VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS


# BEZPEČNOSTNÍ MONITOROVÁNÍ KOMUNIKACE DOMÁCÍCH IOT SÍTÍ

SECURITY MONITORING OF HOME IOT NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS


AUTOR PRÁCE PATRIK KRAJČ
AUTHOR

VEDOUCÍ PRÁCE Ing. PETR MATOUŠEK, Ph.D., M.A.
SUPERVISOR



BRNO 2018

Ústav informačních systémů (UIFS) Akademický rok 2018/2019

# Zadání bakalářské práce

21957

Student: **Krajč Patrik**
Program: Informační technologie
Název: **Bezpečnostní monitorování komunikace domácích IoT sítí**
**Security Monitoring of Home IoT Networks**
Kategorie: Počítačové sítě
Zadání:

1. Seznamte se s architekturou domácích IoT sítí. Popište základní typy komunikace a protokoly, které se používají.
2. Na základě dostupných monitorovacích dat (PCAP soubory, NetFlow záznamy) proveďte analýzu přenášených informací z pohledu bezpečnostního monitorování.
3. Navrhněte a vytvořte systém pro bezpečnostní monitorování domácí IoT sítí. Navrhněte a implementujte vhodné úložiště pro tato data.
4. Popište základní typy útoků na domácí IoT sítě. Podle doporučení vedoucího simulujte některé útoky za pomoci dostupných nástrojů.
5. Vytvořte uživatelské grafické rozhraní pro zobrazení chování sítě a detekci útoků na základě nasbíraných monitorovacích dat.
6. Popište výsledky své práce a zhodnoťte její praktické využití.

Literatura:

- David Hanes, Gonzalo Salqueiro, Patrick Grossetete, Rob Barton, andJereme Henry. IoT Fundamentals. Networking Technologies, Protocoland Use Cases for the Internet of Things. Cisco Press, 2017.
- RYŠAVÝ Ondřej. Analysis of Constrained Application Protocol. FIT-TR-2017-15, Brno: Fakulta informačních technologií VUT v Brně, 2017.
- RYŠAVÝ Ondřej. Security Monitoring of LwM2M communication. FIT-TR-2017-16, Brno: Fakulta informačních technologií VUT v Brně, 2017
- Is Anybody Home? Inferring Activity From Smart Home Network TrafficB Copos, K Levitt, M Bishop, J Rowe,Security and Privacy Workshops (SPW), 2016 IEEE, 245-251

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Matoušek Petr, Ing., Ph.D., M.A.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 25. října 2018

## Abstrakt

Cieľom práce je vytvoriť systém na bezpečnostné monitorovanie IoT sietí a užívateľské grafické rozhranie pre zobrazenie chovania IoT siete, ktoré zahŕňa detekciu anomálií v sieti. Aplikácia umožňuje zobraziť informácie o prevádzke v rôznych časových intervaloch, definovaním konkrétneho dňa a hodiny. Informácie o prevádzke sú získané z IPFIX záznamov uložených v MySQL databáze. Vytvorené IPFIX záznamy obsahujú komunikáciu medzi zariadeniami CoAP, ktoré podstúpili niekoľko útokov.



## Abstract

The main purpose of my bachelor thesis is create system for security monitoring of home IoT networks and user interface for network anomalies detection. Final application shows information about traffic in specified time intervals for specific day and hour. Traffic information are obtained from IPFIX records stored in MySQL database. Used test kits of IPFIX records were created from communication of locally connected devices that used CoAP protocol.



## Kľúčové slová

Internet of Things, Constrained Application Protocol, bezpečnostné monitorovanie, útoky na IoT siete

## Keywords

Internet of Things, Constrained Application Protocol, security monitoring, IoT attacks


## Citácia

KRAJČ, Patrik. *Bezpečnostní monitorování komunikace domácích IoT sítí.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Matoušek, Ph.D., M.A.

# Bezpečnostní monitorování komunikace domácích IoT sítí

## Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D., M.A.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Patrik Krajč
13. mája 2019

## Poďakovanie

Touto cestou by som sa chcel poďakovať pánovi Ing. Petrovi Matouškovi, Ph.D., M.A., ktorý bol vždy ochotný poskytnúť odbornú pomoc pri vypracovaní práce.

# Obsah

# Kapitola 1

# Úvod

Bakalárska práca je zameraná na bezpečnostné monitorovanie komunikácie v sieťach IoT. Takáto sieť obsahuje množstvo menších zariadení, ako sú napríklad senzory alebo rôzne spotrebiče, ktoré komunikujú s ostatnými zariadeniami v sieti bez nutnej interakcie s človekom.

IoT siete sú pomerne rozšírené a v následujúcich rokoch je očakávaný veľký nárast týchto sietí vo firmách, ale aj v domácnostiach a iných organizáciach. Nie každý používateľ je oboznámený stým, či je jeho sieť zabezpečená a tu sa otvárajú dvere útočníkom, pretože zariadenia môžu prenášať pomerne citlivé informácie, ktoré by mohli byť zneužité.

Cieľom bakalárskej práce je navrhnúť a vytvoriť monitorovací systém, ktorý by bol schopný vytvoriť štatistické údaje o prevádzke v sieti. Získaný prehľad o udalostiach v sieti nám umožní vytvoriť prehľad o chovaní zapojených zariadení. K dosiahnutiu tohto cieľa je nutné oboznámiť sa s IoT aplikačnými protokolmi ich vlastnosťami. Medzi najbežnejšie patrí protokol MQTT alebo CoAP.

Práca je zameraná na monitorovanie zariadení, ktoré používajú na komunikáciu aplikačný protokol CoAP (Constrained Application Protocol) [17], ktorému je venovaná kapitola 3, v ktorej sú spísané informácie o formáte správy, metódy a voľby, ktoré poskytuje. Ďalej obsahuje informácie o základných spôsoboch prenosu dát medzi zariadeniami, ale aj o rozšírení blokového prenosu dát. Protokol CoAP poskytuje niekoľko služieb, ktoré umožnujú klientom ale aj serverom jednoduchým spôsobom objaviť ostatné zariadenia CoAP a získať dostupné infromácie, ktoré poskytujú. V poslednej časti kapitoly je popísané bezpečnostné módy, v ktorých sa môžu zariadenia nachádzať.

Kapitola 6 je venovaná vytvoreniu testovacieho prostredia. Obsahuje použité prostriedky na vytvorenie a odchytávanie komunikácie. Sieť obsahuje tri CoAP zariadenia. Zariadenia podstúpili niekoľko útokov z kapitoly 5, ktorých emulácia je popísaná v kapitole 6.4.

Ďalším bodom práce, je oboznámiť sa s dostupnými monitorovacími prostriedkami 4, s ich vlastnosťami a požiadavkami na monitorované zariadenia, kvôli obmedzeniam ktoré vytvára hardware v IoT zariadeniach 2.1.

Posledná kapitola 7 je venovaná vizualizácií získaných dát. V tejto kapitole sú spomenúte spôsoby, akým sa vyhodnocujú rôzne anomálie a sú zobrazené vytvorené testovacie sady z kapitoly 6.4.

# Kapitola 2

# Internet of things

Myšlienkou IoT je pripojiť ešte nepripojené zariadenia k internetu, a tak získať množstvo nových služieb, ktoré nám môžu uľahčiť a spríjemniť život. Takéto zariadenia dokážu vnímať vec reálneho sveta, a premeniť získané informácie na digitálny signál, ktorý spracuje aplikácia, ktorá nám následne môže poskytnúť informácie o danej veci. Okrem vnímania vecí realneho sveta, je možné ich aj kontrolovať naprieč sieťou, napríklad pri regulácií kúrenia a podobne.

## 2.1 IoT zariadenia

Zapojené zariadenia do IoT siete môžeme rozdeľiť do niekoľkých kategórií. Prvou kategóriou podľa ktorej ich môžeme rozdeliť je spôsob napájania. Sú to zariadenia napájane z elektrickej siete alebo batérie. Výhodou zariadení napájaných z batérie je ich jednoduchá manipulácia bez straty pripojenia. Nevýhodou sú obmedzené zdroje na hardware, ktoré vytvára vplyv na vzdialenosť pripojenia k sieti.

Z hardwarovej stránky sú zariadenia rozdelené do troch tried, ktoré sú definované v RFC 7228 [2]. Toto rozdelnie je na základne pamäte RAM a ROM, ktoré súvisí s implementáciou IP stacku.

- Zariadenia patriace do *triedy 0*, obsahujú menej ako 10 KB pamäte RAM a menej ako 100 KB pamäte ROM. Tieto zariadenia sú zvyčajne napájané z batérie. Nemajú dostatočné prostriedky potrebné na implementovanie IP stacku a súvisiacich bezpečnostých mechanizmov [7].

- Ďalšou triedou je *trieda 1*. Definuje zariadenia s oniečo väčšou pamätou RAM a ROM, menej ako 10 KB pre RAM a 100 KB pre flash pamäť. Tieto parametre stále nedosahúju možnosť implementovať kompletný IP stack. A však môžu implementovať optimalizovaný IP stack, špecialne navrhnutý pre tieto zariadenia, napríklad pre Constrained Application Protocol (CoAP). Umožní to zariadeniam zapojiť sa do zmysluplnej konverzácie so sieťou bez pomoci brány a poskytuje podporu bezpečnostných funkcií [7].

- Zariadenia z *triedy 2* sú charakterizované plnou implementáciou IP stacku v stavaných zariadeniach. Obsahujú viacej, ako 50 KB pamäte RAM a 250 KB pamäte ROM. S týmito parametrami môžu byť plne integrované do IP siete [7].

## 2.2 Spôsoby komunikácie v domácich IoT sieťach

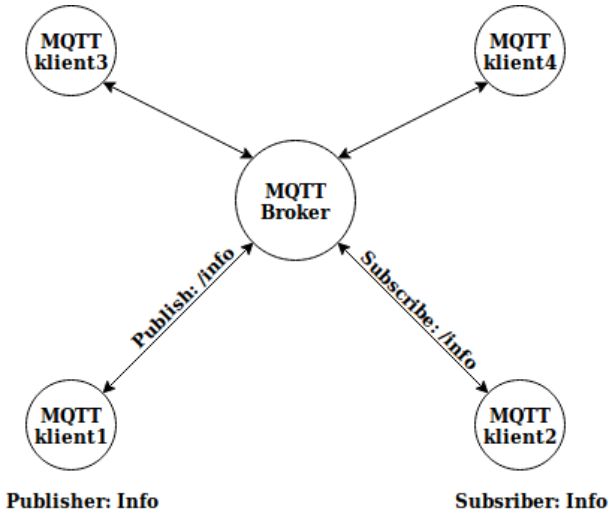
Komunikácia medzi IoT zariadeniami môže mať niekoľko spôsobov, ktorým je venovaná táto kapitola. Spôsob prenosu dát medzi zariadeniami je ovplyvnený najmä od zariadení zapojených v sieti, ktoré možu patriť do jednej z troch tried z kapitoly 2.1, ale aj od ich účelu.

### 2.2.1 Webové protokoly

V prípade, kedy sa nejedná o siete s obmedzením na prenosovú šírku pásma, je možné použiť aj zariadenia z triedy 2, miestami zariadenia z triedy 1. Zariadenia majú väčšinou implementovaný úplny IP stack, ktorý im umožnuje nadviazať spojenie naprieč sieťou, a využívajú na prenos dát webové protokoly HTTP/HTTPS. Prenášané dáta môžu byť napríklad vo formáte XML alebo JSON. Výhodou použitia webových protokolov na IoT zariadeniach, je ich pomerne jednoduchá implemntácia [7].

### 2.2.2 IoT aplikačné protokoly

Rýchlym vývojom IoT sieti, vzniklo aj niekoľko aplikačných protokolov. Každý z protokolov má svoje výhody a nevýhody. Medzi najbežnejšie protokoly patria protokol CoAP a MQTT. Obidva protokoly sú vyvíjané tak, aby mohli byť použiteľné na zariadenia s obmedzeným výkonom a aby vytvárali čo najmenší tok dát.



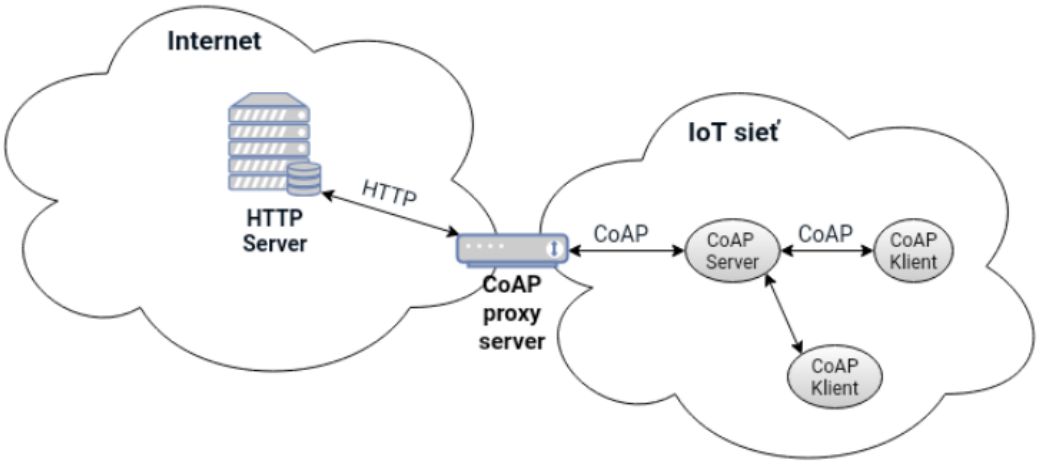

Obr. 2.1: Komunikácia protokolom MQTT

MQTT (Message Queuing Telemetry Transport)[1] je založený na komunikácií publish/subscribe. Komunikácia prebieha medzi klientami a sprostredkúvateľom (MQTT broker). Klienti môžu publikovať alebo sa prihlásiť na odber určitých tém, ktoré sú predmetom správy. Sprostredkovateľ podľa predmetu správy rozhodne o tom, pre ktoré zariadenia bola určená správa. Komunikácia protokolom MQTT je znázornená na obrázku 2.1, obsahuje štyri zariadenia s názvom MQTT klient a jedno zariadenie MQTT broker, ktorý predstavuje sprostredkovateľa. Zariadenie s názvom MQTT klient1 publikuje tému s názvom info,

[1] viď http://mqtt.org/documentation

o ktorý má záujem MQTT klient2. Sprostredkóvateľ na základe získaných informácií o tom ktorý klient odoberá témy, bude smerovať danú tému len na tie konkrétne zariadenia, v tomto prípade sa jedná o MQTT klienta2.

Protokol CoAP používa iný spôsob komunikácie ako MQTT. Jedná sa komunikáciu typu požiadavka/odpoveď (request/respond). Tento protokol je veľmi podobný protokolu HTTP a podporuje multicástove skupiny a ďalšie mechanizmy, ktoré umožňujú získať prehľad o sieti bez ľudského faktora. Podobizeň medzi protokolmi je pri použití metód, ktoré sú napríklad PUT a GET, ale majú aj podobné navrátové kódy. To, že protokol CoAP je podobný protokolu HTTP, má svôj význam, a to aby bol CoAP kompatibilný s protokolom HTTP. Zariadenie označujúce ako CoAP proxy server vykonáva preklad požiadaviek CoAP na HTTP, čo umožnuje zariadeniam s neúplnym IP stackom komunikovať aj s webovým serverom. Na obrázku 2.2 je zobrazená komunikácia medzi zariadeniami CoAP v IoT sieti, a prepojením IoT sieti s internetom.




Obr. 2.2: Komunikácia protokolom CoAP

## 2.3 Zhrnutie

V domácich IoT sieťach sa využívajú hlavne zariadenia, s ktorými je jednoduchá manipulácia a je možné ich flexibilné umiestnenie v domácnosti. Informácie o monitorovaných predmetoch môžu byť uložené nie len v rámci lokálnej siete. Sady pre inteligentnú domácnosť poskytujú vzdialený prístup k zariadeniam cez cloud. Tento spôsob komunikácie je zobrazený na obrázku 2.2, kde zariadenia s názvom CoAP proxy server vytvára preklad medzi webovým protokolom HTTP a CoAP a ukladá informácie o zdrojoch na server mimo IoT sieť.

# Kapitola 3

# Constrained Application Protocol (CoAP)

CoAP [17] je protokol aplikačnej vrstvy určený na prenos dát medzi zariadeniami s obmedzenými vypočtovými zdrojmi. Väčšinou sa jedná o zariadenia pozostávajúce z mikrokontrolérov s malou kapacitou pamätí RAM a ROM. CoAP protokol bol vytvorený pre siete IoT (Internet of Things) za účelom zníženia záťaze siete a zníženia nárokov na hardware pre zariadenia IoT.

Protokol je navrhnutý tak, aby bol kompatibilný s protokolom HTTP, čo umožnuje zariadeniam zapojených v sieti s obmedzenými prostredkami nadviazať spojenie s HTTP serverom. CoAP umožňuje komunikovať aj multicástovými adresami, ktorá môže byť napríklad pri objavovaní ostatných CoAP zariadení. Existuje niekoľko rozšírení, ktoré umožňujú prenášať väčšie množstvo dát alebo publish/subscribe mechanizmus ako pri protokole MQTT.

## 3.1 Správy

CoAP správy sú zakódované v jednoduchom binárnom formáte. Správa obsahuje hlavičku o fixnej veľkosti štyri bajty. Za hlavičkou následuje token, voľby a obsah správy. Formát správy je znázornený v tabuľke 3.1.

Správa je zložená nasledovne:

- V–Verzia (Version) označuje číslo verzie protokolu CoAP.
- T–Typ (Type) označuje o aký typ správy sa jedná. CoAP používa štyri typy správ: Confirmable (0), Non-Confirmable (1), Acknowledgment (2) a Reset (3). Rozdiel medzi Confirmable a Non-Confirmable správou je v tom, že Confirmable správa, vyžaduje od zariadenia prijatie správy. Správa ktorá potvrdzuje prijatie, je označená typom Acknowledgment. Typ správy Reset označuje konkrétnu správu prijatú zariadením, ktorú nie je možné správne spracovať.
- Token slúži na spárovanie požiadavku s odpoveďou.
- TKL–Token length určuje dĺžku tokenu. Rozsah je 0-32 bitov.
- Code (kód) je rozdelený na dve časti v tvare *c.dd*. Prvá časť *c*, tri bity, je určená pre triedu, ktorá indikuje či sa jedná o žiadosť (0), úspešnú odpoveď (1), neúspešnú

odpoveď zo strany klienta (4) alebo neúspešnú odpoveď zo strany servera (5). Druhá časť, *dd*, ktorej dĺžka je päť bitov, slúži pre bližšie špecifikácie. V prípade, že sa jedná o žiadosť, špecifikuje použitú metódu. Metódy sú spomenuté v kapitole 3.3. V ostatných prípadoch sa jedná o odpoveď, kde popisuje stav transakcie.

- Message ID (MID) slúži na detekciu duplicitných správ a na spárovanie správ typu Acknowledgement/Reset k Confirmable/Non-Confirmable. Pre každú novú požiadavku je vygenerovaný nový MID, ktorý je unikátny v rámci prenosu.
- Options (voľby) položka obsahuje zoznam použitých volieb. Jednotlivé voľby sú definované číslom voľby, formátom a dĺžku voľby. Číslo voľby jednoznačne identifikuje o akú voľbu sa jedná. Formát určuje typ dát spojených s voľbou. Ako príklad je použitá voľba Uri-Host, ktorej číslo voľby je tri a formát dát je literárny reťazec (string) o maximálnej dĺžke 255 znakov, viď 3.2.
- Payload reprezentuje dáta (obsah správy), ktoré môžu byť v binárnom alebo textovom formáte.
- 0xFF určuje koniec volieb a začiatok obsahu správy (Payload).

<table>
<tr><td>V</td><td>T</td><td>TKL</td><td>Code</td><td>Message ID</td></tr>
<tr><td colspan="5">Token</td></tr>
<tr><td colspan="5">Options</td></tr>
<tr><td colspan="3">0xFF</td><td colspan="2">Payload</td></tr>
</table>

Tabuľka 3.1: Formát správy CoAP

## 3.2 Voľby

Požiadavky na zariadenia, ale aj odpovede môžu obsahovať zoznam volieb, ktoré bližšie špecifikujú prenášané informácie. Niektoré voľby sú priamo spojené s použitými metódami 3.3, ako napríklad pri metódach PUT a POST je nutné definovať content-formát prenášaných dát. Jednotlivé voľby sú zhrnuté v tabuľke 3.2.

| Názov | Čislo voľby | Formát | Dĺžka |
|---|---|---|---|
| If-Match | 1 | Opaque | 0-8 |
| Uri-Host | 3 | String | 1-255 |
| ETag | 4 | Opaque | 1-8 |
| If-None-Match | 5 | Empty | 0 |
| Observe | 6 | Uint | 0-3 |
| Uri-Port | 7 | Uint | 0-2 |
| Location-Path | 8 | String | 0-255 |
| Uri-Path | 11 | String | 0-255 |
| Content-Format | 12 | Uinit | 0-2 |
| Max-Age | 14 | Uinit | 0-4 |

| Názov | Čislo voľby | Formát | Dĺžka |
|---|---|---|---|
| Uri-Query | 15 | String | 0-255 |
| Accept | 17 | Uinit | 0-2 |
| Location-Query | 20 | String | 1-255 |
| Block2 | 23 | unit | 0-3 |
| Block1 | 27 | unit | 0-3 |
| Proxy-Uri | 35 | String | 1-1034 |
| Proxy-Scheme | 39 | String | 1-255 |
| Size1 | 60 | Uint | 0-4 |

Tabuľka 3.2: Voľby CoAP

### 3.2.1 Voľby Uri-Host, Uri-Port, Uri-Path, Uri-Query

Tieto voľby slúžia na určenie cieľového servera spolu s relatívnou cestou k zdroju. Voľby Uri-Host a Uri-Port majú preddefinovnaé hodnoty, pre Uri-Host je to IP adresa cieľového zdroja, a pre Uri-Port číslo portu 5683, viď RFC 7252 [17] .

- Uri-Host špecifikuje Coap server. Je to dôležité, keď na jednom serveri je prevádzkovaných niekoľko virtuálnych serverov.
- Uri-Port špecifikuje port na transportnej vrstve.
- Uri-Path špecifikuje časť absolútnej cesty k zdroju.
- Uri-Query špecifikuje parameter určený zdroju, napríklad *key=value*.

CoAP definuje URI schému, ktorá má nasledujúci format:

| |
|---|
| "coap:" "//" host [ ":" port ] path [ "?" query ] |
| coap://140.134.200.13:5683/sensor/temp |
| coap://testujem.test.sk:5683/sensor/temp?min=0 |

Tabuľka 3.3: CoAP URI schémy

V tabuľke 3.3 sú znázornené príklady URI schém. V druhom prípade klient poslal požiadavku na server s IPv4 adresou 140.134.200.13 a s portom 5683, kde žiada o zdroj temp. V treťom prípade je za zdrojom pridaná voľba *Uri-Query* s hodnotou *min=10*. Túto voľbu spracuje server ako parameter k prístupu k zdroju.

### 3.2.2 Voľba Accept

Určuje formát obsahu správy, ktorý klient podporuje. Tento formát je rovnaký ako v prípade Content-Format. Server pošle požadovaný obsah vo formáte, aký klient žiada, ak je to možné. V prípade, kedy nieje možné poslať obsah správy vo formáte, aký klient žiada, server musí poslať správu s kódom 4.06, ktorý označuje transakciu ako neakceptovateľnú.

### 3.2.3 Voľba Content-Format

Určuje v akom formáte je reprezentovaný obsah správy. Tento formát je reprezentovaný ako číselný identifikátor. Táto voľba je nevyhnutná pri metódach PUT alebo POST. Ak by chýbala táto voľba, príjemca bude ignorovať obsah správy.

| Formát | ID |
|---|---|
| Text/plain | 0 |
| Application/link-format | 40 |
| Application/xml | 41 |
| Application/octet-stream | 42 |
| Application/exi | 47 |
| Application/json | 50 |

Tabuľka 3.4: Sumarizácia voľby Content-Format

### 3.2.4 Voľba Max-Age

Určuje dobu, po ktorú môžu byť odpovede uložené v pamäti a budú stále označené ako aktuálne. Tento čas je udávaný číselnou hodnotou v sekundách. Maximálne hodnota je $2^{32} - 1$, a preddefinovaná hodnota je šesťdesiat sekúnd.

### 3.2.5 Voľba ETag

ETag je skratka pre *Entity-tag*. ETag je lokálny identifikátor určený pre validáciu zdroja, ktorý sa v priebehu času mení. Je generovaný serverom, ktorý poskytuje daný zdroj. Môže byť generovaný rôznymi spôsobmi, ako je napríklad checksum, hash alebo čas.

### 3.2.6 Voľby Location-Path a Location-Query

Voľby Location-Path a Location-Query plnia rovnakú úlohu ako Voľby Uri-Path a Uri-Query. Rozdiel je však v použití týchto volieb. Tieto voľby sú použíté v odpovediach na požiadavky. Konrétne sa jedná o požiadavky na vytvorenie zdroja alebo zmenu hodnoty zdroja.

### 3.2.7 Voľby Block1 a Block2

Protokol CoAP je navrhnutý tak, aby nezaťažoval sieť veľkými prenosmi dát. Môže nastať situácia, že je potrebné, aby bol prenášaný väčší obsah správy. V takom prípade sa môžu použiť voľby *block1* a *block2*. Väčší obsah správy predstavuje správu CoAP, ktorá presiahne štandardnú veľkosť datagramu CoAP a tieto voľby indikujú, že dáta budú prenášané po blokoch, viď RFC 7959 [16].

Voľby *Block1* a *Block2* musia obsahovať tri informácie:

- Veľkosť bloku, *SZX* (size exponent). Táto informácia je prenášaná v prvých troch bitoch. Veľkosť bloku je počítaná vzorcom $2^{SZX+4}$.
- Bit indikujúci, či bude následovať ďalší blok *M*. Nasleduje za *SZX*.
- Relatívny počet blokov *NUM*, v rámci sekvencie blokov, s rovnakou veľkosťou.

Rozdieľ medzi voľbami *block1* a *block2* je v použití. V prípade, kedy klient CoAP požaduje uloženie dát na server CoAP, je použitá voľba *block1* (metódy PUT a POST). Pri použití voľby *block2* klient požaduje od servera CoAP zdroje (metóda GET).

### 3.2.8 Voľby If-Match a If-None-Match

Tieto voľby slúžia na overenie, či daný zdroj existuje. Využívajú sa na zabránenie či prepísaniu dát v prípadoch, kedy má niekoľko klientov paralelný prístup k daným zdrojom.

- If-Match môže byť použitá k validácií ETag-u. V prípade zhody je podmienka splnená.
- If-None-Match môže byť použitá ako žiadosť pre neexistujúci zdroj. Ak daný zdroj neexistuje, podmienka je splnená.

### 3.2.9 Size1

Táto voľba poskytuje informácie o veľkosti zdroja definovanej v požiadavke. Využíva sa hlavne pri špecifikovaní celkovej veľkosti zdroja v bajtoch pri blokových prenosoch, kedy sú prenášané informácie po blokoch.

### 3.2.10 Voľba Observe

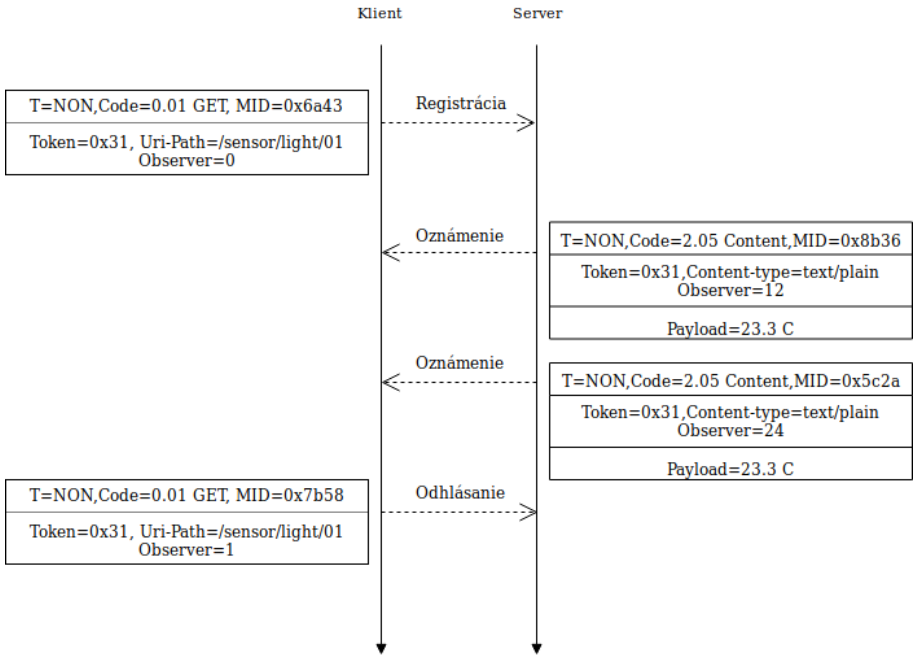
Táto voľba je rozšírenie protokolu CoAP, ktoré umožnuje odoberať zdroje, o ktoré ma kllient záujem. Jedná sa o mechanizmus *publish/subscribe*, ktorý je bližšie špecifikovaný v RFC 7641 [8].

Pozorovateľ (observer) je CoAP klient, ktorý sa registruje voči poskytovateľovi zdrojov (serveru CoAP). Tieto zdroje sa nazývajú v tomto prípade ako predmety (subjects). Pozorovateľ sa registruje/odhlasuje voči poskytovateľovi zdrojov metódou GET s voľbou *Observe=0/Observe=1*. Po úspešnej registrácii poskytovateľ zasiela oznámenia (notification) klientovi o zmene stavu zdroja. Klient potrebuje určit postupnosť týchto oznámení a na to slúži voľba *observe*.

Voľba observe je v tvare *Observe=X*, kde $X$ je číselná hodnota, ktorá slúži ako časová značka. Poskytovateľ na začiatku komunikácie vygeneruje ľubovoľnú hodnotu, ktorá je inkrementovaná každú sekundu s obmedzením $2^{16}$ sekúnd. Transakcia s voľbou *Observe* je znázornená na obrázku 3.1.

Prvou správou sa klient registruje voči poskytovateľovi zdrojov. Registrácia prebieha zaslaním *Non-Confirmable* správy s metódou GET a voľbou *observe* s hodnotou 0. Informácie, ktoré nás v rámci registrácie zaujímajú, sú token, ktorý slúži na spárovanie žiadosti o zdroj s oznámeniami o zmene stavu daného zdroja, a absolútna cesta k zdroju. Ohlásenie klienta prebieha podobne ako registrácia s tým rozdieľom, že je poslaná správa s voľbou *observe*, ktorá má hodnotou 1.

Prvé oznámenie od servera obsahuje voľbu *observe* s hodnotou, ktorú si vygeneruje server a s rovnakou hodnotou tokenu pri registrácié klienta. Nasledujúce oznámenia sa budú líšiť hodnotou voľby *observe*. Na obrázku 3.1 je znázornené, že zdroj zmenil svoj stav po dvanástich sekundách.




Obr. 3.1: Voľba observe

## 3.3 Metódy

V tejto kapitole bude venovaná pozornosť štyrom základným metódam protokolu CoAP. Metódy a ich kódy sú znázornené v tabuľke 3.5.

- Metóda GET slúži na získavanie zdrojov zo servera, ktoré sú špecifikované v požiadavku. Metóda môže obsahovať niekoľko volieb, napríklad voľbu *Accept*, ktorá definue formát obsahu správy pre odpoveď, alebo voľbu *ETag*, ktorá slúži na validáciu zdroja.

- Metóda POST slúži zvyčajne na vytvorenie alebo aktualizovanie cieľového zdroja. V prípade vytvorenia nového zdroja server zašle odpoveď s kódom 2.01 spolu so sekvenciou volieb *Location-Path* a *Location-Query*. V prípade aktualizovania zdroja server pošle správu s kódom 2.04 označujúcim, že boli aplikované zmeny.

- Metóda PUT slúži na vytvorenie alebo aktualizovanie cieľového zdroja. Môže obsahovať napríklad voľby *If-Match* alebo *If-None-Match*. Pri metóde PUT nastáva vytvorenie/aktualizovanie zdroja na definovaje URI. Metóda POST sa používa v prípade, kedy uživateľ nemá prístup/informácie o konkrétnej URI zdroja. Server podľa požiadavky určí o aký zdroj sa jedná.

- Metóda DELETE slúži na vymazanie cieľového zdroja. Server CoAp odpovedá na požiadavku s kódom 200.

| Názov metódy | Kód |
|---|---|
| GET | 0.01 |
| POST | 0.02 |
| PUT | 0.03 |
| DELETE | 0.03 |

Tabuľka 3.5: Kódové označenie metód

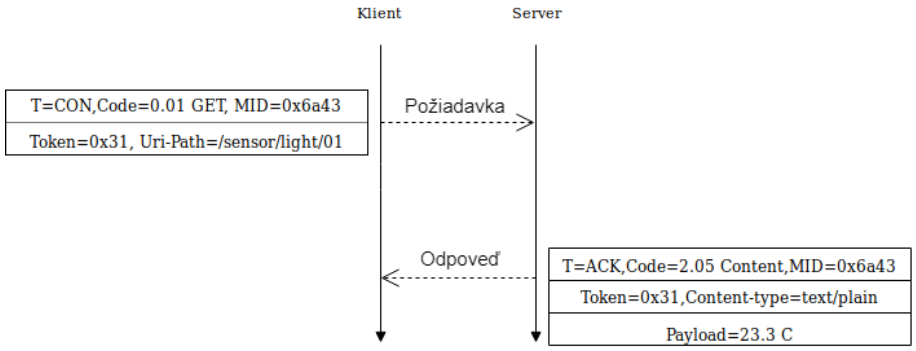
## 3.4 Transakcie

Pri transakcii klienta so serverom môže byť požadovaný spoľahlivý prenos alebo nespoľahlivý prenos. Protokol CoAP taktiež podporuje aj možnosť prenášať väčší objem dát, pri ktorom je možné použiť blokový prenos.

### 3.4.1 Spoľahlivý prenos

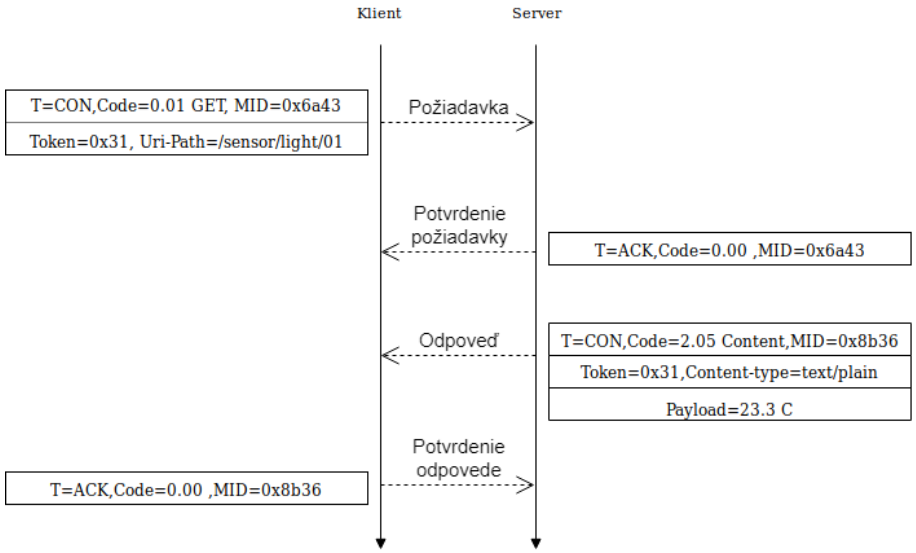
Spoľahlivý prenos zabezpečujú správy typu *Confirmable*, na ktoré musí byť zaslaná potvrdzujúca správa. Transakcie sú označené tokenom, aby bolo možné jednoznačne priradiť odpoveď k požiadavke. Môžu nastať dva spôsoby transakcií:

1. Požadované dáta sú dostupné. V tomto prípade server pošle pozvrdzujúcu správu s rovnakou hodnotou tokenu a MID požiadavku spolu s kódom 2.05 označujúcim, že všetko prebehlo v poriadku. Ďalej nasleduje formát správy a obsah správy. Tento spôsob transakcie sa nazýva aj ako *Piggybacked*. Transakcia je znázornená na obrázku 3.2.




Obr. 3.2: Spoľahlivý prenos

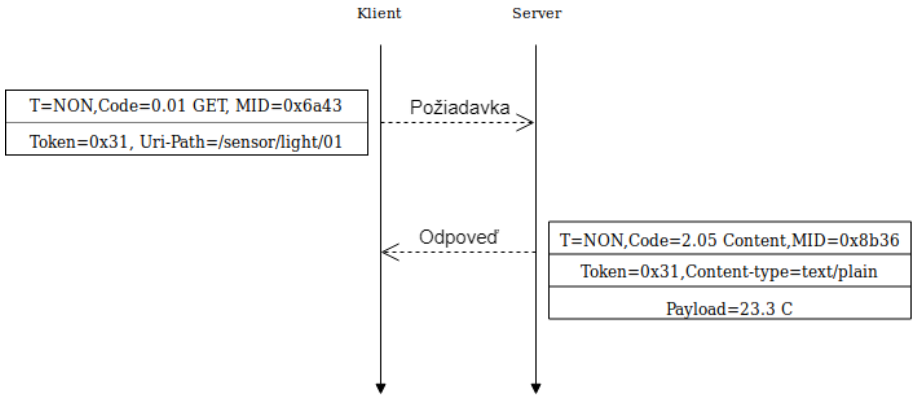
2. V prípade, kedy server nemôže okamžite poskytnúť dáta, nastáva situácia, kedy je odpoveď rozdelená na dve časti. Server pošle klientovi prázdnu potvrdzujúcu správu. Táto správa obsahuje kód 0.00. Správa má rovnaký token a MID, ako požiadavka. Keď bude mať server dostupné požadované dáta, pošle novú *Confirmable* správu s novým MID správy, ale s rovnakým tokenom, aby bolo možné určiť, o ktorú transakciu sa jedná. Následne musí klient poslať prázdnu potvrdzujúcu správu. Transakcia je znázornená na obrázku 3.3.




Obr. 3.3: Spoľahlivý prenos s oneskorením

### 3.4.2 Nespoľahlivý prenos

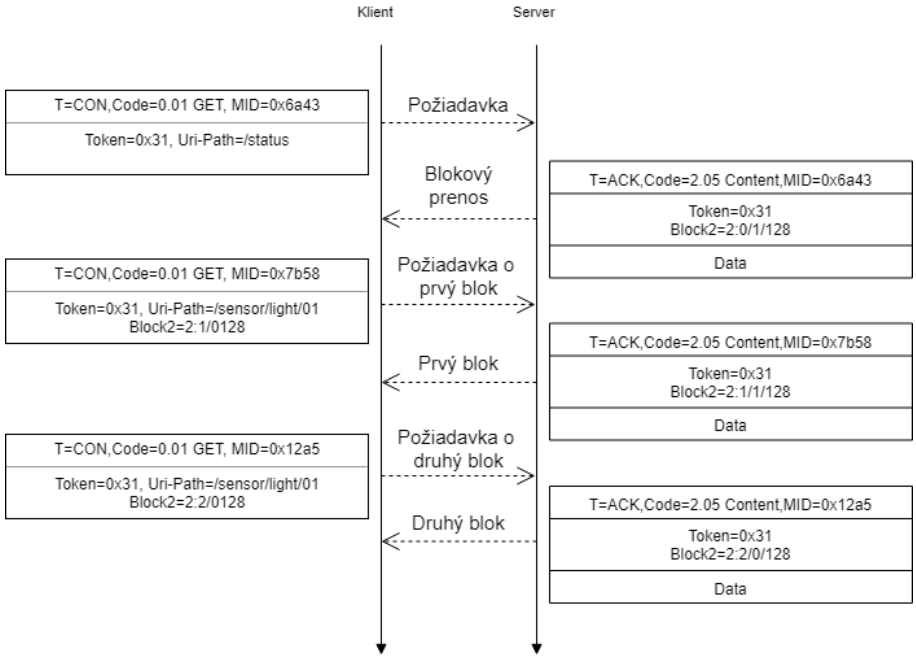
Pri nespoľahlivom prenose sú používané správy typu *Non-Confirmable*. V tomto prípade odpoveď na požiadavku môže mať iný MID správy ako požiadavka. Na spárovanie odpovede s požiadavkou slúži token, ktorý má rovnakú hodnotu. Transakcia je znázornená na obrázku 3.4.




Obr. 3.4: Nespoľahlivý prenos

### 3.4.3 Blokový prenos

Keď klient CoAP požiada o zdroj, ktorý sa prenáša voľbou *block*, server CoAP odpovedá klientovi s voľbou *block*, v ktorej sú uvedené informácie ohľadom zdroja, na ktorý bola vytvorená požiadavka. Odpoveď obsahuje voľbu *block* v tvare $NUM : X/M/SIZE$. Príklad blokového prenosu je znázornený na obrázku 3.5.




Obr. 3.5: Block-wise prenos

Klient požiadal o zdroj */status*. Následne server CoAP odpovedá informačnou správou klientovi, že sa jedná o blokový prenos. Táto správa obsahuje voľbu *block2* s hodnotou *2:0/1/128*. Časť *2:0* zobrazuje informácie o počte blokov a o počte zaslaných blokov. Ďalšia časť s hodnotou *1* je parameter M, viď 3.2, ktorý indikuje, že bude nasledovať ďalší blok. Posledná časť s hodnotou *128* označuje veľkosť bloku v bajtoch. Následne klient žiada o prvú časť dát *2:1* a oznamuje serveru, že nebudú nasledovať bloky hodnotou *0* v parametre *M*. Server odpovedá klientovi s prvou časťou dát *2:1* a parametrom *M* s hodnotou *1*. Prenos sa opakuje pokiaľ server CoAP nepošle posledný blok s parametrom *M*, ktorý bude označovať, že už nenasledujú žiadne bloky. Hodnota parametru *M* je 0.

## 3.5 Multicástová komunikácia

Protokol CoAP definuje jednu multicástovú skupinu a to *All CoAP Nodes*. Táto multicástova skupina sa používa pri objavovaní služieb (service discovery), ktorá je spomenutá v kapitole service discovery 3.6.

- CoAP skupinová komunikácia funguje iba v prípade, keď majú všetky zariadenia v skupine rovnaké číslo portu.
- Používajú sa iba *Non-Confirmable* typ správ.

## 3.6 Service discovery

Protokol CoAP umožnuje zariadeniam CoAP zapojených v IoT sieti jednoduchým mechanizmom objaviť dostupné servery poskytujúce zdroje. Využíva sa pri tomu členstvo v multicástových skupinách, konkrétne sa jedná o multicástovú skupinu *All CoAP Nodes*. Do tejto skupiny by mali patriť všetky zariadenia CoAP. Multicástova IPv4 adresa pre

túto skupinu je 224.0.1.187 [13] a pre IPv6 je to link-local adresa FF02::FD, a site-local FF05::FD [13].

Aby bolo možné využívať túto službu, CoAP server, ktorý poskytuje zdroje, musí podporovať štandardné číslo portu 5683 a pre bezpečnostný mód je číslo portu 5684 [17].

## 3.7 Resource discovery

V prípade, kedy je nevyhnutné poskytnúť informácie o lokalizácií zdrojov bez ľudského faktora, poskytuje CoAP mechanizmy, ktoré dokážu prideliť URI pre cieľové zdroje poskytované serverom CoAP.

Zisťovanie zdrojov prebieha zaslaním požiadavku s metódou GET, kde žiada o zdroj s názvom */.well-known/core* [12], ktorá je preddefinovaná ako vstupný bod. Odpoveďou na túto požiadavku je zoznam všetkých dostupných zdrojov. V prípade, kedy klient potrebuje, len určitú podmnožinu všetkých zdrojov, môže poslať novú požiadavku v tvare *./well - known/core?rt=temp*, kde temp predstavuje názov zdroju.

Mechanizmus na zisťovanie zdrojov využíva webové odkazy. Tieto odkazy sú v tvare *<TARGET-URI>;LINK-PARAM*.

- TARGET-URI predstavuje relatívnu cestu k zdrojom, ktorá musí byť obsahom každého odkazu.
- LINK-PARAM špecifikuje cieľový zdroj. Jednotlivé atribúty sú popísané v RFC 6690 [15]. V nižšie uvedenom príklade sú znázornené dva atribúty, *rt* a *if*. Prvým atribútom je *rt* (resource type), ktorý popisuje typ zdroja. Atribút *if* (interface) popisuje rozhranie zo žiadosti.

```
Request: GET /.well-known/core

Response: 2.05 Content
</sensor>;ct=40

Response: 2.05 Content
</sensor/temp>;rt="temperature-c";if="sensor"
</sensor/light>;rt="light-lux";if="sensor"
```

Tabuľka 3.6: Žiadosť o dostupné zdroje

Tabuľka 3.6 znázorňuje príncíp získavania zdrojov. Klient CoAP posiela požiadavku na */.well-known/core*, na ktorú dostáva dve odpovede. Prvá odpoveď obsahuje relatívnu cestu k zdrojom spolu s atribútom *ct* (content-type) [17], ktorý predstavuje formát obsahu správy. Hodnota atribútu 40 predstavuje *Application/link-format* 3.4. Následná odpoveď obsahuje zoznam všetkých dostupných zdrojov.

## 3.8 Bezpečnosť

Počas fáze poskytovania informácií sú zariadeniam CoAP poskytnuté informácie CoAP serverom ohľadom zabezpečenia. Tieto informácie zahŕňajú kľúče a prístupové záznamy. Tieto kľúče sú generované a inštalované výrobcom. Na konci tejto fáze budú zariadenia v jednom zo štyroch bezpečnostných módov:

- NoSec: v tomto móde neexistuje žiadna bezpečnosť na úrovni protokolu DTLS (Datagram Transport Layer Security), ktoré je bližšie špecifikované v RFC 6347 [14].
- PreSharedKey: v tomto móde je aktivované DTLS a exituje zoznam predbežne zdieľaných kľúčov, kde každý kľúč obsahuje zoznam zariadení, s ktorými je možné nadviazať komunikáciu. Keď bude určený jeden kľúč pre viacej zariadení, tento kľúč slúži ako overenie člena v multicástovej skupine.
- RawPublicKey: v tomto móde je aktivované DTLS a zariadenie má asymetrický pár kľúčov bez certifikátu, ktorý je overený mechanizmom *out-of-band*. Tento mechanizmus je špecifikovaný v RFC 7250 [19]. Identita zariadenia a zoznam zariadení, s ktorými môže komunikovať, sa vypočíta z verejného kľúča.
- Certificate: v tomto móde je aktivované DTLS a zariadenie má asymetrický pár kľúčov s certifikátom X.509 [6]. Pri nadviazaní novej komunikácie musí byť overený certifikát.

Bezpečnosť môžeme rozdeliť na tri hlavné kategérie. Jedná sa o dôveryhodnosť zariadení, integritu dát, a dostupnosť zariadení [11].

Protokolom CoAP sa vo veľkom množstve prenášajú dáta v textovej forme. Aby útočník nemal prístup k citlivím informáciam, je vhodné, aby zariadenia komunikovali iba s oprávnenými zariadeniami. Zariadenia musia zabezpečiť integritu dát. Poskytované alebo získavané informácie zariadeniami musia byť v pôvodnom stave, nemali by byť modifikované.

## 3.9 Zhrnutie

Protokol CoAP je aplikačný protokol nad TCP/UDP určený na prenos dát medzi zariadeniami IoT. Dáta su prenášané nad transportným protokolom UDP, čo umožnuje rýchly prenos paketov bez zaisťovania spoľahlivého prenosu.

Kedže nie je zabezpečený spoľahlivý transport, protokol CoAP obsahuje mechanizmy, vďaka ktorým dokáže detekovať duplicitné správy a spárovať žiadost s odpoveďou. Na detekciu duplicitných správ slúži položka MID (message ID), a na spárovanie žiadosti s odpoveďou slúži položka token v správe CoAP. Aby bol zabezpečný spoľahlivý prenos, správy musia obsahovať typ správy, na ktorý musí byť zaslaná odpoveď, jedná sa o typ *Confirmable*.

V IoT sieťach je vhodné zabezpečiť spôsob, ako sa jednotlivé zariadenia môžu lokalizovať spolu s zdrojmi bez zásahu ľudského faktora. K lokalizácii slúži multicástova skupina *All CoAP nodes*, v ktorej by sa mali nachádzať všetky zariadenia CoAP a mali by používať štandardný aplikačný port. Informácie o tom, aké zdroje poskytuje daný server, môžeme získať zaslaním požiadavky *./well-known/core* na konkrétny server. Server na takúto požiadavku odpovedá zoznamom všetkých dostupných zdrojov a k jednotlivým zdrojom je pridelený atribút, ktorý určuje, v akom formáte je reprezentovaný obsah správy.

# Kapitola 4

# Monitorovanie siete

Monitorovanie siete predstavuje proces ziskávania informácií o sieti. Medzi tieto získavané informácie patria napríklad informácie o zariadeniach zapojených v sieti, štatistické údaje o prenosoch medzi jednotlivými zariadeniami a podobne.

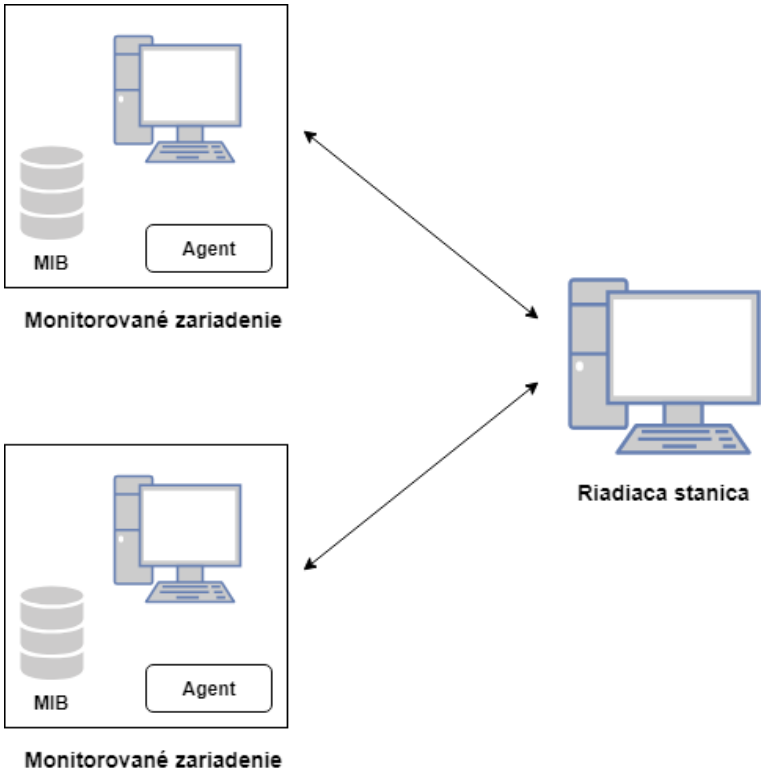
Sú dva typy monitorovania, pasívne a aktívne. Pri pasívnom monitorovaní sú zbierané informácie o sieti, nad ktorými bude neskôr prebiehať analýza. Medzi pasívne monitorovacie prostriedky môžeme zaradiť napríklad protokol Syslog alebo NetFlow a medzi aktívne SNMP. Pri aktívnom monitorovaní, správca môže vytvárať ICMP dotazy na jednotlivé zariadenia, služby a podobne, za účelom získania informácií, o aktuálnom stave zariadenia.

## 4.1 Systém SNMP

Protokol SNMP (Simple Network Management Protocol), RFC 1157 [3], vyžaduje obojstranú komunikáciu medzi riadiacou stanicou a monitorovaným zariadeniam. Riadiaca stanica NMS (Network Management Station) je serverová aplikácia, ktorá tvorí nástroje na zber a analýzu monitorovacích dát, ukladanie štastitík a prezentáciu stavu siete. Riadiaca stanica môže taktiež nastaviť stav monitorovaných objektov podľa požiadaviek správcu siete [10].

Na monitorovanom zariadení beží aktívny proces, agent (Management Agent), ktorý získavá informácie o zariadení a spravuje objetky SNMP. Objekt predstavuje sledovanú informáciu. Objekty sú uložené v databáze MIB (Management Information Base), a na tieto objekty SNMP vytvára riadiaca stanica dotazy. Úlohou agenta je komunikovať s riadiacou stanicou. Agent v bežnej situácii odpovedá na dotazy prichádzajúce z riadiacej stanice. Ak by na zariadení nastala mimoriadna situácia, ktorú by bol agent schopný rozoznať, nemusí čakať na požiadavku od riadiacej stanice, aby informoval spravcu o udalosti. Môže vytvoriť správy typu trap, ktorá bude informovať správcu o mimoriadnej situácii na zariadení.

Na obrázku 4.1 je znázornený príncíp monitorovania protokolom SNMP. Každé monitorované zariadenie zahŕňa bežiaceho agenta a databázu MIB, obsahujúca monitorovacie objekty SNMP, ktoré definujú jeho stav. Riadiaca stanica vytvára požiadavky na objekty SNMP, ktoré sú podrobené analýze.



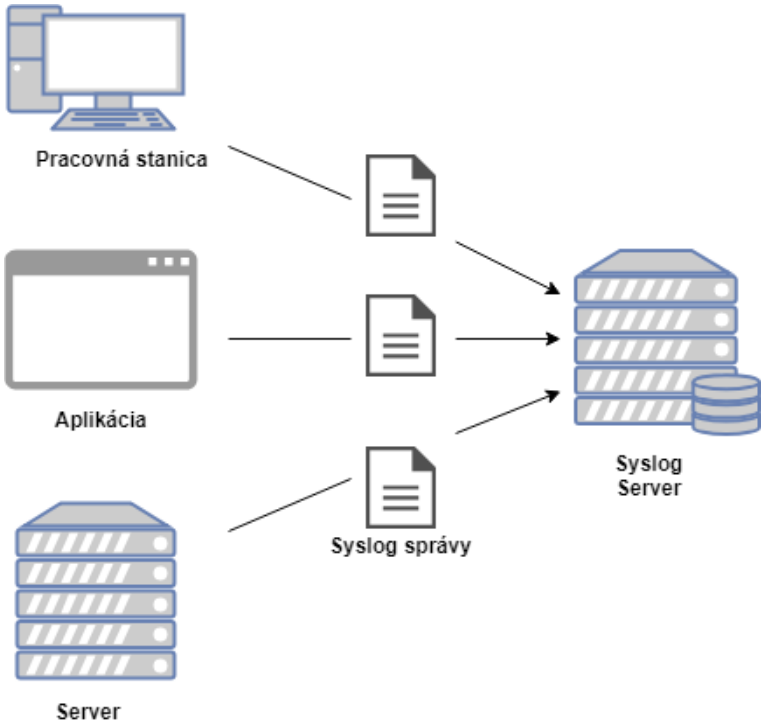

Obr. 4.1: Monitorovanie sieťe pomocou SNMP

## 4.2 Syslog

Syslog je aplikácia ktorá vychádza z Unixu. Slúži k logovaniu systémových udalostí, ktoré sú ukladané do logovacieho súboru. Logovací súbor je nutné zaznamenávať lokálne, ale je vhodné tieto záznamy centralizovať na syslog serveri, kde môže prebiehať analýza získaných dát.

Protokol syslog prenáša informácie medzi klientom a serverom. Klient je zariadenie, ktoré posiela syslog správy s informáciami o udalostiach spojených so zariadením a so službami.

Syslog správa sa skladá zo štyroch položiek. Prvou položkou je severity, dôležitosti správy, ktorá definuje osem úrovní závažnosti správy. Ďalšou položkou je facility. Táto položka definuje typ odosielateľa správy. Môže byť definovaných 24 typov odosielateľov, prvých 16 je definovaných štandardom Syslog, zvyšných osem je určených pre lokálnu špecifikáciu správcom. Treťou položkou je timestamp. Aby mohlo správne fungovať monitorovanie pomocou Syslog-u, je nutné aby všetky zariadenia mali správne nastavený čas, a aby medzi nimi prebiehala synchronizácia, ktorú je možné zabezpečiť NTP serverom. Poslednou položkou je správa, ktorá pozostáva z podrobnejších informácií o udalosti.



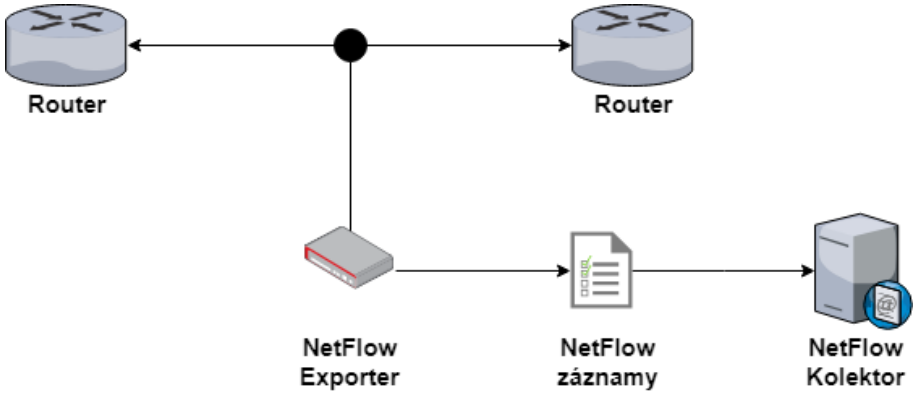

Obr. 4.2: Monitorovanie sieťe pomocou Syslog-u

## 4.3 NetFlow

Protokol NetFlow, RFC 3954 [4], je vyvinútý spološnosťou Cisco Systems. Táto technológia umožňuje monitorovať prenosy v sieti na základe IP tokov. IP tok predstavuje sekvenciu paketov so zhodnou päticou údajov z tretej a štvrtej vrstvy ISO/OSI modelu. Jedná sa o cieľovú a zdrojovú IP adresu, cieľový a zdrojový port a protokol transportnej vrstvy.

NetFlow architektúra sa skladá z dvoch častí. Jedná sa o exporter a kolektor. V sieti môže byť zapojených niekoľko exportérov, ktoré sú pripojené na monitorovacej linke. Zvyčajne je v sieti zapojený jeden kolektor, ale nie je to obmedzenie zo strany exportéra. Exportér môže vysielať dáta na viacej kolektorov.

Úlohou exportéra je vytvoriť NetFlow štatistiky na základe odchytených IP tokov, ktoré sú exportované na NetFlow kolektor. Ako exportér môže byť použitý napríklad router. Takéto zariadenia majú, ale veľké výpočtové nároky, čo sa prejavuje aj na ich cene. Aby boli znížené tieto nároky, používa sa takzvané vzorkovanie, čo znamená že je odchytávaný každý n-tý paket, čo ale skresluje informácie o sieti. Existuje ešte možnosť použiť NetFlow sondu, ktorá plní iba monitorovacie činnosti, a nijako neovyplňuje tok dát. Úlohou NetFlow kolektoru je prijímať prichádzajúce dáta z exportérov a uložiť ich. Tieto záznamy môžu byť ukladané napríklad do lokálnej databáze, nad ktorými prebieha analýza. Monitorovanie protokolom NetFlow je znázornené na obrázku 4.3.




Obr. 4.3: NetFlow monitoring

Technológia NetFlow nám umožňuje vytvoriť si detailný pohľad na aktuálny stav siete v reálnom čase. Vďaka uloženým dátam v databáze môžeme mať prehľad o predchádzajúcich komunikáciach v sieti.

Umožnuje nám získať informácie o množstve prenesených dát v sieti a medzi jednotlivými zariadeniami, vďaka čomu je možné detekovať nezvyčajné udalosti, ktoré môže spôsobiť napríklad nesprávna konfigurácia zariadenia, alebo detekovať DoS útoky a podobne.

## 4.4 Zhrnutie

Zariadenia zapojené v IoT sieti sú zvyčajne mikrokontroléry, ktoré majú obmedzené výpočtové prostriedky a sú určené na jednu úlohu. Kvôli týmto obmedzeniam nieje možné použiť na monitorovanie systém SNMP alebo Syslog.

Monitorovanie založené na IP tokoch nám poskytuje šírší pohľad na sieť. V tomto prípade je vhodnejšie použiť namiesto protokolu NetFlow protokol IPFIX, ktorý je bližšie špecifikovaný v RFC 7011 [5]. IPFIX vychádza z protokolu NetFlow v9. Rozdiel medzi protokolom IPFIX a NetFlow v5 je v tom, že IPFIX umožňuje definovať flexibilný formát monitorovacích dát a prenášať dáta z aplikačnej vrstvy. Informácie z aplikačnej vrstvy nám poskytujú širší prehľad o udalostiach v sieti, ktoré môžu byť použité pri detekovaní anomálií v sieti. Ďalšou výhodou, ktorú nám ponúka tento protokol, je širšie spektrum agregácie tokov, vďaka čomu sa znižuje potrebná pamäť na ukladanie IPFIX záznamov na kolektore.

# Kapitola 5

# Útoky na IoT sieť

IoT sieťe sú pomerne rozšírené, a dopyt po IoT sieťach sa neustále zväčšuje. Tieto sieťe často prenášajú citlivé informácie v otvorenej forme, ktoré by mohol útočník zneužiť, prípadne ich pozmeniť. V tejto kapitole budú spomenúte a priblížené niektoré útoky, ktoré môžu byť aplikované na IoT sieť. Vybrané útoky budú použité ako testovacie sady pre výslednú aplikáciu, ktorej úlohou je detekovať výskyt anonálií v sieti.

## 5.1 Klasifikácia útokov

Útoky môžeme rozdeliť do dvoch kategórií a to na pasívne a aktívne útoky [18]. Pri pasívnom útoku útočník priamo nevstupuje do komunikácie, jeho úlohou je len získať informácie o sieti. Pasívne útoky môžeme rozdeliť na dva typy :

- Odpočúvanie komunikácie – cieľom útoku je zhromažďovať prenášané informácie. Môže sa jednať o zisťovanie uživateľských dát, ako sú prihlasovacie údaje, alebo informácie o prenášaných paketoch.
- Analýza prevádzky – tento typ útoku sa používa v prípadoch, kedy je komunikácia šifrovaná. Prenášané dáta sú zabezpečené. Útočník môže sledovať napríklad prenášané vzory alebo frekvienciu a dĺžku komunikácie.

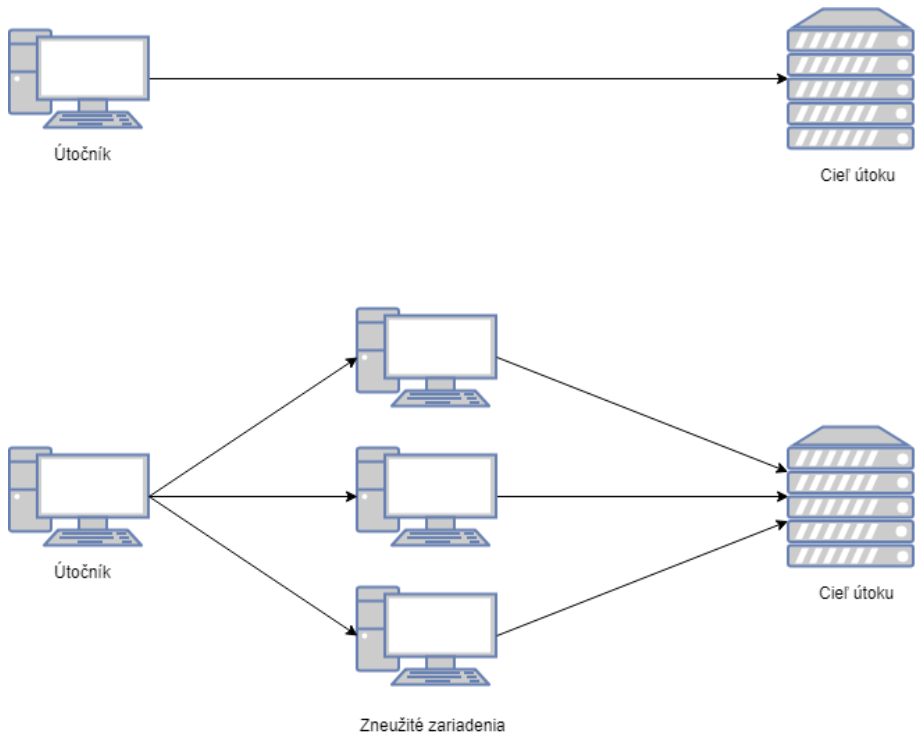
Pri aktívnom útoku sa útočník aktívne zapája do komunikácie. Útoky môžu prebiehať v niekoľkých podobách:

- Podvrhnutie identity – tento typ útoku je založený na napodobnení chovania entity v siete. Útočník sa snaží podvrhnúť zariadeniam správy s falošnou identitou.
- Útok prehrávaním (Replay attack) – podstatou útoku prehrávaním je, že útočnik posiela cieľovému zariadeniu správu, ktorá už bola spracovaná. Táto správa je správne šifrofaná/formátovaná, a tak môže zariadenie túto správu považovať za validnú, a vykonať akcie spojené s danou požiadavkou.
- Modifikácia správy – jedná sa o útok, kedy útočník odchytí komunikáciu a modifikuje validnú správu, odošle neskôr odchytené správy, alebo pošle správy v inom poradí s cieľom získať neautorizovaný prístup.
- Odmietnutie služby – cieľom útoku je obmedziť prístup k požadovanej službe. Tento typ útoku môže mať niekoľko foriem. Môže sa jednať napríklad o útok, pri ktorom

sa snaží útočník vyplýtvať prichádzajúcu alebo odchádzajúcu prenosovú šírku pásma alebo vytvoriť komunikáciu, ktorou by bolo možné vytvoriť odmietnutie služby.

## 5.2 DDoS útok

DDoS (Distributed Denial of Service) je DoS útok, pri ktorom útočník zneužíva iné zariadenia, aby útočili na daný cieľ. Rozdiel medzi DoS a DDoS útokom je znázornený na obrázku 5.1. DoS útoky patria medzi najbežnejšie útoky. Tento typ útoku môže byť použitý na IoT zariadenia a spôsobiť zmeny ich chovania aj pri slabšom útoku, kvôli obmedzeným zdrojom na zariadeniach.




Obr. 5.1: DoS a DDoS útok

DoS útok je pokus útočnika využiť zdroje cieľového zariadenia napr. CPU alebo prenosovú šírku pásma a to posielaním veľkého množstva paketov na cieľové zariadenie. DoS útoky môžeme rozdeliť do niekoľkých typov [9]:

- Záplava UDP - útočník posiela veľké množstvo paketov na náhodny cieľový port, čo spôsobí, že cieľové zariadenie opakovane skontroluje, či aplikácia očakáva spojenie na danom porte a odpovedá správou ICMP, ktorá obsahuje informáciu o nedostupnej službe, *Destination Unreachable*.

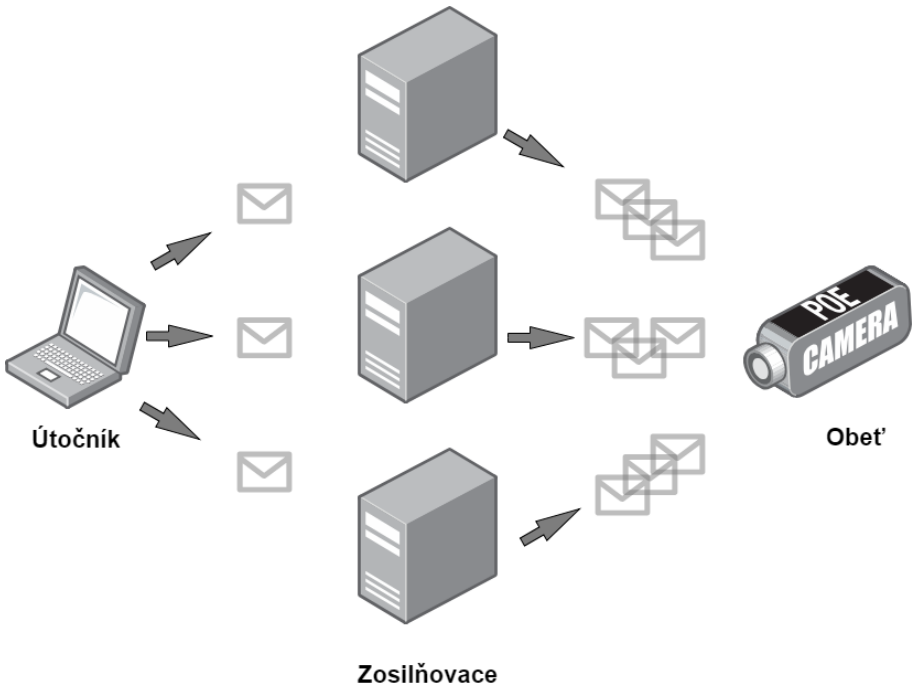
- Záplava ICMP/PING - tento útok je podobný ako záplava UDP s rozdielom, že sa používajú správy ICMP. Používajú sa správy echo, ktoré sa čo najrýchlejšie posielajú na cieľové zariadenie, bez čakania na odpoveď. Takýto útok môže spotrebovať prichádzajúcu, ale aj odchádzajúcu prenosovú šírku pásma. Servery sa často pokúšajú odpovedať na echo správy, čo ma za následok výrazne spomalanie systému.

- Ping of Death - pri tomto type útoku útočník posiela chybne vytvorené ICMP správy (ping). Maximálna veľkosť paketu je definovaná v IP hlavičke a predstavuje 65 535 bajtov, ale nastáva obmedzenie na linkovej vrstve, ktoré definuje maximálnu veľkosť prenášaného rámca, napríklad pre ethernetové sieťe je maximálna veľkosť rámca 1500 bajtov. V prípade, kedy by bol IP paket väčší ako 1500 bajtov, je tento paket rozdelený na viacero IP paketov, nastáva fragmentácia, a príjmateľ zostavuje fragmenty do pôvodného paketu. Pri tomto útoku útočník úmyslne manipulujú s obsahom fragmentov a prijímateľ dostane paket, ktorý presahuje maximálnu veľkosť IP paketu, čo môže spôsobiť pretečenie v alokovanej pamäti pre paket, a tak spôsobiť odmietnutie služby.

## 5.3 Útok zosilnením

V tomto útoku vystupujú tri typy účastníkov. Jedná sa o útočnika, zosilňovače útoku, ktoré predstavujú verejne dostupné servery, a obeť, na ktorú je cielený útok. Cieľom útočníka je vyčerpať prenosovú šírku pásma obete. Útočník zneužíva verejne dostupné servery, ktoré ponúkaju sieťové služby nad transportným protokolom UDP.

Útočník priamo nevyčerpáva šírku pásma obete, ale zneužíva zosilňovače. Útočník vytvorí žiadosť smerovanú na servery s podvrhnutou IP adresou obete. Cieľom tejto žiadosti je vytvoriť niekoľko násobne väčšiu odpoveď, ktorá je určená pre obeť. Na obrázku 5.2 je znázornený amplification útok, ktorý znázornuje komunikáciu medzi útočníkom, zosilňovačmi a obeťou.




Obr. 5.2: Útok zosilnením

### 5.3.1 Faktor zosilnenie útoku

Faktor zosilnenia šírky pásma (bandwidth amplification factor, ďalej už len ako BAF) predstavuje mieru zosilnenia útoku. Mieru zosilnenia útoku je možné vypočítať ako pomer prenesených bajtov poslaných zosilňovačom k preneseným bajtom poslaných servermi ako odpoveď na žiadosti, ktoré boli vytvorené útočníkom [1].

$$BAF = \frac{len(UDP\,payload)\;amplifier\;to\;victim}{len(UDP\,payload)\;attacker\;to\;amplifier}$$

Pri tomto výpočte nie je zahrnutá ethernetová hlavička, IP hlavička ani UDP hlavička. Jedná sa len o zosilnenie, ktoré vznikne na aplikačnej vrstve. Zosilnenie útoku závisí od viacerých faktorov, ako je napríklad počet dostupných zosilňovačov alebo implementácia danej služby.

### 5.3.2 Útok protokolom CoAP

Protokol CoAP umožnuje niekoľko spôsobov, ako vytvoriť viacnásobne väčšiu odpoveď na podvrhnutú požiadavku.

Jedným zo spôsobov, ako zneužiť server CoAP, môže byť dotazovanie na všetky dostupné zdroje, ktoré poskytuje. Útočník vytvorí požiadavku na zdroj s názvom */.well-known/core*, na ktorú server odpovedá obeti s informáciami ohľadom dostupných zdrojov. Tieto informácie môžu obsahovať napríklad relatívnu cestu k zdroju, rozhranie alebo typ zdroja. Faktor zosilnenia útoku v tomto prípade závisí od počtu zdrojov a od počtu atribútov priradených k jednotlivým zdrojom.

Ďalším spôsobom, ako vytvoriť veľku odpoveď na požiadavku je použitie blokového prenosu, ktorý je spomenutý v kapitole 3.4.3. Kvôli tomuto rozšíreniu je možné prenášať sieťov IoT väčšie dátove bloky. Pred zahájaním komunikácie sú špecifikované prenosové parametre, počet dátových blokov a veľkosť jedného dátového bloku v bajtoch. Maximálna veľkosť jedného dátového bloku je 1024 bajtov [16], čím vzniká pomerne veľké zosilnenie.

## 5.4 Zhrnutie

V tejto kapitole sme si klasifikovali útoky do dvoch skupín a to na pasívne a aktívne útoky. Pri pasívnom útoku útočník nevstupuje priamo do komunikácie, ale zhromažďuje informácie. Pri aktívnom útoku útočník vstupuje do komunikácie a snaží sa ovplyvniť chovanie chovanie siete.

Ďalším bodom kapitoly boli tri útoky DDoS a Amplification útok. DDoS útokom sa snaží útočník o spomaliť systém prípadne donútiť systém, aby nemohol poskytovať služby.

K DoS útokom patrí aj amplification útok. Podstatou tohto útoku je zneužitie iných zariadení, ktoré budú predstavovať zosilňovače útoku. Útočník vytvára požiadavky s podvrhnutou IP adresou obete na zasilňovače, ktoré smerujú niekoľko násobne väčšiu odpoveď na obeť útoku.

# Kapitola 6

# Testovacie prostredie

Pre získanie monitorovacích dát bolo nutné vytvoriť sieť, ktorá obsahuje zariadenia CoAP. Sieť bola vytvorená na lokálnom počítači, na ktorom boli vytvorené tri virtuálne počítace s operačným systémom Ubuntu[1] spolu s nainštalovanou knižnicou libcoap[2]. Na všetkých troch virtuálnych strojoch je nutné nastaviť sieťový adapter, na hodnotu *bridged adapter*, aby bolo možné prepojiť virtuálny stroj so sieťou na fyzickom stroji, a tak vytvoriť pre jednotlivé stroje vlastnú identitu. Jednotlivé stroje predstavujú v sieti jedno zariadenia CoAP.

Vytvorená sieť je znázornená na obrázku 6.1. Obsahuje dve zariadenia s názvom IoT node, server a klientov. Zariadenia označené IoT node predstavujú senzory, ktoré slúžia na generovanie zdrojov, na ktoré sa môže dotazovať uživateľ. Hodnota týchto zdrojov sa mení s malou pravdepodobnosťou, aby bolo možné znázorniť komunikáciu medzi serverom a senzorom. V prípade, kedy sa hodnota na senzore zmení, senzor vytvorí požiadavku na server s metódou PUT a uloží novú hodnotu na server.

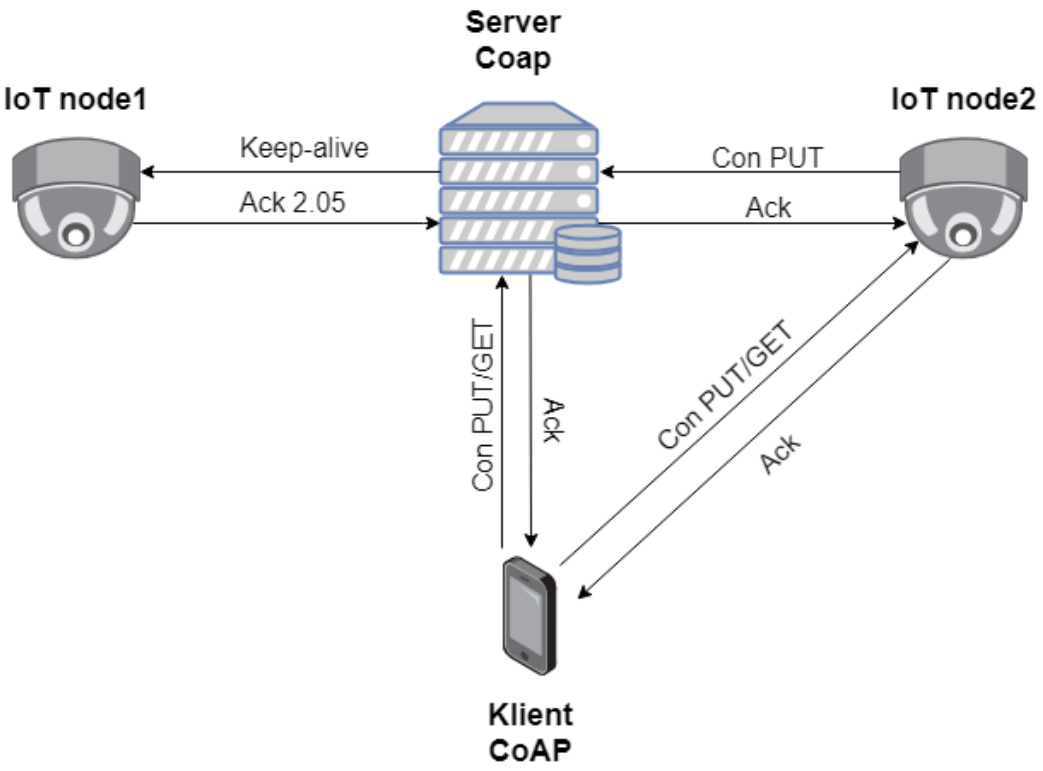
Ako už bolo spomenuté, ďalšíe zariadenie CoAP predstavuje server. Server môže plniť niekoľko úloh. Prvou úlohou, ktorú môže plniť, je že si ukladá dostupné zdroje v lokálnej pamäti. Druhú úlohu, ktorú môže plniť, je smerovanie jednotlivých požiadaviek na konkrétne zariadania, senzory. Taktiež takýto server vytvár tzv. *keep-alive* správy.

Keep-alive správa je generovaná v sekundových intervaloch. Táto správa predstavuje *prázdnu* požiadavku na zariadenia CoAP. Požiadavka sa posiela s typom *confirmable*, ale nevytvára požiadavku na konkretny zdroj. Posiela sa s kódom 2.05, ktorý predstavuje úspešne spracovanú požiadavku a vyžaduje od zariadenia potvrdenie prijatie správy. Odpoveďou je prázdny správa (empty message). Týmto mechanizmom si server udržiava informácie o dostupnosti zariadení, prípadne označuje zariadenie ako nedostupné.

Posledné zariadenia zapojené v sieti predstavuje klient CoAP. Klient je koncové zariadenie, ktoré vytvára rôzne požiadavky. Môže sa jednať o požiadavku napríklad typu GET alebo PUT, kedy by chcel uživateĺ získať prípadne zmeniť hodnotu zdroja. Klient sa nemusí dotazovať, iba na server. Môže vytvárať požiadavky priamo na jednotlivé senzory.

Na obrázku 6.1 je znázornená komunikácia medzi zariadeniami CoAP. Komunikácia medzi zariadením s názvom IoT node1 znázorňuje serverom generované keep-alive správy. Komunikácia medzi IoT node2 a serverom zobrazuje zmenu stavu zdroja na senzore. IoT node2 vytvorí požiadavku na uloženie novej hodnoty na serveri CoAP. Ostatné znázornené komunikácie znázorňujú akcie tvorené použivateľom.

[1] viď https://www.ubuntu.com/
[2] viď https://libcoap.net/




Obr. 6.1: Topológia IoT siete

## 6.1 Použité nástroje pri vytváraní siete

Na vytvorenie zariadení CoAP bola použitá spomenutá knižnica libcoap. Github repozitár knižnice libcoap[3] obsahuje niekoľko príkladov zariadení, napríklad server CoAP alebo klienta CoAP. Tieto príklady boli v malej miere upravené a použité pri vytváraní komunikácie medzi zariadeniami, konkrétne bol upravený súbor coap-server.c v dvoch modifikáciach. Bol upravený tak, aby spĺňal funkcie jednak servera, ale aj aby pracoval ako senzor. Senzor obsahuje generátor s malou pravdepodobnosťou, že nastane zmena hodnoty zdroja. Jedná sa o zdroje s nazvom *floor_1_temp* a *floor_1_light*.

Senzor so zdrojom *floor_1_temp* obsahuje generátor náhodnej hodnoty teploty v rozsahu $10 - 30\,°C$. V prípade, kedy nastane zmena teploty, senzor posiela správu serveru s informáciou o zmene teploty spolu so zmenenou hodnotou.

Druhý senzor, ktorého zdroj má názov *floor_1_light*, sa môže nachádzať v dvoch stavoch, konkrétne v stave *DISABLED* alebo *ENABLED*. Tieto dva stavy predstavujú, či je svetlo zapnuté alebo vypnuté. Aby boli zmeny o niečo realistickejšie, pravdepodobnosť na zmenu stavu je iba 10%. Pri zmene stavu sa tento senzor chová podobne ako senzor určený pre monitorovanie teploty, s tým rozdielom, že sa neposiela hodnota, v akom stave sa nachádza, ale posiela sa správa s hodnotou *ON/OFF*, prípadne s hodnotou *ENABLE/DISABLE*.

Na generovanie požiadaviek na zdroje a získavanie informácií ohľadom IoT siete bol použitý coap-client z príkladov knižnice libcoap, ale aj mobilná aplikácia *IoT CoAP*, ktorá je voľne dostupná z Google play[4]. Mobilná aplikácia umožnuje jednoducho a rýchlo vytvo-

[3] viď https://github.com/obgm/libcoap/tree/develop/examples
[4] viď https://play.google.com/store/apps/details?id=ch.buedev.iot_coap

riť základne požiadavky na koncové zariadenia. Umožnuje vytvoriť si zoznam koncových zariadení v IoT sieti, na ktoré chce vytvárať požiadavky.

Príklady použitia klienta CoAP z knižnice libcoap sú znázornené v tabuľke 6.1.

```
coap-client -m get coap://[192.168.10.106]/.well-known/core
coap-client -m get coap://[192.168.10.106]/floor_1_light -N
coap-client -m get coap://[192.168.10.106]/floor_1_light -T 3a4b -s
coap-client -m put coap://[192.168.10.106]/floor_1_light -t 0 -e ON -T 3a4b
```

Tabuľka 6.1: Príklady použitia klienta CoAP

Klient CoAP ponúka množstvo volieb použitia. V tabuľke 6.1 sú znázornené niektoré voľby, ktoré boli použité pri vytvárani monitorovacích dát. Prvý príklad použitia znázornuje vytvorenie požiadavky na všetky dostupné zdroje servera. Druhý príklad ukazuje spôsob získania stavu svetla s voľbou -N, ktorá inicializuje typ správy ako *Non-Confirmable*. Narozdiel od mobilnej aplikácie, klient CoAP poskytuje rozšírenie *Observe* 3.2.10, ktoré je možné spustiť voľbou -s. Spolu s voľbou Observe je vhodné priradiť k registrácii aj hodnotu tokenu pomocou voľby -T, aby bolo možné spárovať požiadavku s odpoveďou. Posledným príkladom použitia je zmena stavu zdroja. Je použitá metodá PUT spolu s ďalšími dvomi voľbami. Voľba -t 0 určuje *content-format* text/plain pre prenášané dáta, ktoré sú zadávané za voľbou -e.

## 6.2 Vytvorenie monitorovacích dát

Vytvárenie monitorovacích dát prebiehalo na sieti z kapitoly 6. Monitorovanie siete je založené na IP tokov nad protokolom IPFIX. V sieti neboli použité zariadenia určené priamo na odchytávanie dát ako sú sondy IPFIX. Bol použitý nástroj Wireshark, ktorý umožňuje vytvoriť záznamy komunikácie.

Prevádzka protokolom CoAP prebiehala medzi serverom a senzorom CoAP, alebo serverom a klientom CoAP. V oboch prípadoch sa jedná o komunikáciu so serverom CoAP. Odchytávanie prebiehalo na virtuálnom stroji so serverom CoAP, čo umožnilo zachytiť prevádzku vytvorenú protokolom CoAP.

Vytvorené záznamy komunikácie boli transformované na IPFIX záznamy. IPFIX záznamy sú vo formáte CSV.

## 6.3 Analýza prenášaných dat

Analýza komunikácie medzi zariadeniami CoAP prebiehala nad IPFIX záznamami. IPFIX záznamy poskytujú informácie o tokoch aj z aplikačnej vrstve.

Informácie z IPFIX záznámov nám môžu poskytunúť širší prehľad o spôsobe komunikácie medzi jednotlivými zariadeniami. Pod spôsobom komunikácie sú myslené použité voľby v jednotlivých správach, ktoré definujú napríklad formát prenášaných dát, status transakcie alebo špecifikáciu relatívnej cesty k zdroju.

### 6.3.1 Agregácia

Vytvorené IPFIX záznamy použili na agregáciu identifikátor správy (MID). Tento spôsob agregácie nieje vhodný v prípade, kedy správy medzi zariadeniami nevynucujú potrvrdenie

prijatia správy. Pri tomto spôsobe nemusia mať odpovede na požiadavky rovnaký identifikátor, čo znemožňuje správnu agregáciu. Táto agregácia nieje vhodná pri multicástovej komunikácii, pri komunikácii s voľbou Observe, alebo pri blokovom prenose. Prvé dva spôsoby komunikácie používajú správy typu *Non-confirmable*, čo nezaručuje rovnakú hodnotu identifikátoru správy. Pri blokovom prenose sa síce používajú *Confirmable* správy, ale pre jednotlivé prenášané bloky je generovaný nový identifikátor správy.

Vhodnejšou položkou pre agregáciu predstavuje Token. Na základe tejto položky je možné spárovať všetky požiadavky s odpoveďami. Položku Token by mali mať zhodnú všetky spôsoby komunikácie. Platí to aj pre voľbu Observe, ktorá používa *Non-confirmable* správy, ale aj pre blokový prenos. Tento spôsob agregácie by mohol umožniť zredukovať množstvo uložených IPFIX záznamov v databáze.

## 6.4 Emulácia útokov na IoT sieť

V kapitole 5 boli spomenúte niektoré útoky, ktoré môžu ovplyvniť bežný chod siete. Prvým útokom, ktorý bol spomenutý je DoS útok (odmietnutie služby). V tejto kapitole sa budeme venovať emulácií útokov na zariadenia a pozorovať ich chovanie počas aplikovania útoku.

### 6.4.1 Útok záplavou ICMP správ

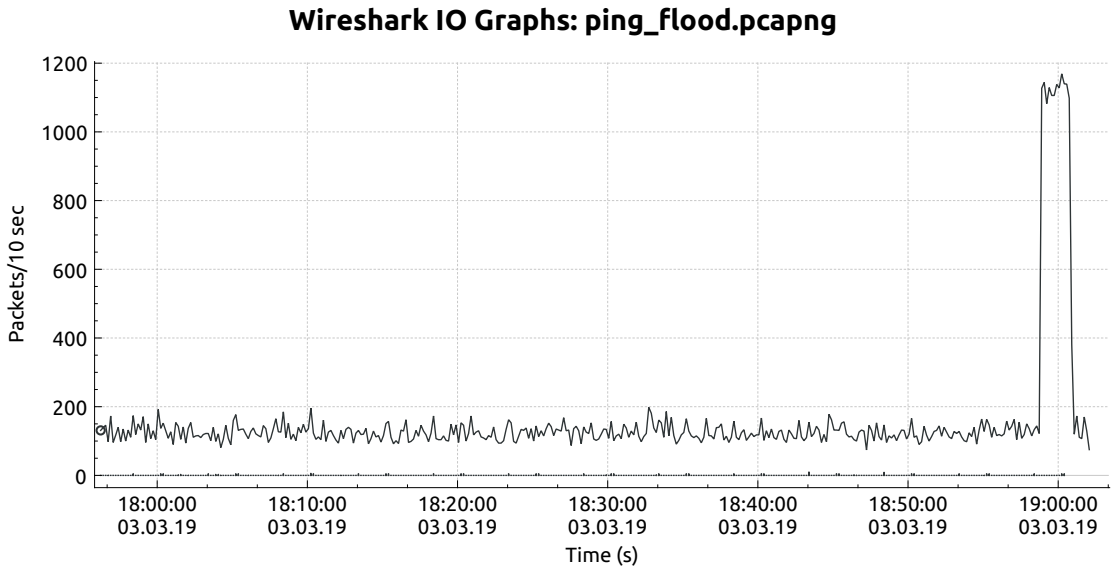
Pri vytváraní ping flood útoku bol použitý voľne dostupný program, ktorý je dostupný v github repozitári ping_flood[5].

Cieľom útoku bolo vytvoriť čo najväčší tok ICMP správ smerovaných na server CoAP a donútiť ho odpovedať na prichádzajúce echo správy a tak spotrebovať prichádzajúcu a odchádzajúcu prenosovú šírku pásma. Podstatou útoku bolo vytvoriť dostatočne veľký tok ICMP správ, aby server CoAP nebol schopný odpovedať na prichádzajúce požiadavky.

Útok bol aplikovaný v rôznych trvaniach, ale ani pri jednom sa nepodarilo obmedziť server. Server generoval v pravidelných intervaloch keep-alive správy a ani prichádzajúce požiadavky neboli ovplyvnené útokom.

Server nezmenil svoje chovanie počas útokov, vplyv na to malo najmä to, že server nebol prevádzkovaný na mikrokontrolery, ale na virtuálnom stroji s oveľa väčším výkonom. Na obrázku 6.2 je zobrazená hodinová prevádzka, počas ktorej bol tri minúty spustený útok na server CoAP.

[5] viď https://github.com/johnboulder/ping_flood/blob/master/ping_flood.c

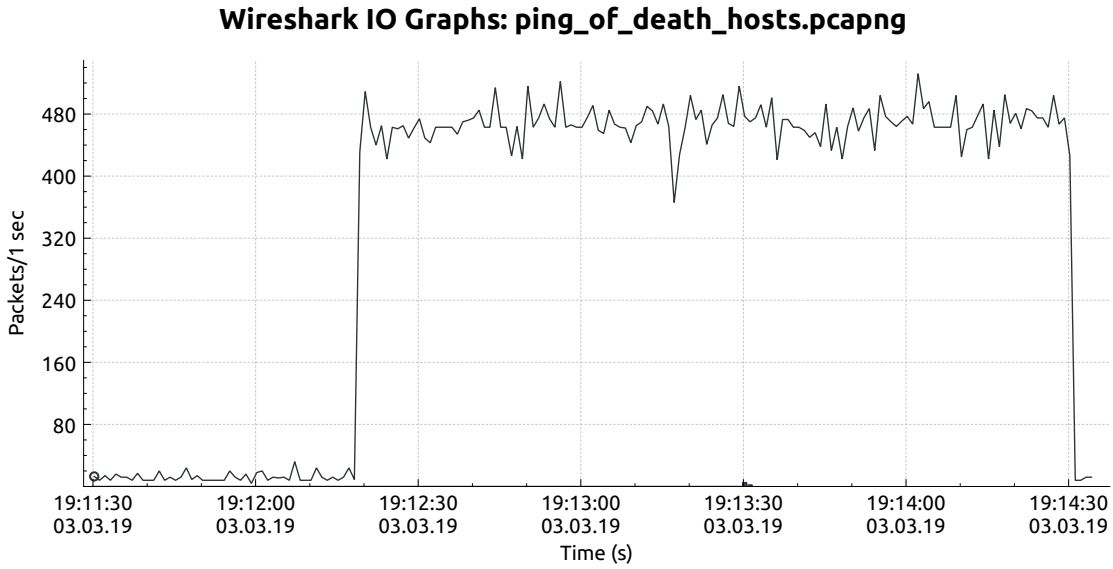



Obr. 6.2: Prevádzka pri útoku ping záplava

### 6.4.2 Útok Ping of death

Pri vytváraní útoku bol použitý voľne dostupný program *pod.py*, ktorý je dostupný v github repozitári ping_of_death[6]. Tento program neponúkal vstupné parametre, a tak bolo nutné manuálne upraviť rozsah generovaných adries.

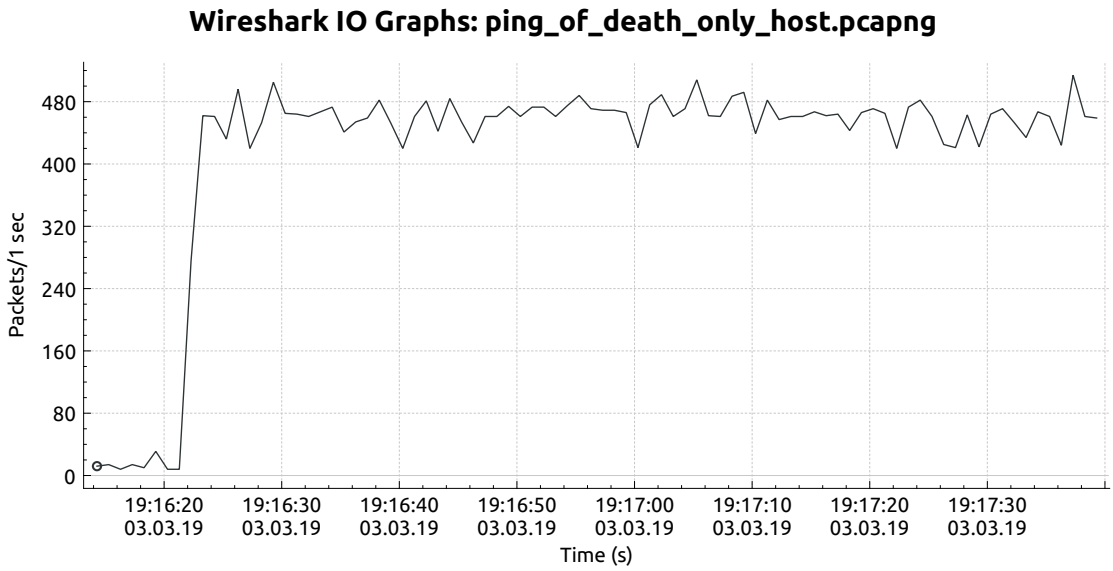
Útok prebiehal v dvoch spôsoboch. Pri prvom spôsobe boli generované ICMP správy o veľkosti 65 000 bajtov. Správa obsahovala podvrhnutú IP adresu odosielateľa, ktorá bola náhodne generovaná v rozsahu 192.168.10.0-192.168.10.255. Na obrázku 6.3 je zobrazená prevádzka.




Obr. 6.3: Prevádzka pri útoku ping of death tvorená niekoľkými zariadeniami

[6] viď https://github.com/Capt-Slow/ping_of_death

Pri druhom spôsobe boli taktiež posielané ICMP správy o veľkosti 65 000 bajtov stým rozdielom, že bola podvrhnutá iba jedna adresa odosielateľa. Na obrázku 6.4 je zobrazená prevádzka.




Obr. 6.4: Prevádzka pri útoku ping of death tvorená jedným zariadením

Obidva útoky boli zamerané na server za účelom z časti obmedziť jeho chovanie. Útokmi sa ale nepodarilo obmedziť server v jeho funkčnosti. Server generoval keep-alive správy v pravidelných intervaloch, taktiež aj odpovede na prichádzajúce požiadavky boli odbavené v rovnakom čase, ako za bežnej prevádzky.

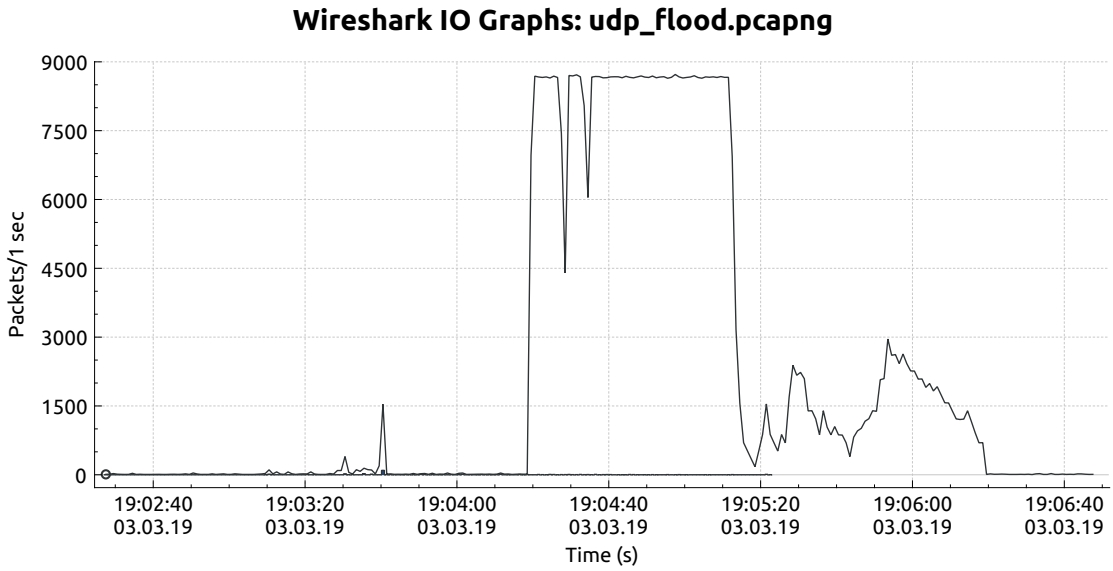
Server nezmenil svoje chovanie počas útokov. Ako už bolo spomenuté, vplyv na to malo najmä to, že server nebol prevádzkovaný na mikrokontrolery, ale na virtuálnom stroji.

### 6.4.3 Útok UDP záplava

Pri vytváraní útoku bol použitý php skript, ktorý je voľne dostupný z github repozitára DDoS-PHP-Script[7]. Skript môže byť spustený s niekoľkými voľbami, ktoré špecifikujú cieľ útoku, trvanie útoku v sekundách, počet prenesených bajtov, počet prenesených paketov, alebo špecifikovať číslo portu.

Podobne ako u predchádzajúcich útokov, bol cieľom útoku server CoAP. Pomocou skriptu boli generované pakety, ktorým sa podarilo ovplyvniť chovanie servera. Počas aplikovania útoku nebol server schopný generovať keep-alive správy a ani odpovedať na prichádzajúce požiadavky. Podarilo sa spotrebovať prichádzajúcu prenosovú šírku pásma, a tak spôsobiť odmietnutie služby. Na obrázku je 6.5 je zobrazená prevádzka vytvorená útokom UDP záplava.

[7] viď https://github.com/drego85/DDoS-PHP-Script




Obr. 6.5: Prevádzka pri útoku UDP záplava

### 6.4.4 Útok zosilnením

Pre vytvorenie amplification útoku bolo nutné vytvoriť program, ktorým by bolo možné podvrhnúť požiadavky zariadeniam CoAP s podvrhnutou IP adresou obete a číslom portu. Program *CoAP_amplification_attack* je napísany v jazyku C a využíva schránky typu *SOCK_RAW*. Tieto schránky je možné použiť na zapisovanie alebo čítanie dat prokolu IP, vďaka čomu je možné modifikovať IP adresy spolu s číslom portu, a tak špecifikovať zariadenia, ktoré sú určené ako zosilňovač a cieľ útoku. Program obsahuje odchytenú správu CoAP, ktorá obsahuje požiadavku na všetky dostupné zdroje. Táto požiadavka je modifikovaná podľa vstupných parametrov programu. Program *CoAP_amplification_attack* je možné spustiť s piatimi voľbami, z toho dve sú povinné. Medzi povinné voľby patrí IP adresa zosiľnovača a IP adresa obete. Nepovinnou voľbou je číslo aplikačného portu. Číslo portu je preddefinované na štandardnú hodnotu 5683. Ďalšie nepovinné voľby slúžia na definovanie trvania útoku v sekundách a na určenie časového intervalu medzi jednotlivými požiadavkami.

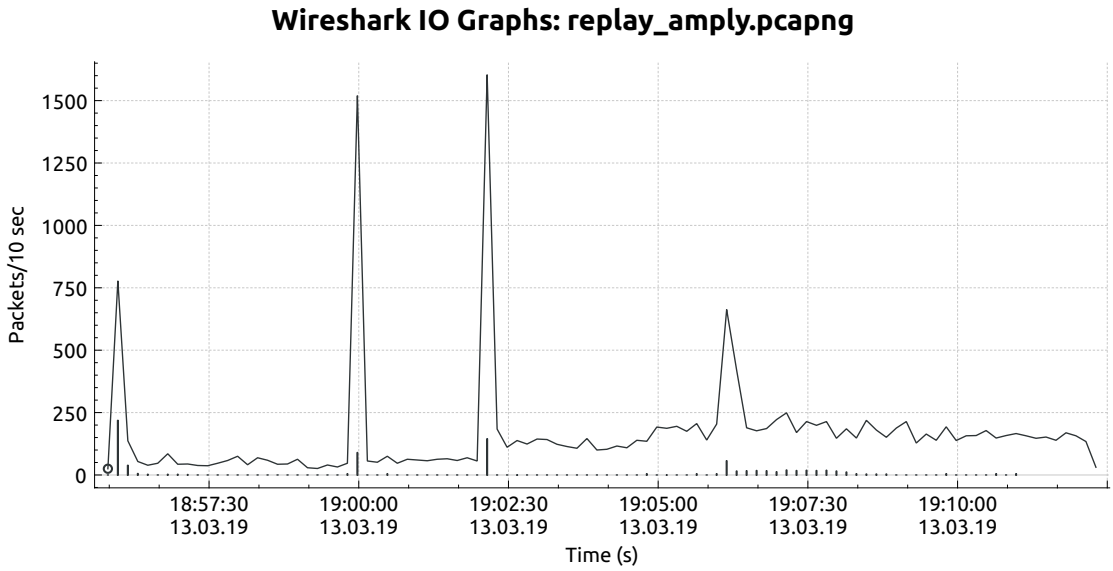
```
./CoAP_amplification_attack -s 192.168.10.101 -d 192.168.10.106
./CoAP_amplification_attack -s 192.168.10.101 -d 192.168.10.106 -i 3 -t 120
```

Tabuľka 6.2: Príklad použitia CoAP_amplification_attack

Útok bol testovaný na sieti, ktorá je špecifikovaná v kapitole 6. Pri útoku bola podvrhnutá adresa klienta 192.168.10.101 a požiadavky boli posielané na adresu 192.168.10.106, ktorá patrila serveru CoAP. Útoky boli generované v rôznych časových intervaloch a rôznych trvaniach. Pri veľkom počte požiadaviek server CoAP neodpovedal na generované požiadavky, pretožé metóda GET je typu *cacheable*. Dôsledku toho bol zväčšený časový interval medzi jednotlivými správami, aby bolo možné odychytiť komunikáciu generovanú podvrhnutými požiadavkami. Server CoAP bol schopný odpovedať na všetky generované požiadavky, keď interval medzi jednotlivými správami bol väčší ako dve sekundy.

Pri aplikovaní útoku nedošlo k spomaleniu serveru CoAP. Server bol schopný v rovnakom časovom intervale generovať keep-alive správy a taktiež odpovedať na prichádzajúcu požiadavky od ostatných zariadení. Pomocou vzorca na vypočet faktoru zosilnenia útoku z kapitoly 5.3.1, môžeme vypočítať hodnotu zosilnenia, ktorá v tomto prípade dosahuje hodnoty $\approx 4,5$. Veľkosť zosilnenia je v tomto prípade závisla od počtu zdrojov a ich špecifikácií na servery CoAP.

Informácie ohľadom dostupných zdrojov môžu byť prenášané blokovým prenosom. Využíva sa to najme v prípade, kedy server CoAP je nútený posielať väčšie množstvo dát. Pri blokovom prenose je ale nutná interakcia klienta so serverom, kedy si predávajú informácie o prenesených blokoch. Môžeme ale nastaviť veľkosť prenášaného bloku na maximálnu hodnotu 1024 bajtov a počet očakávaných blokov, na jeden blok. Takýmto spôsobom sme schopný vytvoriť zosilnenie útoku $\approx 15,05$ krát väčšie. Na obrázku 6.6 je zobrazený prevádzka, pri ktorej boli podvrhované požiadavky na všetky dostupné zdroje.




Obr. 6.6: Prevádzka pri útoku zosilnením

### 6.4.5 Skenovanie IoT siete

Skenovanie siete patrí medzi základné metódy, ktoré používajú útočníci na získanie informácií o sieti. Pomocou skenovaniu siete si útočník vytvára profil sieti. Alebo metódou takzvaného inverzného skenovania si môže útočník vytvoriť prehľad o neaktívnych IP adresách.

Útočník môže získať informácie napríklad o dostupných zariadeniach CoAP, špecifikovať ich úlohy v sieti, a tak tiež môže získať informácie o jednotlivých senzoroch a zdrojoch. Skenovanie môžeme rozdeliť v tomto prípade na dva typy, a to na multicástove a unicástové skenovanie.

Protokol CoAP umožnuje priradiť jednotlivé zariadenia CoAP do multicástových skupín, čo umožňuje vyhľadať v sieti ostatné zariadenia CoAP. Všetky zariadenia CoAP by mali patriť do multicástovej skupiny *All CoAP Nodes*, čo umôžňuje útočníkovy jednoducho získať informácie o dostupných zariadeniach CoAP.

Pri skenovaní siete boli použité príklady z voľne dostupnej knižne FreeCoAP[8]. Repozitár obsahuje time_server a time_client, ktoré podporujú IPv4 multicástovú skupinu *All CoAP Nodes*.

Multicástové skenovanie prebiehalo na troch virtuálnych strojoch. Na dvoch virtuálanych strojoch bol spustený time_server, a na treťom stroji bol spustený time_client. Pred spustením monitorovania bolo nutné pripojiť virtuálne stroje do multicástovej skupiny. Bola použitá IP adresa 224.0.1.187, ktorá predstavuje multicástovú skupinu *All CoAP Nodes*. K pripojeniu zariadenia do multicástovej skupiny bol použitý nástroj *smcroute*[9], ktorý umožnuje jednoducho sa pripojiť prípadne odpojiť zariadenie z multicástovej skupiny. Príkazy použité pri vytváraní skenovania siete sú znázornené v tabuľke 6.3.

| **Príkaz** | **Popis** |
|---|---|
| smcroute -j <interface><address> | Pripojenie k multicástovej skupine |
| netstat -g | Zobrazenie informácií o členstve v multicástovej skupine |
| time_client 224.0.1.187 5683 | Požiadavka o všetky dostupné zdroje zariadení v multicástovej skupine |
| time_server 0.0.0.0 5683 | Spustenie servera s členstvom v multicástovej skupine |

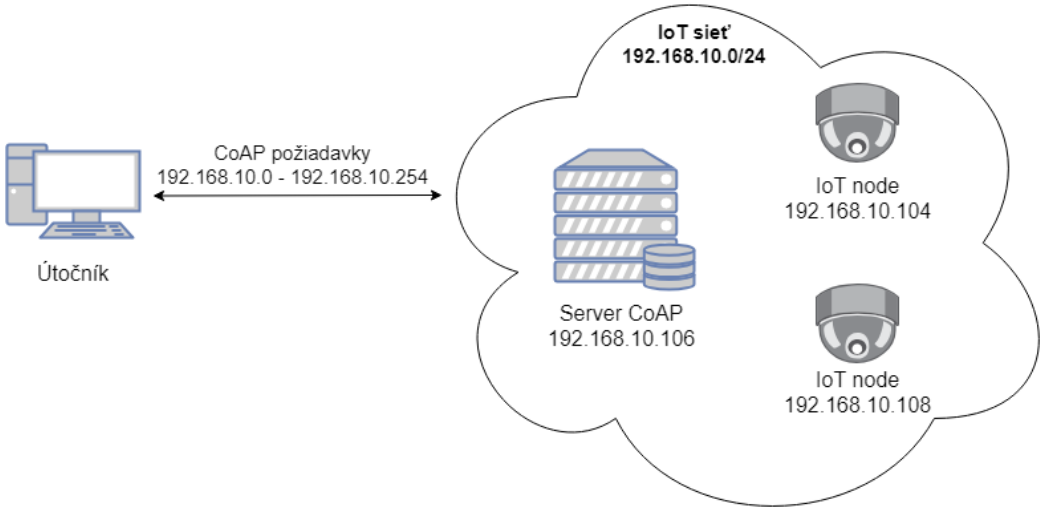
Tabuľka 6.3: Príkazy použité pri vytvorení skenovania pomocou multicástovej skupiny

Unicastové skenovanie siete bolo vykonávané z lokálneho počítača. Sieť určená na skenovanie bola vytvorená na lokálnom počítači, ako bolo spomenúté v kapitole 6. Kedže v sieti neboli zapojené zariadenia určené na odchytávanie dát, bolo nutné odchytávať dáta na lokálnom počítači, aby bolo možné odchytiť všetky prichádzajúce požiadavky.

Pri skenovaní siete bol použitý program *CoAP_amplification_attack* z kapitoly 6.4.4. Týmto programom je možné špecifikovať zdrojovú a cieľovú IP adresu, čo umôžňuje jednuduchým spôsobom prechádzať definovaný adresný rozsah. Pri skenovaní siete bol vytvorený skript, ktorý generoval IP adresy v rozsahu 192.168.10.1 - 192.168.10.254 a spúštal program s podvrhnutou zdrojovou IP adresou a s vygenerovanou IP adresou cieľa z daného rozsahu spolu s parametrami, ktoré určovali interval medzi jednotlivými správami a počet odoslaných správ na jednu IP adresu.

Cieľom skenovania sieťe bolo vytvoriť záznam komunikácie vo formáte PCAP, ktorý obsahuje spôsoby, ako je možné objaviť zariadenia CoAP v domácej IoT sieti. Tieto pcap záznamy sú pomocou IPFIX sondy transformované na IPFIX záznamy, ktoré sú určené na bezpečnostnú analýzu.

[8] viď https://github.com/keith-cullen/FreeCoAP/tree/master/sample
[9] viď https://github.com/troglobit/smcroute




Obr. 6.7: Skenovanie siete

### 6.4.6 Zhrnutie

Kapitola bola zameraná na vytvorenie testovacích sád, obsahujúcu bežnú prevádzku, ale aj prevádzku pri ktorej došlo k určitej anomálií vytvorenou jedným zo spomenutých útokov.

Aby bolo možné splniť tento účel bolo nutné najprv vytvoriť požadovanú sieť, a vybrať vhodný spôsob monitorovania. Pre monitorovanie bol použitý spomenutý nástroj wireshark, ktorý umožňuje vytvárať záznamy kuminikácia. Monitorovanie primárne prebiehalo na zariadení, na ktorom bol spustený server CoAP.

Pri skenovaní siete, nebolo možné odcyhtávať komunikáciu na zariadenia so serverom CoAP, pretože by neboli zachytené požiadavky smerované na ostatné zariadenia.

Vytvorené záznamy komunikácie, boli neskôr transformované IPFIX sondov na IPFIX záznamy. Tieto záznamy budú slúžiť ako testovacie dáta, pre vytvorenú aplikáciu, určenú na monitorovanie IoT sieti. Zoznam vytvorených záznamov komunikácie je zobrazený v tabuľke A.1.

| Názov súboru | Veľkosť | Trvanie |
|---|---|---|
| 17_hodin.pcapng | 37,7 MB | 17 h 30 min |
| DTLS.pcapng | 164,4 kB | 4 min |
| ipv4_multicast.pcapng | 14,0 kB | 2 min |
| multicast_discovery.pcapng | 2,0 kB | <1 min |
| observe.pcapng | 82,8 MB | 50 min |
| ping_flood.pcapng | 5,8 MB | 1 h 5 min |
| ping_of_death_hosts.pcapng | 91,7 MB | 3 min |
| ping_of_death_only_host.pcapng | 53,2 MB | 2 min |
| replay_amply.pcapng | 5,0 MB | 18 min |
| scanning.pcapng | 1,2 MB | 15 min |
| udp_flood.pcapng | 801,9 MB | 5 min |

Tabuľka 6.4: Zoznam vytvorených PCAP súborov

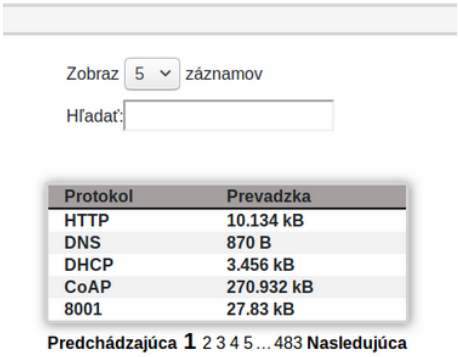
# Kapitola 7

# Implementácia grafického rozhrania

Uživateľske rozhranie je vytvorené ako webová aplikácia, ktorá umožnuje grafickými prostriedkami zobrazovať informácie o sieti ako je právadzka v sieti, ale aj informácie získané z aplikačnej vrstvy, zamerané na protokol CoAP. Nižšie časti kapitoly opisujú jednotlivé časti webovej aplikácie, ako fungujú, aký druh informácie zobrazúju a k čomu môžu byť použité získané informácie. Informácie o prevádzke sú získavané z uložených IPFIX záznamov, nad ktorými prebiehajú databázové operácie. Grafické rozhranie je implementované PHP frameworkom Laravel[1] a knižnicou na vytváranie grafických prostriedkov ConsoleTVs/Charts[2].

## 7.1 Zobrazenie prevádzky

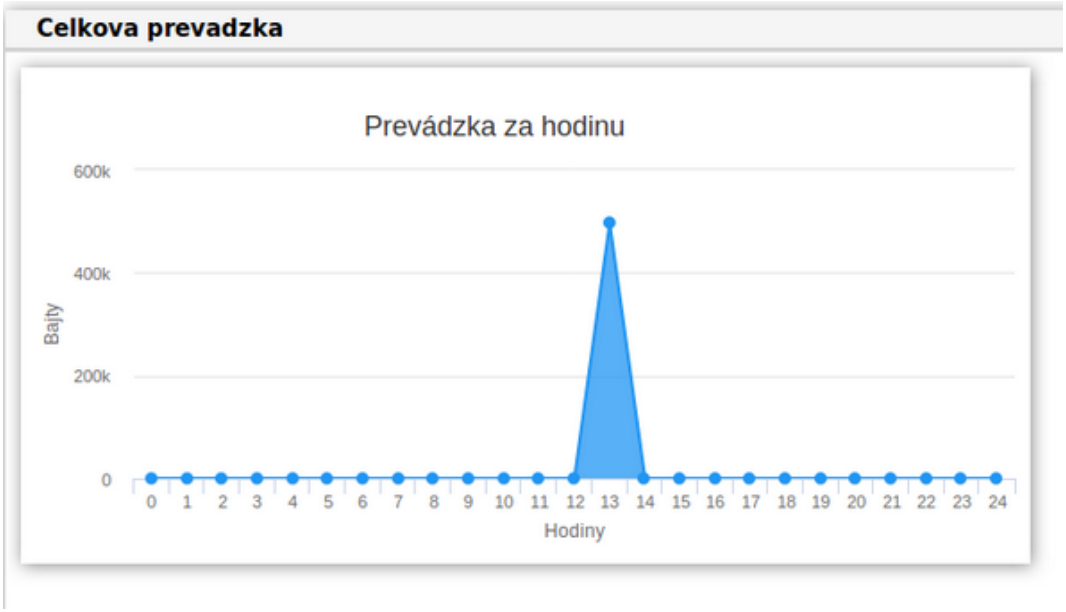
Uživateľské rozhranie zobrazuje niekoľko grafov, ktoré zobrazujú prehľad o udalostiach v sieti. Ako prvé si spomenieme grafy, ktoré sú určené na zobrazenie prevádzky v sieti. Jedná sa konkrétne o tri grafy. Graf znázornujúci celkový tok a graf znázornujúci tok iba protokolom CoAP. Posledný graf znázorňuje množstvo použitých metód protokolu CoAP.




Obr. 7.1: Tabuľka použitých protokolov

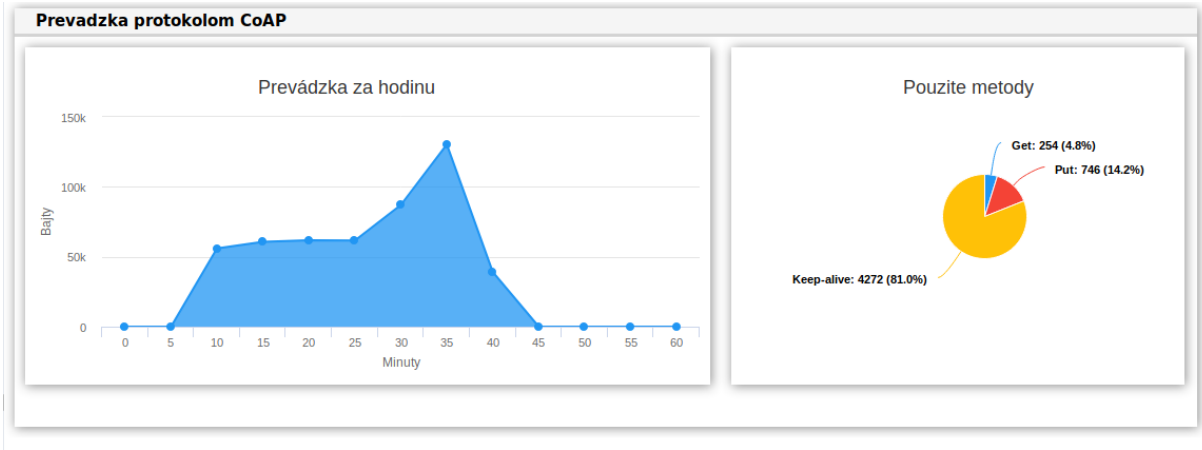
[1] viď https://laravel.com/
[2] viď https://github.com/ConsoleTVs/Charts

S grafom pre celkovú prevádzku, ktorý je zobrazený na obrázku 7.2 je spojená tabuľka zobrazená na obrázku 7.1, ktorá obsahuje informácie o protokoloch (ak je schopná aplikácia rozpoznať, o aký protokol sa jedná) a počte prenesených bajtov. V tabuľke je možné vyhľadávať konkrétne položky a nastaviť počet riadkov tabuľky. Na obrázku 7.2 je zobrazená prevádzka, medzi 12 a 14 hodinou počas ktorej bolo prenesených približne 500 kB dát.




Obr. 7.2: Celková prevádzka

Ďalšie dva grafy znázorňujú prehľad o prevádzke vytvorenú protokolom CoAP, ako je možné vidieť na obrázku 7.3. Na obrázku je znázornená komunikácia za jednu hodinu. V tomto časovom rozsahu bola vytvorená komunikácia v intervale piatej a štyridsaťej piatej minúty vybranej hodiny. Graf napravo znázornuje množstvo použitách metód rozdelených podľa kódu. Z obrázku je vidieť, že najväčšiu prevádzku vytvorili keep-alive správy.




Obr. 7.3: Prevádzka protokolom CoAP

## 7.2 Detekcia zariadení CoAP

Aplikácia umožňuje priradiť zariadenia CoAP do jednej z troch kategórií. Jedná sa o katégorie senzor, server alebo klient.

Na priradenie zariadenia do jednej zo skupín je možné použiť generované keep-alive správy serverom CoAP. Zariadenia, ktoré vytvárajú keep-alive správy, môže zaradiť do skupiny server, a zariadenia, ktorým sú určené do kategórie senzor. Poslednou skupinou, ktorá nám ostáva, sú klienti. Do tejto skupiny patria všetky ostatné zariadenia, ktoré komunikujú protokolom CoAP.

**Mapovanie CoAP zariadení**

| Č. | Typ zariadenia | IP adresa | Zdroj |
|---|---|---|---|
| 1 | Server | 192.168.10.106 | |
| 2 | Senzor | 192.168.10.105 | /floor_1_light |
| 3 | Senzor | 192.168.10.108 | /floor_1_temp |
| 4 | Klient | 192.168.10.104 | |

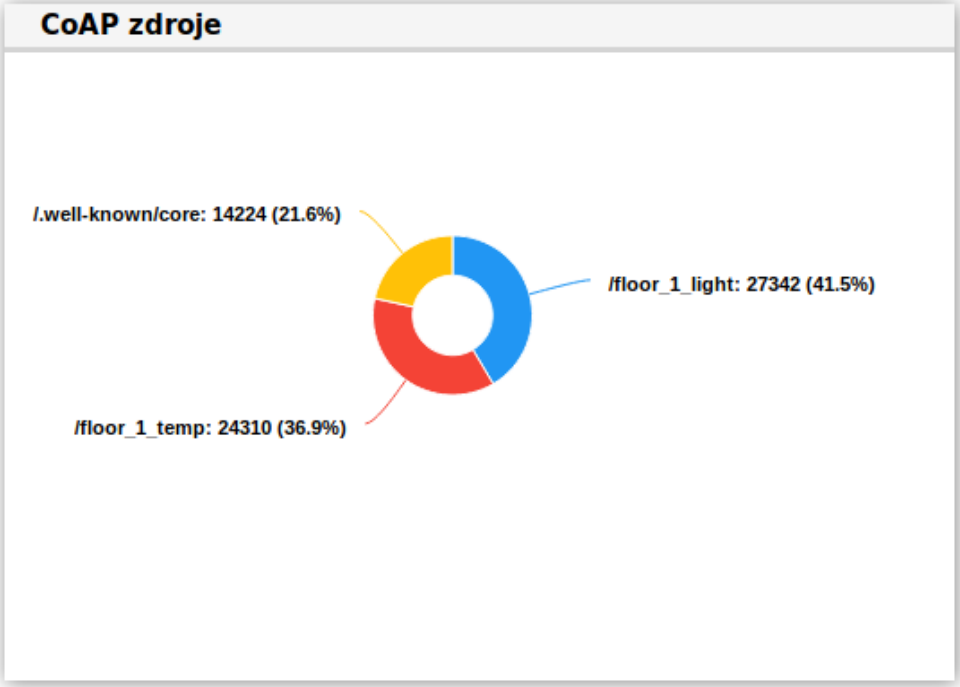
Obr. 7.4: Mapovanie zariadení CoAP

Na obrázku 7.4 je zobrazené mapovanie zariadení CoAP. Každý riadok zobrazuje kategóriu zariadenia, IP adresu zariadenia a ak sa jedná o zariadenie zo skupiny senzor, zobrazí sa aj názov zdroja, ktorý poskytuje.

## 7.3 Dostupné zdroje

V IoT sieti môže byť zapojené veľké množstvo senzorov, ktoré si uchovávajú stavové informácie o svojom zdroji v lokálnej pamäti. Tieto informácie predstavujú zdroje. Na senzory sú tvorené požiadavky ohľadom konkrétnych zdrojov alebo požiadavky na získanie informácií o zdrojoch na senzore. Z dostupných záznamov je možné získať informácie o počte prenesených bajtov tvorených požiadavkami a odpoveďami. Umožnuje nám to získať prehľad o vyťaženosti jednotlivých zdrojov, čo môže slúžiť napríklad na detekciu poškodenia zariadenia, ktoré vytvára oveľa väčší alebo menší dátovy tok.

Graf na obrázku 7.5 znázorňuje prevádzku vytvorenú dvomi zdrojmi */floor_1_light* a */floor_1_temp* spolu s prevádzkou vytvorenou požiadavkami na získanie informácií o zdrojoch */.well-known/core*.



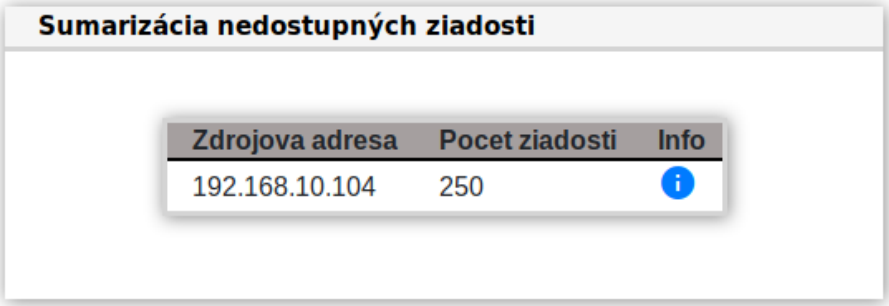

Obr. 7.5: Graf zobrazujúci požiadavky na zdroje

## 7.4 Detekcia nedostupných požiadaviek

Testovacie prostredie podstúpilo dva spôsoby skenovania. Jednalo sa o multicástové a unicástové skenovanie. Pri unicástovom skenovaní siete útočník postupne vytvára požiadavky na konkrétne IP addresy a získava tak informácie o sieti a pripojených zariadeniach.

Aplikácia umožňuje zobraziť nedostupné požiadavky. Tieto informácie boli získané z IP tokov, ktoré obsahovali cieľovú IP adresu, ktorá nevytvorila dátovy tok.

Získané informácie sú sumarizované na základe zdrojovej IP adresy, ku ktorej sa radí celkový počet vytvorených nedostupných požiadaviek. Náhľad na sumarizované informácie je znázornený na obrázku 7.6. Jedná sa o záznam komunikácie, ktorý bol vytvorený pri skenované siete.




Obr. 7.6: Sumarizácia nedostupných ziadosti

Po rozklinutí linku Info z tabuľky sú zobrazené podrobnejšie informácie o vytvorených požiadavkach. Tieto informácie pozostávajú z času, kedy bola vytvorená požiadavka, počiatočná a cieľová IP adresa, a počet opakovaní danej požiadavky v sieti.

**Tabulka nedostupných sprav**

Zobraz 15 záznamov

Hľadať:

| Zaciatok | Zdrojova addresa | Cielova addresa | Pocet ziadosti |
|---|---|---|---|
| 04/19/2019 01:04:59 | 192.168.10.104 | 192.168.10.78 | 1 |
| 04/19/2019 01:04:59 | 192.168.10.104 | 192.168.10.61 | 1 |
| 04/19/2019 01:04:59 | 192.168.10.104 | 192.168.10.11 | 1 |
| 04/19/2019 01:04:59 | 192.168.10.104 | 192.168.10.123 | 1 |
| 04/19/2019 01:04:58 | 192.168.10.104 | 192.168.10.148 | 1 |
| 04/19/2019 01:04:57 | 192.168.10.104 | 192.168.10.150 | 1 |
| 04/19/2019 01:04:57 | 192.168.10.104 | 192.168.10.115 | 1 |
| 04/19/2019 01:04:57 | 192.168.10.104 | 192.168.10.65 | 1 |
| 04/19/2019 01:04:57 | 192.168.10.104 | 192.168.10.198 | 1 |
| 04/19/2019 01:04:56 | 192.168.10.104 | 192.168.10.10 | 1 |
| 04/19/2019 01:04:56 | 192.168.10.104 | 192.168.10.161 | 1 |
| 04/19/2019 01:04:56 | 192.168.10.104 | 192.168.10.206 | 1 |
| 04/19/2019 01:04:56 | 192.168.10.104 | 192.168.10.144 | 1 |
| 04/19/2019 01:04:55 | 192.168.10.104 | 192.168.10.145 | 1 |
| 04/19/2019 01:04:55 | 192.168.10.104 | 192.168.10.66 | 1 |

Záznamy 1 až 15 z celkom 250

Predchádzajúca **1** 2 3 4 5 ... 17 Nasledujúca

Obr. 7.7: Zobrazené nedostupné požiadavky

Protokol CoAP má vyhradenú jednu multicástovú skupinu, viď kapitolu 3.5, ale umožnuje priradiť zariadenia CoAP do ľubovolnej skupiny. Aplikácia zobrazuje multicástovú prevádzku medzi zariadeniami. Informácie o prevádzke sú zhrnuté do tabuľky. Tabuľka je zobrazená na obrázku 7.8. Obsahuje informácie a začatí toku, zdrojovú IP adresu, cieľovú multicástovú adresu, počet vytvorených správ a počet prenesených bajtov. V tomto prípade sa jedná o multicástovú skupinu FF02::FD, *All CoAP nodes*.

**Tabulka multicastovej prevadzky**

**Tabulka nedostupných sprav**

Zobraz 15 záznamov

Hľadať:

| Zaciatok | Zdrojova addresa | Cielova addresa | Pocet ziadosti | Pocet bajtov |
|---|---|---|---|---|
| 04/23/2019 09:43:44 | fe80::bea9:a73e:48c9:5029 | [ff02::fd]:5683 | 1 | 57 |

Záznamy 1 až 1 z celkom 1

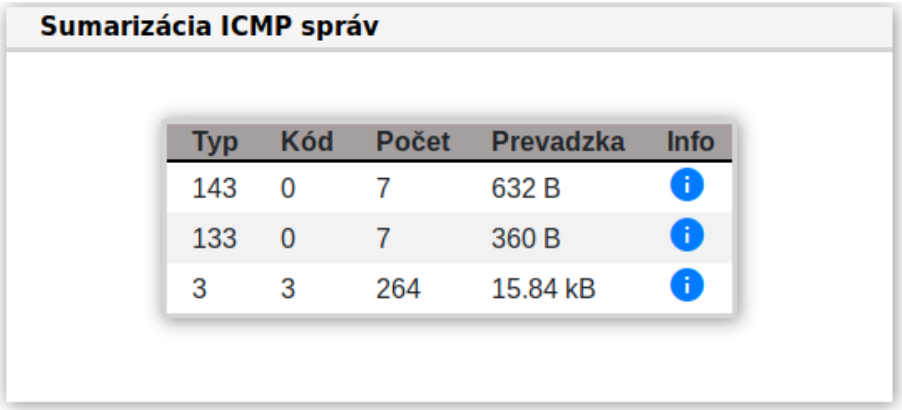
Predchádzajúca **1** Nasledujúca

Obr. 7.8: Tabuľka zobrazujúca multicástovú prevádzku protokolom CoAP

## 7.5 Sumarizácia ICMP správ

Množstvo podstatných informácií o sieti sa dá získať z ICMP správ. Jedným z príkladov môže byť časté generovanie ICMP správ, ktoré označujú zariadenia ako nedostupné. Tieto správy môžu poukazovať napríklad na poruchy na zariadení, alebo že jeho umiestnenie je na hranici dosahu bezdrôtového signálu.

Na obrázku 7.9 je znázornená tabuľka, ktorá obsahuje summarizáciu ICMP správ. Tabuľka pozostáva z typu a kódu ICMP správy, z počtu vytvorených správ, z vytvorenej prevádzky v bajtoch a linkom, ktorý zobrazí podrobnejšie informácie o zvolených správach.




| Typ | Kód | Počet | Prevadzka | Info |
|---|---|---|---|---|
| 143 | 0 | 7 | 632 B | i |
| 133 | 0 | 7 | 360 B | i |
| 3 | 3 | 264 | 15.84 kB | i |

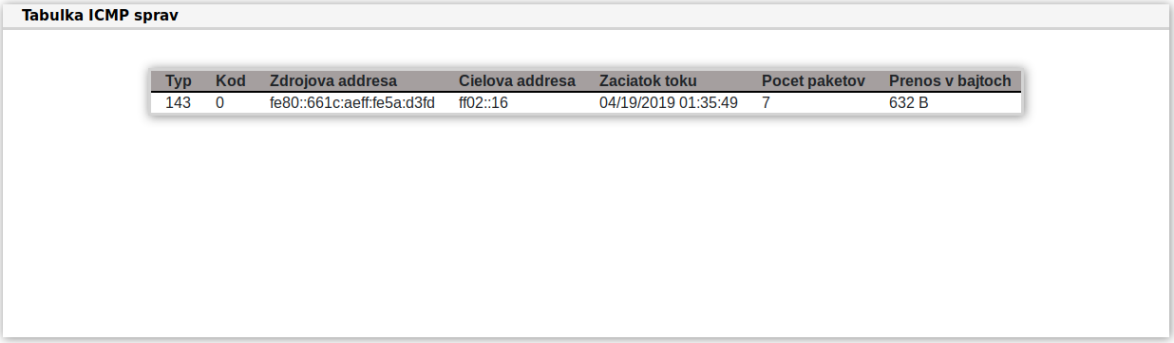
Obr. 7.9: Summarizácia ICMP správ

Tabuľka na obrázku 7.9 obsahuje dva typy ICMPv6 správ s typom 143 a kódom 0, ktorá označuje správu definovanú ako *Multicast Listener Report* a správu s typom 133 a kódom 0 ktorá je definovaná ako *Router Solicitation*

. Správa s typom 3 a kódom 3 označuje správu *Time Exceeded*[3].

Podrobnejšie informácie pozostávajú taktiež z typu a kódu ICMP správy spolu s informáciami o zdrojovej IP adresy, cieľovej IP adresy, začiatok toku, počet prenesených paketov, a počet prenesených bajtov. Tabuľka ICMP správ je znázornená na obrázku 7.10. Na obrázku je vidieť, že sa jedná o správu typu 143 s kódom 0, ktorá je definovaný ako *Multicast Listener Discovery*[4]. ICMP správu vygenerovalo zariadenie s IPv6 adresou fe80::661c:aeff:fe5a:d3fd, a bola smerované na multicástovú adresu ff02::16. Celkovo bolo poslaných sedem paketov, ktorých veľkosť predstavuje 632 bajtov.

[3] viď https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml
[4] viď https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol_for_IPv6




| Typ | Kod | Zdrojova addresa | Cielova addresa | Zaciatok toku | Pocet paketov | Prenos v bajtoch |
|---|---|---|---|---|---|---|
| 143 | 0 | fe80::661c:aeff:fe5a:d3fd | ff02::16 | 04/19/2019 01:35:49 | 7 | 632 B |

Obr. 7.10: Tabuľka ICMP správ

## 7.6 Prevadzka medzi zariadeniami CoAP

Navigácia obsahuje položku CoAP, ktorá zobrazí informácie o prevádzke medzi jednotlivými zariadeniami CoAP. Zobrazí sa komunikácia medzi všetkými zariadeniami CoAP. Jedná sa o komunikáciu vytvorenú protokolom CoAP. Komunikácia je znázornená čiarovým grafom, ktorý popisuje prevádzku vytvorenú smerom k cieľu, požiadavky a prevádzku vytvorenú smerom k zdroju, ktorá pozostáva z odpovedí na požiadavky.

Komunikácia medzi zariadeniami s IP adresou 192.168.10.108 a 192.168.10.106 je zobrazená na obrázku 7.11. Komunikačný graf neznázornuje celkovú prevádzku medzi zariadeniami, ale znázornuje prevádzku vytvorenú požiadavky z IP adresy 192.168.10.108 a odpoveďami na vytvorené požiadavky zariadením s IP adresou 192.168.10.106.

Ďalší graf na obrázku 7.11 zobrazuje počet použitých metód medzi zariadeniami. Poslednou častou tohto zobrazenia je tabuľka, ktorá obsahuje informácie o maximálnej a priemernej veľkosti paketu smerom od zdroja k zdroju a počte použitých volieb.


| # | Velkost paketu od zdroja | Velkost paketu k zdroju | Pocet volieb |
|---|---|---|---|
| Maximum | 49 B | 32 B | 2 |
| Priemer | 49 B | 4.15 B | 0.52 |

Obr. 7.11: Komunikácia medzi zariadeniami CoAP

### 7.6.1 Detekcia útoku Ping of Death

Detekcia útok Ping of Death je založená na získaných informáciach z ICMP správ. Jedná sa o typ o kód ICMP správy, počet paketov a prevádzku v bajtoch. Na obrázku 7.12 sú zobrazené tabuľky obsahujúce informácie o ICMP správach a nedostupných požiadavkách.

Trvanie útoko bolo približne tri minúty. Z obrázku je vidieť, že je vytvorená veľká prevádzka ICMP správami s typom osem a kódom nula čo definuje správu *echo request*.

**Sumarizácia ICMP správ**

| Typ | Kód | Počet | Prevadzka | Info |
|---|---|---|---|---|
| 0 | 0 | 2025 | 77.304 kB | |
| 8 | 0 | 59722 | 88.605 MB | |
| 3 | 3 | 54 | 3.47 kB | |

**Sumarizácia nedostupných ziadosti**

| Zdrojova adresa | Pocet ziadosti | Info |
|---|---|---|
| 192.168.10.105 | 276 | |
| 192.168.10.107 | 290 | |

Obr. 7.12: Detekcia Ping of Death útoku

Tabuľka zobrazujúca nedostupné požiadavky poukazuje na nedoručené požiadavky počas útoku. Zdrojové adresa 192.168.10.105 a 192.168.10.107 predstavujú senzory CoAP. Na obrázku 7.13 je zobrazená tabuľka s podrobnejšími informáciami o daných požiadavkách. Senzor CoAP s adresou 192.168.10.105 sa snažil nadviazať spojenie so zariadením s adresou 192.168.10.106 276 krát. Zariadenie s adresou 192.168.10.106 predstavuje server CoAP, na ktorý bol aplikovaný útok.

**Tabulka nedostupných sprav**

Zobraz 15 záznamov

Hľadať:

| Zaciatok | Zdrojova addresa | Cielova addresa | Pocet ziadosti |
|---|---|---|---|
| 03/04/2019 02:03:29 | 192.168.10.105 | 192.168.10.106 | 276 |

Záznamy 1 až 1 z celkom 1

Predchádzajúca **1** Nasledujúca

Obr. 7.13: Nedostupné požiadavky pri útoku Ping of Death

### 7.6.2 Detekcia útoku záplavou ICMP správ

Podobne ako pri útoku je Ping of Death detekcia záplavou ICMP správ založená na získaných informáciach z ICMP správ. Na obrázku 7.14 sú znázornené tabuľky ohľadom ICMP správ a nedostupných požiadaviek. Z ICMP tabuľky je vidieť, že pri útoku boli použité echo ICMP správy. Jednalo sa o 12 000 správ, ktoré vytvorili prevádzku o veľkosti 349 kB.

**Sumarizácia ICMP správ**

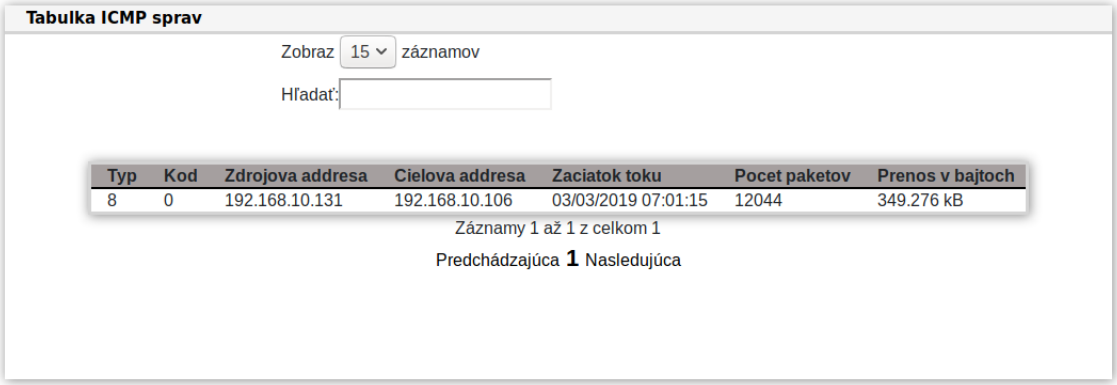
| Typ | Kód | Počet | Prevadzka | Info |
|---|---|---|---|---|
| 143 | 0 | 4 | 304 B | i |
| 135 | 0 | 1 | 64 B | i |
| 133 | 0 | 7 | 392 B | i |
| 8 | 0 | 12044 | 349.276 kB | i |
| 3 | 3 | 122 | 19.852 kB | i |

**Sumarizácia nedostupných ziadosti**

| Zdrojova adresa | Pocet ziadosti | Info |
|---|---|---|
| 192.168.10.106 | 10270 | i |

Obr. 7.14: Detekcia útoku záplavou ICMP správ

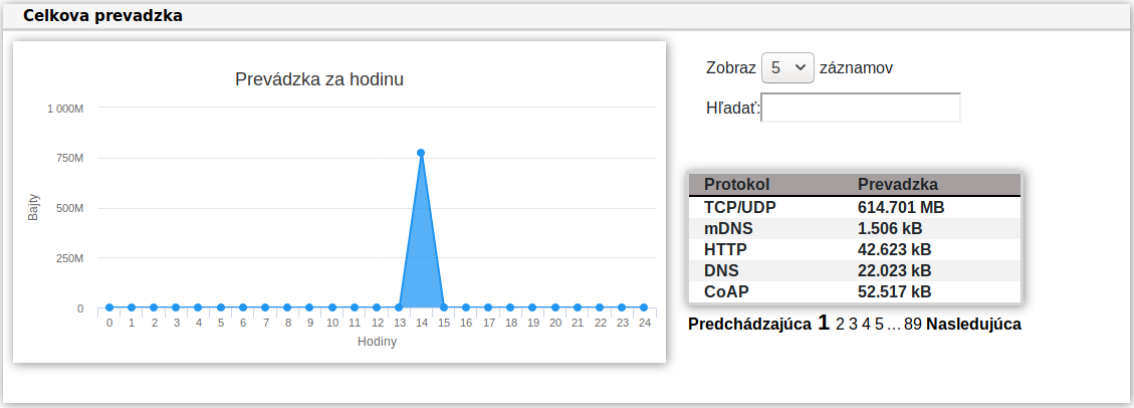
Na obrázku 7.15 sú zobrezené informácie o daných ICMP správach. Jednalo sa echo správy, ktoré generovalo zariadenie s IP adresou 192.168.10.131, ktoré boli cielené na CoAP server s adresou 192.168.10.106.




Obr. 7.15: Informácie z ICMP tabuľky

### 7.6.3 Detekcia útoku záplavou UDP

Pri tomto útoku je tvorená veľká prevádzka protokolom UDP. Informácie o prevádzke sú zobrazené grafom, ale aj tabuľkou obsahujúcou informácie o počte prenesených bajtov jednotlivými protokolmi. Na obrázku 7.16 je znázornená komunikácia počas útoku. Na grafe je zobrazená prevádzka medzi 13 a 15 hodinou. V tomto intervale bola dosiahnutá prevádzka viacej ako 750 MB. Z tabuľky je vidieť, že najväčšiu časť vytvorili TCP/UDP pakety a to konkrétne 614,701 MB.




Obr. 7.16: Zobrazenie útoku záplavou UDP paketov

## 7.7 Zhrnutie

Vytvorená aplikácia pre zobrazovanie monitorovacích dát protokolom CoAP bola testovaná testovacími sadami z kapitoly 6.4. Jednalo sa útok Ping of Death, záplava UDP, záplava ICMP a skenovanie siete. Jednotlivé útoky boli rozdelné do samostatných tabuliek v databáze za účelom jednoduchého prístup k dátam z konkrétneho útoku.

# Kapitola 8

# Záver

Cieľom bakalárskej práce bolo vytvoriť monitorovací systém pre domácie IoT siete, ktorý by bol schopný detekovať anomálie v sieti.

Prvým čiastkovým cieľom bolo oboznámenie s jedným z aplikačných protokolov určených pre IoT siete. Zameral som sa na protokol CoAP, ktorému je venovaná kapitola 3, v ktorej je popísaný formát prenášaných správ a spôsoby komunikácie medzi zariadeniami.

Ďalším cieľom práce bola analýza PCAP súborov a IPFIX záznamov. Aby som mohol tento cieľ splniť, bolo nutné vytvoriť testovacie prostredie obsahujúce zariadenia CoAP. Testovaciemu prostrediu je venovaná kapitola 6. Snažil som sa simulovať chovanie servera CoAP a senzorov CoAP, ktoré zahrňovalo generovanie *keep-alive* správ a zmenu stavu zdroja na zariadeniach. Pri analýze IPFIX záznamov som zistil, že protokol CoAP obsahuje dve položky podľa ktorých by bola vhodná agregácia IP tokov. Jedná sa o identifikátor správy (MID) a token, ktoré ale nie sú povinné. Pri vytváraní IPFIX záznamov bola použitá IPFIX sonda, ktorá použila na agregáciu položku MID. Vhodnejšou položkou by bola položka token, ktorá slúži na spárovanie požiadaviek s odpoveďami. Týmto spôsob by sa zredukoval počet záznamov pri komunikácii voľbou Observe alebo pri blokovom prenose.

Testovacie prostredie podstúpilo útoky z kapitoly 5. Pri útokoch prebiehalo monitorovanie nástrojom Wireshark, ktorého výstup boli záznamy komunikácie. Záznamy komunikácie boli transformované na IPFIX záznamy, ktoré boli samostatne uložené v MySQL databáze.

Výstupmi prácu sú záznamy komunikácie vo formáte PCAP, ktoré obsahujú komunikáciu počas aplikovania útokov, ale aj bežnú komunikáciu a webová aplikácia, ktorá umožňuje zobrazovať informácie o prevádzke. Aplikácia bola testovaná vytvorenými IPFIX záznami, ktorých výsledok je znázornený v kapitole 7.7.

Práca bola súčasťou výzkumného projektu IRONSTONE[1], kde sa využila vytvorená sonda FlowMon pre zber a vytváranie rozšírených záznamov IPFIX.

[1] viď https://www.vutbr.cz/vav/projekty/detail/27735

# Literatúra


[1] administrator: DDoS Amplification Attacks. [Online; navštívené 5-Február-2019]. URL https://www.noction.com/blog/ddos-amplification-attacks

[2] Bormann, C.; Ersue, M.; Keranen, A.: Terminology for Constrained-Node Networks. RFC 7228, Máj 2014.

[3] Case, J. D.; Fedor, M.; Schoffstall, M. L.; aj.: A Simple Network Management Protocol (SNMP). RFC 1157, Máj 1990.

[4] Claise, B.; Sadasivan, G.; Valluri, V.; aj.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, October 2004.

[5] Claise, B.; Trammell, B.; Aitken, P.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, September 2013.

[6] Cooper, D.; Santesson, S.; Farrell, S.; aj.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Technická správa, Máj 2008.

[7] Hanes, D.; Salqueiro, G.; Grossetete, P.; aj.: IoT Fundamentals. Technická správa, August 2017.

[8] Hartke, K.: Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641, September 2015.

[9] Incorporated, T. I.: REST Layers and REST Applications. https://www.academia.edu/10172972/A_Survey_DDOS_Attack_on_Internet_of_Things, 2008, [Online; navštívené 24-Január-2019].

[10] Matoušek, P.: *Síťové aplikace a jejich architektura*. Brno: Akademické nakladatelství, VUTIUM, 2014, ISBN 97-8802-214-37661.

[11] Ninja, S.: CIA Triad. [Online; navštívené 20.06.2018]. URL https://resources.infosecinstitute.com/cia-triad/

[12] Nottingham, M.; Hammer-Lahav, E.: Defining Well-Known Uniform Resource Identifiers (URIs). RFC 5785, April 2010.

[13] Rahman, A.; Dijk, E.: Group Communication for the Constrained Application Protocol (CoAP). RFC 7390, October 2014.

[14] Rescorla, E.: Datagram Transport Layer Security Version 1.2. RFC 6347, January 2012.

[15] Shelby, Z.: Constrained RESTful Environments (CoRE) Link Format. RFC 6690, August 2012.

[16] Shelby, Z.; Bormann, C.: Block-Wise Transfers in the Constrained Application Protocol (CoAP). RFC 7959, August 2016.

[17] Shelby, Z.; Hartke, K.; Bormann, C.: The Constrained Application Protocol (CoAP). RFC 7252, Jún 2014.

[18] Stallings, W.: *Network security essentials: applications and standards fourth edittion.* Brno: Prentice Hall, 2010, ISBN 0-13-610805-9.

[19] Wouters, P.; Tschofenig, H.; Gilmore, J.; aj.: Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7250, Jún 2014.

# Príloha A

# Obsah DVD

- dataset-pcap/ – adresár obsahujúci vytvorené záznamy komunikácie vo formáte PCAP.
- text-prace – obsahuje súbory k textu práce.
- src/ – adresár obsahujúcí zdrojové kódy k aplikácii.
- xkrajc17.pdf – text bakalárskej práce.

| Názov súboru | Veľkosť | Trvanie |
|---|---|---|
| 17_hodin.pcapng | 37,7 MB | 17 h 30 min |
| DTLS.pcapng | 164,4 kB | 4 min |
| ipv4_multicast.pcapng | 14,0 kB | 2 min |
| multicast_discovery.pcapng | 2,0 kB | <1 min |
| observe.pcapng | 82,8 MB | 50 min |
| ping_flood.pcapng | 5,8 MB | 1 h 5 min |
| ping_of_death_hosts.pcapng | 91,7 MB | 3 min |
| ping_of_death_only_host.pcapng | 53,2 MB | 2 min |
| replay_amply.pcapng | 5,0 MB | 18 min |
| scanning.pcapng | 1,2 MB | 15 min |
| udp_flood.pcapng | 801,9 MB | 5 min |
| minimalna$_a$*ktivita.pcapng* | 38,3 MB | 4 h |
| normálny.pcapng | 3,8 MB | 1h 30 min |

Tabuľka A.1: Zoznam vytvorených záznamov komunikácie

# Príloha B

# Spustenie aplikácie

**Operačný systém:**
Uživateľ: xkrajc17
Heslo: demo123

**Aplikácia:**
Uživateľ: demo@demo.sk
Heslo: demo123456

Spustenie

- sudo /opt/lampp/lampp restart
- cd Documents/monitoring/
- php artisan serve
- adresa stránky: 127.0.0.1:8000

Ak by nastala chyba:

- php artisan route:clear
- php artisan config:clear
- php artisan cachce:clear