3.

Výpis 2.2: Komunikačný scenár *I* - spustenie procesu objektu typu *Subscriber*.

```
pi@raspberrypi:~/OpenDDS-3.13.3/build/target/tests/DCPS/Messenger$ ./subscriber -DC
PSConfigFile rtps-nezabezpecena-67.ini
```

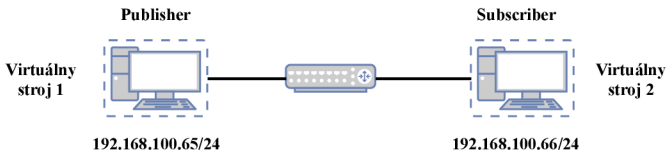
Výpis 2.3 z terminálu na zariadení Raspberry Pi 3 zobrazuje úvodnú časť prijatých dát od objektu typu *Publisher*, ktorý je spustený na virtuálnom stroji. Ako je možné vidieť po spustení procesu objektu typu *Subscriber* je v úvode objektom typu *DataReader* oznámené, že prijaté dáta sú spracovávané spoľahlivo a teda, že

služba by mala všetky dáta prijať bez straty. Následne už vo výpise nasleduje prijatá správa, z ktorej je možné odčítať predmet správy, identifikačné číslo správy, odosielateľa správy, poradové číslo správy a na záver samotný text správy.

Výpis 2.3: Komunikačný scenár *I* - simulácia nezabezpečenej komunikácie.

```
Transport is RELIABLE
Reliable DataReader
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 0
         text       = Worst. Movie. Ever.
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 1
         text       = Worst. Movie. Ever.
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 2
         text       = Worst. Movie. Ever.
```

Druhou variantou je komunikačný scenár využívajúci časť zabezpečenia, pričom komunikácia prebieha medzi dvojicou virtuálnych strojov. Na obrázku 2.4 je zobrazená komunikačná topológia s priradenými IP adresami jednotlivých virtuálnych strojov.



Obr. 2.4: Komunikačný scenár *I* so zabezpečením - komunikačná topológia.

Využitie časti zabezpečenia je zadefinované v konfiguračnom súbore, kedy je parametru *DCPSSecurity* priradené číslo 1. Nasledujúci výpis 2.4 zobrazuje spustenie procesu objektu typu *Subscriber* na virtuálnom stroji 2.

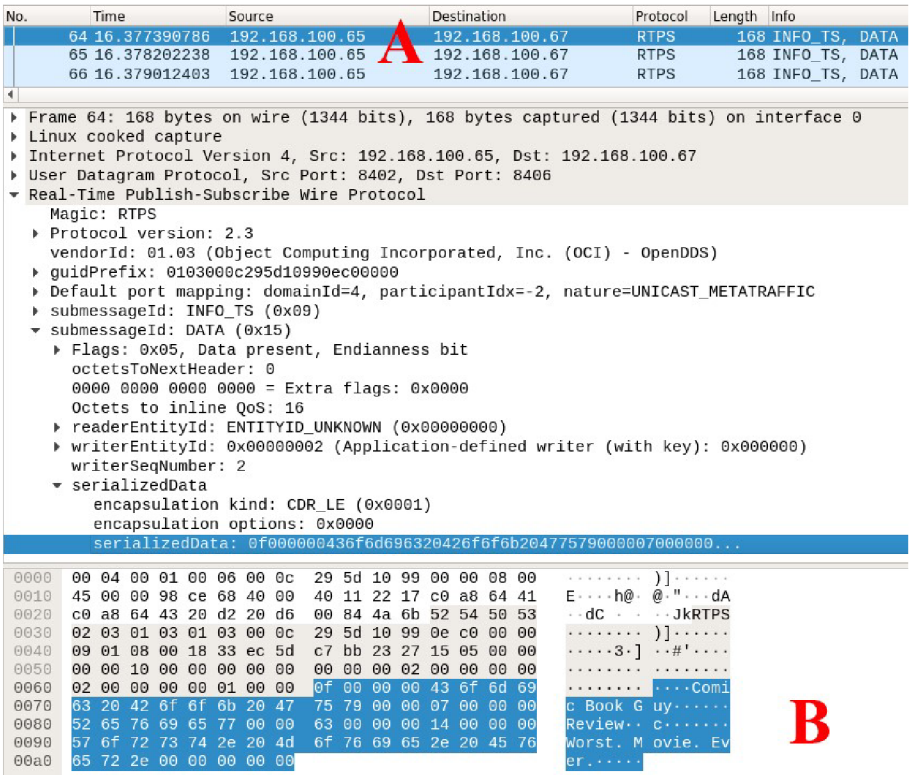
Výpis 2.4: Spustenie procesu zabezpečeného objektu typu *Subscriber*.

```
student@ubuntu:~/OpenDDS-3.13.1/tests/DCPS/Messenger$ ./subscriber -DCPSConfigFile
rtps-zabezpecena-66.ini
```

Ako je možné vidieť z výpisu 2.5 z terminálu na virtuálnom stroji 2, v úvode je rovnako ako pri nezabezpečenej komunikácii objektom typu *DataReader* oznámené, že prijaté dáta sú spracovávané spoľahlivo a teda, že služba by mala všetky dáta prijať bez straty. Nasleduje pokus o autentifikáciu participujúcich entít, ktorej postup je rozobratý v kapitole 1.5.3. Následne už sú prijímané zabezpečené dáta.

Výpis 2.5: Komunikačný scenár *I* - simulácia zabezpečenej komunikácie.

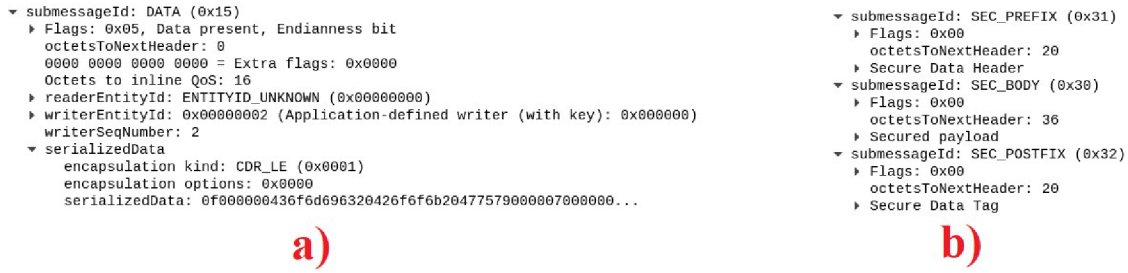
```
Transport is RELIABLE
Reliable DataReader
(2758|2763) DEBUG: Spdp::attempt_authentication() - Attempting authentication
(expecting reply) for participant:   97bf6f66.343b5464.cf29d647.000001c1(86146a80)
(2758|2760) RPCH 97bf6f66.343b5464.cf29d647.000001c1(86146a80) = 12
(2758|2760) DWCH 97bf6f66.343b5464.cf29d647.ff0202c3(9eb68af0) = 13
(2758|2760) DRCH 9ef3a9f4.7de654dd.b9771ace.ff0202c4(5251fefd) = 3
(2758|2760) RPCH 97bf6f66.343b5464.cf29d647.000001c1(86146a80) = 12
(2758|2760) DWCH 97bf6f66.343b5464.cf29d647.ff0202c4(00d21f53) = 2
(2758|2760) DRCH 9ef3a9f4.7de654dd.b9771ace.ff0202c3(cc356b5e) = 14
(2758|2760) RPCH 97bf6f66.343b5464.cf29d647.000001c1(86146a80) = 12
(2758|2760) DWCH 97bf6f66.343b5464.cf29d647.ff0200c2(db87d8e4) = 15
(2758|2760) DRCH 9ef3a9f4.7de654dd.b9771ace.ff0200c7(f96ecdc5) = 5
```



Obr. 2.5: Zachytená RTPS správa nezabezpečenej komunikácie.

V oboch prípadoch bola komunikácia zachytená pomocou paketového analyzátoru Wireshark. Na obrázku 2.5 je zachytená RTPS správa pri nezabezpečenej komunikácii. Časť **A** na obrázku 2.5 zobrazuje zdrojovú a cieľovú IP adresu komunikácie,

použitý protokol RTPS a typ podsprávy RTPS (DATA). V časti **B** na obrázku 2.5 je zachytená odoslaná správa, vrátane odosielateľa a predmetu správy.

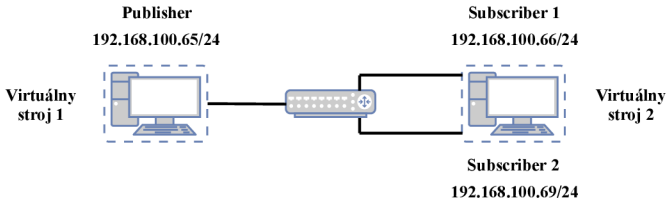


Obr. 2.6: Porovnanie zachytených RTPS správ.

Na obrázku 2.6 sa nachádza porovnanie RTPS správ pri nezabezpečenej a zabezpečenej komunikácii. Časť **a)** na obrázku 2.6 zachytáva nezabezpečenú RTPS správu a časť **b)** zachytáva zabezpečenú RTPS správu. Ako je možné vidieť štruktúra zabezpečenej RTPS správy je odlišná oproti nezabezpečenej RTPS správe a skladá sa z troch podspráv, ktoré sú rozobraté v kapitole 1.5.5. Najpodstatnejším rozdielom je, že v prípade zabezpečenej RTPS správy nie je možné rozkódovať užitočné zaťaženie a nie je teda možné zistiť informácie o správe ako je predmet, odosielateľ, poradové číslo správy a mnoho ďalších informácií, ktoré by mohli viesť k zneužitiu.

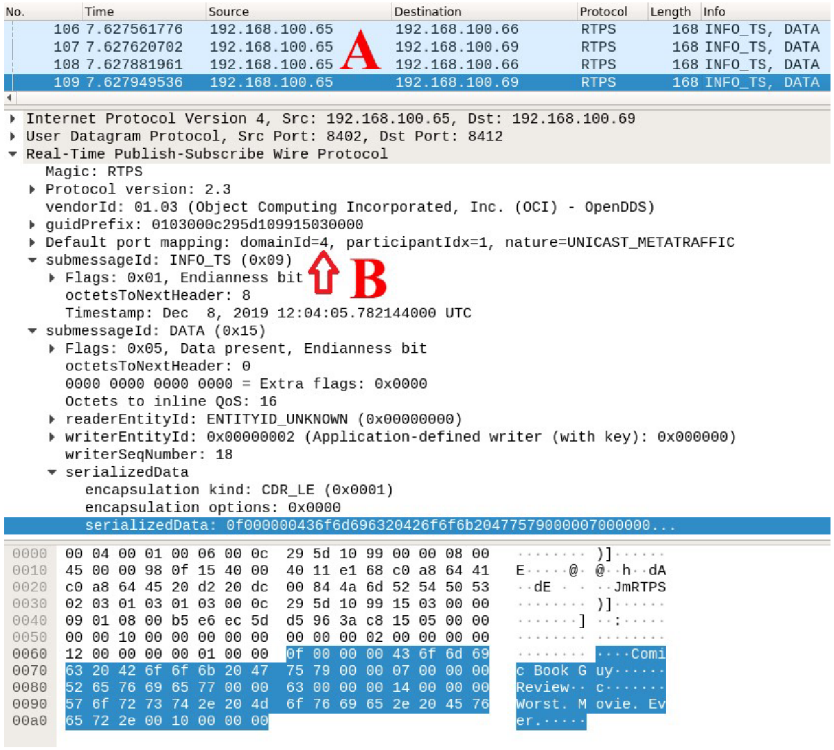
2.4.4 Simulácia komunikačného scenára *II*

Simulácia komunikačného scenára *II*, kedy komunikácia prebieha medzi jedným objektom typu *Publisher* a dvomi objektami typu *Subscriber* využíva rovnako ako v prípade komunikačného scenára *I* aplikáciu Messenger, ktorá je implementovaná v DDS implementácii OpenDDS. Komunikačný scenár *II* je simulovaný bez časti zabezpečenia. Komunikácia prebieha medzi dvomi virtuálnymi strojmi, pričom na virtuálnom stroji kde sú spustené procesy objektov typu *Subscriber* sú dve sieťové rozhrania. Komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní virtuálnych strojov je na obrázku 2.7.



Obr. 2.7: Komunikačný scenár *II* - komunikačná topológia.

Výpis v termináloch oboch objektov typu *Subscriber* je totožný s tým na výpise 2.3. Oba objekty typu *Subscriber* v rovnakom čase odoberajú správy, ktoré publikuje objekt typu *Publisher*. Súčasné odoberanie dát oboma objektami typu *Subscriber* je zjavné zo zachytenej komunikácie z programu Wireshark, ktorej časť sa nachádza na obrázku 2.8. Časť **A** poukazuje práve na odosielanie dát k objektom typu *Subscriber*. Z časti **B** je možno odčítať identifikátor domény v ktorej sa komunikujúce entity nachádzajú, v tomto prípade je identifikačné číslo domény 4. Tento identifikátor domény je vždy definovaný pri vytváraní objektu typu *DomainParticipant*.

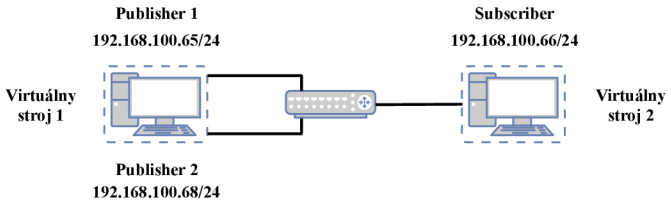


Obr. 2.8: Komunikačný scenár *II* - zachytená komunikácia.

2.4.5 Simulácia komunikačného scenára *III*

Komunikačný scenár *III* je inverzný voči komunikačnému scenáru *II* nakoľko komunikácia prebieha medzi dvomi objektami typu *Publisher* a jedným objektom typu *Subscriber*. Pre tento komunikačný scenár je využitá aplikácia LargeSample, ktorá je implementovaná v DDS implementácii OpenDDS. Jedná sa o aplikáciu založenú na aplikácii Messenger, ktorá bola využitá v predchádzajúcich komunikačných scenároch. Aplikácia LargeSample je primárne založená na otestovaní fragmentácie paketov, pri postupnom zväčšovaní veľkosti odoslaných správ. Pre túto prácu je využitá vzhľadom k tomu, že implementuje dva objekty typu *DataWriter*, ktoré

sú jednoznačne identifikované identifikačným číslom. Komunikácia prebieha medzi dvomi virtuálnymi strojmi, pričom na virtuálnom stroji kde sú spustené procesy objektov typu *Publisher* sa nachádzajú dve sieťové rozhrania. Komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní virtuálnych strojov je na obrázku 2.9.



Obr. 2.9: Komunikačný scenár *III* - komunikačná topológia.

Každý z objektov typu *Publisher* je identifikovaný identifikačným číslom procesu. Vo výpise objektu typu *Subscriber* v terminály je možné podľa čísla procesu definovať, ktorý objekt typu *Publisher* je zdrojom dát.

Výpis 2.6 definuje spustenie procesu objektu typu *Publisher1*, ktorý je spustený na virtuálnom stroji 1.

Výpis 2.6: Komunikačný scenár *III* - spustenie procesu objektu typu *Publisher1*.

```
adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/LargeSample$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-65.ini
```

Nasledujúci výpis 2.7 z terminálu na virtuálnom stroji 1 zobrazuje publikovanie dát objektom typu *Publisher1*. Z výpisu 2.7 je zjavné, že najskôr dôjde k spárovaniu objektov typu *DataWriter*, ktoré zapisujú dáta. Vo výpise z terminálu je následne možné vidieť informáciu o odoslaní správy, identifikačné číslo procesu, identifikátor objektu typu *DataWriter*, identifikačné číslo zapisovanej vzorky a veľkosť odosielaných dát.

Výpis 2.7: Komunikačný scenár *III* - publikovanie dát - Publisher 1.

```
adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/LargeSample$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-65.ini
(5560|5560) Writers matched
(5560|5560)Writer.cpp:146: Sending Message: process_id =  5560 writer_id = 1
sample_id = 0 extra data length = 10240
(5560|5560)Writer.cpp:175: Sending Message: process_id =  5560 writer_id = 2
sample_id = 0 extra data length = 20480
(5560|5560)Writer.cpp:146: Sending Message: process_id =  5560 writer_id = 1
sample_id = 1 extra data length = 20480
(5560|5560)Writer.cpp:175: Sending Message: process_id =  5560 writer_id = 2
sample_id = 1 extra data length = 30720
(5560|5560)Writer.cpp:146: Sending Message: process_id =  5560 writer_id = 1
sample_id = 2 extra data length = 40960
(5560|5560)Writer.cpp:175: Sending Message: process_id =  5560 writer_id = 2
sample_id = 2 extra data length = 51200
```

Výpis 2.8 zobrazuje publikovanie dát objektom typu *Publisher2*.

Výpis 2.8: Komunikačný scenár *III* - publikovanie dát - Publisher 2.

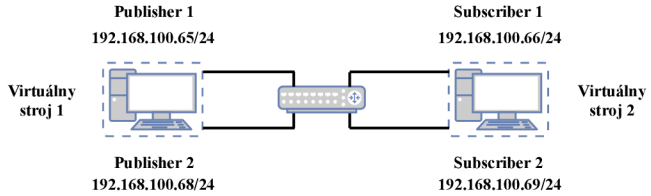
```
(5552|5552) Writers matched
(5552|5552)Writer.cpp:146: Sending Message: process_id =  5552 writer_id = 1
sample_id = 0 extra data length = 10240
(5552|5552)Writer.cpp:175: Sending Message: process_id =  5552 writer_id = 2
sample_id = 0 extra data length = 20480
(5552|5552)Writer.cpp:146: Sending Message: process_id =  5552 writer_id = 1
sample_id = 1 extra data length = 20480
(5552|5552)Writer.cpp:175: Sending Message: process_id =  5552 writer_id = 2
sample_id = 1 extra data length = 30720
(5552|5552)Writer.cpp:146: Sending Message: process_id =  5552 writer_id = 1
sample_id = 2 extra data length = 40960
(5552|5552)Writer.cpp:175: Sending Message: process_id =  5552 writer_id = 2
sample_id = 2 extra data length = 51200
```

Odoberanie dát objektom typu *Subscriber* je zachytené na výpise v prílohe A.1. V tomto prípade existujú dve možnosti spôsobu odoberania dát. V prvom prípade sa objekt typu *Subscriber* prihlási na ober dát pred ich publikovaním a dáta bude prijímať postupne od objektu typu *Publisher* s nižším číslom identifikujúcim proces a následne po odobratí všetkých publikovaných dát od daného objektu typu *Publisher* odoberá dáta od druhého publikujúceho objektu. V opačnom prípade prijíma objekt typu *Subscriber*, pomocou dvoch jemu asociovaných objektov typu *DataReader*, dáta od oboch objektov typu *Publisher* súčasne. Ako vyplýva z výpisu v prílohe A.1, v tomto prípade sa objekt typu *Subscriber* prihlási na ober dát pred ich publikovaním.

2.4.6 Simulácia komunikačného scenára *IV*

Rovnako ako pri komunikačnom scenári *III* aj pri simulácii komunikačného scenára *IV* je využitá aplikácia LargeSample, ktorá je implementovaná v DDS implementácii OpenDDS. V tomto komunikačnom scenári prebieha komunikácia medzi dvomi objektami typu *Publisher* a dvomi objektami typu *Subscriber*. Komunikačný scenár *IV* je simulovaný bez časti zabezpečenia. Komunikácia prebieha medzi dvomi virtuálnymi strojmi, pričom na oboch sú pripojené dve sieťové rozhrania. Komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní virtuálnych strojov sa nachádza na obrázku 2.10.

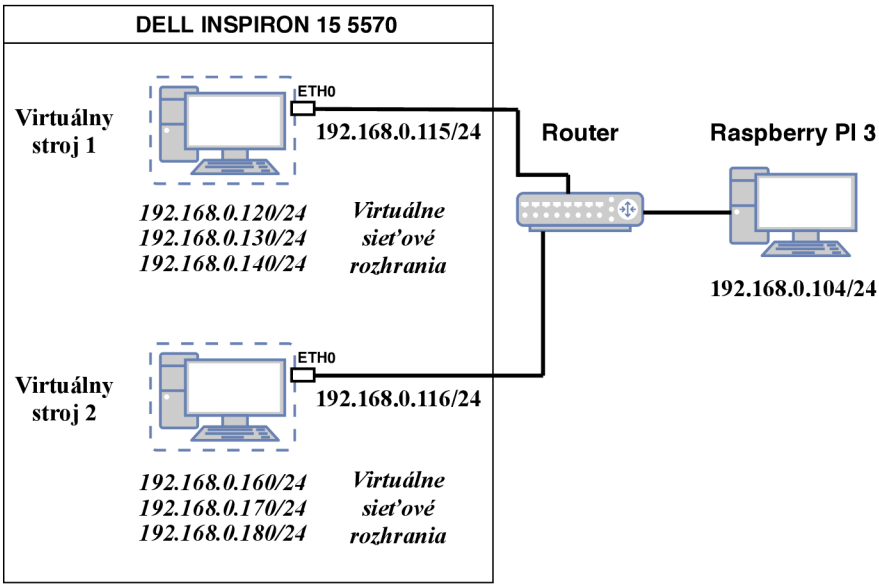
Výpisy z terminálov pri publikovaní dát sú v tomto komunikačnom scenári totožné s tými na výpisoch 2.7 a 2.8, pričom jediným rozdielom sú identifikačné čísla jednotlivých procesov, ktoré je možné vyčítať z výpisov objektov typu *Subscriber*. Výpis prvého objektu typu *Subscriber* (Subscriber 1) sa nachádza na výpise v prílohe A.2 a je z neho viditeľné odoberanie dát od oboch publikujúcich entít. Výpis v prílohe A.3 zobrazuje odoberanie dát druhého objektu typu *Subscriber*.



Obr. 2.10: Komunikačný scenár *IV* - komunikačná topológia.

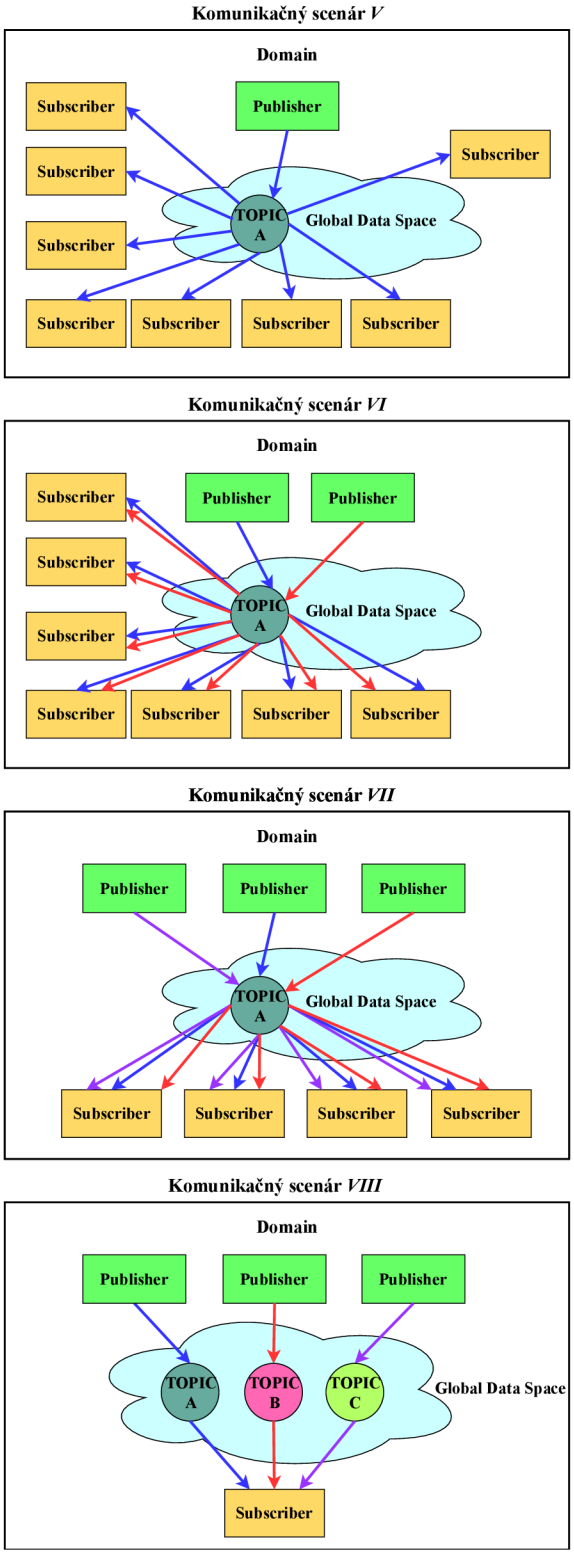
2.5 Návrh komplexných komunikačných scenárov

Pre návrh komplexných komunikačných scenárov je mierne pozmenená sieťová topológia použitých zariadení, nakoľko je nevyhnutné zaobstarať väčší počet sieťových rozhraní. Na každom z virtuálnych strojov sa nachádza jedno fyzické sieťové rozhranie a tri virtuálne sieťové rozhrania. Pre lepšiu prehľadnosť sú pri komplexných komunikačných scenárov nastavené jednotlivým sieťovým rozhraniam rozdielne IP adresy oproti základným testovacím komunikačným scenárom. Schéma topológie siete sa nachádza na obrázku 2.11.



Obr. 2.11: Topológia siete - komplexné komunikačné scenáre.

Na obrázku 2.12 sú zobrazené schémy štyroch navrhnutých komplexných komunikačných scenárov.



Obr. 2.12: Schémy navrhnutých komplexných komunikačných scenárov.

Komunikačný scenár *V*. Tento komunikačný scenár definuje komunikáciu medzi jedným objektom typu *Publisher* a ôsmimi objektami typu *Subscriber*. Objekt typu *Publisher* má priradený jeden objekt typu *DataWriter*, každý z objektov typu *Subscriber* má priradený jeden objekt typu *DataReader* a definovaný je jeden objekt typu *Topic*. Schéma komunikačného scenára *V* je zobrazená na obrázku 2.12.

Komunikačný scenár *VI*. Komunikačný scenár *VI* rieši komunikáciu medzi dvomi objektami typu *Publisher* a siedmymi objektami typu *Subscriber*. Každému objektu typu *Publisher* je priradený jeden objekt typu *DataWriter*, každému objektu typu *Subscriber* sú priradené dva objekty typu *DataReader* a definovaný je jeden objekt typu *Topic*. Všetky objekty typu *Subscriber* odoberajú dáta od oboch objektov typu *Publisher*. Schéma komunikačného scenára *VI* je zobrazená na obrázku 2.12.

Komunikačný scenár *VII*. Komunikačný scenár *VII* sa zaoberá komunikáciou medzi tromi objektami typu *Publisher* a štyrmi objektami typu *Subscriber*. Každý objekt typu *Publisher* má priradený jeden objekt typu *DataWriter*, každý objekt typu *Subscriber* má priradené tri objekty typu *DataReader* a definovaný je jeden objekt typu *Topic*. Schéma komunikačného scenára *VII* je zobrazená na obrázku 2.12.

Komunikačný scenár *VIII*. Tento komunikačný scenár sa zaoberá komunikáciou medzi tromi objektami typu *Publisher* a jedným objektom typu *Subscriber*. Každý objekt typu *Publisher* má priradený jeden objekt typu *DataWriter*, objekt typu *Subscriber* má priradené tri objekty typu *DataReader* a definované sú tri objekty typu *Topic*, pričom každý z objektov typu *DataWriter* má priradený jeden objekt typu *Topic*. Schéma komunikačného scenára *VIII* je zobrazená na obrázku 2.12.

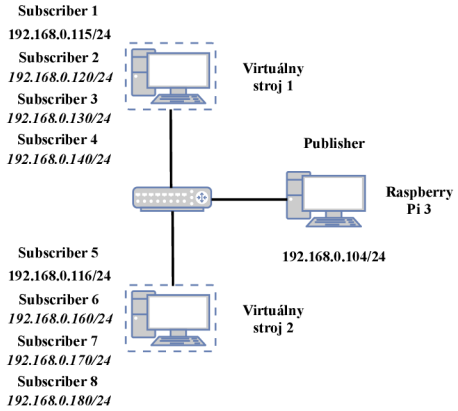
2.6 Simulácia komplexných komunikačných scenárov

V tejto kapitole sú prevedené simulácie navrhnutých komplexných komunikačných scenárov. Vo všetkých štyroch prípadoch je využitá pre simuláciu aplikácia Messenger, ktorá je implementovaná v DDS implementácii OpenDDS. Upravený je počet odosielaných správ, ktorý je nastavený pre všetky objekty typu *DataWriter* na tisíc správ.

2.6.1 Simulácia komunikačného scenára *V*

Komunikačný scenár *V*, ktorý rieši komunikáciu medzi jedným objektom typu *Publisher* a ôsmimi objektami typu *Subscriber* je simulovaný bez časti zabezpečenia. Komunikácia prebieha medzi jednodoskovým počítačom Raspberry Pi 3 a dvomi virtuálnymi strojmi. Proces objektu typu *Publisher* je spustený na jednodoskovom

počítači Raspberry Pi 3 a procesy objektov typu *Subscriber* sú spustené na virtuálnych strojoch, pričom na každom z nich sú spustené štyri procesy. Komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení je na obrázku 2.13.

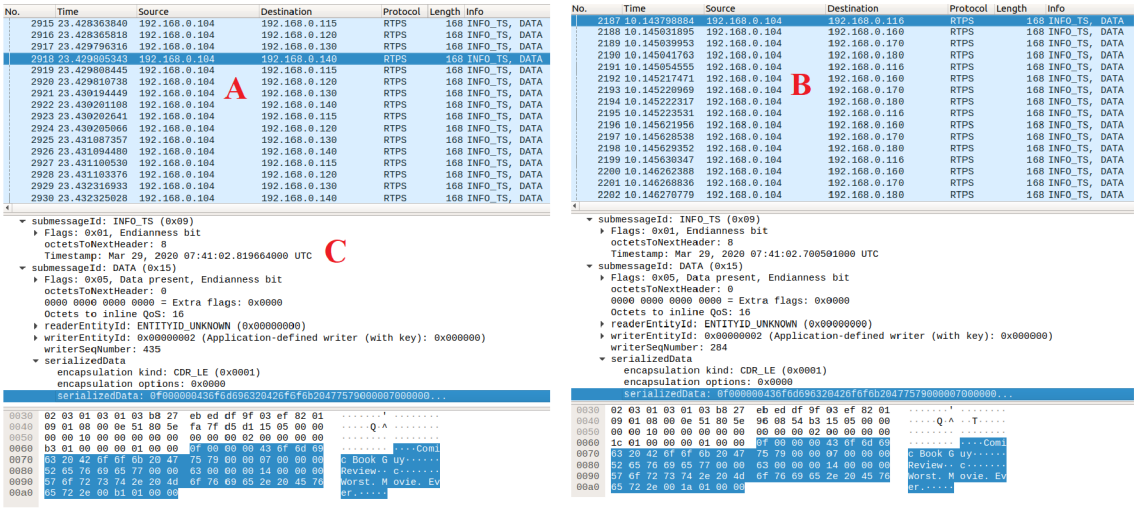


Obr. 2.13: Komunikačný scenár V - komunikačná topológia.

Výpis 2.9 zobrazuje spustenie procesu objektu typu *Publisher*.

Výpis 2.9: Komunikačný scenár V - spustenie procesu objektu typu *Publisher*.

```
pi@raspberrypi:~/OpenDDS-3.13.3/build/target/tests/DCPS/Messenger $ ./publisher -DC
PSConfigFile rtps-nezabezpecena-104.ini
```



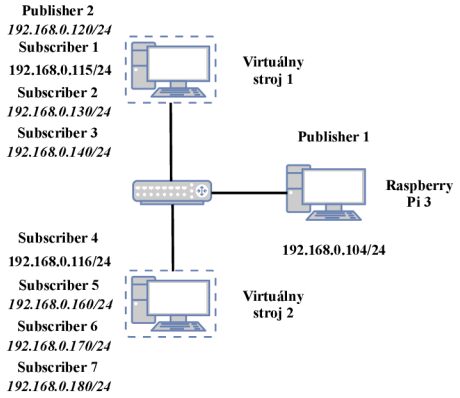
Obr. 2.14: Komunikačný scenár V - odoberanie dát.

Z obrázku 2.14 je možné vidieť súčasné odoberanie dát všetkými objektami typu *Subscriber*, pričom vľavo v časti A je zachytené odoberanie dát z virtuálneho stroja

1, vpravo v časti B je zachytené odoberanie dát všetkými objektami typu *Subscriber* z virtuálneho stroja 2 a časť C dokumentuje časové razítko, kedy je možné vidieť, že sa jedná o súčasné odoberanie dát všetkými odoberajúcimi entitami.

2.6.2 Simulácia komunikačného scenára *VI*

V komunikačnom scenári *VI* participujú dva objekty typu *Publisher* a sedem objektov typu *Subscriber*. Nakoľko v komunikačnom scenári *VI* vystupujú dva objekty typu *Publisher* je pre objekt typu *Publisher2*, ktorého proces je spustený na virtuálnom stroji 1 upravená aplikácia Messenger, aby bolo možné lepšie rozoznať odoberanie dát od dvoch rôznych objektov typu *Publisher*. Úprava spočíva v zmene odosielateľa správy, predmetu správy a textu správy. Identifikácia prijatých správ začína na čísle 1000. Proces objektu typu *Publisher1* je spustený na jednodoskovom počítači Raspberry Pi 3, na virtuálnom stroji 1 sú tiež spustené tri procesy objektov typu *Subscriber* a na virtuálnom stroji 2 sú opäť spustené štyri procesy objektov typu *Subscriber*. Na obrázku 2.15 je zobrazená komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení.



Obr. 2.15: Komunikačný scenár *VI* - komunikačná topológia.

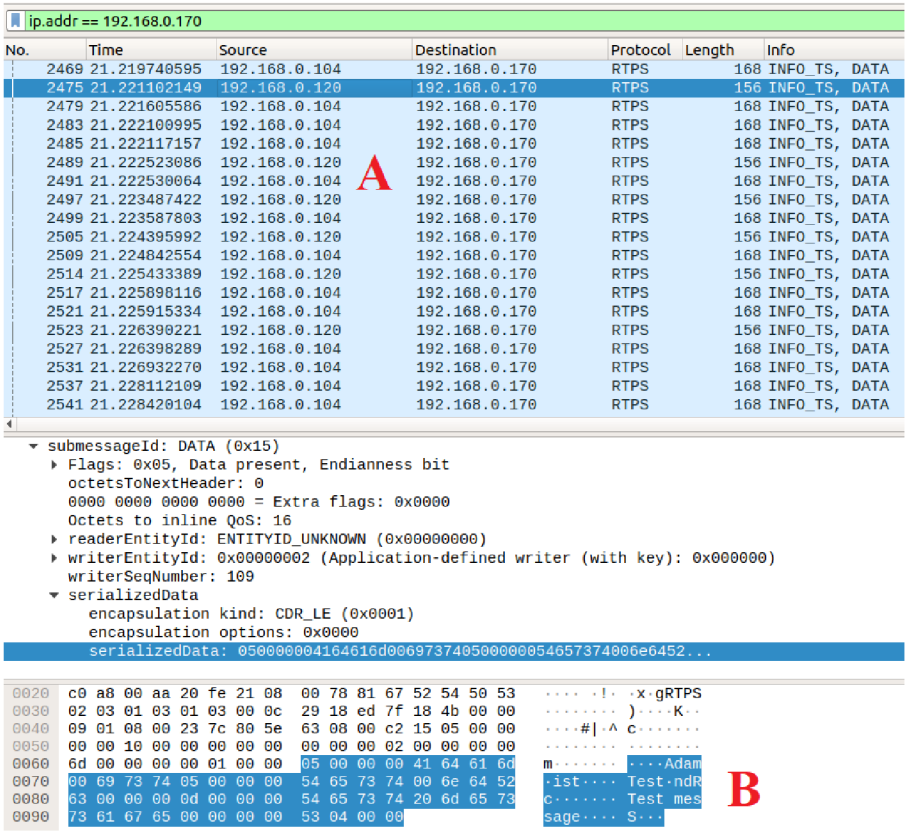
Súčasné odoberanie dát od oboch publikujúcich entít je možné vidieť na výpise 2.10, ktorý zobrazuje ukážku výpisu z terminálu objektu typu *Subscriber6* spusteného na virtuálnom stroji 2.

Výpis 2.10: Komunikačný scenár *VI* - odoberanie dát - Subscriber 6.

```
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 255
         text       = Worst. Movie. Ever.
```

```
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject     = Test
         subject_id  = 99
         from        = Adam
         count       = 1088
         text        = Test message
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject     = Review
         subject_id  = 99
         from        = Comic Book Guy
         count       = 256
         text        = Worst. Movie. Ever.
```

Obrázok 2.16 v časti A ukazuje striedavé odoberanie dát objektom typu *Subscriber6* od oboch publikujúcich entít a v časti B je možné vidieť zachytenú správu, vrátane predmetu a odosielateľa správy.

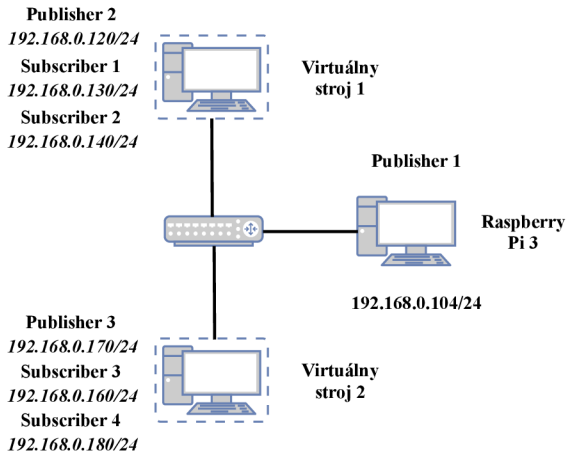


Obr. 2.16: Komunikačný scenár *VI* - odoberanie dát.

2.6.3 Simulácia komunikačného scenára *VII*

Komunikačný scenár *VII* rieši komunikáciu medzi tromi objektami typu *Publisher* a štyrmi objektami typu *Subscriber* a je simulovaný bez časti zabezpečenia. Rovnako

ako pre komunikačný scenár *VI* je aj v tomto prípade upravená aplikácia Messenger, aby bolo možné lepšie rozoznať odoberanie dát od troch rôznych objektov typu *Publisher*. Dáta publikované objektom typu *Publisher2* sú rovnaké ako pri komunikačnom scenári *VI* a jeho proces je opäť spustený na virtuálnom stroji 1. V tomto scenári sú upravené aj dáta publikované objektom typu *Publisher3*, ktorého proces je spustený na virtuálnom stroji 2. Identifikácia prijatých správ od objektu typu *Publisher3* začína na čísle 2000. Proces objektu typu *Publisher1* je spustený na jednodoskovom počítači Raspberry Pi 3. Na oboch virtuálnych strojoch sú taktiež spustené po dva procesy objektov typu *Subscriber*. Komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení sa nachádza na obrázku 2.17.



Obr. 2.17: Komunikačný scenár *VII* - komunikačná topológia.

Spustenie procesov všetkých troch objektov typu *Publisher* zobrazuje výpis 2.11.

Výpis 2.11: Komunikačný scenár *VII* - spustenie procesov objektov typu *Publisher*.

```
pi@raspberrypi:~/OpenDDS-3.13.3/build/target/tests/DCPS/Messenger $ ./publisher -DC
PSConfigFile rtps-nezabezpecena-104.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger1$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-120.ini

student@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger2$ ./publisher -DCPSConfig
File rtps-nezabezpecena-170.ini
```

Výpis 2.12 zobrazuje ukážku výpisu z terminálu objektu typu *Subscriber1* spusteného na virtuálnom stroji 1, kde je možné vidieť súčasné odoberanie správ od všetkých troch publikujúcich entít.

Výpis 2.12: Komunikačný scenár *VII* - odoberanie dát - Subscriber 1.

```
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Test1
         subject_id = 99
         from       = Student
         count      = 2219
         text       = Test message1
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 577
         text       = Worst. Movie. Ever.
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Review
         subject_id = 99
         from       = Comic Book Guy
         count      = 578
         text       = Worst. Movie. Ever.
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = Test
         subject_id = 99
         from       = Adam
         count      = 1028
         text       = Test message
```

2.6.4 Simulácia komunikačného scenára *VIII*

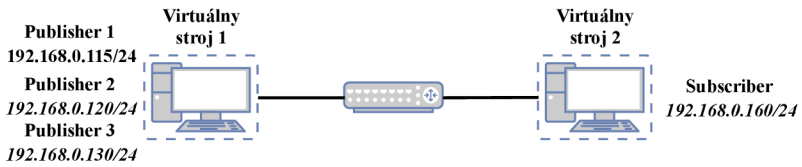
V komunikačnom scenári *VIII* komunikujú tri objekty typu *Publisher* a jeden objekt typu *Subscriber*. Keďže v tomto komunikačnom scenári participujú tri objekty typu *Publisher* a každý z nich má priradený jeden objekt typu *DataWriter* s rozdielnym objektom typu *Topic* je nutné upraviť zdrojový kód objektu typu *Subscriber* a priradiť jednotlivým objektom typu *DataReader* konkrétne objekty typu *Topic*. Na výpise 2.13 sa nachádza ukážka vytvorenia objektu typu *TopicB* a jeho následne priradenie objektu typu *DataReader*. Okrem toho sú upravené pre lepšiu identifikáciu aj objekty typu *DataWriter* podobne ako v predchádzajúcich prípadoch.

Výpis 2.13: Vytvorenie objektu typu *TopicB* a priradenie objektu typu *DataReader*.

```
// Create TopicB
CORBA::String_var type_name1 = ts->get_type_name();
DDS::Topic_var topicB =
participant->create_topic("TopicB",
                          type_name1.in(),
                          TOPIC_QOS_DEFAULT,
                          DDS::TopicListener::_nil(),
                          OpenDDS::DCPS::DEFAULT_STATUS_MASK);
```

```
// Create DataReader
DDS::DataReader_var reader1 =
sub->create_datareader(topicB.in(),
                       dr_qos1,
                       listener1.in(),
                       OpenDDS::DCPS::DEFAULT_STATUS_MASK);
```

Procesy objektov typu *Publisher* sú spustené na virtuálnom stroji 1 a proces objektu typu *Subscriber* je spustený na virtuálnom stroji 2. Na obrázku 2.18 je zobrazená komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení.



Obr. 2.18: Komunikačný scenár *VIII* - komunikačná topológia.

Výpis 2.14 zobrazuje spustenie procesov objektov typu *Publisher*.

Výpis 2.14: Komunikačný scenár *VIII* - spustenie procesov objektov typu *Publisher*.

```
adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-A$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-115.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-B$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-120.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-C$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-130.ini
```

Odoberanie dát objektom typu *Subscriber* od všetkých troch objektov typu *Publisher*, ktoré publikujú dáta s rozdielnymi objektami typu *Topic* je zachytené na výpise 2.15.

Výpis 2.15: Komunikačný scenár *VIII* - odoberanie dát - Subscriber.

```
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = TopicA
         subject_id = 19
         from       = Pub1
         count      = 287
         text       = AAAAAAAAA
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = TopicB
         subject_id = 29
         from       = Pub2
         count      = 1274
```

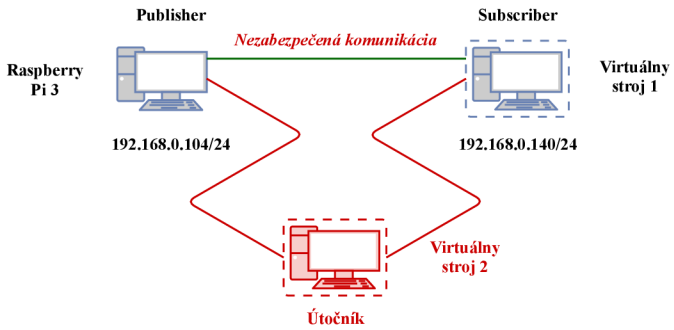
```
        text       = BBBBBBBBB
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = TopicC
        subject_id = 39
        from       = Pub3
        count      = 2252
        text       = CCCCCCCCC
```

2.7 Overenie bezpečnosti implementácie

Táto kapitola sa zaoberá overením bezpečnosti implementácie služby distribúcie dát OpenDDS. Z hľadiska overenia bezpečnosti akejkoľvek implementácie služby distribúcie dát je kľúčové, či komunikácia medzi participujúcimi entitami prebieha zabezpečene alebo nezabezpečene. Mohlo by sa zdať, že pri zabezpečenej komunikácii nie je útočník schopný získať dáta k svojmu účelu, avšak v tomto prípade záleží na tom, či sú šifrované aj výsledné dáta alebo nie. Tieto tri scenáre sú použité pre overenie bezpečnosti OpenDDS. Na simuláciu komunikačných scenárov je opäť využitá aplikácia Messenger. Vo všetkých troch scenároch prebieha komunikácia medzi jednodoskovým počítačom Raspberry Pi 3 (objekt typu *Publisher*) a jedným z virtuálnych strojov (objekt typu *Subscriber*). Druhý z virtuálnych strojov pôsobí v sieti ako útočník, ktorý odchytáva komunikáciu, ktorá pre neho nie je určená. Pre útok je využitý softvér Ettercap, pomocou ktorého je nasadený útok muža uprostred (man in the middle) a konkrétne je využitá technika ARP spoofing.

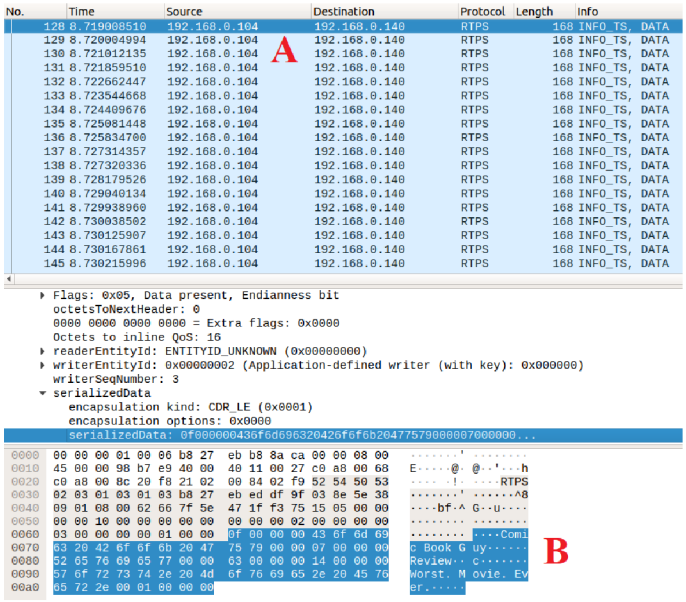
2.7.1 Nezabezpečená komunikácia

Komunikačná topológia tohoto komunikačného scenára sa nachádza na obrázku 2.19.



Obr. 2.19: Overenie bezpečnosti - nezabezpečená komunikácia.

V rámci tohoto scenára prebieha komunikácia bez akejkoľvek časti zabezpečenia medzi jedným objektom typu *Publisher* a jedným objektom typu *Subscriber*. Pomocou útoku muža uprostred je útočník schopný odchytiť danú komunikáciu a môže tak získať citlivé informácie. Obrázok 2.20 zobrazuje komunikáciu zachytenú útočníkom. Časť A na obrázku 2.20 zobrazuje IP adresy komunikujúcich entít a v časti B sa nachádza zachytená správa, ktorú je ako je možné vidieť zobraziť vzhľadom na absenciu časti zabezpečenia v tomto komunikačnom scenári.



Obr. 2.20: Overenie bezpečnosti - zachytená nezabezpečená komunikácia.

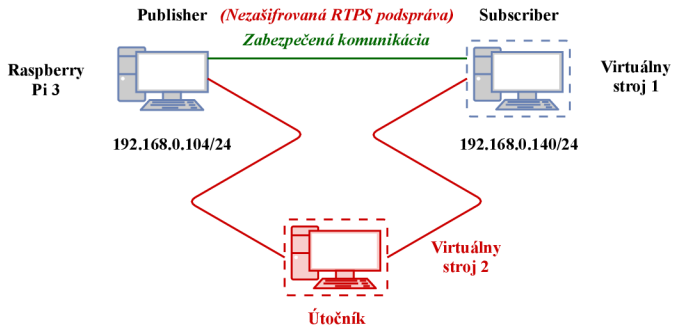
2.7.2 Nezašifrovaná RTPS podspráva

V prípade využitia časti zabezpečenia je nevyhnutné v prvom rade vygenerovať bezpečnostné certifikáty podľa [22]. Na generovanie bezpečnostných certifikátov je použitý nástroj OpenSSL. V rámci tejto práce sú všetky bezpečnostné certifikáty generované s využitím 2048-bitového RSA algoritmu. Okrem bezpečnostných certifikátov sú pre časť zabezpečenia dôležité aj dokumenty *governance.xml* a *permissions.xml*, ktoré boli predstavené v rámci kapitoly 1.5.4. V dokumente *governance.xml* je okrem iného definované, či majú byť dáta zašifrované. Na výpise 2.16 sa nachádza nastavenie parametru pre šifrovanie RTPS podspráv. Tento parameter definuje akým spôsobom sú RTPS podsprávy chránené a pre tento scenár je nastavený na hodnotu NONE z čoho vyplýva, že RTPS podsprávy šifrované nie sú. Zoznam najdôležitejších nastaviteľných parametrov v rámci oboch dokumentov je možné nájsť v [22]. Obidva dokumenty musia byť následne podpísané certifikačnou autoritou.

Výpis 2.16: Nastavenie nezašifrovanej RTPS podsprávy.

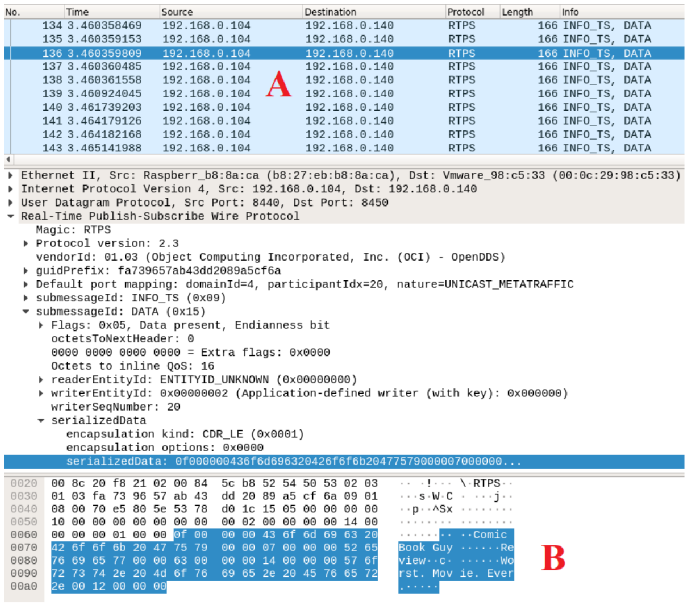
```
<metadata_protection_kind>NONE</metadata_protection_kind>
```

Komunikačná topológia scenára simulujúceho zabezpečenú komunikáciu bez šifrovania RTPS podsprávy sa nachádza na obrázku 2.21.



Obr. 2.21: Overenie bezpečnosti - zabezpečená komunikácia bez šifrovania.

Vzhľadom k tomu, že komunikácia síce prebieha zabezpečene, ale dáta nie sú šifrované môže útočník aj v tomto prípade získať citlivé informácie. Obrázok 2.22 zobrazuje v časti A IP adresy komunikujúcich entít a v časti B poukazuje na zachytenú správa, ktorá je zobraziteľná pre útočníka, nakoľko RTPS podspráva nie je šifrovaná.



Obr. 2.22: Overenie bezpečnosti - zachytená zabezpečená komunikácia bez šifrovania.

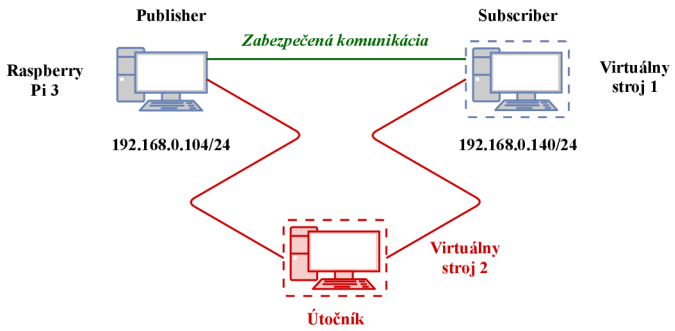
2.7.3 Zašifrovaná RTPS podspráva

Pre tento komunikačný scenár platia rovnaké náležitosti ako pre vyššie definovaný scenár zabezpečenej komunikácie. Jediným rozdielom je, že v rámci tohoto komunikačného scenára je RTPS podspráva šifrovaná. Výpis 2.17 zobrazuje nastavenie parametru pre šifrovanie RTPS podspráv, pričom pre tento scenár je jeho hodnota nastavená na ENCRYPT.

Výpis 2.17: Nastavenie zašifrovanej RTPS podsprávy.

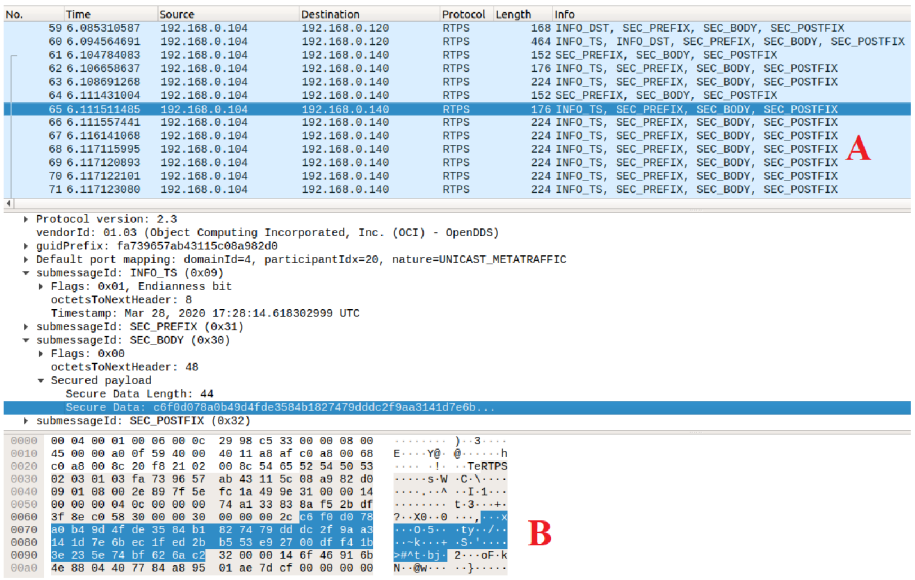
```
<metadata_protection_kind>ENCRYPT</metadata_protection_kind>
```

Komunikačná topológia využitá v tomto scenári je na obrázku 2.23.



Obr. 2.23: Overenie bezpečnosti - zabezpečená komunikácia.

V tomto prípade útočník zo zachytenej komunikácie nie je schopný zistiť žiadne citlivé dáta, keďže sú šifrované. Z toho vyplýva, že táto varianta so zabezpečením je z hľadiska bezpečnosti najvhodnejšia a najbezpečnejšia. Napriek tomu, že implementácia OpenDDS doposiaľ využíva iba beta verziu OMG DDS Security, zabezpečuje v rámci možností bezpečnú komunikáciu medzi participujúcimi entitami. Obrázok 2.24 v časti A zobrazuje štruktúru zabezpečených RTPS správ a v časti B preukazuje nemožnosť získania citlivých informácii útočníkom.



Obr. 2.24: Overenie bezpečnosti - zachytená zabezpečená komunikácia.

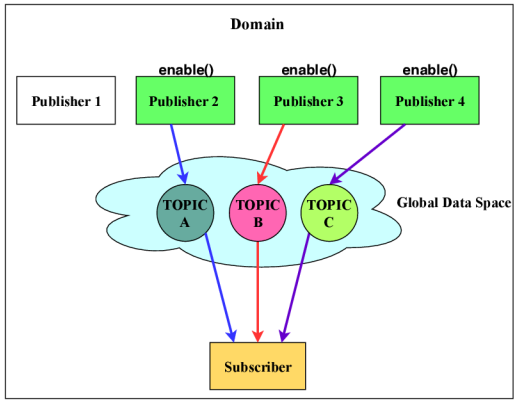
2.8 Návrh bezpečnostných incidentov

Pri absencii časti zabezpečenia u služby distribúcie dát existuje niekoľko potencionálnych bezpečnostných hrozieb, ktoré sú popísané v kapitole 1.5.2. Analýzou bezpečnosti DDS sa zaoberá výskum [31], v ktorom je definovaných 60 bezpečnostných problémov. Táto práca čerpá z analýzy spomínaných bezpečnostných problémov definovaných v [31]. Ako už vyplýva z analýzy DDS implementácií v kapitole 1.6, niektoré z DDS implementácií majú licenciu platenú, zatiaľ čo niektoré DDS implementácie sú s otvoreným zdrojovým kódom, ako napríklad OpenDDS, čo je prvý predpoklad toho, že potencionálny útočník môže svoj útok ľahšie prispôsobiť na konkrétny cieľový systém.

2.8.1 Zneužitie QoS politiky ENTITY_FACTORY

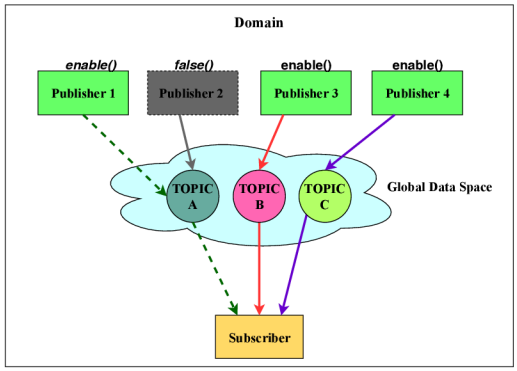
QoS politika ENTITY_FACTORY slúži k riadeniu povolenia jednotlivých entít DCPS modelu po ich vytvorení. Prednastavenou hodnotou tejto politiky je automatické povolenie entít po ich vytvorení [8]. Ak chce napríklad aplikácia zakázať špecifickému objektu typu *Publisher* jej povolenie po vytvorení, nastaví QoS politiku ENTITY_FACTORY na hodnotu *false* a manuálne povolí entity, ktoré môžu publikovať dáta. Hodnota tejto QoS politiky sa môže meniť kedykoľvek a z toho vyplýva aj jej možné zneužitie. Možné zneužitie spočíva napríklad v prípade ak si aplikácia, ktorá pomocou svojho objektu typu *Subscriber* odoberá dáta neželá

odoberať dáta od konkrétneho objektu typu *Publisher* a dôjde k napadnutiu tejto aplikácie odoberajúcej dáta. Následne môže dôjsť k povoleniu daného nežiadaného objektu typu *Publisher* a aplikácia odoberajúca dáta nebude prijímať dáta iba od povolených objektov typu *Publisher*, ale aj od nežiadanej publikujúcej entity.
Na obrázku 2.25 je zobrazený požadovaný komunikačný scenár, kedy je QoS politika ENTITY_FACTORY nastavená na hodnotu *false* a povolené (*enable()*) publikujúce entity sú *Publisher2*, *Publisher3*, *Publisher4*. Entita *Publisher1* je nevyžiadanou a keďže nie je povolená nemôže publikovať svoje dáta smerom k objektu typu *Subscriber*.



Obr. 2.25: Komunikačný scenár bez zneužitia (ENTITY_FACTORY).

V prípade napadnutia aplikácie, ktorá dáta odoberá a povoleniu objektu typu *Publisher1*, môže daná entita dáta publikovať a objekt typu *Subscriber* bude odoberať aj pôvodne nevyžiadané dáta. Ďalším typom zneužitia tejto politiky kvality služby v prípade napadnutia je zakázanie publikujúcej entity, ktorá je pôvodne vyžiadaná. Tieto dve situácie zobrazuje obrázok 2.26.

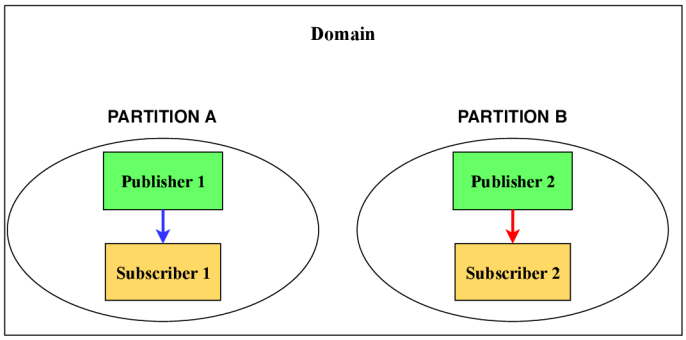


Obr. 2.26: Komunikačný scenár po zneužití (ENTITY_FACTORY).

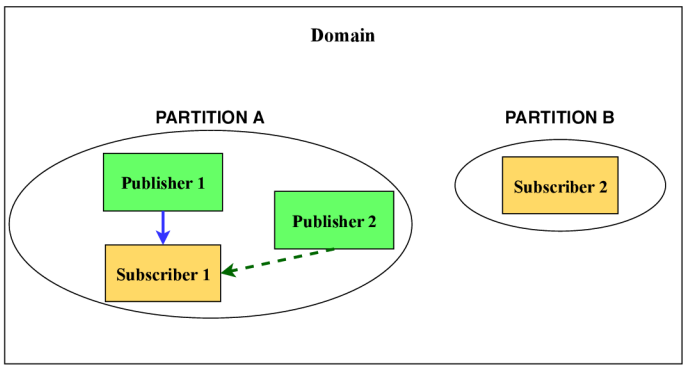
2.8.2 Modifikácia QoS politiky PARTITION

Vďaka QoS politike PARTITION je možné logicky rozdeliť doménu na menšie časti. Jedná sa o QoS politiku, ktorá sa priraďuje iba objektom typu *Publisher* a *Subscriber*. Táto QoS politika umožňuje asociáciu objektov typu *DataWriter* a *DataReader* iba ak sa zhodujú v názve PARTITION, pričom názov tejto QoS politiky je daný dátovým typom String. Prednastavenou hodnotou tejto politiky je prázdny dátový typ String, čo znamená, že daná entita participuje v rámci celej domény [8]. Hodnota QoS politiky PARTITION sa môže kedykoľvek meniť, čo môže mať za následok vytvorenie alebo rušenie asociácii a z toho vyplýva aj možné zneužitie tejto QoS politiky.

Na obrázku 2.27 je zobrazený komunikačný scenár, kedy je doména rozdelená logicky na dve časti. V PARTITION **A** komunikuje jeden objekt typu *Publisher1* a jeden objekt typu *Subscriber1*. V PARTITION **B** komunikuje jeden objekt typu *Publisher2* a jeden objekt typu *Subscriber2*.



Obr. 2.27: Komunikačný scenár bez zneužitia (PARTITION).

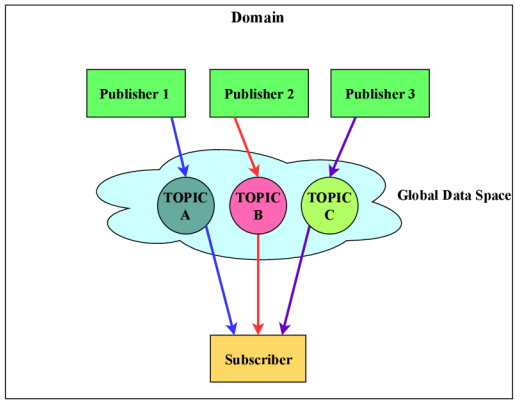


Obr. 2.28: Komunikačný scenár po zneužití (PARTITION).

Možné zneužitie QoS politiky PARTITION je naznačené na obrázku 2.28. V prípade ak dôjde k napadnutiu entít komunikujúcich v PARTITION **A** a útočník (*Publisher2*) zistí názov QoS politiky PARTITION, ktorou sú asociované ich objekty typu *DataWriter* a *DataReader*, môže zmeniť názov svojej QoS politiky PARTITION na zistený názov z PARTITION **A** a objekt typu *Subscriber* začne prijímať aj nevyžiadané dáta od objektu typu (*Publisher2*).

2.8.3 Zneužitie QoS politiky LIFESPAN

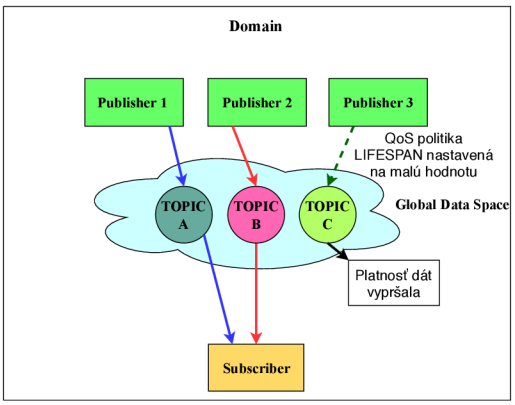
QoS politika LIFESPAN umožňuje aplikácii určiť časovú platnosť konkrétnych dát. V prípade, ak daným dátam vyprší platnosť nebudú doručené entite, ktorá tieto dáta odoberá. QoS politika LIFESPAN sa vzťahuje na objekty typu *DataWriter* a *Topic* [8]. Prednastavená hodnota pre túto QoS politiku je nekončená, čo znamená, že dáta nikdy nestratia platnosť. Hodnota tejto QoS politiky sa môže kedykoľvek zmeniť rovnako ako v predchádzajúcich prípadoch a taktiež z toho plynie možné zneužitie tejto QoS politiky.



Obr. 2.29: Komunikačný scenár bez zneužitia (LIFESPAN).

Na obrázku 2.29 je zobrazený požadovaný komunikačný scenár, kedy má QoS politika LIFESPAN nastavenú prednastavenú hodnotu a teda časová platnosť publikovaných vzoriek je nekonečná. Komunikácia prebieha medzi tromi objektami typu *Publisher* a jedným objektom typu *Subscriber*.

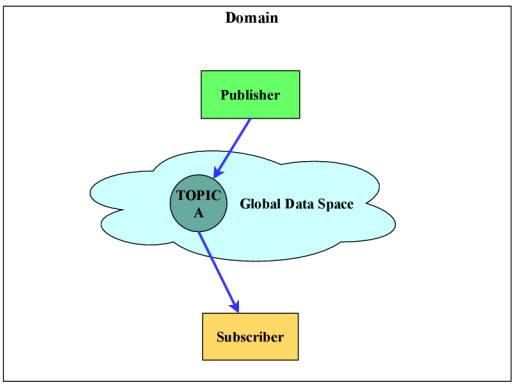
Zneužitie QoS politiky LIFESPAN je naznačené na obrázku 2.30. V tomto prípade objekt typu *Publisher3* vďaka konfiguračnému súboru, pomocou ktorého je možné zmeniť hodnotu QoS politiky LIFESPAN, nastaví hodnotu QoS politiky LIFESPAN na tak malú hodnotu, že platnosť dát vyprší ešte pred prijatím dát objektom typu *Subscriber* a táto entita sa tak k dátam, ktoré chce odoberať nedostane.



Obr. 2.30: Komunikačný scenár po zneužití (LIFESPAN).

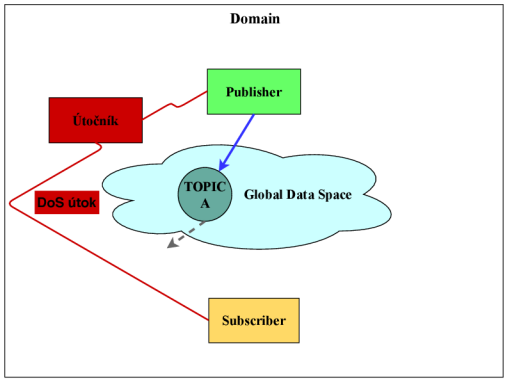
2.8.4 Útok odopretia služby na systém služby distribúcie dát

Útok odopretia služby (DoS) je typ útoku, ktorý má za cieľ znefunkčniť komunikáciu v rámci internetového priestoru. Pre túto prácu je jeho cieľom znemožniť komunikáciu medzi dvomi komunikujúcimi entitami využívajúcimi službu distribúcie dát, ktoré za normálnych okolností môžu spolu komunikovať. Na obrázku 2.31 sa nachádza jednoduchý komunikačný scenár, kde medzi sebou komunikujú jeden objekt typu *Publisher* a jeden objekt typu *Subscriber*.



Obr. 2.31: Komunikačný scenár bez zneužitia (DoS).

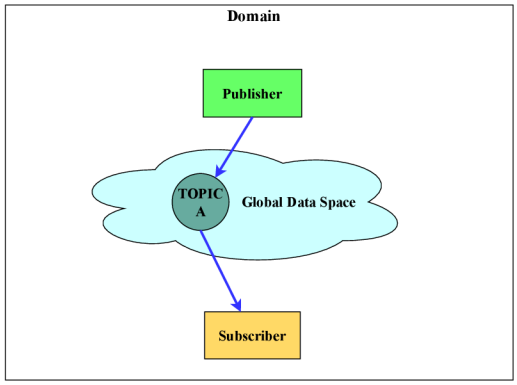
Útok odopretia služby je naznačený na obrázku 2.32, kedy útočník pomocou DoS útoku znefunkční komunikáciu medzi dvomi participujúcimi entitami a objekt typu *Subscriber* tak nemôže odoberať dáta, ktoré mu boli pôvodne určené.



Obr. 2.32: Komunikačný scenár po zneužití (DoS).

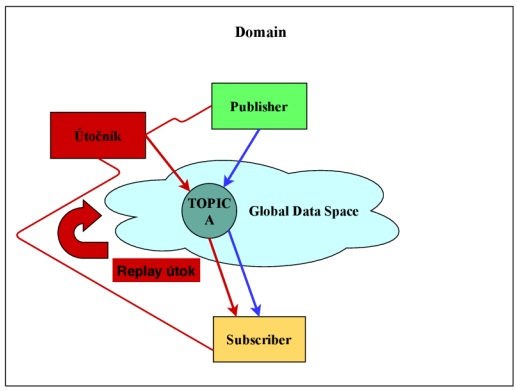
2.8.5 Replay útok na systém služby distribúcie dát

Útok s opakovaním dát (replay) je typ útoku, ktorý má za cieľ zachytiť komunikáciu medzi komunikujúcimi entitami a následne zachytenú komunikáciu opäť umiestniť do siete. V rámci tejto práce je replay útok využitý s cieľom zachytenú komunikáciu pozmeniť a publikovať ju smerom k objektu typu *Subscriber*. Na obrázku 2.33 sa nachádza komunikačný scenár, kedy medzi sebou komunikujú jeden objekt typu *Publisher* a jeden objekt typu *Subscriber*.



Obr. 2.33: Komunikačný scenár bez zneužitia (Replay útok).

Útok s opakovaním dát je naznačený na obrázku 2.34, kedy útočník nachádzajúci sa v sieti zachytáva komunikáciu medzi dvojicou komunikujúcich objektov, následne ju pozmení a publikuje opäť k objektu typu *Subscriber*, ktorý tak prijíma dáta vo väčšom objeme ako očakáva a zároveň sú tieto dáta v určitej miere zmanipulované.



Obr. 2.34: Komunikačný scenár po zneužití (Replay útok).

2.9 Simulácia bezpečnostných incidentov

V tejto kapitole sa nachádzajú simulácie komunikačných scenárov s bezpečnostnými incidentami, ktoré sú navrhnuté v rámci kapitoly 2.8. Postupne sú simulované všetky navrhnuté bezpečnostné incidenty. Na simulácie je použitá rovnako ako pri predchádzajúcich simuláciách aplikácia Messenger, ktorá je implementovaná v DDS implementácii OpenDDS, avšak v rámci každého bezpečnostného incidentu je podľa potreby upravená. Topológia siete je totožná s tou na obrázku 2.11.

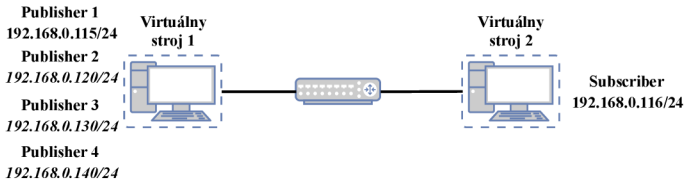
2.9.1 Zneužitie QoS politiky ENTITY_FACTORY - simulácia

Simulácia zneužitia QoS politiky ENTITY_FACTORY navrhnutej v rámci kapitoly 2.8.1 predpokladá zneužitie aplikácie publikujúcej dáta. Úprava aplikácie Messenger pri tejto simulácii spočíva v zmenení prednastavenej hodnoty tejto politiky kvality služby. Ako je uvedené v 2.8.1 prednastavenou hodnotou pri tejto politike kvality služby je, že všetky entity sú automaticky povolené po ich vytvorení. V scenári, ktorý zobrazuje obrázok 2.25 sú povolené iba konkrétne komunikujúce entity. Z toho vyplýva nutnosť nastaviť QoS politiku ENTITY_FACTORY na požadovaný scenár. Na výpise 2.18 sa nachádza ukážka nastavenia QoS politiky ENTITY_FACTORY pre objekt typu *Subscriber* a zároveň aj jeho povolenie. Obdobným spôsobom sú nastavené aj ostatné povolené komunikujúce entity.

Výpis 2.18: Nastavenie QoS politiky ENTITY_FACTORY.

```
DDS::DomainParticipantQos part_qos;
dpf->get_default_participant_qos(part_qos);
part_qos.entity_factory.autoenable_created_entities = false;
sub->enable();
```

Obrázok 2.35 zobrazuje komunikačnú topológiu spolu s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení. Procesy objektov typu *Publisher* sú spustené na virtuálnom stroji 1 a proces objektu typu *Subscriber* je spustený na virtuálnom stroji 2.



Obr. 2.35: Zneužitie QoS politiky ENTITY_FACTORY - komunikačná topológia.

V rámci komunikačného scenára bez zneužitia, ktorý sa nachádza na obrázku 2.25 sú povolené tri publikujúce entity. Objekt typu *Publisher1* nemá povolenie na publikovanie dát. Zneužitím tejto politiky kvality služby, ktorého návrh sa nachádza na obrázku 2.26, je povolenie objektu typu *Publisher1* a zároveň zakázanie objektu typu *Publisher2*, ktorý je pôvodne vyžiadanou komunikujúcou entitou. Výpis 2.19 poukazuje na spustenie všetkých štyroch procesov objektov typu *Publisher*. Na simuláciu sú použité rovnaké zdrojové súbory ako pri simulácii komunikačného scenára *VIII* v 2.6.4, pričom objekty typu *Publisher1* a *Publisher2* publikujú dáta s rovnakým objektom typu *Topic*.

Výpis 2.19: Zneužitie QoS politiky ENTITY_FACTORY - spustenie procesov objektov typu *Publisher*.

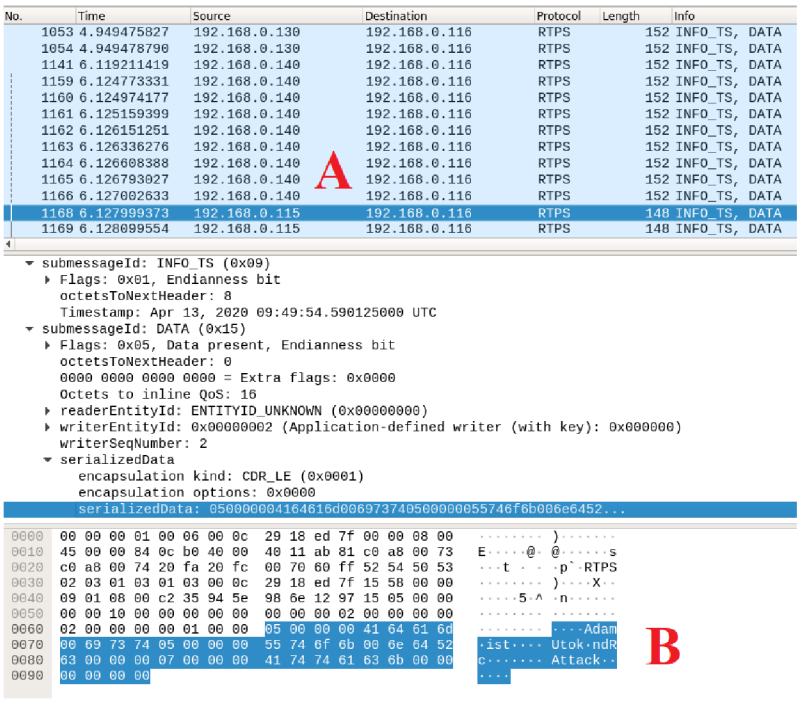
```
adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-entityfactory$ ./publisher
-DCPSConfigFile rtps-nezabezpecena-115.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-A$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-120.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-B$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-130.ini

adam@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/Messenger-C$ ./publisher -DCPSConfigFile
rtps-nezabezpecena-140.ini
```

Na obrázku 2.36 v časti A je možné vidieť, že jednou z publikujúcich entít je objekt typu *Publisher1* a zároveň, že objekt typu Publisher2 nie je schopný dáta publikovať. Objekt typu *Subscriber* tak odoberá dáta od pôvodne nevyžiadanej entity a naopak dáta, o ktoré mal záujem neodoberá. Zároveň časť B na obrázku 2.36 poukazuje, že hoci sú dáta publikované objektom typu *Publisher1* s rovnakým objektom typu *Topic* tak tieto dáta sa líšia od tých, ktoré mal pôvodne publikovať objekt typu *Publisher2*.



Obr. 2.36: Bezpečnostný incident - zneužitie QoS politiky ENTITY_FACTORY.

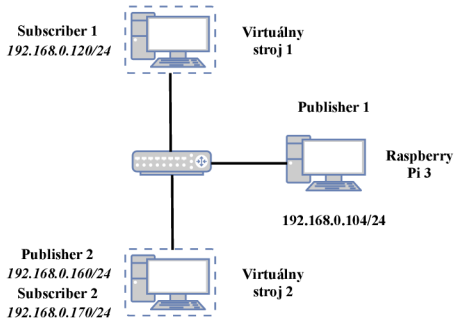
2.9.2 Modifikácia QoS politiky PARTITION - simulácia

Táto podkapitola sa zaoberá simuláciou bezpečnostného incidentu, ktorý je navrhnutý v kapitole 2.8.2. Aby boli jednotlivé entity v rámci domény logicky oddelené tak ako ich zobrazuje obrázok 2.27 je potrebné upraviť aplikáciu Messenger. Konkrétne sa úprava týka zdrojových súborov objektov typu *Publisher* a *Subscriber*. Na výpise 2.20 sa nachádza ukážka nastavenia QoS politiky PARTITION pre objekt typu *Publisher1*. Obdobným spôsobom je nastavená QoS politika PARTITION aj pre ostatné komunikujúce entity.

Výpis 2.20: Nastavenie QoS politiky PARTITION.

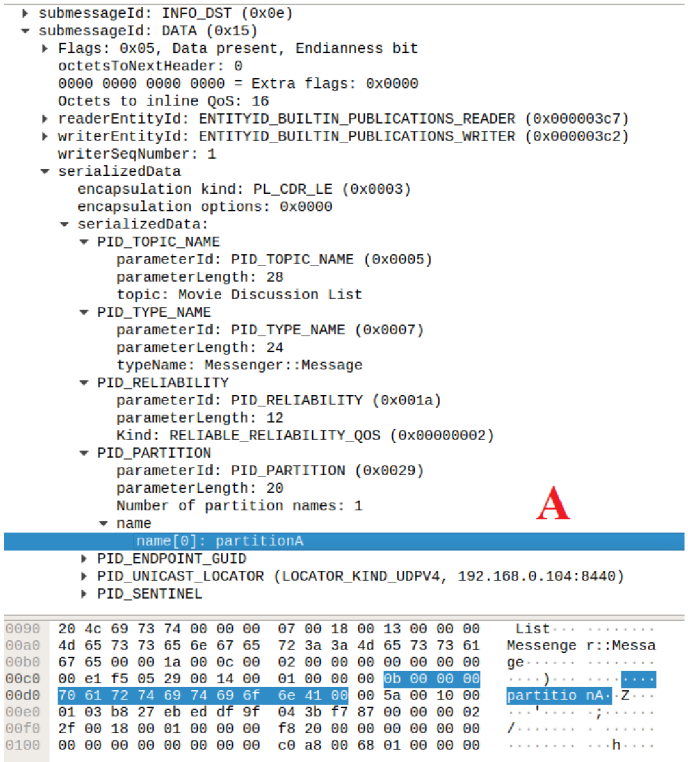
```
DDS::PublisherQos pub_qos;
participant->get_default_publisher_qos (pub_qos);
pub_qos.partition.name.length(1);
pub_qos.partition.name[0] = CORBA::string_dup("partitionA");
```

Proces objektu typu *Publisher1* je spustený na jednodoskovom počítači Raspberry Pi 3 a proces objektu typu *Subscriber1* je spustený na virtuálnom stroji 1. Procesy objektov komunikujúcich v PARTITION B sú spustené na virtuálnom stroji 2. Na obrázku 2.37 sa nachádza komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení.



Obr. 2.37: Modifikácia QoS politiky PARTITION - komunikačná topológia.

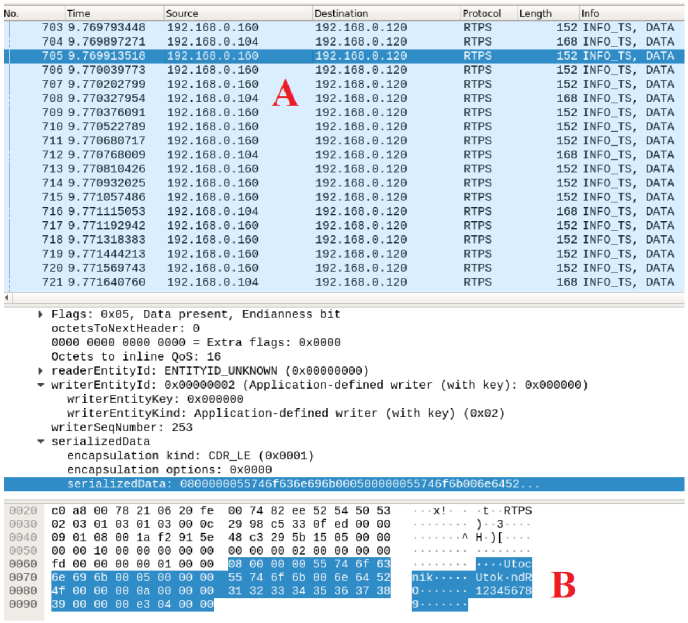
Ako je zobrazené na obrázku 2.27 objekty typu *Publisher1* a *Subscriber1* komunikujú v PARTITION A. Bezpečnostný incident je zobrazený na obrázku 2.28. Útočník, ktorým je objekt typu *Publisher2* pomocou útoku muža uprostred a techniky ARP spoofing odchytí komunikáciu medzi objektami typu *Publisher1* a *Subscriber1*. Obrázok 2.38 zobrazuje zachytenú komunikáciu a v časti A poukazuje na útočníkom zistený názov QoS politiky PARTITION.



Obr. 2.38: Bezpečnostný incident - zistenie názvu QoS politiky PARTITION.

Následne môže útočník po nastavení QoS politiky PARTITION a vďaka zisteným informáciám ako sú nastavenia ostatných politík kvality služby, dátový typ

správy alebo názov objektu typu *Topic*, publikovať správy smerom k objektu typu *Subscriber1*. Na obrázku 2.39 je zachytené v časti A odoberanie správ objektom *Subscriber1* od objektu *Publisher1* a zároveň od objektu *Publisher2*, ktorý je v tomto scenári útočiacim objektom a objekt typu *Subscriber1* odoberá teda aj pôvodne nevyžiadané správy. Časť B na obrázku 2.39 zobrazuje správu od útočníka odoberanú objektom typu *Subscriber1*.



Obr. 2.39: Bezpečnostný incident - modifikácia QoS politiky PARTITION.

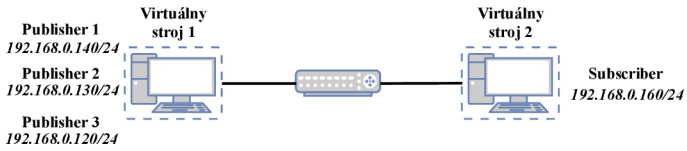
2.9.3 Zneužitie QoS politiky LIFESPAN - simulácia

V kapitole 2.8.3 sa nachádza návrh bezpečnostného incidentu zneužitia QoS politiky LIFESPAN, ktorého simuláciou sa zaoberá táto podkapitola. Obrázok 2.30 zobrazuje návrh zneužitia QoS politiky LIFESPAN, ktoré spočíva v nastavení hodnoty tejto QoS politiky na tak malú hodnotu, že dáta publikované objektom typu *Publisher* stratia platnosť ešte pred ich odobratí objektom typu *Subscriber*. Pre simuláciu sú využité rovnaké upravené zdrojové súbory ako pre simuláciu komunikačného scenára *VIII*. Zmena je v nastavení QoS politiky LIFESPAN pre objekt typu *Topic* v rámci zdrojového súboru objektu typu *Publisher3* a zachytená je na výpise 2.21. V tomto prípade je hodnota QoS politiky LIFESPAN nastavená na dve milisekundy. V rámci DDS implementácie OpenDDS je funkčnosť tejto QoS politiky podporovaná za predpokladu, že QoS politika DURABILITY, ktorá definuje či si objekt typu *DataWriter* uchováva vzorky dát aj po ich odoslaní odoberajúcej entite, má nastavenú inú hodnotu ako je jej prednastavená volatilná hodnota [8].

Výpis 2.21: Nastavenie QoS politiky LIFESPAN.

```
DDS::TopicQos topic_qos;
participant->get_default_topic_qos(topic_qos);
topic_qos.durability.kind = DDS::TRANSIENT_LOCAL_DURABILITY_QOS;
topic_qos.lifespan.duration.sec = 0;
topic_qos.lifespan.duration.nanosec = 2000000;
```

Obrázok 2.40 zobrazuje komunikačnú topológiu s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení. Pre túto simuláciu sú využité oba virtuálne stroje. Procesy objektov typu *Publisher* sú spustené na virtuálnom stroji 1 a proces objektu typu *Subscriber* je spustený na virtuálnom stroji 2.



Obr. 2.40: Zneužitie QoS politiky LIFESPAN - komunikačná topológia.

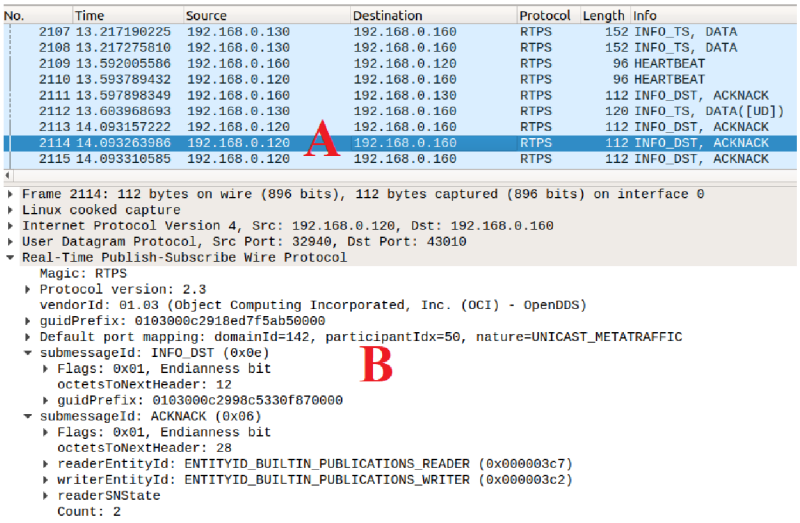
Z výpisu 2.22 je možné vidieť, že objekt typu *Subscriber* odoberá dáta od objektov typu *Publisher1* a *Publisher2* a zároveň nadviazal komunikáciu aj s objektom typu *Publisher3*, avšak dáta od objektu typu *Publisher3* vzhľadom na nastavenú QoS politiku LIFESPAN a ich expiráciu neodoberie.

Výpis 2.22: Zneužitie QoS politiky LIFEPSAN - odoberanie dát.

```
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = TopicA
         subject_id = 19
         from       = Pub1
         count      = 796
         text       = AAAAAAAAA
SampleInfo.sample_rank = 0
SampleInfo.instance_state = 1
Message: subject    = TopicB
         subject_id = 29
         from       = Pub2
         count      = 1835
         text       = BBBBBBBBB
DataReaderListener.cpp:146: INFO: on_subscription_matched()
DataReaderListener.cpp:139: INFO: on_liveliness_changed()
DataReaderListener.cpp:139: INFO: on_liveliness_changed()
DataReaderListener.cpp:146: INFO: on_subscription_matched()
ERROR: received 0 messages, but expected 1000
```

Túto skutočnosť dokumentuje aj obrázok 2.41 kde je možné v časti A vidieť, že objekt typu *Publisher3* sa v sieti nachádza a publikuje svoje dáta, ale vzhľadom na nastavenú hodnotu ich platnosti objekt typu *Subscriber* ich nie je schopný odoberať

a nedostane sa tak k dátam, ktoré mal pôvodne záujem odobrať. Obrázok 2.41 v časti B zobrazuje správu INFO_DST, ktorá poskytuje informácie o predpokladanom cieli jednotlivých následne prenášaných RTPS podspráv.



Obr. 2.41: Bezpečnostný incident - zneužitie QoS politiky LIFESPAN.

2.9.4 Útok DoS na systém služby distribúcie dát - simulácia

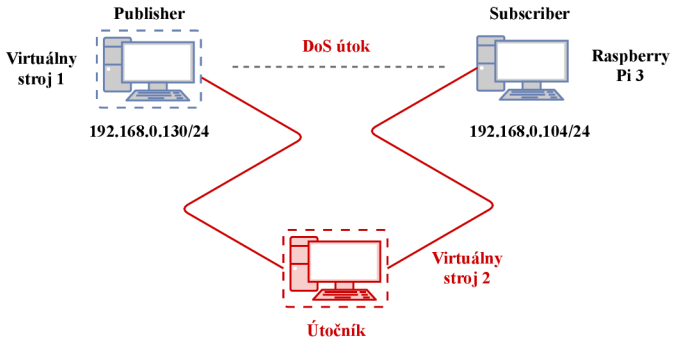
V tejto podkapitole je prevedený útok odopretia služby na systém služby distribúcie dát navrhnutý v 2.8.4. Softvér Ettercap je využitý pre tento útok a v rámci neho je nasadený skript, ktorý sa nachádza na výpise 2.23. V rámci skriptu sú definované zachytené IP adresy komunikujúcich entít, pričom pri komunikácii medzi definovanými IP adresami dôjde k zahodeniu paketov a následne k zrušeniu spojenia medzi nimi.

Výpis 2.23: Skript využitý pre DoS útok.

```
if (ip.src == '192.168.0.130' || ip.dst == '192.168.0.104')
{
drop();
kill();
}
```

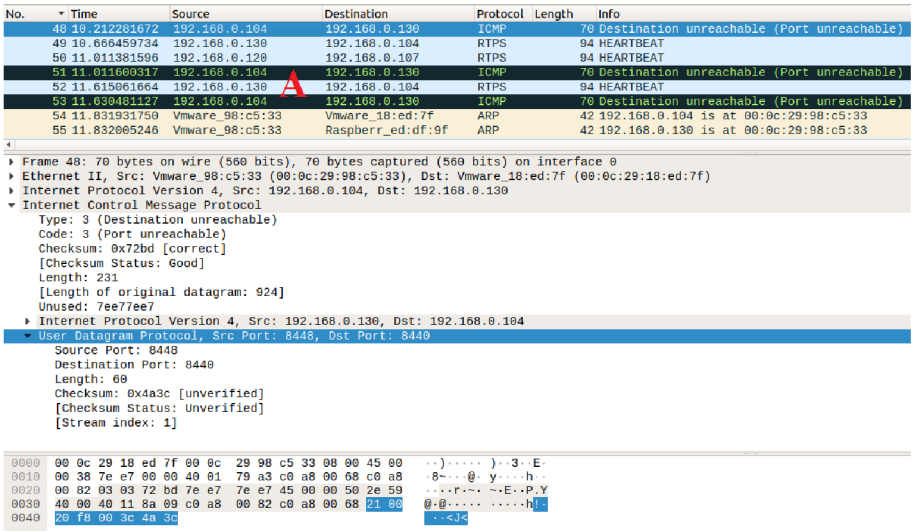
V rámci komunikačného scenára, ktorého návrh sa nachádza na obrázku 2.31 komunikujú jeden objekt typu *Publisher* a jeden objekt typu *Subscriber*. Napadnutie tejto komunikácie zobrazuje obrázok 2.32. Proces objektu typu *Publisher* je spustený na virtuálnom stroji 1 a proces objektu typu *Subscriber* je spustený na jednodoskovom počítači Raspberry Pi 3. Útočník v sieti pracuje na virtuálnom stroji

2. Na obrázku 2.42 sa nachádza komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení.



Obr. 2.42: Útok odopretia služby - komunikačná topológia.

Obrázok 2.43 v časti A zobrazuje nemožnosť naviazania spojenia medzi dvomi participujúcimi entitami v dôsledku útoku odopretia služby. Objekt typu *Publisher* zasiela správu HEARTBEAT, v ktorej sa nachádzajú informácie, ktoré sú k dispozícii v rámci objektu typu *DataWriter* a následne dostáva správu o nedostupnosti cieľového zariadenia. Objekt typu *Subscriber* tak neodoberie žiadnu správu.



Obr. 2.43: Bezpečnostný incident - útok odopretia služby.

Na výpise 2.24 sa nachádza výstup z programu iftop, ktorý je spustený na jednodoskovom počítači Raspberry Pi 3, z ktorého je možné vidieť, že medzi dvojicou participujúcich entít neprechádza takmer žiadna komunikácia. Výpis 2.25 zobrazuje pre porovnanie ako vyzerá tok dát pri komunikácii bez DoS útoku, ktorej návrh sa nachádza na obrázku 2.31

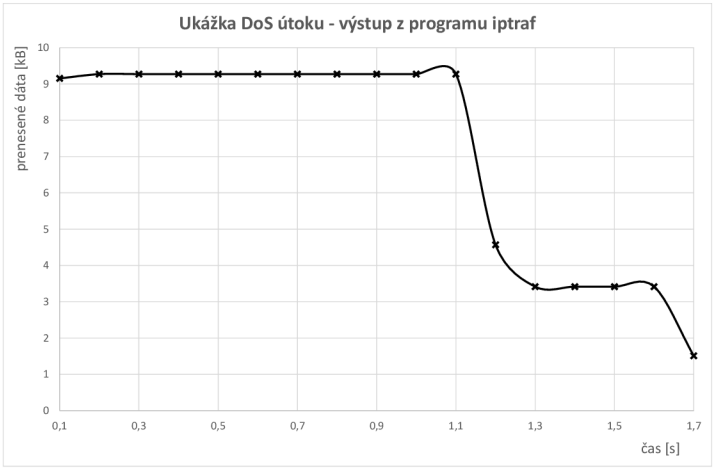
Výpis 2.24: Výstup z programu iftop - DoS útok.

```
Listening on eth0
# Host name (port/service if enabled)       last 2s   last 10s
--------------------------------------------------------------------
1 192.168.0.104                          =>      0b        22b
  192.168.0.130                          <=      0b        45b
--------------------------------------------------------------------
Total send rate:                                 0b        22b
Total receive rate:                              0b        45b
Total send and receive rate:                     0b        67b
====================================================================
```

Výpis 2.25: Výstup z programu iftop - komunikácia bez DoS útoku.

```
Listening on eth0
# Host name (port/service if enabled)       last 2s   last 10s
--------------------------------------------------------------------
1 192.168.0.104                          =>     496b        99b
  192.168.0.130                          <=    595Kb      119Kb
--------------------------------------------------------------------
Total send rate:                                496b        99b
Total receive rate:                            595Kb      119Kb
Total send and receive rate:                   595Kb      119Kb
====================================================================
```

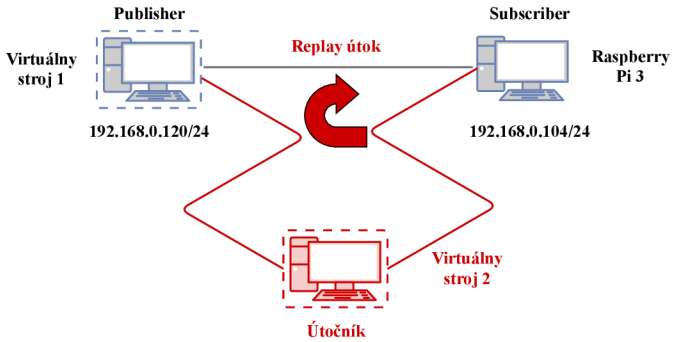
Na obrázku 2.44 sa nachádza graf, ktorý zobrazuje ukážku vplyvu útoku odopretia služby na množstvo prenesených dát v čase.



Obr. 2.44: Ukážka vplyvu DoS útoku na množstvo prenesených dát.

2.9.5 Replay útok na systém služby distribúcie dát - simulácia

Návrh tohoto bezpečnostného incidentu sa nachádza v 2.8.5. Pre replay útok je využitých niekoľko softvérov. Postupne sa jedná o softvér Ettercap pomocou, ktorého je nasadený útok muža uprostred s využitím metódy ARP spoofing, softvér tcpdump je využitý pre zachytenie komunikácie a softvér tcpreplay nasadený v editovacej verzii tcpreplay-edit, pre realizáciu útoku s opakovaním dát.
Návrh komunikačného scenára, v rámci ktorého komunikuje jeden objekt typu *Publisher* s jedným objektom typu *Subscriber* sa nachádza na obrázku 2.33. Návrh útoku s opakovaním dát je zobrazený na obrázku 2.34. Proces objektu typu *Publisher* je spustený na virtuálnom stroji 1 a na jednodoskovom počítači Raspberry Pi 3 je spustený proces objektu typu *Subscriber*. Útočník v sieti pracuje na virtuálnom stroji 2. Na obrázku 2.45 sa nachádza komunikačná topológia s priradenými IP adresami jednotlivých sieťových rozhraní použitých zariadení.



Obr. 2.45: Replay útok - komunikačná topológia.

Výpis 2.26 zobrazuje spustenie softvéru tcpreplay-edit na virtuálnom stroji 2 po zachytení komunikácie. Následne podľa zistení zo zachytenej komunikácie medzi dvojicou participujúcich entít je v rámci tohoto útoku nastavené čo najrýchlejšie odoslanie paketov, zmena zdrojového portu z 8446 na 50000 a zmena zdrojovej IP adresy.

Výpis 2.26: Spustenie softvéru tcpreplay-edit pre replay útok.

```
student@ubuntu:~$ sudo tcpreplay-edit -i ens33 -t --portmap=8446:50000
--srcipmap=192.168.0.120:192.168.0.1 replayutok.pcap
```

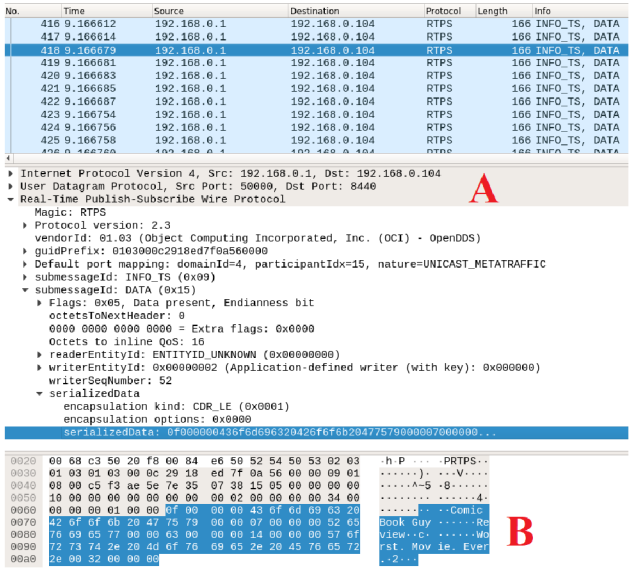
Výpis z programu iftop, ktorý je spustený na jednodoskovom počítači Raspberry Pi 3 sa nachádza na výpise 2.27 a je z neho možné vidieť, že objekt typu *Subscriber* prijíma dáta z dvoch zdrojových IP adries, pričom IP adresa 192.168.0.1 je zmanipulovaná útočníkom, ktorý pomocou útoku s opakovaním dát najskôr zachytené dáta pozmení a následne publikuje smerom k objektu typu *Subscriber*. Zároveň

je možné vidieť, že zatiaľ čo pri komunikácii s autorizovaným objektom typu *Publisher*, odosiela aj objekt typu *Subscriber* menšie množstvo dát, ktoré predstavujú správy odosielané pri vzájomnom objavovaní sa a potvrdzovaní prijatých dát, tak komunikácia zo strany objektu *Subscriber* smerom k útočiacej entite je nulová.

Výpis 2.27: Výstup z programu iftop - Replay útok.

```
Listening on eth0
# Host name (port/service if enabled)        last 2s   last 10s
--------------------------------------------------------------------
1 192.168.0.104                          =>     384b      124b
  192.168.0.120                          <=     595Kb     127Kb
2 192.168.0.104                          =>       0b        0b
  192.168.0.1                            <=     375Kb     116Kb
--------------------------------------------------------------------
Total send rate:                                 384b      124b
Total receive rate:                              970Kb     243Kb
Total send and receive rate:                     970Kb     243Kb
====================================================================
```

V časti A na obrázku 2.46 je možné vidieť, že smerom k objektu typu *Subscriber* sú odosielané dáta s pozmenenou zdrojovou IP adresou aj zdrojovým portom. V časti B obrázok 2.46 zobrazuje, že správa je identická s tou, ktorá je zasielaná od pôvodného objektu typu *Publisher*. Objekt typu *Subscriber* tak prijíma dáta v neporovnateľne väčšom množstve ako očakáva a tieto dáta sú zároveň v určitej miere zmanipulované.



Obr. 2.46: Bezpečnostný incident - replay útok.

2.10 Zhrnutie - OpenDDS

Táto časť v úvode predstavuje hardware využitý pre praktickú realizáciu a následnú kompiláciu implementácie OpenDDS na použitých zariadeniach. Ďalšie kapitoly sa postupne venujú návrhu a simulácii základných komunikačných scenárov, ktoré predchádzajú návrhu a simulácii komplexných komunikačných scenárov. Celkovo je navrhnutých osem komunikačných scenárov. Nasledujúca časť praktickej realizácie je zameraná na overenie bezpečnosti implementácie služby distribúcie dát OpenDDS. Ďalšie dve kapitoly sa zameriavajú na návrh bezpečnostných incidentov a ich následnú simuláciu. Celkovo je v práci navrhnutých päť bezpečnostných incidentov, z toho tri bezpečnostné incidenty sú zamerané na zneužitie a modifikáciu QoS politík a zvyšné dva sú zamerané na znefunkčnenie alebo zásadné ovplyvnenie komunikácie útočníkom. Tabuľka 2.1 predstavuje súhrn simulovaných komunikačných scenárov, pričom v rámci každého komunikačného scenára je definované na akom zariadení je spustený proces objektu typu *Publisher* a na akom zariadení je spustený objekt typu *Subscriber*. V prípade väčšieho množstva spustených procesov daného objektu v rámci jedného zariadenia sa v tabuľke nachádza pri názve zariadenia v zátvorke aj číslo, ktoré definuje počet spustených procesov daného objektu.

Kom. scenár	***Publisher***	***Subscriber***	**Security časť**
Kom. scenár *I*	Virtuálny stroj 1 Virtuálny stroj 1	Raspberry Pi 3 Virtuálny stroj 2	Nie Áno
Kom. scenár *II*	Virtuálny stroj 1	Virtuálny stroj 2 (2x)	Nie
Kom. scenár *III*	Virtuálny stroj 1 (2x)	Virtuálny stroj 2	Nie
Kom. scenár *IV*	Virtuálny stroj 1 (2x)	Virtuálny stroj 2 (2x)	Nie
Kom. scenár *V*	Raspberry Pi 3	Virtuálny stroj 1 (4x) Virtuálny stroj 2 (4x)	Nie
Kom. scenár *VI*	Raspberry Pi 3 Virtuálny stroj 1	Virtuálny stroj 1 (3x) Virtuálny stroj 2 (4x)	Nie
Kom. scenár *VII*	Raspberry Pi 3 Virtuálny stroj 1 Virtuálny stroj 2	Virtuálny stroj 1 (2x) Virtuálny stroj 2 (2x)	Nie
Kom. scenár *VIII*	Virtuálny stroj 1 (3x)	Virtuálny stroj 2	Nie

Tab. 2.1: Zhrnutie simulovaných komunikačných scenárov.

Zhrnutie simulovaných bezpečnostných incidentov sa nachádza v tabuľke 2.2, v ktorej sa nachádza typ bezpečnostného incidentu, softvér využitý pri simulácii daného bezpečnostného incidentu a výsledok, ktorý daný bezpečnostný incident spôsobí.

Bezp. incident	**Využitý softvér**	**Výsledok bezp. incidentu**
Zneužitie Entity_Factory	OpenDDS	publikovanie dát neautorizovaným objektom typu *Publisher*
Modifikácia Partition	OpenDDS Ettercap	publikovanie dát neautorizovaným objektom typu *Publisher*
Zneužitie Lifespan	OpenDDS	znemožnené odoberanie pôvodne vyžiadaných dát objektom typu *Subscriber*
DoS útok	OpenDDS Ettercap	zahodenie posielaných paketov a následné prerušenie komunikácie
Replay útok	OpenDDS Ettercap TCP replay	príjem zmanipulovaných dát v množstve väčšom ako objekt typu *Subscriber* očakáva

Tab. 2.2: Zhrnutie simulovaných bezpečnostných incidentov.

3 Nástroj pre realizáciu útokov

V rámci tejto práce je navrhnutý nástroj, ktorý umožňuje realizáciu určitého typu útokov. Nástroj je navrhnutý v programovacom jazyku Python, pričom je využitá verzia Python 2.17.15 a je navrhnutý pre používanie na operačnom systéme Linux. Pred využitím tohoto nástroja je nevyhnutná inštalácia nasledujúceho softvéru:

- **Ettercap** - tento softvér je využitý pri **ARP spoofing** a **DoS útoku**,
- **TCP dump** - tento softvér je využitý pri **TCP dump**, avšak jeho využiteľnosť je aj v spolupráci s ďalším softvérom pri **ARP spoofing** a **Replay útoku**.
- **TCP replay** - tento softvér je využitý pri **Replay útok**.

Tento nástroj umožňuje užívateľovi vybrať si zo štyroch možností útokov:

- **ARP spoofing** - tento typ útoku umožňuje užívateľovi vykonať útok muža uprostred,
- **DoS útok** - tento typ útoku umožňuje užívateľovi vykonať útok odopretia služby,
- **TCP dump** - nástroj určený k zachytávaniu komunikácie,
- **Replay útok** - tento typ útoku umožňuje užívateľovi vykonať útok s opakovaním dát a s ich prípadnou modifikáciou.

Na výpise 3.1 sa nachádza ukážka výberu útoku v rámci navrhnutého nástroja pre realizáciu jednej z vyššie spomenutých možností útoku.

Výpis 3.1: Ukážka nástroja pre realizáciu útokov - menu.

```
1: ARP spoofing
2: DoS útok
3: TCP dump
4: Replay útok

Vyberte typ útoku:
```

ARP spoofing - v prípade, že si užívateľ vyberie túto možnosť, je vyzvaný na zadanie sieťového rozhrania, na ktorom chce počúvať a následne postupne na zadanie dvoch cieľových IP adries, medzi ktorými chce zachytávať komunikáciu. Následne dôjde k spusteniu softvéru Ettercap v textovej forme a k spusteniu útoku muža uprostred so zadanými parametrami.

DoS útok - pri výbere DoS útoku je nevyhnutné, aby užívateľ pred využitím tohoto útoku umiestnil súbor *DoS.ef* do zložky, kde sa nachádza softvér Ettercap. Tento súbor definuje, že komunikácia medzi dvojicou participujúich účastníkov bude zahodená a následne úplne prerušená. Po výbere tohoto útoku je užívateľ rovnako ako v prípade ARP spoofing vyzvaný na zadanie sieťového rozhrania, na ktorom chce počúvať a následne zadanie dvoch cieľových IP adries, medzi ktorými má dôjsť k útoku odopretia služby. Následne dôjde k spusteniu softvéru Ettercap v textovej forme a k vykonaniu DoS útoku so zadanými parametrami.

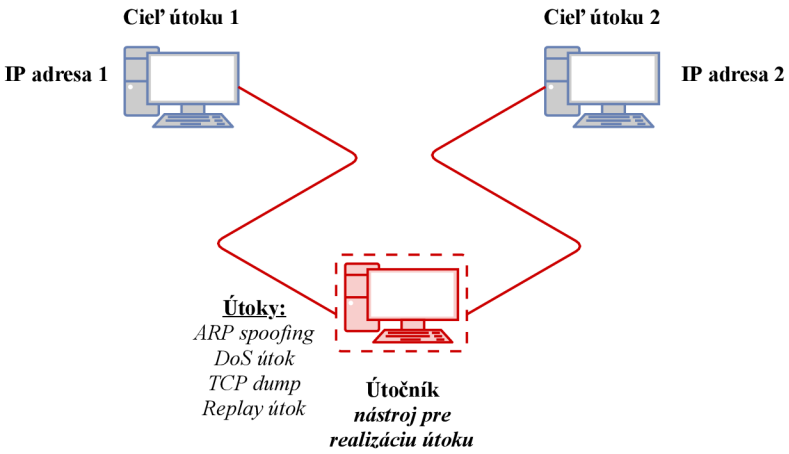
TCP dump - výberom tejto možnosti spustí užívateľ zachytávanie komunikácie. Jediným parametrom na zadanie, ktorého je užívateľ vyzvaný je názov súboru, pod ktorým má byť daná zachytená komunikácia uložená. Následne dôjde k spusteniu softvéru TCP dump.

Replay útok - v prípade výberu Replay útoku je nutné, aby užívateľ vlastnil súbor so zachytenými paketmi, ktorý chce v rámci tohoto útoku použiť. Jedným zo spôsobov je využitie softvéru TCP dump v rámci tohoto nástroja. V úvode je užívateľ vyzvaný pre zadanie sieťového rozhrania a názvu súboru so zachytenými paketmi. Následne má užívateľ na výber dve možnosti. V prípade ak nechce meniť zachytenú komunikáciu je táto komunikácia pomocou softvéru TCP replay najrýchlejším možným spôsobom zopakovaná smerom k cieľovej IP adrese. V prípade ak má užívateľ záujem manipulovať dáta má na výber pozmeniť zdrojový port a zdrojovú IP adresu. Postupne je vyzvaný k zadaniu zdrojového portu a následne k zadaniu portu, na ktorý má byť pôvodný port zmenený. Rovnaký postup platí aj pre zdrojovú IP adresu. Následne dôjde k spusteniu softvéru TCP replay a zmanipulované dáta sú zopakované smerom k cieľovej IP adrese.

Cieľom tohoto nástroja je umožniť vykonať útoky a využiť softvér, ktorý je do určitej miery využitý v rámci simulácie bezpečnostných incidentov v kapitole 2.9. Preto sú aj možnosti využitia tohoto nástroja obmedzené na zadanie určitých konkrétnych parametrov. Tabuľka 3.1 sumarizuje, aké parametre sú zadávané pre daný typ útoku. Hviezdička u replay útoku definuje, že je to voliteľná možnosť závislá na výbere užívateľa v rámci tohoto útoku.

ARP spoofing	**DoS útok**	**TCP dump**	**Replay útok**
rozhranie 1. IP adresa 2. IP adresa	rozhranie 1. IP adresa 2. IP adresa	názov súboru	rozhranie názov súboru * pôvodný port * nový port * pôvodná IP adresa * nová IP adresa

Tab. 3.1: Parametre zadávané pri použití nástroja.



Obr. 3.1: Topológia využitia nástroja pre realizáciu útokov.

Obrázok 3.1 zobrazuje topológiu využitia nástroja pre realizáciu útokov. Útočník, ktorý využíva tento nástroj si môže vybrať zo štyroch typov útokov, pričom v troch prípadoch sú kľúčovými parametrami zadanie IP adries potencionálnych cieľov útokov a zadanie rozhrania, na ktorom chce útočník počúvať.

Záver

Cieľom tejto diplomovej práce bolo analyzovať a porovnať dostupné implementácie služby distribúcie dát, preštudovať možnosti systémov publikovanie-odber, overiť bezpečnosť vybranej implementácie služby distribúcie dát, navrhnúť komplexné komunikačné scenáre a bezpečnostné incidenty. V neposlednom rade bolo cieľom diplomovej práce navrhnuté komunikačné scenáre a bezpečnostné incidenty simulovať. V teoretickej časti bola predstavená služba distribúcie dát a následujúce kapitoly sa zameriavali na jednotlivé súčasti služby distribúcie dát. Postupne bol rozobratý model Data-Centric Publish-Subscribe, vrátane definície entít, ktoré participujú v tomto modeli. Následne bola zadefinovaná kvalita služby v službe distribúcie dát. Ďalšia kapitola v rámci teoretickej časti predstavila interoperabilný protokol RTPS. Zabezpečeniu služby distribúcie dát sa táto práca venovala najobsiahlejšie v rámci teoretickej časti. Posledná kapitola teoretickej časti analyzovala tri vybraté DDS implementácie: OpenDDS, Vortex OpenSplice DDS a RTI Connext DDS. Na záver teoretickej časti boli zhrnuté bezpečnostné možnosti troch analyzovaných DDS implementácií.

V časti zameranej na praktické využitie služby distribúcie dát bol spočiatku predstavený hardware využitý pre túto diplomovú prácu a spôsob kompilácie využitej DDS implementácie OpenDDS na použitých zariadeniach, pričom v rámci tejto práce boli použité pre realizáciu simulácií dva virtuálne stroje a jednodoskový počítač Raspberry Pi 3. Následne boli navrhnuté štyri základné komunikačné scenáre a v neposlednom rade ich samotná simulácia. Komunikačný scenár *I* bol simulovaný v nezabezpečenej aj zabezpečenej variante. V rámci simulácie tohto scenára bola okrem samotnej simulácie porovnaná RTPS správa nezabezpečenej a zabezpečenej komunikácie. Zvyšné komunikačné scenáre boli simulované v nezabezpečenej variante. V rámci komunikačného scenára *II* bola zachytená komunikácia pomocou programu Wireshark, na ukážku súčasného odoberania dát dvomi objektami typu *Subscriber*. Komunikačný scenár *III* predstavoval komunikáciu medzi dvomi objektami typu *Publisher* a jedným objektom typu *Subscriber*, v rámci čoho bolo ukázané publikovanie dát obomi publikujúcimi entitami a samotné odoberanie dát objektom typu *Subscriber*. Komunikačný scenár *IV* obsahoval narozdiel od komunikačného scenára *III* naviac jeden objekt typu *Subscriber* a v rámci neho bolo preukázané súčasné odoberanie dát oboch objektov typu *Subscriber* od oboch publikujúcich entít. Ďalšia kapitola praktickej časti riešila návrh štyroch komplexných komunikačných scenárov. Komunikačný scenár *V* predstavoval komunikáciu jedného objektu typu *Publisher* a súčasné odoberanie dát publikovaných týmto objektom ôsmimi objektami typu *Subscriber*. V komunikačnom scenári *VI* spolu participovali dva objekty typu *Publisher* a sedem objektov typu *Subscriber*, pričom z dôvodu lepšieho rozoznania

odoberania dát objektami typu *Subscriber* od oboch objektov typu *Publisher* bola mierne upravená aplikácia Messenger, ktorá bola využívaná v rámci simulácií. Ďalšie mierne modifikácie aplikácie boli realizované aj v rámci komunikačného scenára *VII* a komunikačného scenára *VIII*. Komunikačný scenár *VII* riešil komunikáciu troch objektov typu *Publisher* a štyroch objektov typu *Subscriber* avšak všetky tri publikujúce entity publikovali dáta s rovnakým typom objektu typu *Topic*. V rámci komunikačného scenára *VIII* bola aplikácia Messenger upravená z dôvodu participácie troch objektov typu *Publisher*, pričom každý z objektov publikoval dáta s iným typom objektu typu *Topic*. Dáta od všetkých troch publikujúcich entít v komunikačnom scenári *VIII* odoberal jeden objekt typu *Subscriber*, ktorý mal priradené tri objekty typu *DataReader*. Siedma kapitola v rámci praktickej časti sa zameriavala na overenie bezpečnosti DDS implementácie OpenDDS, pričom pre ukážku boli využité tri typy komunikácie. Prvým typom bola nezabezpečená komunikácia, kedy bol útočník schopný odchytiť komunikáciu a získal tým prístup ku všetkým citlivým informáciám danej komunikácie. Druhým typom bola zabezpečená komunikácia bez zašifrovanej RTPS podsprávy. Vzhľadom k tomu, že dáta neboli zabezpečené sa situácia pre útočníka v zásade nezmenila a mal prístup k všetkým pre neho potrebným informáciám o danej komunikácii. Posledným typom bola zabezpečená komunikácia vrátane zabezpečenej RTPS podsprávy. V tomto prípade bolo preukázané, že aj vzhľadom k implementácii beta verzie OMG DDS Security v OpenDDS je možná v rámci možností bezpečná komunikácia. Posledné dve kapitoly praktickej časti sa venovali návrhu bezpečnostných incidentov a ich následnej simulácii. Celkovo bolo navrhnutých a simulovaných päť bezpečnostných incidentov. Bezpečnostný incident zneužitia politiky kvality služby ENTITY_FACTORY mal za následok publikovanie dát neautorizovaným objektom typu *Publisher* a zároveň zakázanie publikovania dát pre autorizovaný objekt typu *Publisher*. V rámci bezpečnostného incidentu modifikácie QoS politiky PARTITION bolo výsledkom publikovanie dát neautorizovaným objektom typu *Publisher*, ktorý bol schopný po zachytení a analýze predmetnej komunikácie nastaviť svoju QoS politiku PARTITION, tak aby mohol publikovať dáta smerom k objektu typu *Subscriber*. Zneužitie QoS politiky LIFESPAN znemožnilo odoberanie dát objektu typu *Subscriber*. Útok odopretia služby spôsobil zahodenie paketov a následné prerušenie komunikácie medzi dvojicou komunikujúcich entít. Piatym bezpečnostným incidentom bol útok s opakovaním dát, kedy bol útočník schopný odchytiť komunikáciu modifikovať ju a následne ju opätovne odoslať smerom k odoberajúcej entite.

Záverečná časť práce predstavila nástroj navrhnutý v programovacom jazyku Python pre realizáciu určitého typu útokov. Tento nástroj konkrétne umožňuje realizovať štyri typu útokov. Konkrétne sa jedná o útok ARP spoofing, DoS útok, zachytenie komunikácie pomocou TCP dump a replay útok.

Literatúra

[1] PARDO-CASTELLOTE, G.: *OMG Data-Distribution Service: architectural overview* [online]. 2003, 28s [cit. 14. 10. 2019]. DOI: 10.1109/MILCOM.2003.1290110. Dostupné z URL: <https://www.researchgate.net/publication/4070720_OMG_Data-Distribution_Service_architectural_overview>

[2] DDS Foundation: *What is DDS?* [online]. [cit. 14. 10. 2019]. Dostupné z URL: <https://www.dds-foundation.org/what-is-dds-3/>.

[3] OpenDDS: *Introduction to OpenDDS* [online]. [cit. 14. 10. 2019]. Dostupné z URL: <https://opendds.org/about/articles/Article-Intro.html>.

[4] LEE, I., LEUNG, J. Y-T., SON, S. H.: *Handbook of Real-Time and Embedded Systems* 1st. Chapman Hall/CRC, [online]. 2007, 800s. [cit. 14. 10. 2019]. ISBN 978-1584886785. Dostupné z URL: <https://books.google.cz/books?id=Bz7DTDZQaGIC&printsec=frontcover&hl=sk&source=gbs_atb#v=onepage&q&f=false>.

[5] Object Management Group: OMG: *Data Distribution Service (DDS)* [online]. 2015, 180s. [cit. 14. 10. 2019]. Dostupné z URL: <https://www.omg.org/spec/DDS/1.4/PDF>.

[6] RTI: *What is DCPS?* [online]. 2015, [cit. 16. 10. 2019]. Dostupné z URL: <https://community.rti.com/static/documentation/connext-dds/5.2.0/doc/manuals/connext_dds/html_files/RTI_ConnextDDS_CoreLibraries_UsersManual/index.htm#UsersManual/What_is_DCPS_.htm>.

[7] CALVO, I., ETXEBERRIA-AGIRIANO, I., PÉREZ GONZÁLEZ, F., GARCIA DE ALBENIZ, O.: *Designing High Performance Factory Automation Applications on Top of DDS* [online]. International Journal of Advanced Robotic Systems. 2013, 1-12 [cit. 16. 10. 2019]. DOI: 10.5772/56341. Dostupné z URL: <https://www.researchgate.net/publication/237052462_Designing_High_Performance_Factory_Automation_Applications_on_Top_of_DDS>

[8] OpenDDS: *OpenDDS Developer's Guide* [online]. [cit. 16. 10. 2019]. Dostupné z URL: <http://download.objectcomputing.com/OpenDDS/OpenDDS-latest.pdf>.

[9] PARDO-CASTELLOTE, G.: *OMG Data-Distribution Service: architectural overview* [online]. 2003, 200-206 [cit. 16. 10. 2019]. DOI: 10.1109/ICDCSW.2003.1203555. ISBN: 0-7695-1921-0. Dostupné z URL: <https://ieeexplore.ieee.org/document/1203555>

[10] RTI: *DDS Samples, Instances, and Keys* [online]. 2016, [cit. 16. 10. 2019]. Dostupné z URL: <https://community.rti.com/static/documentation/connext-dds/5.2.3/doc/manuals/connext_dds/html_files/RTI_ConnextDDS_CoreLibraries_UsersManual/index.htm#UsersManual/DDS_Samples__Instances__and_Keys.htm>.

[11] RTI: *DDS Domains and DomainParticipants* [online]. 2015, [cit. 16. 10. 2019]. Dostupné z URL: <https://community.rti.com/static/documentation/connext-dds/5.2.0/doc/manuals/connext_dds/html_files/RTI_ConnextDDS_CoreLibraries_UsersManual/index.htm#UsersManual/DDS_Domains_and_DomainParticipants.htm>.

[12] DDS Foundation: *Technical Benefits* [online]. [cit. 16. 10. 2019]. Dostupné z URL: <https://www.dds-foundation.org/key-technical-benefits/>.

[13] RTI: *QoS* [online]. 2015, [cit. 21. 10. 2019]. Dostupné z URL: <https://community.rti.com/glossary/qos>.

[14] CORSARO, A., QUERZONI, L., SCIPIONI, S., TUCCI-PIERGIOVANNI, S., VIRGILLITO, A.: *Quality of Service in Publish/Subscribe Middleware* [online]. 2006, 1-19 [cit. 21. 10. 2019]. Dostupné z URL: <https://www.researchgate.net/publication/237100885_Quality_of_Service_in_PublishSubscribe_Middleware>

[15] ADLINK TECHNOLOGY: *The Data Distribution Service Tutorial - Quality of Service* [online]. [cit. 21. 10. 2019]. Dostupné z URL: <http://download.prismtech.com/docs/Vortex/html/ospl/DDSTutorial/qos.html>.

[16] ADLINK TECHNOLOGY: *Welcome to Vortex Café's User Guide - Quality of Service - Support of DDS Security* [online]. [cit. 28. 10. 2019]. Dostupné z URL: <http://download.prismtech.com/docs/Vortex/html/cafe/user-guide/09-Security.html>.

[17] OpenDDS: *Revolutionizing Data Distribution with an Open and Secure DDS* [online]. 2018, [cit. 28. 10. 2019]. Dostupné z URL: <https://objectcomputing.com/files/1615/3504/7743/DDS_Security_Webinar.pdf>.

[18] Object Management Group: OMG: *DDS Security* [online]. 2018, 285s. [cit. 28. 10. 2019]. Dostupné z URL: <https://www.omg.org/spec/DDS-SECURITY/1.1/PDF>.

[19] eProsima: *RTPS Introduction* [online]. [cit. 31. 10. 2019]. Dostupné z URL: <https://www.eprosima.com/index.php/resources-all/rtps>.

[20] Object Management Group: OMG: *The Real-time Publish-SubscribeProtocol (RTPS) DDS Interoperability Wire Protocol Specification* [online]. 2018, 185s. [cit. 31. 10. 2019]. Dostupné z URL: <https://www.omg.org/spec/DDSI-RTPS/2.3/Beta1/PDF>.

[21] OpenDDS: *About* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <https://opendds.org/about/>.

[22] OpenDDS: *Using DDS Security in OpenDDS* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <http://download.ociweb.com/OpenDDS/Using_DDS_Security_in_OpenDDS_3_13.pdf>.

[23] ADLINK Technology: *Vortex OpenSplice and the DDS Standard* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <https://www.adlinktech.com/en/standards-vortex-data-distribution-service-opensplice>.

[24] ADLINK Technology: *The Vortex OpenSplice Deployment Guide* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <http://download.prismtech.com/docs/Vortex/html/ospl/DeploymentGuide/index.html>.

[25] ADLINK Technology: *Tools* [online]. [cit. 23. 11. 2019]. Dostupné z URL: <https://www.adlinktech.com/en/vortex-opensplice-tools>.

[26] RTI: *RTI Connext DDS Professional* [online]. [cit. 26. 11. 2019]. Dostupné z URL: <https://info.rti.com/hubfs/Datasheets/RTI_Datasheet_10017_Connext-DDS-Professional_V29_Web_0718.pdf>.

[27] RTI: *RTI Connext DDS Secure* [online]. [cit. 26. 11. 2019]. Dostupné z URL: <https://info.rti.com/hubfs/Datasheets/RTI_Datasheet_10018_Connext-DDS-Secure_V7_Web_0718.pdf>.

[28] OpenDDS: *Linux/Solaris/MacOSX* [online]. [cit. 8. 12. 2019]. Dostupné z URL: <https://opendds.org/quickstart/GettingStartedLinux.html>.

[29] OpenDDS: *Raspberry Pi* [online]. [cit. 8. 12. 2019]. Dostupné z URL: <https://opendds.org/quickstart/GettingStartedPi.html>.

[30] ADLINK Technology: *Secure Networking Configuration Release 6.x* [online]. [cit. 10. 12. 2019]. Dostupné z URL: <http://download.prismtech.com/docs/Vortex/pdfs/OpenSplice_SecureNetworkingGuide.pdf>.

[31] MICHAUD, M. J.: *Malicious Use of OMG Data Distribution Service (DDS) In Real Time Mission Critical Systems* [online]. M.A.Sc, Royal Military College of Canada, 2017, [cit. 12. 12. 2019]. Dostupné z URL: <https://espace.rmc.ca/bitstream/11264/1241/1/Lt(N)_Michaud-MASCThesis-MalicousUseDDSinRTMCSystems.pdf>.

Zoznam symbolov, veličín a skratiek

AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
CORBA	Common Object Request Broker Architecture
DCPS	Data-Centric Publish-Subscribe
DDS	Data Distribution Service
DH	Diffie-Hellman
DLRL	Data Local Reconstruction Layer
DoS	Denial of Service
ECDH	Elliptic-Curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GCM	Galois/Counter Mode
GDS	Global Data Space
GMAC	Galois Message Authentication Code
HMAC	Hash-based Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
IDL	Interface Description Language
IOT	Internet of Things
JNI	Java Native Interface
OCI	Object Computing, Inc.
OMG	Object Managment Group
PKI	Public Key Infrastructure
QoS	Quality of Service
RAM	Random Access Memory
RSA	Rivest–Shamir–Adleman
RTI	Real-Time Innovations
RTPS	Real-Time Publish-Subscribe
SPDP	Simple Participant Discovery Protocol
SPI	Service Plugin Interface
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
XML	eXtensible Markup Language

Zoznam príloh

A Prílohy

A.1 Simulácie základných komunikačných scenárov

Výpis A.1: Komunikačný scenár *III* - odoberanie dát.

```
student@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/LargeSample$ ./subscriber -DCPSConfi
rtps-nezabezpecena-66.ini
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=1, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=2, not alive=0
Message: process_id =  5552 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 1 sample_id = 0
Message: process_id =  5552 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 2 sample_id = 0
Message: process_id =  5552 writer_id = 1 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 1 sample_id = 1
Message: process_id =  5552 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 2 sample_id = 1
Message: process_id =  5552 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 1 sample_id = 2
Message: process_id =  5552 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  5552 writer_id = 2 sample_id = 2
                                        .
                                        .
                                        .
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=3, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=4, not alive=0
Message: process_id =  5560 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 1 sample_id = 0
Message: process_id =  5560 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 2 sample_id = 0
Message: process_id =  5560 writer_id = 1 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 1 sample_id = 1
Message: process_id =  5560 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 2 sample_id = 1
Message: process_id =  5560 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 1 sample_id = 2
Message: process_id =  5560 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  5560 writer_id = 2 sample_id = 2
```

Výpis A.2: Komunikačný scenár *IV* - odoberanie dát - Subscriber 1.

```
student@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/LargeSample$ ./subscriber -DCPSConfi
rtps-nezabezpecena-66.ini
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=1, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=2, not alive=0
Message: process_id =  6085 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 0
Message: process_id =  6085 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 0
Message: process_id =  6085 writer_id = 1 sample_id = 1
```

```
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 1
Message: process_id =  6085 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 1
Message: process_id =  6085 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 2
Message: process_id =  6085 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 2
                                        .
                                        .
                                        .
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=3, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=4, not alive=0
Message: process_id =  6093 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 0
Message: process_id =  6093 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 0
Message: process_id =  6093 writer_id = 1 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 1
Message: process_id =  6093 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 1
Message: process_id =  6093 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 2
Message: process_id =  6093 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 2
```

Výpis A.3: Komunikačný scenár *IV* - odoberanie dát - Subscriber 2.

```
student@ubuntu:~/OpenDDS-3.13.1-sec/tests/DCPS/LargeSample$ ./subscriber -DCPSConfi
rtps-nezabezpecena-69.ini
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=1, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=2, not alive=0
Message: process_id =  6093 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 0
Message: process_id =  6093 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 0
Message: process_id =  6093 writer_id = 1 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 1
Message: process_id =  6093 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 1
Message: process_id =  6093 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 1 sample_id = 2
Message: process_id =  6093 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6093 writer_id = 2 sample_id = 2
                                        .
                                        .
                                        .
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:179: INFO: on_subscription_matched()
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=3, not alive=0
DataReaderListener.cpp:172: INFO: on_liveliness_changed() alive=4, not alive=0
Message: process_id =  6085 writer_id = 1 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 0
Message: process_id =  6085 writer_id = 2 sample_id = 0
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 0
```

```
Message: process_id =  6085 writer_id = 1 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 1
Message: process_id =  6085 writer_id = 2 sample_id = 1
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 1
Message: process_id =  6085 writer_id = 1 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 1 sample_id = 2
Message: process_id =  6085 writer_id = 2 sample_id = 2
DataReaderListener.cpp:80: Message: process_id =  6085 writer_id = 2 sample_id = 2
```

B Obsah priloženého CD

- Diplomová práca: obsahuje elektronickú verziu práce vo formáte pdf,
- VS 1: obsahuje aplikácie využité pri simuláciách, ktoré boli spustené na virtuálnom stroji 1,
- VS 2: obsahuje aplikácie využité pri simuláciách, ktoré boli spustené na virtuálnom stroji 2,
- RPI: obsahuje konfiguračné súbory využité pri simuláciách pre dané komunikačné scenáre, ktoré boli využité na Raspberry Pi 3,
- RSA_certifikáty: obsahuje vygenerované RSA certifikáty a podpísané dokumenty governance a permissions,
- Nástroj: obsahuje nástroj pre realizáciu útokov a skript pre realizáciu DoS útoku,
- Excel: obsahuje graf DoS útoku v programe Excel,
- README súbor.