



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**  
BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**  
FACULTY OF INFORMATION TECHNOLOGY

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**  
DEPARTMENT OF INFORMATION SYSTEMS

**GENEROVANIE PREVÁDZKY IOT SIETÍ A DETEKCIÁ  
BEZPEČNOSTNÝCH INCIDENTOV**

IOT TRAFFIC GENERATION AND DETECTION OF SECURITY INCIDENTS

**BAKALÁŘSKÁ PRÁCE**  
BACHELOR'S THESIS

**AUTOR PRÁCE**  
AUTHOR

**JÁN PRISTAŠ**

**VEDOUCÍ PRÁCE**  
SUPERVISOR

**Ing. PETR MATOUŠEK, Ph.D., M.A.**

**BRNO 2018**

Zadání bakalářské práce/21299/2017/xprist06

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav informačních systémů

Akademický rok 2017/2018

**Zadání bakalářské práce**

Řešitel: **Pristaš Ján**

Obor: Informační technologie

Téma: **Generování provozu IoT sítí a detekce bezpečnostních incidentů**  
**IoT Traffic Generation and Detection of Security Incidents**

Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se s komunikací pro průmyslové IoT sítě a SCADA systémy. Zaměřte se na protokoly IEC 104, DLMS a MQTT
2. Na základě doporučení od vedoucího prozkoumejte dostupné emulátory a generátory provozu pro IoT sítě. Vyhodnoťte použitelnost těchto aplikací.
3. Prostudujte známé útoky na průmyslové IoT sítě a SCADA systémy. Zaměřte se na možnosti emulace těchto útoků.
4. Výstupem práce bude datová sada ve formátu PCAP, která bude obsahovat typickou komunikaci IoT sítí a komunikaci obsahující známé útoky.
5. Popište možnosti detekce útoků a způsob ochrany IoT sítí vůči těmto útokům.

Literatura:

- Gordon Clarke, Deon Reinders: Practical Modern SCADA Protocols, Elsevier, 2004.
- David Hanes, Gonzalo Salqueiro, Patrick Grossetete, Rob Barton, and Jereme Henry: IoT Fundamentals. Networking Technologies, Protocol and Use Cases for the Internet of Things, Cisco Press, 2017.
- Dokumentace ke konkrétním emulátorům.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese

<http://www.fit.vutbr.cz/info/szz/>

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Matoušek Petr, Ing., Ph.D., M.A., UIFS FIT VUT**

Datum zadání: 1. listopadu 2017

Datum odevzdání: 16. května 2018

**VYSOKÉ UČENÍ TEHNICKÉ V BRNĚ**

Fakulta informačních technologií

Ústav informačních systémů

612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář  
vedoucí ústavu

## **Abstrakt**

Hlavným cieľom mojej bakalárskej práce je navrhnúť a vytvoriť systém generujúci prevádzku IoT sietí, vytváranie bezpečnostných incidentov na túto siet a ich detekovanie. Prvá časť popisuje základné princípy IoT sietí, SCADA systémov a komunikačné protokoly IEC 60870-5-104 a DLMS/COSEM. Následne sú popísané priemyselné nástroje na emuláciu prevádzky SCADA systémov využívajúc skúmané protokoly. V ďalšej časti je popísaná bezpečnosť týchto systémov, spolu s rizikami, ktorým čelia. Na základe týchto informácií bol navrhnutý a implementovaný nástroj umožňujúci simulať rozne typy útokov na SCADA systémy a sledovať ich reakcie. Posledná časť práce je venovaná možnostiam detektie rôznych typov útokov a spôsobu ochrany sietí pred nimi.

## **Abstract**

The main purpose of my bachelor thesis is to design and create operation generating system of IoT networks, creating security incidents and their detection. First part describes basic principles of IoT networks, SCADA systems and communication protocols IEC 60870-5-104 and DLMS/COSEM. Subsequently there are described industrial tools emulating operation of SCADA systems using investigated protocols. The next section describes the security of these systems, along with the risks they face. Based on this information, a tool was designed and implemented to simulate various types of attacks on SCADA systems and track their reactions. The last part of the work is devoted to the possibilities of detecting different types of attacks and how to protect the networks from them.

## **Kľúčové slová**

IoT - Internet vecí, SCADA, IEC 104, DLMS/COSEM

## **Keywords**

IoT - Internet of Things, SCADA, IEC 104, DLMS/COSEM

## **Citácia**

PRISTAŠ, Ján. *Generovanie prevádzky IoT sietí a detekcia bezpečnostných incidentov*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Petr Matoušek, Ph.D., M.A.

# **Generovanie prevádzky IoT sietí a detekcia bezpečnostných incidentov**

## **Prehlásenie**

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Petra Matouška Ph.D., M.A. Uviedol som všetky literárne pramene a publikácie z ktorých som čerpal.

.....  
Ján Pristaš  
13. mája 2018

## **Poděkování**

Rád by som sa poděkoval pánovi Ing. Petrovi Matouškovi Ph.D., M.A., za odborné rady a vedenie pri vypracovaní tejto bakalárskej práce.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>3</b>
1.1	Motivácia . . . . .	3
1.2	Postup práce . . . . .	3
1.3	Rozdelenie práce . . . . .	3
1.4	Prínos práce . . . . .	4
<b>2</b>	<b>SCADA systémy</b>	<b>5</b>
2.1	Základný prehľad IoT . . . . .	5
2.2	SCADA systémy . . . . .	6
2.2.1	Komunikačné protokoly . . . . .	8
<b>3</b>	<b>Emulátory prevádzky SCADA systémov</b>	<b>13</b>
3.1	IEC 60870-5-104 - DLMS/COSEM . . . . .	13
3.1.1	DMLSDirector . . . . .	13
3.1.2	XmlDemo . . . . .	16
3.2	IEC 60870-5-104 . . . . .	18
3.2.1	WinPP104 . . . . .	18
3.2.2	Knižnica OpenMUC - j60870 . . . . .	22
3.2.3	IEC 60870-5-104 Client/Server Simulator . . . . .	24
3.2.4	QTester 104 . . . . .	27
3.3	Zhrnutie . . . . .	29
3.3.1	DLMS/COSEM . . . . .	29
3.3.2	IEC 60870-5-104 . . . . .	30
<b>4</b>	<b>Bezpečnosť SCADA systémov</b>	<b>32</b>
4.1	Bezpečnostné riziká . . . . .	33
4.1.1	Útoky cez SCADA kanále . . . . .	34
4.1.2	Útoky cez koncové zariadenia . . . . .	35
4.2	Typy útokov . . . . .	35
4.2.1	Man-in-the-middle útok . . . . .	36
4.2.2	Denial-of-Service útok . . . . .	36
4.3	Zaznamenané útoky . . . . .	37
4.3.1	Vlakový signalizačný systém CSX (2003) . . . . .	37
4.3.2	Výpadok prúdu v elektrárni v Northeast (2003) . . . . .	37
4.3.3	Stuxnet červ (2010) . . . . .	38
4.3.4	Útok na ukrajinskú elektráreň (2016) . . . . .	38
4.4	Zhrnutie . . . . .	41

<b>5 Implementácia</b>	<b>42</b>
5.1 Návrh prostredia . . . . .	42
5.2 Implementácia simulačného prostredia . . . . .	42
5.2.1 Popis behu programu . . . . .	44
5.2.2 Návod na použitie . . . . .	44
5.3 Testovanie . . . . .	45
5.3.1 DLMS/COSEM . . . . .	47
5.3.2 IEC 104 . . . . .	50
<b>6 Detekcia útokov na SCADA systémy</b>	<b>53</b>
6.1 Kontrola a riadenie zraniteľnosti systému . . . . .	54
6.2 Detekcia zneužitia . . . . .	54
6.3 Monitorovanie prístupu . . . . .	55
6.4 Kontrola logovacích záznamov . . . . .	56
<b>7 Záver</b>	<b>58</b>
<b>Literatúra</b>	<b>59</b>
<b>A Inštalácia monitorovacích nástrojov</b>	<b>60</b>
A.1 DMLS Director . . . . .	60
A.2 XmlDemo . . . . .	61
A.3 WinPP104 . . . . .	61
A.4 Knižnica OpenMUC - j60870 . . . . .	62
A.5 QTestler 104 . . . . .	62
A.6 IEC 60870-5-104 Client/Server Simulator . . . . .	62
<b>B Inštalácia C++ knižnice pre protokol DLMS</b>	<b>64</b>

# Kapitola 1

## Úvod

### 1.1 Motivácia

Priemyslené IoT siete, tzv. SCADA systémy sú dnes veľmi využívané mnohými spoločnosťami. V Českej republike sú SCADA systémy využívané napríklad spoločnosťami ako RWE, E.ON alebo skupinou ČEZ. Útoky na takéto veľké komplexy môžu mať obrovský dopad nielen na spoločnosť ako takú, ale aj na bežných užívateľov. Napríklad útok na elektrárenský komplex môže spôsobiť výpadok prúdu pre tisíce domácností. Preto si myslím, že je v dnešnej dobe naozaj dôležité, aby boli tieto systémy dostatočne chránené, nakoľko sú veľmi často súčasťou kritickej infraštruktúry a majú veľký dopad na spoločnosť. Vid. napríklad zákon č. 181/2014 Sb. o kybernetických útokoch, ktorý radí energetiku do kritickej infraštruktúry<sup>1</sup>.

### 1.2 Postup práce

Hlavným cieľom tejto bakalárskej práce je naštudovať si komunikáciu pre priemyselné siete IoT a systémy SCADA so zameraním na komunikačné protokoly IEC 60870-5-104 a DLMS/COSEM. Následne je cieľom preskúmať známe útoky na tieto siete a zamerať sa na možnosti emulácie takýchto útokov. Výstupom práce je datová sada súborov vo formáte .pcap, ktorá obsahuje typickú komunikáciu IoT sietí a komunikáciu obsahujúcu známe útoky. V poslednej časti práce sú popísané možnosti detekcie jednotlivých útokov spolu s možnosťami ochrany sietí pred nim.

### 1.3 Rozdelenie práce

V druhej kapitole je uvedený teoretický základ sietí IoT a systémov SCADA. Kapitola popisuje princíp fungovania sietí, ich výhody a nevýhody. Ďalej nasleduje popis komunikácie v sietach so zameraním na protokoly IEC 60870-5-104 a DLMS/COSEM. V tretej kapitole sú podrobne popísané existujúce nástroje na simulovanie a emuláciu prevádzky SCADA systémov. Programy komunikujú pomocou vyššie spomínaných protokolov. Pri každom sú uvedené stručné informácie o výrobcovi, popis nástroja, potrebné zariadenia pre správnu funkcionality, typ topológie, ktorú umožňujú vytvoriť a prípadová štúdia, ktorá popisuje ako bol daný nástroj testovaný. Bola vytvorená sada .pcap súborov, ktoré sú uložené v

---

<sup>1</sup>Zákon o kybernetických útokoch <https://www.zakonyprolidi.cz/cs/2014-181> [Online: Máj 2018]

Github repozitári. Na konci kapitoly je vzájomné porovnanie jednotlivých programov a vyhodnotenie ich použiteľnosti pre účely tejto práce.

Štvrtá kapitola popisuje bezpečnosť v priemyselných sietach IoT a je v nej uvedených niekoľko zaznamenaných útokov na takéto siete z dôvodu ilustrácie možného dopadu na okolie a bežných občanov.

Piata kapitola je rozdelená na dve časti. V prvej, je uvedený podrobnyj popis vytvárania emulačného prostredia na testovanie rôznych typov útokov. Druhá časť je zameraná na popis testovania reakcií systému na rôzne druhy narušenia a na sledovanie reakcií systému na ne. V poslednej kapitole sú preberané rôzne postupy monitorovania sietí a detekcie jednotlivých typov útokov. Útoky sú rozdelené do niekolkých kategórií na základe spôsobu prieniku do systému a jeho ohrozenia.

## 1.4 Prínos práce

Práca obsahuje popis a testovanie dostupných emulačných nástrojov priemyselnej komunikácie spolu s popisom komunikačných protokolov využívaných v systémoch SCADA. Z poznatkov o protokoloch bol vytvorený nástroj umožňujúci emulovať rôzne typy útokov na priemyselné siete. Pomocou môjho nástroja a dostupných emulačných nástrojov bolo vytvorené testovacie prostredie na testovanie jednotlivých typov útokov a na sledovanie reakcií systému na ne. Výstupom práce je vygenerovaný .pcap dataset obsahujúci rôzne typy útokov a narušení. Jednotlivé .pcap súbory spolu s emulačným nástrojom sú k dispozícii v github repozitáry<sup>2</sup>.

Výsledky tejto práce môžu taktiež pomôcť mnohým spoločnostiam predísť nežiadúcim ohrozeniam ich systémov a efektívne sa brániť väčšine typov známych útokov na ich siete. Výstupy práce, konkrétnie datová sada .pcap súborov, je súčasťou projektu IRONSTONE<sup>3</sup> vo výzkumnej skupine NES@FIT.

---

<sup>2</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

<sup>3</sup>Projekt Ironstone <http://www.fit.vutbr.cz/units/UIFS/grants/index.php.cs?id=1101>

# Kapitola 2

## SCADA systémy

V tejto kapitole je uvedený teoretický základ o internete vecí (IoT - Internet of Things), jeho výhody a nevýhody. Podrobnejšie je popísané využitie v priemysle, tzv. SCADA systémy a komunikačné protokoly pomocou ktorých jednotlivé zariadenia v sieti komunikujú.

### 2.1 Základný prehľad IoT

Internet vecí je dnes veľmi aktuálna téma. Ide o siet zariadení (príručné zariadenia, vozidlá, domáce spotrebiče, a pod.) ktoré sú vybavené elektronikou, rôznymi senzormi a sieťovou konektivitou, ktorá umožňuje vzájomné prepojenie a výmenu dát medzi týmito zariadeniami. Ide o komunikáciu tzv. inteligentných (smart) zariadení cez internet. Prepojenie je väčšinou bezdrôtové, cez Wi-Fi, alebo Bluetooth. Internet vecí umožňuje vzdialené snímanie, alebo riadenie v rámci celej sietovej infraštruktúry. Umožňuje to tak užívateľom dosiahnuť väčšiu automatizáciu, lepšiu analýzu a integráciu v rámci celého systému. Internet vecí môže mať spotrebiteľské využitie, napr. v chytrej domácnosti (smart home), kde sú jednotlivé domáce spotrebiče vzájomne prepojené cez Wi-Fi a užívateľ môže centrálnie ovládať všetky svoje zariadenia. Internet vecí má ale hlavné podnikové využitie, prevažne z dôvodov zvýšenia automatizácie a úspory nákladov na prevádzku[2].

#### Výhody internetu vecí sú:

- Čas - jednotlivé zariadenia, snímače a riadiace prvky nám šetria veľké množstvo času. Údaje je možné sledovať a riadiť na diaľku, bez nutnosti neustáleho cestovania na konkrétné miesta a ručného zapisovania údajov
- Peniaze - možnosť vzdialého prístupu a riadenia tak tiež šetrí veľké množstvo finančí, je potrebný menší počet zamestnancov
- Zariadenia nám poskytujú presnejšie analýzy a dát, bez nutnosti zásahu ľudského faktoru
- Sledovanie - automatické sledovanie hodnôt, pripomienok a pod.

#### Nevýhody sú:

- Bezpečnosť - bezpečnosť je veľký problém v internete vecí. Dáta v zariadeniach musia byť dôkladne zašifrované. Zariadenia totiž môžu obsahovať osobné informácie o užívateľoch, ktoré môžu byť zneužité. Jednotlivé siete sú tak tiež náchylné k rôznym

útokom, prípadne môžu byť napadnuté a použité pri útoku samotnom, napr. rôzne DDoS útoky

- Nekompatibilita - jednotlivé zariadenia nemusia spolu navzájom komunikovať, napr. ak každé zariadenie používa komunikačné protokoly vytvorené svojím výrobcom[2]

## 2.2 SCADA systémy

Vo svojej práci sa budem bližšie venovať priemyselnému využitiu internetu vecí, konkrétnie tzv. SCADA systémom. SCADA (Supervisory Control And Data Acquisition) systém je typ architektúry riadiaceho systému využívajúceho počítače, sieťové prepojenie a rôzne vzdialene riadené objekty. SCADA systémy sú využívané prevažne v rôznych výrobných (výrobné linky, baliace linky, skladové systémy) a energetických (elektrárne, teplárne, výmenníkové stanice) závodoch, ale aj v technológiach budov (vzduchotechnika, zabezpečenie, dochádzkové systémy) a v ekológii (emisný monitoring, čističky odpadných vôd). V Českej republike sú SCADA systémy využívané napríklad spoločnosťami ako RWE, E.ON alebo skupinou ČEZ.

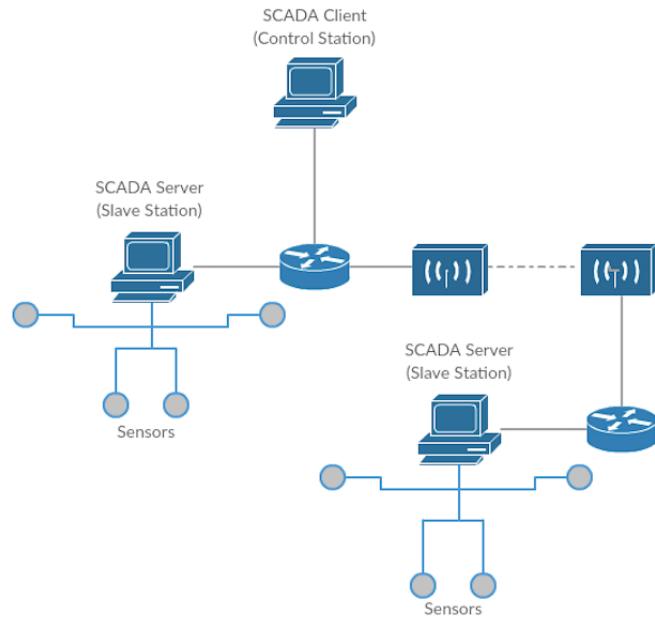
SCADA systémy sa radia medzi tzv. OT (Operational Technology) siete. Na rozdiel od bežných IT sietí, ktoré slúžia hlavne na prepojenie zariadení cez internet, OT siete slúžia na monitorovanie a riadenie zariadení v rôznych priemyselných odvetviach. Čo sa bezpečnosti týka, IT siete sa zameriavajú hlavne na dôkladnú autentizáciu jednotlivých zariadení, zatiaľ čo v OT sietach ide do popredia fyzická ochrana systému. OT siete sú väčšinou izolované od bežnej TCP/IP prevádzky a využívajú proprietárne protokoly. Sú veľmi často súčasťou kritickej infraštruktúry a preto môžu mať útoky na ne veľký dopad na biznis, energetické dodávky (elektrina, voda, plyn) ap.

SCADA systémy sa skladajú z dvoch častí: strany klienta a strany servera. Strana klienta zastrešuje riadiacu centrálu, ktorá vzdialene monitoruje a riadi pripojené prvky. Strana servera je vzdialená stanica, ktorá obsahuje pripojené rôzne inteligentné meracie zariadenia (snímače, teplomery, elektromery ap.), ktoré sú vzdialene čítané alebo riadené (nastavované) stranou klienta. V SCADA systémoch sú úlohy klienta a servera presne naopak ako je štandardne používaný model komunikácie klient-server. Na obrázku 2.1 je ukážka jednoduchej topológie SCADA systému.

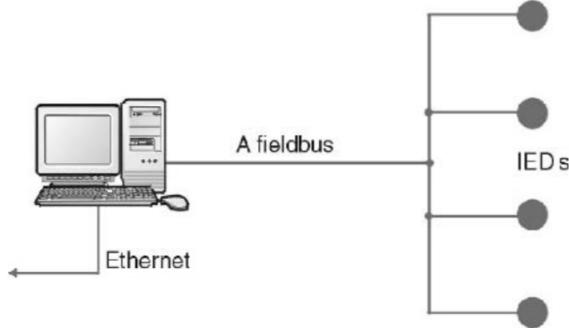
Systémy slúžia na vzdialenú kontrolu a zber údajov. Využívajú sa v nich tzv. intelligentné merače - IED (Intelligent Electronic Devices). Jednotlivé zariadenia sú prepojené a vzájomne komunikujú cez aplikačnú zbernicu (fieldbus). Taktiež sú pripojené k riadiacemu centru, ktoré ich vzdialene kontroluje a ovláda. Ukážka prepojenia je na obrázku 2.2. Vzdialenosť riadiacich staníc (master) a koncových zariadení (slave) môže byť od niekoľkých metrov po tisíce kilometrov. Každé zariadenie môže obsahovať niekoľko rôznych senzorov, analógový vstup/výstup senzora, systém na komunikáciu s riadiacim počítačom a zariadeniami, a programovú pamäť[1].

### Výhody SCADA systémov sú:

- Je potrebná minimálna kabeláž - oproti prvotným verziám SCADA systémov kedy sa nevyužívali IED zariadenia a každý senzor musel byť samostatne pripojený ku kontrolným staniciam
- Prijímané dátá môžu obsahovať veľkú škálu informácií - seriové čísla, čísla modelu, informácie o inštalácii zariadenia (kedy bolo nainštalované a kým)
- Zariadenia sú jednoduché na inštaláciu, prípadne výmenu



Obr. 2.1: Topológia SCADA systému



Obr. 2.2: Zapojenie zariadení v SCADA[1]

- Celý systém potrebuje málo fyzického priestoru - jednotlivé koncové zariadenia sú relatívne malé

#### Nevýhody sú:

- Využívanie sofistikovanejších systémov je náročnejšie a vyžaduje lepšie vyškolený a kvalifikovaný personál
- Ceny senzorov sú vyššie
- Zariadenia sú závislé na komunikačnom systéme - je nutné aby podporovali rovnaký komunikačný protokol
- Postupný prechod systémov nad IP vrstvu znižuje ich bezpečnosť[1]

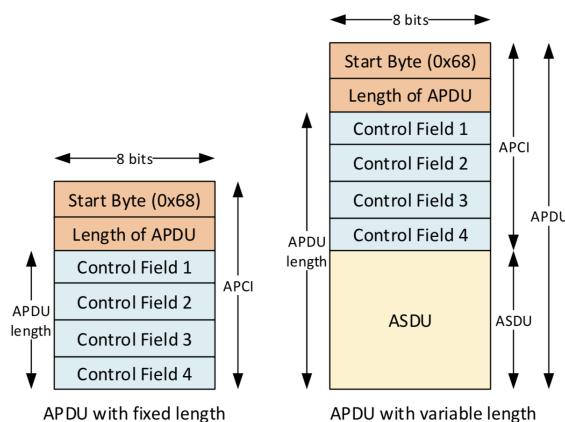
## 2.2.1 Komunikačné protokoly

Aby mohli medzi sebou koncové zariadenia a riadiaca stanica komunikovať, musia podporovať rovnaký spôsob komunikácie. Na to slúžia rôzne štandardizované protokoly, ktoré takúto komunikáciu popisujú. V tejto práci sa budem venovať dvom z nich. Konkrétnie to sú protokoly IEC 60870-5-104 a DLMS/COSEM.

### IEC 60870-5-104

Protokol IEC 60870-5-104 je časť protokolu IEC 60870-5. IEC 60870-5 je komunikačný protokol a je súčasťou protokolu IEC 60870, ktorý je určený pre systémy diaľkového riadenia. IEC 60870-5 špecifikuje prenosové protokoly pre diaľkové ovládanie zariadení, ktoré slúžia k prenosu dát a riadeniu geograficky vzdialených procesov. Pozostáva z niekoľkých častí, z ktorých nás najviac zaujíma IEC 60870-5-104. Protokol IEC 60870-5-104 špecifikuje sieťový prístup pre IEC 60870-5-101. Protokol IEC 60870-5-101 je spoločná norma pre základné úlohy diaľkového ovládania. Jeho cieľom je zaistiť vzájomnú spoluprácu medzi jednotlivými (kompatibilnými) zariadeniami pre diaľkové riadenie. Tzn. že IEC 60870-5-101 špecifikuje mechanizmy prenosu a IEC 60870-5-104 stanovuje ich použitie v bežných komunikačných sieťach[1][8].

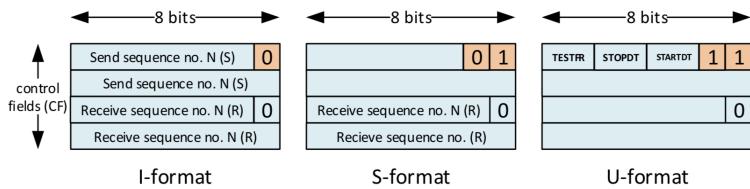
Protokol IEC 60870-5-104 je umiestnený na aplikačnej vrstve modelu ISO/OSI. Špecifikácia IEC 60870-5-104 kombinuje aplikačnú časť protokolu IEC 60870-5-101 a prenosové funkcie poskytované TCP/IP. Protokol poskytuje funkcie pre prenos dat aplikačným funkciám užívateľského procesu. Prenášaná správa je vo formáte APDU (Application Protocol Data Unit). Je to tzv. aplikačná jednotka, ktorá pozostáva z dvoch častí - APCI (Application Protocol Control Information) a ASDU (Application Service Data Unit). APCI je záhlavie aplikačnej jednotky, ktoré určuje jej dĺžku, typ, sekvenčné čísla a pod. ASDU sú samotné prenášané dátá, tj. príkazy posielané medzi klientom a serverom. APDU správa môže mať pevnú alebo variabilnú dĺžku. Správa s pevnou dĺžkou neobsahuje ASDU časť. Formát rámca je ukázaný na obrázku 2.3. Formát je určený poslednými dvoma bitmi prvého



Obr. 2.3: Formát APDU rámca[7]

kontrolného pola APCI časti (CF1). Štandard definuje tri typy rámcov, viď obrázok 2.4.

- I formát - je určený k bežnému prenosu dát medzi aplikačnými funkciami
- S formát - je používaný pre dohľad nad prebiehajúcou komunikáciou

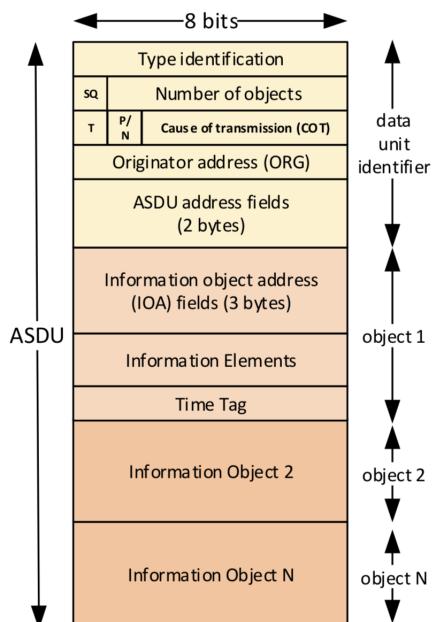


Obr. 2.4: Typy rámsov APDU[7]

- U formát - je určený k riadeniu samotnej komunikácie

APCI vždy začína "štart bytom"s hodnotou 0x68. Za ňou nasleduje 8-bitové APDU a štyri 8-bitové kontrolné polia (CF).

ASDU pozostáva z dvoch hlavných častí - identifikátor datovej jednotky (má vždy pevnú dĺžku šesť bytov) a dátá samotné, ktoré môžu pozostávať z jedného, alebo viacerých objektov. Identifikátor datovej jednotky definuje špecifický typ dát, poskytuje adresovanie na identifikáciu špecifickej identity dát a obsahuje dodatočné informácie ako napr. dôvod prenosu. Každé ASDU môže preniesť najviac 127 objektov. Na obrázku 2.5 je ukážka formátu ASDU[7].



Obr. 2.5: Formát ASDU[7]

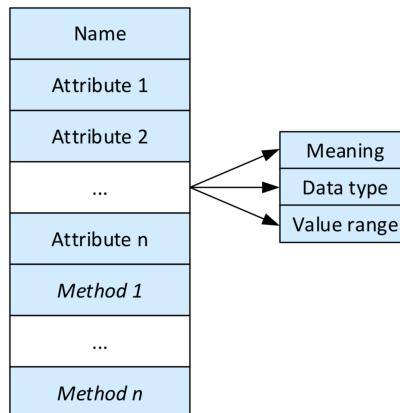
Riadenie prebieha pomocou niekolkých príkazov - TESTFR, STARTDT a STOPDT. STARTDT a STOPDT sú príkazy používané riadiacimi stanicami na riadenie datového prenosu z riadenej stanice. Ked započne komunikácia, prenos dát nie je ešte povolený. Riadiaca stanica musí povoliť datový prenos odoslaním príkazu STARTDT act (activate). Riadená stanica odpovie príkazom STARTDT con (confirm). Ak nie je príkaz STARTDT potvrdený, riadiaca stanica automaticky ukončí spojenie. Príkaz STARTDT posiela iba riadiaca stanica a je odoslaný iba raz, po prvotnej inicializácii spojenia. Po povolení prenosu dát prebieha ľubovoľná komunikácia medzi riadiacou a riadenou stanicou. Spojenie sa ukončuje príkazom STOPDT.

Komunikácia vždy pozostáva medzi riadiacim (master) a riadeným (slave) systémom. Riadiaci systém inicializuje spojenie, začína a ukončuje výmenu aplikačných dát. Ukončenie komunikácie môže iniciovať riadiaca, ale aj riadená stanica. Riadená stanica (server) načíva na port 2404 na príkazy od riadiacej stanice (klienta). Riadiaca stanica môže naraz komunikovať s viacerými riadenými stanicami[7][8].

## DLMS/COSEM

Protokol DLMS/COSEM je, podobne ako protokol IEC 60870, súbor štandardov pre výmenu údajov o spotrebe v energetike. Protokol pozostáva z niekoľkých častí, pričom každá špecifikuje určitú časť problematiky, ktorú protokol rieši. Prvá časť, DLMS (Device Language Message Specification) je špecifikácia aplikačnej vrstvynezávislá od nižších vrstiev, vytvorená na podporu prenosu zpráv do a z vzdialených (energetických) zariadení. Cieľom je poskytnúť interoperabilné prostredie pre výmenu dát. Špecifikácia podporuje funkcie na vzdialé čítanie a nastavovanie hodnôt pre všetky odvetvia energetiky (elektrárne, vodárne, teplárne, atp.). DLMS je používané na popis rozhrania tried jednotlivých objektov spolu s ich atribútmi. Špecifikácia je podobná protokolu IEC 62056, ten je ale zamenaný na meranie spotreby elektrickej energie na rozdiel od DLMS, ktorý je použiteľný na všetky energetické odvetvia. Druhá časť protokolu, COSEM (Companion Specification for Energy Metering) je rozhranie modelu komunikujúceho zariadenia na meranie energetickej spotreby, ktoré poskytuje pohľad na funkčnosť dostupnú cez komunikačné rozhranie. Poskytuje taktiež sémantiku na aplikáciu jednotlivých meraní. COSEM model je založený na objektovo-orientovanom prístupe. Instancia COSEM rozhrania sa nazýva **COSEM interface object**. Sada jednotlivých objektov v logických zariadeniach fyzického zariadenia modeluje funkcionality meracích zariadení rovnako ako je to možné vidieť v jeho komunikačnom rozhraní[6].

Model reprezentuje meracie zariadenie ako server využívaný aplikáciami klienta. Aplikácie slúžia na získavanie dát, poskytovanie riadiacich informácií a spuštanie funkcií objektu pomocou riadeného prístupu k jeho atribútom a špecifickým metódam jednotlivých typov objektov.

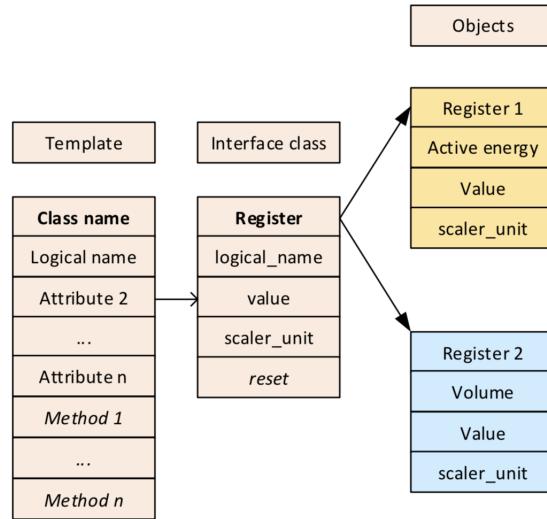


Obr. 2.6: COSEM objekt[6]

COSEM modeluje fyzické zariadenie ako súbor logických zariadení, pričom každé má svoj jednoznačný identifikátor. Identifikátor určuje typ (funkcionalitu) daného zariadenia. Každé zariadenie môže byť jednoznačne identifikované pomocou jeho logického mena (OBIS

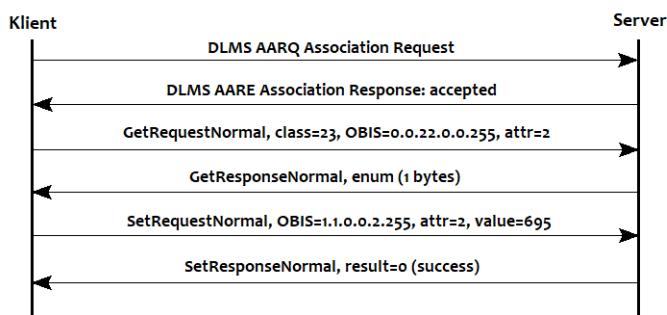
kódom). Ide o šesť 8-bitových čísl. Informácie obsiahnuté v jednotlivých logických zariadeniach sú modelované objektami rozhrania. Objekty rozhrania sú špecifické pre danú doménu merania spolu s ich atribútmi a metódami. V atribútoch sú usporiadane informácie, ktoré daný objekt uchováva. Vlastnosti objektu sú určené pomocou hodnôt atribútov. Každý atribút pozostáva z typu, hodnoty a určenia. Ukážka objektu je na obrázku 2.6.

Prvý atribút každého objektu je jeho logické meno, ktoré slúži na identifikáciu objektu. Objekty, ktoré majú rovnakú charakteristiku sú generalizované a identifikované pomocou ID ich triedy (class\_id). Vzťah medzi šablónami a rozhraniami tried je ukázaný na obrázku 2.7. Štandard protokolu definuje približne 70 rozhraní tried pričom každé je určené jeho menom, ID triedy a verziou.



Obr. 2.7: Šablóny a rozhrania tried[6]

Jednotlivé metódy umožňujú identifikáciu, vyhľadávanie a interpretáciu informácií uchovávaných v objektoch meracích zariadení[6][3].

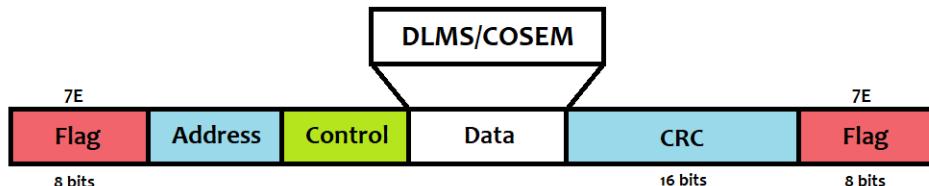


Obr. 2.8: Komunikácia protokolu DLMS/COSEM

Na obrázku 2.8 je ukážka komunikácie protokolu DLMS/COSEM medzi klientom a serverom. Ukážka obsahuje niekoľko bežných príkazov. Ako prvé nnačítanie informácií o jednotlivých pripojených meračoch (Association Request/Response), tento príkaz je vždy zasielaný ako prvý na začiatku komunikácie. Druhá je žiadosť o navrátenie hodnoty druhého atribútu objektu s OBIS kódom 0.0.22.0.0.255. Ide o hodnotu prenosovej rýchlosťi

(baudrate) objektu typu `IecHd1cSetup`. Nakoniec je zaslaný príkaz na nastavenie druhého atribútu objektu s OBIS kódom 1.1.0.0.2.255. Je to atribút nesúci údaje o spotrebe v objekte typu `Data`.

Dáta prenášané protokolom DLMS/COSEM sú väčšinou zapúzdroňe do tzv. HDLC rámca definovaného štandardom ISO/IEC 13239[6]. Štandard používa formát HDLC rámca typu 3, ukážka je na obrázku 2.9.



Obr. 2.9: HDLC rámec

Rámec neobsahuje informačné pole, kontrolnú sekvenciu hlavičky (HCS) iba kontrolnú sekvenciu rámca. Prvé štyri byty slúžia na identifikáciu HDLC rámca. Pre DLMS je to 1010 (0xA). Kontrolné pole (CTRL) označuje typ príkazu alebo odpovede a obsahuje patričné sekvenčné číslo. Ukážka kontrolného poľa je na obrázku 2.10.

HDLC Control Fields							
0	1	2	3	4	5	6	7
N(R)	P/F		N(S)		0		
N(R)	P/F		type	0	1		
type	P/F	type		1	1		

I-frame  
S-frame  
U-frame

Obr. 2.10: Kontrolné pole HDLC rámca[6]

Sú tri typy rámcov:

- I formát - informačný (information), prenáša užívateľské dátu zo sietovej vrstvy
- S formát - kontrolný (supervisory), je používaný na kontrolu datového toku a chýb. Neobsahuje informačné polia. Používajú sa tri typy kontrolného rámca:
  - 00: Pripravený na prijímanie, RR (Receive ready)
  - 01: Prijímanie, nie čítanie, RNR (Receive not read)
  - 10: Zamietnuté, REJ (Reject)
  - 11: Selektívne domietnutie, SREJ (Selective reject)
- U formát - nečíslovaný (unnumbered) je používaný na správu odkazov (nastavenie režimu, zotavenie) a môže byť tiež použitý na prenos užívateľských dát

Pre DLMS/COSEM je potrebné parsovanie rámcov na identifikáciu vonkajšieho zapúzdroenia. HDLC môže byť identifikované začiatočným a koncovým príznakom (flag), ktorý má hodnotu 0x7e[6].

## Kapitola 3

# Emulátory prevádzky SCADA systémov

V tejto kapitole uvediem základný prehľad a porovnanie bežne používaných monitorovacích a emulačných nástrojov pre protokoly IEC 60870-5-104 a IEC 60870-5-104. Programy, ktorým sa budem venovať, slúžia nielen na monitorovanie komunikácie reálnych IEC 60870-5-104 a DLMS/COSEM meracích zariadení, ale aj na ich emuláciu. Popis inštalácie jednotlivých programov je uvedený v prílohe A.

### 3.1 IEC 60870-5-104 - DLMS/COSEM

#### 3.1.1 DMLS Director

**Výrobca:** Firma GuruX Ltd. je fínska spoločnosť špecializujúca sa na protokol DLMS využívaný v inteligentných meračoch<sup>1</sup>. Firma poskytuje produkty pre automatické zaznamenávanie hodnôt z meracích zariadení a umožnuje tým vytvorenie vlastných AMR systémov (systémov pre automatickú správu meracích zariadení).

**Popis produktu:** DLMS Director je open source software navrhnutý na DLMS komunikáciu a smart metering s DLMS/COSEM zariadeniami.

**Protokoly:** Program DLMS Director pokrýva niekoľko komunikačných štandardov. Je zamenaný na čítanie údajov a rozposielanie príkazov. Jednotlivé protokoly sú:

- IEC 62056-21 Priama lokálna výmena dát
- IEC 62056-42 Služby a procedúry pre asynchronnú spojovú výmenu dát na fyzickej vrstve
- IEC 62056-46 Linková vrstva využívajúca HDLC protokol
- IEC 62056-47 COSEM transportná vrstva pre IPv4
- IEC 62056-53 COSEM aplikačná vrstva
- IEC 62056-61 OBIS systém na identifikáciu objektov
- IEC 62056-62 Interface

---

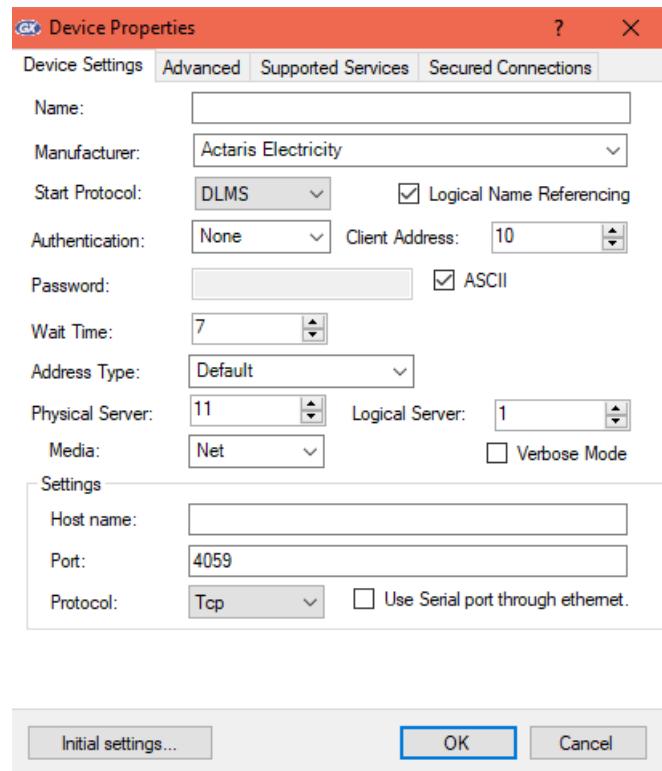
<sup>1</sup>GuruX Ltd. <http://www.gurux.fi> [Online: Október 2017]

**Zariadenia:** Program je schopný pracovať iba s reálnymi fyzickými zariadeniami, prípadne so zariadeniami emulovanými iným programom. Sám ich však emulovať nedokáže. So zariadeniami môže byť pripojený cez TCP/IP spojenie, terminál, alebo cez sériovú linku.

**Typ:** DLMS Director je open source software, ktorý je voľne dostupný na stránkach výrobcu. Zdrojové kódy sú taktiež voľne dotupné v pratričnom github repozitári<sup>2</sup>.

**Platforma:** Program je vytvorený pre OS Windows. Na iných platformách s ním nie je možné pracovať.

**Prípadová štúdia:** Program samotný slúži ako jedna klientská stanica umožňujúca pripojenie niekoľkých meracích zariadení. Pri pridávaní nového zariadenia je možnosť nastavenia jednotlivých parametrov spolu s preddefinovanými hodnotami. Ukážka pridania nového zariadenia je na obrázku 3.1. Odkaz na popis konfigurácie rôznych zariadení je medzi prí-



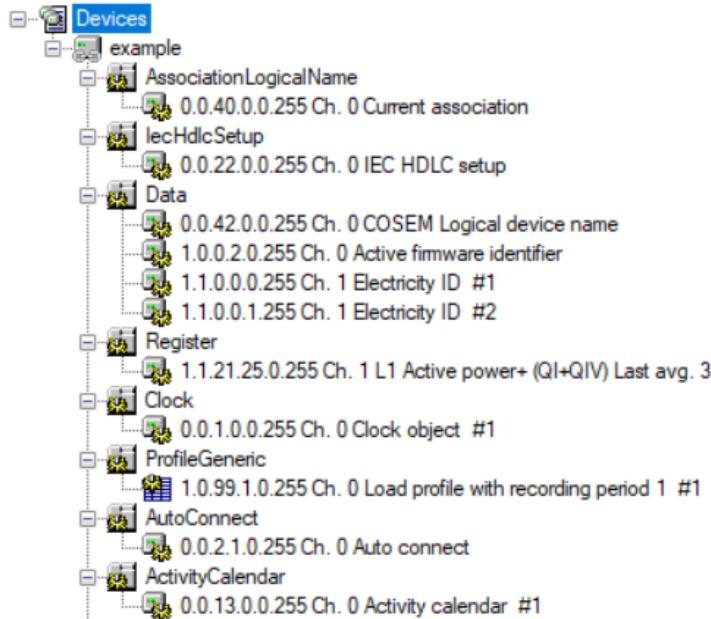
Obr. 3.1: Konfigurácia nového zariadenia v programe DLMS Director

lohami, časť A.1. Po pripojení k serveru zobrazí program zoznam všetkých objektov, ktoré server obsahuje. Objekty sú rozdelené do množín podľa ich typu, napr. Data, Register, Clock, MacAddress Setup. Ukážka je na obrázku 3.2. Je možné čítať hodnoty atribútov jednotlivých objektov, avšak iba všetkých hodnôt naraz. Nie je možné odoslať samostatný príkaz na prečítanie napríklad logického mena objektu. Je tiež možné prečítať hodnoty celej skupiny objektov v jednom príkaze, ale iba v rámci jedného typu. Napríklad hodnoty objektov typu Data.

Pri testovaní bola použitá knižnica v jazyku C++, ktorú poskytuje spoločnosť GuruX. Knižnica je voľne k dispozícii v Github repozitári<sup>3</sup> a umožňuje vytvorenie vlastných aplikácií pre DLMS. Spolu s knižnicou sú k dispozícii vzorové programy pre klienta a server.

<sup>2</sup>DLMS Director - zdrojové kódy <https://github.com/Gurux/GXDLMSSDirector> [Online: Október 2017]

<sup>3</sup>DLMS knižnica - Github <https://github.com/Gurux/Gurux.DLMS.cpp> [Online: Marec 2018]



Obr. 3.2: Zoznam pripojených objektov

Programy sú vytvorené pre OS Linux. Návod na inštaláciu knižnice je medzi prílohami B.  
Vzorový program servera vytvorí pri spustení štyri rôzne DLMS servery:

- port 4060 - DLMS server využívajúci krátke mená (Short Names - SN)
- port 4061 - DLMS server využívajúci logické mená (Logical Names - LN)
- port 4062 - DLMS server využívajúci krátke mená spolu s protokolom IEC 62056-47
- port 4063 - DLMS server využívajúci logické mená spolu s protokolom IEC 62056-47

Samotné testovanie pozostávalo vo vytvorení spojenia medzi programom DLMS Director a vzorovým programom pre server. Program bol ale čiastočne upravený aby bol užívateľsky o niečo prívetivejší a aby podporoval väčšiu škálu možností. Po spustení sa vytvorí iba jeden server na porte 4060 využívajúci logické mena. Bola pridaná možnosť hodnoty vzdialene meniť, nie iba čítať a bola pridaná možnosť využívať autentizáciu heslom typu low. Zmenený program bol umiestnený na github repozitár<sup>4</sup>. Na jeho úspešnú kompliaciu a spustenie je potrebné mať nainštalovanú vyššie spomínanú knižnicu.

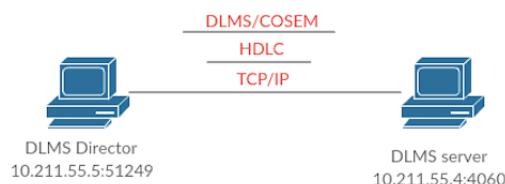
Vytvorenie spojenia a testovanie pozostávalo z niekoľkých krokov:

1. Bol spustený DLMS Director a program servera. Server nebolo nutné nijako ďalej konfigurovať. Bolo možné sa s ním priamo spojiť.
2. V DLMS Directore bolo potrebné nakonfigurovať server s ktorým sa ide program spojiť:
  - (a) File → Add Device
  - (b) Zvoliť meno pre server

<sup>4</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

- (c) Zvoliť výrobcu → Gurux,
  - (d) Zvoliť autentizáciu → Low a heslo → password (nastavené v programe servera)
  - (e) Nastaviť IP adresu servera a port 4060
3. Po úspešnom nakonfigurovaní bolo možné vytvoriť spojenie so serverom. Spojenie inicializuje program DLMS Director. V ľavom paneli bolo potrebné rozkliknúť zoznam Devices a kliknúť pravým tlačítkom na server, následne na Connect.
  4. Po vytvorení spojenia sa zobrazil zoznam pripojených zariadení, z ktorých bolo možné čítať a zapisovať hodnoty, príkazy Read a Write.

Testovaciu topológiu je možné vidieť na obrázku 3.3. Testovací server obsahoval nie-



Obr. 3.3: Ukážka testovacej topológie pre program DLMS Director

koľko rôznych objektov. Testovanie spočívalo v dotazovaní sa na hodnoty atribútov týchto objektov. Objekty boli napríklad typu Data, Clock, SapAssignment, ActivityCalendar. Komunikácia bola zachytená pomocou nástroja Wireshark a bol vytvorený súbor DLMS-Director.pcap uložený v github repozitári.<sup>5</sup>

**Zhrnutie:** DLMS Director je dobre spracovaný program určený hlavne na čítanie údajov a riadenie meracích zariadení. Výhodou je, že je typu open source a možno s ním zdarma pracovať. Nevýhodou je chýbajúca možnosť emulovania koncových zariadení a tým aj prevádzky samotnej.

### 3.1.2 XmlDemo

**Výrobca:** iCube je švajčiarska softwarová firma zaoberajúca sa, podobne ako vyššie spomínaná firma GuruX, vývojom softwaru pre komunikačný protokol DLMS<sup>6</sup>. Firma ponúka DLMS vývojové komunikačné balíky pre jazyky C++, C# a Java, zjednodušujúce implementáciu DLMS/COSEM klientských aplikácií. Vo variante C++ sú jednotlivé komponenty implementované ako DLL, v C# a Java variantách sú implementované ako triedy. Jednotlivé balíky ale nebudem bližšie rozoberať.

**Popis produktu:** XmlDemo je jednoduchý DLMS "čítač" hodnôt uložených v DLMS/COSEM meracích zariadeniach. Implementácia využíva vyššie spomenuté DLMS balíky, konkrétnie komponenty xmlpdu na výstup vo formáte xml, ezhdlc pri využití HDLC spojenia a wrapper pri spojení cez TCP/IP. Program XmlDemo je v úlohe klienta, ktorý zasiela žiadosti serveru (zariadeniam) a prijíma odpovede. Napríklad, klient pošle žiadosť read the object 1.0.1.8.0.255 a server odpovie 1789.8 kWh.

**Protokoly:** Program je vytvorený na komunikáciu so zariadeniami pomocou protokolu DLMS/COSEM.

<sup>5</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

<sup>6</sup>iCube <https://www.icube.ch/index.html> [Online: Október 2017]

**Zariadenia:** Program je schopný pracovať iba s reálnymi fyzickými zariadeniami, prípadne so zariadeniami emulovanými iným programom. Sám ich emulovať nedokáže. Podporuje komunikáciu medzi zariadeniami a aplikáciou cez TCP/IP a HDLC.

**Typ:** XmlDemo je zdarma poskytovaný program spoločnosťou iCube, nakoľko sa jedná o ukážku programu vytvoreného vo vyššie spomínaných vývojových balíkoch.

**Platforma:** Obdobne ako predchádzajúci program je aj XmlDemo prispôsobený iba na OS Windows. Na obrázku 3.4 je ukážka užívateľského rozhrania. Spôsob prijímania správ a logovací záznam vo formáte xml je na obrázku 3.5.

Logical Name	ClassId	Description
00002A0000FF	1	Ch. 0 COSEM Logical device name
0100020806FF	3	Ch. 0 Sum Li Active power-(Qll+Qlll) Time integral 1 Rate 6 (...)
010007060065	4	Ch. 0 Sum Li Reactive power Qlll Max. 1 Rate 0 (0 is total) 1. I...
01000F0400FF	5	Ch. 0 Sum Li Active power (abs(Ql+Qlv)+abs(Qll+Qlll)) Curre...
00000E0001FF	6	Ch. 0 Register activation #2
0003180300FF	7	Ch. 3 M-Bus profile generic #1
0000010000FF	8	Ch. 0 Clock object #1
00000A0067FF	9	Ch. 0 Set output signals
00000C0000FF	10	Ch. 0 Schedule #1
00000B0000FF	11	Ch. 0 Special days table #1

Obr. 3.4: Užívateľské rozhranie XmlDemo

```

1'19.332 +0      0001001000010013D960000001000101000100002A0000FF020
1'19.393 +60     0001000100100015DA600000010000010908494355303030303
RESPONSE:
<AccessResponse>
  <LongInvokeIdAndPriority Value="60000001" />
  <DateTime Value="" />
  <AccessResponseBody>
    <AccessResponseListOfData Qty="0001" >
      <OctetString Value="\ICU00000" />
    </AccessResponseListOfData>
    <AccessResponseSpecification Qty="0001" >
      <AccessResponseGet>
        <Result Value="Success" />
      </AccessResponseGet>
    </AccessResponseSpecification>
  </AccessResponseBody>
</AccessResponse>

```

Obr. 3.5: Užívateľské rozhranie XmlDemo

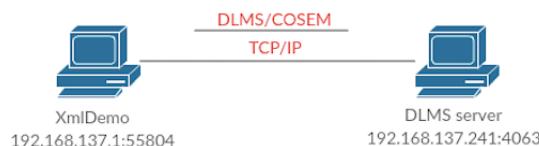
**Prípadová štúdia:** Čo sa topológie týka, program funguje ako vyššie popísaný DLMS Director. Program samotný slúži ako klientská stanica, ku ktorej je možné pripojiť jednotlivé monitorovacie zariadenia.

Pri testovaní bol použitý vzorový program pre server komunikujúci na porte 4063 využívajúc spojenie cez TcpUdp. Program XmlDemo súčasne umožňuje komunikáciu so serverom cez HDLC, avšak iba ak sú zariadenia prepojené cez sériovú linku. To som však nemal pri testovaní možnosť vyskúšať, nakoľko som nemal k dispozícii reálne fyzické zariadenie.

Vytvorenie spojenia medzi klientom a serverom opäť pozostávalo z niekolkých krokov:

1. Bol spustený program XmlDemo a pôvodný vzorový program servera, neobsahujúci nijaké zmeny.
2. Bolo potrebné nakonfigurovať program XmlDemo:
  - (a) Show → Settings
  - (b) Select TCP profile
  - (c) Nastaviť IP adresu servera a port 4063
  - (d) Nastaviť referenčný model → LN
  - (e) Nastaviť timeout, obe hodnoty Connect a Response na prijateľné hodnoty v milisekundách, v našom prípade 1000
  - (f) Nastaviť adresy pre klienta aj server, v našom prípade obe na 0
3. Po nakonfigurovaní bolo možné vytvoriť spojenie. Do → Connect a Do → Read object-list.

Testovacia topológia je obdobná ako pri testovaní programu DLMS Director. Je možné ju vidieť na obrázku 3.6. Po vytvorení spojenia medzi klientom a serverom, program (Xml-



Obr. 3.6: Ukážka testovacej topológie pre program XmlDemo

Demo) automaticky zobrazí zoznam všetkých pripojených objektov. Objekty sú taktiež uložené do nového .txt súboru. Zoznam je zobrazený vo formáte - logické meno | id triedy | popis. Zápis objektu vyzerá napríklad → 0000010000FF | 8 | Ch. 0 Clock object #1. Následne je možné z jednotlivých objektov čítať hodnoty atribútov. Je možné čítať hodnoty iba vybraných atribútov, na rozdiel od programu DLMS Director, ktorý umožňuje iba čítanie všetkých hodnôt daného objektu naraz. Po kliknutí na jednotlivé objekty sa zobrazí zoznam ich atribútov. Po kliknutí na jednotlivé atribúty sa prečíta ich hodnota. Testovanie prebiehalo obdobne ako pri programe DLMS Director. Komunikácia bola zachytená nástrojom Wireshark a bol vytvorený súbor XmlDemo.pcap uložený v github repozitári.<sup>7</sup> **Zhrnutie:** Čo sa týka samotnej funkcionality, poskytuje program obdobné funkcie a možnosti ako program DLMS Director, avšak hlavnou nevýhodou je absencia komunikácie pomocou HDLC cez IP vrstvu. Je to možné iba na fyzickej vrstve. Na testovacie účely je preto vhodnejší program DLMS Director. XmlDemo taktiež ponúka o niečo horšie GUI, v ktorom sa nepracuje tak dobre a intuitívne ako v DLMS Directore.

## 3.2 IEC 60870-5-104

### 3.2.1 WinPP104

**Výrobca:** Firma Berthold Boeser Ingenieurbüro je nemecká spoločnosť, ktorá vyvíja a distribuuje testovacie programy pre protokoly diaľkového riadenia<sup>8</sup>. Jednotlivé protokoly sú:

<sup>7</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

<sup>8</sup>Berthold Boeser Ingenieurbüro <http://www.ppfink.de/> [Online: Október 2017]

- IEC: 60870-5-101, 60870-5-102, 60870-5-103, 60870-5-104
- AEG: GEADAT 81, GEATRANS F202, F203, F220, F202KC, SEAB 1-F/N, PMF234C
- BBC: ZM20, I21
- Landis&Gyr: TELEGYR 800
- Mauell: ME 8008, 8018 PDM
- SAT: 1703 PCM
- SEL: ZPC3600/SPC17
- Siemens: SINAUT ST1, 4-FW, SINAUT 8-FW DPDM, SINAUT 8-FW PCM, FW 517/535/537, EFD 400, SAS/Z70

**Popis produktu:** WinPP104 je testovací a emulačný nástroj pracujúci nad protokolom IEC 60870-5-104. Okrem bežnej komunikácie tiež umožňuje odosielanie a prijímanie správ iba simuloval a tým sledovať rôzne faktory ako napr. oneskorenie príkazu. Výsledný záznam komunikácie sa dá exportovať do .csv súboru. Program rovnako umožňuje záznamy z .csv súborov aj nahrávať a následne ich spracovať.

**Protokoly:** Program je uspôsobený na komunikáciu výhradne nad protokolom IEC 60870-5-104.

**Zariadenia:** WinPP104 umožňuje sledovať reálne fyzické meracie zariadenia, ale aj vytvorenie vlastného klienta alebo servera pre simulačné účely. Prepojenie medzi zariadeniami je možné cez LAN alebo TCP/IP.

The screenshot shows the WinPP104 application window. At the top is a menu bar with File, Mode, Send, View, Parameterize, Filter, Help, and several status indicators. Below the menu is a table of network connections:

	Received	Error	Transmitted	Error	Status	IP Partner	Cl.,Se-Port	Function
Rec/Tr 1	2	0	2	0	Connected	127.0.0.1	55250,2404	Master = TCP client
Rec/Tr 2	2	0	2	0	Connected	127.0.0.1	55250,2404	Substation = TCP :

Below the table is a large text area displaying a log of online messages:

```

Online Messages, logical, with time, with link
9      R1 12:51:03,794 d=0,001s
IP:Port : 127.0.0.1:55250      <    127.0.0.1: 2404
Ctrl   : StartDT: con
10     T1 12:51:07,538 d=3,744s
IP:Port : 127.0.0.1:55250      >    127.0.0.1: 2404
Ctrl   : s:0      r:0
Type   : Step position information w.date=32
Cause   : spontaneous=3 ori=0
Station : 0- 0
Object  : 0- 0 1 29/04/2018 12:51:07,537
11      R2 12:51:07,539 d=0,001s
IP:Port : 127.0.0.1:55250      >    127.0.0.1: 2404
Ctrl   : s:0      r:0
Type   : Step position information w.date=32
Cause   : spontaneous=3 ori=0
Station : 0- 0
Object  : 0- 0 1 29/04/2018 12:51:07,537
12      T2 12:51:17,416 d=9,877s
IP:Port : 127.0.0.1:55250      <    127.0.0.1: 2404
Ctrl   : r:1
13      R1 12:51:17,417 d=0,001s
IP:Port : 127.0.0.1:55250      <    127.0.0.1: 2404
Ctrl   : r:1

```

At the bottom of the window are tabs for Online, Online messages, Log filter: Off, Output filter: Off, Log: Log.lg4, and Text: BspText4.csv.

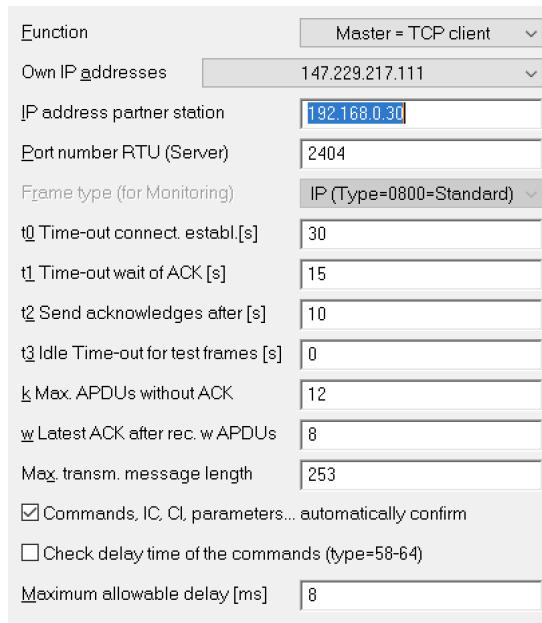
Obr. 3.7: Užívateľské rozhranie WinPP104

**Typ:** Výrobca poskytuje zdarma demo verziu, ktorá má však určité obmedzenia. Pri každom spustení aplikácie je nutné si celý systém nanovo nastavovať, naše predošlé nastavenia

nebudú uložené. Navyše je možné priať a odoslať iba 20 správ, čo striktne zamedzí dlhodobému monitorovaniu systému.

**Platforma:** Program je vytvorený na platformu OS Windows. Na obrázku 3.7 je ukážka užívateľského rozhrania programu WinPP104, konkrétnie ide o ukážku pripojenia zariadení cez TCP/IP a o ich monitorovanie.

**Prípadová štúdia:** WinPP104 umožňuje v rámci jedného spustenia vytvorenie jednej stanice klienta a jednej stanice servera. Je ale možné program spustiť niekoľkokrát a emulovať niekoľko klient/server spojení. Pri vytváraní nového uzlu poskytuje program defaultné nastavenia, ktoré je možné vidieť na obrázku 3.8.



Obr. 3.8: Konfigurácia nového objektu

Vytvorenie testovacieho prostredia pozostávalo z niekoľkých častí:

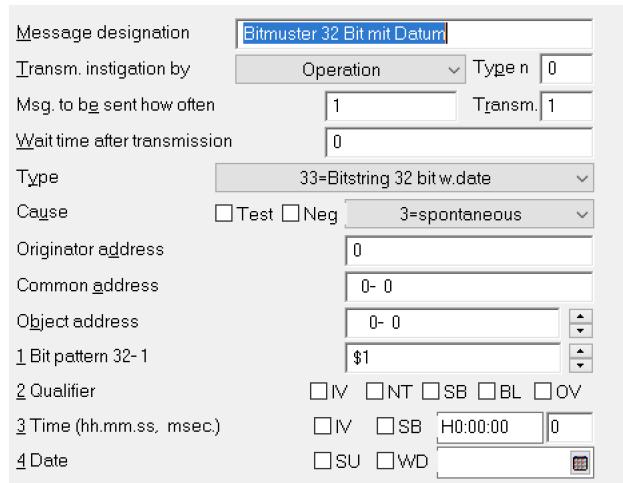
1. Spustenie programu WinPP104
2. Konfigurácia nového uzlu klienta:
  - (a) Parameterize → Receiver/Transmitter 1
  - (b) Function = Master = TCP Client
  - (c) IP address partner station = 127.0.0.1
  - (d) Port number RTU (Server) = 2404
  - (e) Zvyšné parametre mali defaultné hodnoty
  - (f) OK
3. Konfigurácia nového uzlu servera:
  - (a) Parameterize → Receiver/Transmitter 2
  - (b) Function = Substation = TCP Server
  - (c) IP address partner station = 127.0.0.1

- (d) Port number RTU (Server) = 2404  
 (e) Zvyšné parametre mali defaultné hodnoty  
 (f) OK
4. Mode → Online, následne sa vytvorilo spojenie medzi jednotlivými uzlami a bolo možné zasielať príkazy

Pri samotnej komunikácii poskytuje program 12 preddefinovaných správ a 12 zoznamov. Je možné ich dodatočne prekonfigurovať podľa potreby:

- Parameterize → Messages → 1,2,3,...
- Parameterize → Lists → 1,2,3,...

Je možné nastaviť napr. typ správy, dôvod (cause) odoslania správy, zdrojovú, cieľovú adresu a pod. Ukážka konfigurácie správy je na obrázku 3.9.



Obr. 3.9: Konfigurácia novej správy

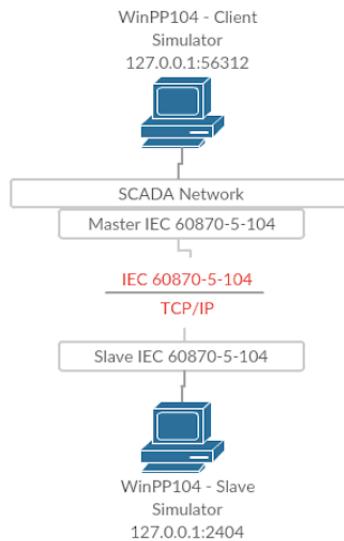
Zasielanie správ alebo zoznamov je možné cez Send → Messages/Lists → 1,2,3,...

Testovanie pozostávalo vo vytvorení jednoduchej topológie a v zaslaní/prijatí niekolkých preddefinovaných správ aby sa overilo, či komunikácia odpovedá štandardom protokolu IEC 60870-5-104. Ukážka testovej topológie je na obrázku 3.10. Program bol spustený na jednom zariadení a komunikácia bola zachytená nástrojom RawCap na rozhraní loopback. Z komunikácie bol vytvorený .pcap súbor WinPP104.pcap uložený v github repozitári<sup>9</sup>. Z následnej analýzy odchytenej dát bolo overené, že komunikácia bola validná a odpovedala protokolu IEC 104.

No.	Time	RTU	Objekt addr.	last Type	last Value	last Cause	Ori.	cyc	back	spon	req	IR
1	17:36:31	0- 0	0- 0	Single-point information w.date=30	ON=1	17:36:31,117	spon=3	0	0	0	20	0 0
2	17:36:45	0- 0	0- 0	Double-point information w.date=31	OFF=1	17:36:45,869	spon=3	0	0	0	7	0 0

Obr. 3.11: Obraz procesov programu WinPP104

<sup>9</sup>GitHub <https://github.com/janpristas/bakalarska-praca>



Obr. 3.10: WinPP104 - Testovacia topológia

**Zhrnutie:** WinPP104 je veľmi dobre vytvorený emulačný a monitorovací nástroj. Veľkou výhodou je, že dokáže jednotlivé zariadenia aj emulovať. Navyše pri monitorovaní alebo emulovaní zariadenia program automaticky vytvára tzv. obraz procesov (process image) o jednotlivých zariadeniach a komunikácii s nimi. Obraz procesov je dobrý na rýchly prehľad stavu jednotlivých objektov. Tiež je možné vyfiltrovať iba určitú komunikáciu podľa nami zadaného filtra. Ukážka je na obrázku 3.11. Obraz procesov je možné zobraziť cez **View → Process image**.

Počas behu programu vytvára systém tabuľku jednotlivých komunikácií, nazývanú tiež tabuľka TCP 104, viď obrázok 3.12. TCP 104 tabuľka je vhodná na rýchly prehľad aktívnych spojení, na filtrovanie spojení a sledovanie napr. nezabezpečených spojení (cez WiFi) alebo zariadení, ktoré sa nechovajú podľa očakávaní.

TCP 104										
CNo.	NCn.	IP Client	Port	IP Server	Port	SYN	Messg.	Start	Ende	Duration
1	1	127.0.0.1	50059	127.0.0.1	2404	0	9	12/05/2018 17:36:40	17:36:45	00:00:

Obr. 3.12: Prehľad TCP spojení programu WinPP104

Nevýhodou však zostáva, že program je voľne dostupný iba v rámci demoverzie, ktorá neumožňuje uloženie konfigurácií systému a je možné zaslanie a prijatie iba 20 správ medzi uzlami. Toto obmedzenie značne komplikuje konfiguráciu a dlhodobé sledovanie rozsiahlejších systémov. Taktiež je práca s programom relatívne komplikovaná a prvotné zorientovanie sa v systéme je o to náročnejšie. Pre porovnanie, nie je také intuitívne ako napr. v programoch Client/Server Simulator, ktoré budú popísané v predposlednej podkapitole.

### 3.2.2 Knižnica OpenMUC - j60870

**Výrobca:** OpenMUC je framework založený na jazyku Java a OSGi (špecifický interface pre jazyk Java), ktorý umožňuje jednoduchší vývoj špecifických monitorovacích, zaznamenávacích a riadiacich systémov. Framework bol vytvorený na Fraunhoferovom inštitúte pre

solárne energetické systémy vo Freiburgu. Podporuje vývoj softwaru pre niekoľko komunikačných protokolov, napr.:

- M-Bus
- IEC 61850
- IEC 62056-21
- IEC 60870-5-104

**Popis produktu:** OpenMUC - j60870 je knižnica pre jazyk Java JDK<sup>10</sup>. Umožňuje vytvorenie (naprogramovanie) koncových staníc typu klient/server. Výsledný program môže slúžiť ako emulačný prostriedok ale aj ako reálne monitorovacie zariadenie. Vytvorené simulačné programy monitorovacích/meracích zariadení by podľa špecifikácie výrobcu mali byť schopné komunikovať s akýmkolvek zariadením využívajúcim protokol IEC 60870-5-104.

**Protokoly:** Knižnica implementuje komunikačný štandard IEC 60870-5-104.

**Zariadenia:** Po naprogramovaní vlastného programu klient/server je možné pripojiť alebo sa pripojiť k reálnemu fyzickému IEC monitorovaciemu/meraciemu zariadeniu, prípadne k zariadeniam emulovaným iným programom.

**Typ:** Knižnica j60870 je licencovaná pod GPLv3. Je možné si od výrobcu zakúpiť proprietárnu licenciu.

**Platforma:** S knižnicou je možné pracovať pod OS Windows alebo pod akýmkolvek UNIX-like OS. Ja osobne som s knižnicou pracoval pod Mac OSX. Bohužiaľ nie je k dispozícii nijaké GUI, ani vývojové prostredie na vytvorenie vlastných zariadení, kedže ide iba o knižnicu rozširujúcu možnosti jazyka Java.

**Prípadová štúdia:** V rámci knižnice sú už predpripravené dva ukážkové programy, jeden pre klienta, druhý pre server. Programy sú uložené v zložke j60870/run-scripts. Pomocou programov je ale možné vytvoriť iba jednoduché spojenie obsahujúce jedného klienta a server obsahujúci jedno koncové zariadenie. Počas testovania bola použitá rovnaká topológia ako pri programe WinPP104. Viď obrázok 3.10. Pre zložitejšie topológie je potrebné si naprogramovať vlastné stanice klient/server. Pri programovaní je možné využiť množstvo funkcií, ktoré knižnica poskytuje práve na takúto implementáciu. Documentáciu jednotlivých funkcií, ktoré knižnica obsahuje je možné nájsť na <sup>11</sup>. V rámci testovania boli spustené oba predpripravené programy na jednom zariadení. Boli spustené v termináli (každý v samostatnom okne) daného zariadenia príkazom ./j60870-sample-server/j60870-console-client. Program servera neumožňoval žiadnu interakciu od užívateľa, iba prijímal príkazy od klienta. V rámci programu klienta bolo možné posielat príkazy na synchronizáciu času a na čítanie hodnoty zo servera, posielat príkazy na zmenu hodnôt na servery nebolo možné. Testovanie slúžilo k preukázaniu, že programy naprogramované v knižnici j60870 sú schopné bežnej komunikacie pomocou protokolu IEC 60870-5-104. Na základe zachytenej komunikácie sa to preukázalo. Komunikácia bola zachytená pomocou nástroja Wireshark na rozhraní loopback. Bol vytvorený súbor j60870.pcap uložený v github repozitári <sup>12</sup>.

**Zhrnutie:** Výhodu v knižnici j60870 vidíme v tom, že si užívateľ môže vytvoriť monitorovacie alebo meracie zariadenia presne na mieru, nakoľko si ich vytvorí priamo v zdrojovom kóde. Nevýhodou ale je, že práca s touto knižnicou už vyžaduje pokročilejšie znalosti a zručnosti s jazykom Java a s programovaním ako takým.

<sup>10</sup>OpenMUC - j60870 <https://www.openmuc.org/openmuc/> [Online: Október 2017]

<sup>11</sup>JavaDoc <https://www.openmuc.org/iec-60870-5-104/javadoc/> [Online: Október 2017]

<sup>12</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

### 3.2.3 IEC 60870-5-104 Client/Server Simulator

**Výrobca:** FreyrSCADA & Embedded Solution je indická spoločnosť so zameraním na vývoj softwaru pre rôzne komunikačné protokoly<sup>13</sup>. Mimo iné medzi ne patria:

- IEC 60870-5-101
- IEC 60870-5-103
- IEC 60870-5-104
- DLMS/COSEM Client

**Popis produktu:** Vyššie uvedená spoločnosť ponúka hned dva produkty pre protokol IEC-60870-5-104. Jeden na emuláciu klient (master) staníc a druhý na emuláciu server (slave) staníc. Každý program umožňuje emulovať až 50 uzlov klient/server a každý uzol navyše niekoľko koncových staníc. Jednotlivé nástroje umožňujú prenos dát a príkazov spolu s uložením .log správy s prehľadom aktuálnej komunikácie, ktorá na nich prebieha. Zdrojové kódy sú písané v jazyku C.

**Protokoly:** Oba programy sú určené na prácu výhradne nad protokolom IEC 60870-5-104.

**Zariadenia:** Programy si emulujú svoje vlastné koncové zariadenia, dokážu sa ale pripojiť aj k zariadeniam emulovaným iným programom, prípadne k reálnym fyzickým zariadeniam.

**Typ:** Program nie je open source, ale výrobca poskytuje zdarma demo verziu oboch programov. Demo verzia má však nevýhodu v tom, že po spustení má užívateľ iba 15 minút na prácu. Po uplynutí tohto času je nutné program vypnúť a spustiť znova, spolu s opäťovnou konfiguráciou klient/server staníc. Je to veľká nevýhoda, ak chcete testovať väčšiu sieť, kde iba samotné vytvorenie zaberie veľa času a na testovanie už nijaký nezostane.

**Platforma:** Samotné programy sú určené iba na OS Windows. Výrobca ale poskytuje aj tzv. SDK (Software Development Kit), pre klienta aj server, ktoré obsahujú programy v jazyku C a umožňujú naprogramovanie koncových staníc s presnými špecifikáciami požadovanými zákazníkom. Programy SDK sú dostupné na OS Windows aj Linux.

**Prípadová štúdia:** Program klienta môže mať v rámci jedného uzlu pripojený iba jeden server, pričom server môže obsahovať niekoľko koncových zariadení. Je možné ale vytvoriť až 50 rôznych spojení klient/server. Pri vytváraní nového klienta/servera ponúka emulátor defaultné nastavenia, ktoré je možné vidieť na obrázku 3.13 a 3.14. Prvotná konfigurácia je o niečo náročnejšia, nakoľko má program viac možností konfigurácie ako vyššie popísané programy. K dispozícii je ale niekoľko inštrukčných videí<sup>14</sup>, ktoré podrobne popisujú prácu s jednotlivými programami. Oba programy sú veľmi dobre spracované a umožňujú užívateľovi podrobnejšiu konfiguráciu jednotlivých staníc podľa jeho potreby. V rámci klienta je možné nastaviť parametre ako ip adresu, port, počet pripojených staníc, adresy jednotlivých staníc a pod. V rámci servera takisto ip adresu, port, ip adresu klienta, dĺžku trvania jednotlivých typov príkazov a pod. Podrobnejšie je to možné vidieť na obrázku 3.13 a 3.14.

Vytvorenie spojenia a testovanie programov pozostávalo z niekoľkých krokov:

1. Spustenie programu servera
2. Kliknutie na Add Server

<sup>13</sup>FreyrSCADA & Embedded Solution <http://freyrscada.com> [Online: November 2017]

<sup>14</sup>Tutoriály <http://freyrscada.com/iec-60870-5-104-Client-Simulator.php> [Online: November 2017]

Server IP Address	127.0.0.1	Source IP Address	127.0.0.1
Port Number	2404	Port Number	2404
Auto Generate IEC104 Data Objects	FALSE	Remote IP address	0.0.0.0
Total Number of Stations(Common Address)	1	Enable UTC	FALSE
Station Address - 1 (CommonAddress 1)	1	K Value	12
Station Address - 2 (CommonAddress 2)	0	W Value	8
Station Address - 3 (CommonAddress 3)	0	t0	30
Station Address - 4 (CommonAddress 4)	0	t1	15
Station Address - 5 (CommonAddress 5)	0	t2	10
K Value	12	t3	20
W Value	8	Long Pulse Time	10000
t0	30	Short Pulse Time	5000
t1	15	Event Buffer Size	50
t2	10	Clock Synchronisation Period	0
t3	20	Transmit Spontaneous Measured Value	TRUE

Obr. 3.13: Defaultná konfigurácia Obr. 3.14: Defaultná konfigurácia nového klienta nového servera

3. Komunikácia prebiehala na rozhraní loopback, nebolo treba nič nastavovať čo sa týka spojenia

4. Configuration → Add Row:

- Group → Measured Short Float
- Type Id → M\_ME\_NC\_1 = 13
- Starting IOA → 1
- Range → 5

5. Add Row

- Group → Set Point Command - Float Value
- Type Id → C\_SE\_NC\_1 = 50
- Starting IOA → 10
- Range → 5

6. Load Configuration

7. Data\_Objects

8. Následne bolo potrebné všetky objekty typu Set Point (začínajúce číslom 10) nama-povať na objekty uchovávajúce namerané hodnoty:

- (a) Pravý klik na objekt
- (b) Map
- (c) V zozname vybrať objekt na ktorý sa ide mapovať, ideálne 10 → 1, atp.

9. Start Communication

10. Spustenie programu klienta

11. Add Client

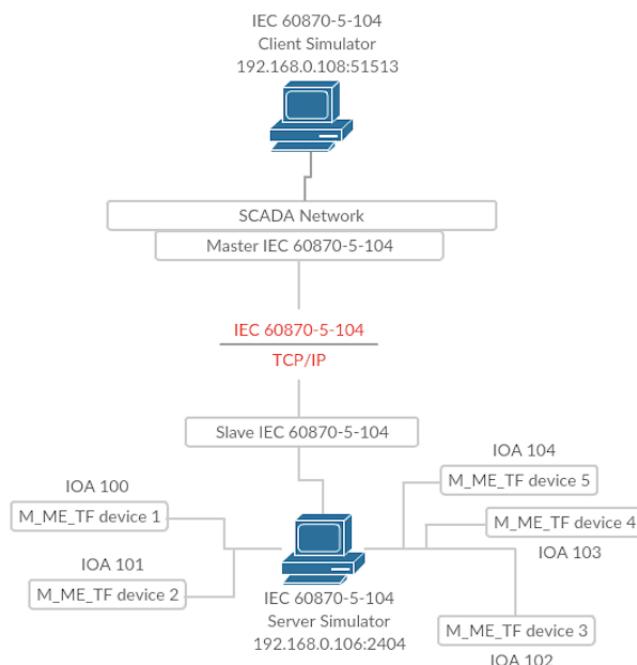
12. Nebolo potrebné nastavovať IP adresu, ani port, komunikácia prebiehala na rozhraní loopback na štandardnom porte 2404

13. **Data\_Objects → Start Communication**

14. Klient sa pripojil k serveru a zobrazil pripojené objekty

Po kliknutí pravým tlačítkom na jednotlivé objekty sa zobrazili príkazy, ktoré bolo možné zasielať serveru. **Point Command** na vzdialené nastavenie hodnôt, **Station Commands** na zaslanie štandardných príkazov typu **General Interrogation**, **Counter Interrogation**, **Directory Read** atď. Priebeh komunikácie je možné vidieť v logovacej správe, ktorá je automaticky generovaná oboma programami. Správu je možné zobraziť po rozkliknutí záložky **Log**.

Testovanie spočívalo v zaslaní niekoľkých štandardných príkazov medzi klientom a serverom na overenie, že komunikácia odpovedá štandardom protokolu IEC 60870-5-104. Zaslané príkazy boli zamerané hlavne na vzdialené čítanie a nastavenie hodnôt. Na obrázku 3.15 je ukázaná topológia použitá pri testovaní. Programy boli spustené na jednom zariadení. Ko-



Obr. 3.15: Ukážka testovacej topológie programov Client/Server Simulator

munikácia bola zachytávaná pomocou nástroja RawCap na rozhraní loopback. Zachytená komunikácia je v súbore C/SSimulator.pcap, ktorý je uložený v github repozitári <sup>15</sup>.

**Zhrnutie:** Client/Server Simulator programy sú asi najprehladnejšie varianty s najväčšou škálou možností na emuláciu prevádzky SCADA systémov využívajúcich protokol IEC 60870-5-104. Veľkou nevýhodou ale zostáva, že demoverzie, s ktorými som pracoval, poskytujú iba 15 minút na prácu a následne sa vypnú. Celý systém je potom nutné konfigurovať odznova.

<sup>15</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

### 3.2.4 QTester 104

**Výrobca:** Autorom produktu je Ricardo Olsen, majiteľ spoločnosti DSC Systems. Jeho prácou je konfigurácia a vývoj riadiacich systémov pre komunikáciu klient/server.

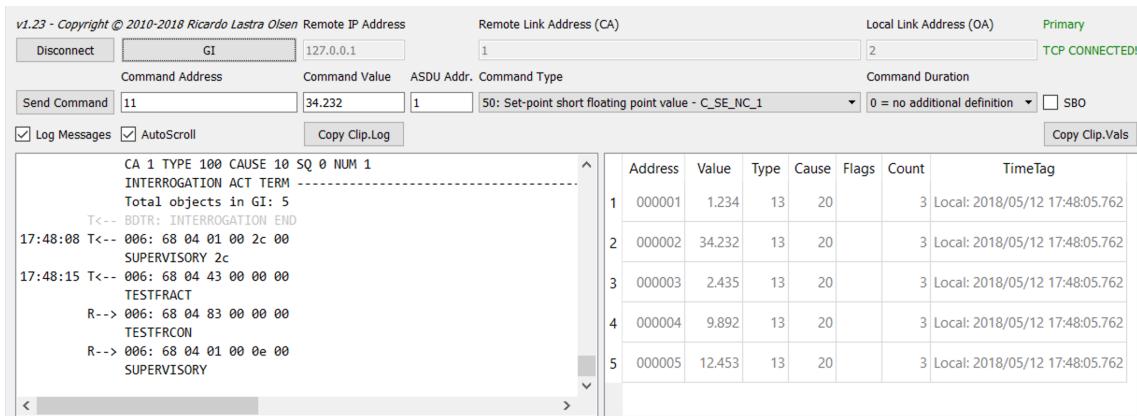
**Popis produktu:** QTester 104 je open source software, ktorý slúži na získavanie dát z koncových staníc a ich riadenie. Je možné si ho zdarma stiahnuť na stránke [sourceforge.net<sup>16</sup>](https://sourceforge.net/projects/qtester104/). Umožňuje sledovať stav koncových zariadení, stahovať z nich dátu a posielat príkazy. Pracuje iba ako klient(master). Je vytvorený v C++ za pomoci QT. Záznam komunikácie je možné uložiť vo forme textového dokumentu. Software je iba časťou väčšieho HMI projektu<sup>17</sup> a slúži ako modul pre HMI. Môže byť ale použitý aj samostatne ako protokol tester.

**Protokoly:** Program využíva na komunikáciu protokol IEC 60870-5-104.

**Zariadenia:** Kedže QTester 104 funguje iba ako klient, nie je schopný emulovať koncové zariadenia. Je možné ale pripojiť reálne, alebo iným programom emulované zariadenia. Komunikácia prebieha cez TCP/IP spojenie.

**Typ:** Program je open source a je spolu so zdrojovými kódmi voľne k dispozícii. Taktiež ho je možné voľne distribuovať ďalej.

**Platforma:** S programom je možné pracovať pod OS Windows a Linux. Na obrázku 3.16 je ukážka užívateľského rozhrania programu.



Obr. 3.16: Ukážka užívateľského rozhrania programu QTester 104

**Prípadová štúdia:** QTester 104 umožňuje v rámci monitorovania mať vytvorenú iba jednú stanicu pre klienta, čo je vlastne program samotný, a umožňuje pripojenie jednej stanice servera. Testovanie prebiehalo pod OS Windows za pomoci programu IEC 104 Server Simulator, ktorý bol popísaný v predchádzajúcej podkapitole.

Vytvorenie testovacej topológie prebiehalo v niekoľkých krokoch:

- Spustenie a konfigurácia servera, rovnako ako v predchádzajúcej podkapitole
- Spustenie programu QTester 104: `qtester104 → bin → QTester104`
- Pri testovaní bolo použité rozhranie loopback na ktorom sa program ihneď po spustení snažil spojiť so serverom, nebola potrebná dodatočná konfigurácia
- Po úspešnom spojení, kliknúť na GI (General Interrogation) aby sa načítali objekty pripojené k serveru

<sup>16</sup>QTester 104 <https://sourceforge.net/projects/qtester104/> [Online: Október 2017]

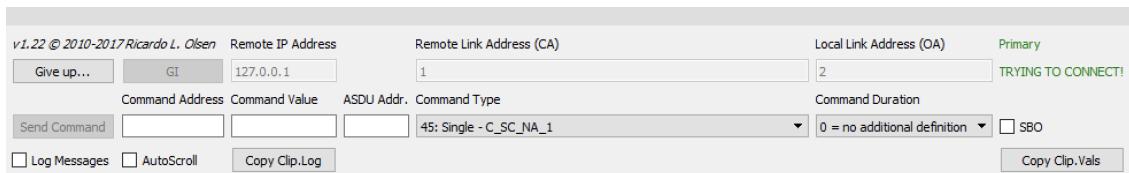
<sup>17</sup>OSHMI <https://sourceforge.net/projects/oshmiopensubstationhmi/> [Online: November 2017]

## 5. Zakliknúť Log Messages, prípadne AutoScroll na zobrazenie logovacej správy

Po úspešnej konfigurácii a prepojení je možné zasielať serveru príkazy. Pre zaslanie príkazu je potrebné nastaviť niekolko hodnôt:

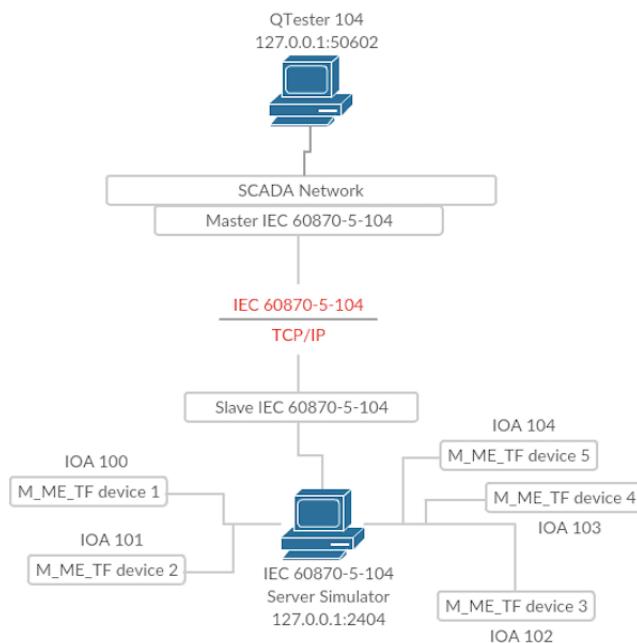
- **Command Address**, adresa Set-point objektu namapovaného na objekt, ktorý chceme nastaviť, napr. 10
- **Command Value**, hodnota na ktorú chceme objekt nastaviť, musí byť typu čísla s pohyblivou rádovou čiarkou, napr 123.456
- **ASDU address**, hodnotu nastaviť na 1

Práca s programom je vo všeobecnosti veľmi intuitívna, po spustení stačí iba zadať IP adresu servera, program sa s ním automaticky spojí a je možné s ním ihneď pracovať. Na obrázku 3.17 je ukážka stavu programu ihneď po spustení. Pri každej zmene hodnôt na



Obr. 3.17: Defaultné nastavenie programu pri spustení

strane servera sú automaticky aktualizované v QTestery. Pri testovaní bolo vytvorených päť koncových zariadení typu M\_ME\_TF na zaznamenávanie hodnôt s pohyblivou rádovou čiarkou. Topológiu je možné vidieť na obrázku 3.18. Oba programy boli spustené na jed-



Obr. 3.18: QTestler 104 - Testovacia topológia

nom zariadení a komunikácia medzi nimi bola následne odchytiavaná v nástroji RawCap

na rozhraní loopback. Z komunikácie bol vygenerovaný súbor QTester104.pcap uložený v github repozitári <sup>18</sup>.

**Zhrnutie:** QTester 104 je celkom prepracovaný a užívateľsky prívetivý monitorovací nástroj. Výhodou je, že je open source a je možné s ním volne pracovať a distribuovať ho. Ďalšiu výhodu vidím v jednoduchom a intuitívnom GUI v ktorom sa veľmi dobre pracuje. Nevýhodou je menšia škála možností oproti programu Client Simulator z predchádzajúcej kapitoly, na testovacie účely bol ale postačujúci.

### 3.3 Zhrnutie

V tejto kapitole bol uvedený základný prehľad priemyselných monitorovacích a emulačných nástrojov pre SCADA systémy. Bola popísaná najmä ich základná funkcia a schopnosť komunikácie odpovedajúcej patričným štandardom.

#### 3.3.1 DLMS/COSEM

Porovnanie emulátorov pre protokol DLMS/COSEM		
	DLMS Director	XmlDemo
Platforma	OS Windows	OS Windows
Klient/Server	Klient	Klient
Kom. protokoly	TCP/IP, HDLC	TCP/IP, HDLC iba cez sériovú linku
Licencia	Open Source	Open Source
Vlastná konfig.	Nie, iba objekty pripojené k serveru	Nie, iba objekty pripojené k serveru
GUI	Veľmi dobré a intuitívne	Veľmi jednoduché
Celkový dojem	Dobre spracovaný program s veľkou škálou možností	Dobre spracovaný program podporujúci základnú potrebnú funkčnosť

Pre protokol DLMS/COSEM boli popísané programy DLMS Director a XmlDemo. Oba programy sú dobre spracované a poskytujú potrebnú funkčnosť a umožňujú vytvoriť komunikáciu odpovedajúcu štandardu DLMS/COSEM, avšak program DLMS Director poskytuje navrch možnosť komunikácie pomocou HDLC, väčšiu škálu možných nastavení a taktiež má oveľa prívetivejšie a intuitívnejšie GUI. Pre protokol DLMS/COSEM budem ďalej využívať program DLMS Director.

<sup>18</sup>GitHub <https://github.com/janpristas/bakalarska-praca>

### 3.3.2 IEC 60870-5-104

Porovnanie emulátorov pre protokol IEC 104, 1. časť

	WinPP104	Knižnica OpenMUC - j60870
Platforma	OS Windows	OS Windows/Unix like OS
Klient/Server	Klient/Server	Klient/Server
Kom. protokoly	TCP/IP	TCP/IP
Licencia	Demoverzia	Licencovaná pod GPLv3
Vlastná konfig.	Možnosť konfigurácie vlastných šablón správ	Možnosť vytvoriť si vlastný program klient/server podľa potreby užívateľa
GUI	Trochu komplikované	Vzorové programy bez GUI, iba ako terminálové aplikácie
Celkový dojem	Program je relatívne náročný na prvotné zorientovanie a pochopenie, taktiež je nevýhodou obmedzenie demoverzie na zaslanie/prijatie iba 20 správ	Vzorové programy jednoduché (slúžia iba na ilustráciu možností knižnice), knižnica je dobre spracovaná, umožňuje veľkú škálu možností na tvorbu vlastných staníc

Porovnanie emulátorov pre protokol IEC 104, 2. časť

	Client/Server Simulator	QTester 104
Platforma	OS Windows, možnosť získať SDK pre klienta a server od výrobcu na OS Windows/Linux	OS Windows/Linux
Klient/Server	Klient/Server	Klient
Kom. protokoly	TCP/IP	TCP/IP
Licencia	Demoverzia	Open Source
Vlastná konfig.	Nie, možnosť využívať iba štandardné objekty	Nie, iba objekty pripojené k serveru
GUI	Dobre spracované, miestami trochu komplikované	Veľmi dobré, jednoduché a intuitívne
Celkový dojem	Veľmi dobre spracované programy s veľkou škálou možností, nevýhodou je obmedzenie na 15 minút pri demoverzii	Veľmi dobre spracovaný program, veľmi jednoduchý, podporujúci základnú požadovanú funkcialitu

Pre protokol IEC 60870-5-104 boli popísané programy WinPP104, QTester 104, IEC 60870-5-104 Client/Server Simulator a knižnica pre jazyk Java OpenMUC - j60870. Každý z programov poskytuje inú funkcionality a škálu možností konfigurácie. Všetky programy umožňujú vytvoriť komunikáciu odpovedajúcu štandardu IEC 60870-5-104. Užívateľsky najprívetivejšie a pre ďalšie testovanie sú najvhodnejšie programy Client/Server Simulátor a QTester 104. Programy podporujú veľkú škálu možností na konfiguráciu staníc a špecifikáciu komunikácie. Taktiež poskytujú veľmi intuitívne prostredie a pracuje sa s nimi jednoducho. Oproti tomu je program WinPP104 oveľa komplikovaný a práca s ním nie je jednoduchá.

Rovnako komplikovaná je práca s knižnicou OpenMuc - j60870, ktorá súčasťou umožňuje vytvorenie vlastného programu klient/server, vyžaduje však vyššiu znalosť programovacieho jazyka Java a problematiky samotnej. Výhodou programu QTester 104 je jeho voľná dostupnosť a kvalitné spracovanie. Nevýhodou ale zostáva možnosť vytvorenia spojenia iba medzi jedným klientom a serverom. Tento problém riešia programy IEC 60870-5-104 Client/Ser-

ver Simulator, ktoré umožňujú vytvorenie väčšej topológie. Avšak nevýhodou demoverzie je časové obmedzenie, ktoré umožňuje s programami pracovať maximálne 15 minút.

V praktickej časti svojej práce budem vychádzať z programov DLMS Director, QTester 104 a IEC 60870-5-104 Client/Server Simulator.

## Kapitola 4

# Bezpečnosť SCADA systémov

Počítačové útoky sú v dnešnej dobe skutočnou hrozbou. Jedným z hlavných dôvodov ohrozenia SCADA systémov je ich postupný prechod na IP vrstvu. Systémy sa stále rozrasťajú a vzdialenosť jednotlivých koncových staníc od riadiacej je čoraz väčšia, čo znemožňuje systému komunikovať na fyzickej vrstve. Je preto potrebné vytvoriť komunikačný kanál (napr. VPN spojenie), na vzdialéne monitorovanie, medzi jednotlivými stanicami. Na jednej strane je to užívateľsky veľmi prívetivé, je možné vzdialene sledovať a riadiť veľmi veľké množstvo zariadení. Taktiež je možné centrálnie vykonávať rôzne aktualizácie ap. Avšak vytvorenie komunikačného kanálu v značnej miere zjednodušuje útočníkom napadnúť a infikovať danú sieť, nakoľko komunikačný kanál môže byť použitý ako zadný vchod (backdoor) do systému. Preto je potrebné komunikáciu v sieti neustále monitorovať (flow monitorovanie) a mať k dispozícii nástroje (rôzne sondy ap.), ktoré sú schopné zachytiť neštandardnú komunikáciu v sieti a včas varovať pred bezpečnostným incidentom. Dôsledky jednotlivých incidentov sa môžu pohybovať od relatívne malých, ako je napríklad prerušenie prebiehajúcej operácie alebo zmena operačného procesu po veľké, ako úmyselné sabotáže s cieľom spôsobiť systému citelnú ujmu. Jednotlivé útoky nemusia mať vplyv iba na systém samotný, ale aj na okolitú spoločnosť. Napríklad útoky na veľké komplexy ako elektrárne, rafinérie, čističky môžu mať veľký dopad aj na životné prostredie (únik ropy, uvoľnenie toxických látok). Niektoré útoky môžu mať dokonca za následok zranenia alebo straty na životoch, potencionálne katastrofické výbuchy ap[4].

Úspešný útok môže mať mnoho rôznych následkov zahŕňajúc:

- zmeny alebo blokovanie zamýšľaného procesu, napr. zmeny v zamýšlanom množstve vyrobenej el. energie,
- zmeny, oneskorenie alebo blokovanie v infomácií o nameraných hodnotách, veľký dôsledok môže mať napríklad pri obchodovaní s energiami,
- neautorizované zmeny v inštrukciách, prípadné vypnutie jednotlivých zariadení ako generátory.

Konečným dôsledkom útoku môže byť čokoľvek od finančnej straty až po stratu fyzického zabezpečenia systému s dopodom na okolný ekosystém, miestnu komunitu, prípadne dokonca aj štát. Stručný prehľad dopadu útokov rôznych veľkostí je na obrázku 4.1.



Obr. 4.1: Dopad rôznych druhov útokov[4]

## 4.1 Bezpečnostné riziká

Pri posudzovaní bezpečnosti a bezpečnostných rizík v SCADA systémoch sa rozlišuje medzi dvoma typmi útokov. Útoky cez SCADA kanále (SCADA channels) a útoky cez podporné kanále (maintenance channels)[4].

SCADA kanále sú komponenty slúžiace na primárne účely SCADA systémov - zber a prenos dát, skladovanie a spracovanie údajov a informácií z komunikácie medzi jednotlivými komponentami systému. Medzi typické komponenty SCADA systémov patria:

- Systém riadenia distribúcie - súbor aplikácií na sledovanie a riadenie systému
- SCADA servery
- Vzdialé koncové zariadenia
- Inteligentné meracie zariadenia
- Ochranné články

Podporné kanále sú systémy slúžiace na inštaláciu a údržbu vyššie spomenutých súčasťí systému. Taktiež slúžia na sprostredkovanie komunikácie medzi nimi. Typicky to sú:

- Inžinierske stanice
- Spúšťacie (commissioning) servery
- Servery na synchronizáciu času v systému
- Monitorovacie a logovacie servery (SNMP, syslog)

Hlavným dôvodom rozlišovania medzi jednotlivými súčasťami systému je to, že každý so sebou nesie rozličné bezpečnostné riziká. Komunikácia cez SCADA kanály je väčšinou typu počítač-počítač a prenášajú sa iba prevádzkové data, nie konfiguračné zmeny alebo binárky. Narušenie alebo zneužitie sa v monitorovanej komunikácii dá ľahko skryť. To znamená, že

je väčšina útokov relatívne rýchlo detekovateľná. Avšak pri podporných kanáloch je to o niečo komplikovanejšie. Autorizované zmeny na jednotlivých komponentoch vykonané pracovníkom spoločnosti sa moc nelisia od neautorizovaných zmien vykonaných útočníkom. A pretože podporné služby vyžadujú privilegovaný prístup do systému, je to velmi lákavé pre útočníkov, ktorí chceú získať permanentnú kontrolu nad systémom. Je ale možné spoloahlivo monitorovať aj túto časť siete, avšak iba ak sa všetky podporné procesy vykonávajú "disciplinovane".

V počítačových sieťach obecne existuje mnoho rôznych druhov útokov. Avšak pri SCADA systémoch sa väčšina z nich neberie do úvahy, nakoľko je pri nich predpoklad dostatočného zabezpečenia systému a dôkladná konfigurácia firewallov na prepúšťanie iba najdôležitejšej komunikácie. To znamená, že očividne "diery" do systému by mali byť uzavreté. Napriek tomu ale zostáva niekolko typov útokov, ktorými sú SCADA systémy ohrozené[4]:

- Útok cez SCADA kanál
- Útok cez podporu SCADA serverov
- Útok cez podporu koncových zariadení
- Útok "zvnútra"

Táto práca je zameraná najmä na útoky cez SCADA kanále a cez podporu koncových zariadení.

#### 4.1.1 Útoky cez SCADA kanále

Pri samotnom získavaní prístupu do riadiaceho systému SCADA siete existuje iba niekoľko možností ako ho infikovať. Nakoľko je na prístup do systému vyžadovaná autentizácia, útočníci sa môžu do systému dostať dvoma spôsobmi. Prvý je získanie prístupu cez platné autentizačné údaje niekoho iného. Rozhrania na výmenu údajov cez IT siet by však mali byť obmedzené na export dát, takže dopad útoku na systém nie je taký veľký. Druhý spôsob je do systému rozšíriť malware cez servery prístupné z IT domén. Ak bude software využívať zraniteľnosti systému, umožní tak útočníkovi robiť veľké zmeny v jadre systému a môže výrazne ovplyvniť a ohroziť celý systém. Pravdepodobnosť úspechu takéhoto útoku výrazne závisí na zabezpečení a počte zraniteľných miest, ktoré systém má.

Ďalším vstupným bodom do SCADA systému je útok na koncové stanice. Nakoľko systém väčšinou využíva veľa koncových staníc, niektoré z nich dokonca na odľahlých miestach, ani dobrá fyzická ochrana nemôže plne zabrániť príniku do zariadenia a následne do systému. Či už ide o fyzický alebo kybernetický útok, takéto útoky väčšinou nespravia v systéme veľké škody, nakoľko ide o napadnutie jedného zariadenia a na ovplyvnenie viacerých by bolo potrebné viesť útok cez riadiacu stanicu. Samozrejme to ale záleží aj od použitej topológie, ktorá je využívaná na prepojenie riadiacej stanice a koncových staníc.

Útočníci môžu využiť zariadenia na koncových staniciach na prístup do systému s platnými autentizačnými údajmi, avšak ak je systém dostatočne zabezpečený, môžu nanajvýš posielat nesprávne/podvrhnuté informácie o nameraných hodnotách. Aby boli schopní spraviť väčšie škody, museli by napr. rozšíriť po systéme software, ktorý bude schopný ovládať väčšie množstvo koncových staníc.

Ďalší typ útoku je pripojenie do systému cez spojenie s riadiacim strediskom. Sú dva typy hrozieb, ktoré môžu byť takýmto útokom napáchané. Prvý je pripojenie cez vlastnú riadiacu stanicu s jednotlivými koncovými zariadeniami s využitím platných autentizačných údajov.

Útočník tak môže napr. posielat neautorizované príkazy jednotlivým stanicam. Druhý útok spočíva vo využití zraniteľnosti systému, čo môže byť zneužité napr. na rozšírenie rôznych typov malware do siete.

Ked' už útočník prenikne do systému, je schopný využívať jeho plnú funkciu a rôznymi spôsobmi ho tak poškodit[4].

#### 4.1.2 Útoky cez koncové zariadenia

Vybavenie koncových stanic ako RTU (Remote Terminal Units - mikroprocesorom riadené zariadenia) alebo IED (Intelligent Electronic Devices - inteligentné merače) môžu byť väčšinou spravované z jednej centrálnej stanice. Ak útočník dokáže získať prístup k týmto hostom, napr. cez platné autentizačné údaje, môže zmeniť konfiguráciu jednotlivých zariadení. Ak zariadenie nie je konfigurované lokálne, môže byť konfigurované cez nejaké externé zariadenie ako napr. laptop, USB kľúč, pamäťovú kartu ap. To je opäť riziko na rozšírenie malware po sieti.

Ďalšia cesta, ako je možné napadnúť systém, je využiť zraniteľnosti jednotlivých zariadení na stanici. Nakolko väčšinou využívajú normálny IT software, poskytujú tým útočníkom jednoduchú cestu do SCADA systému. Koncové stanice môžu byť napadnuté zo sieťovej strany ale aj cez koncové zariadenia, ak do nich niekto vloží malware, ktorý sa môže ďalej šíriť do siete.

Riziko architektúry SCADA systému je aj v tom, že je možné si vytvoriť vstup do systému pripojením vlastného zariadenia. To sa týka najmä koncových stanic, ktoré sú menej chránené ako riadiaca stanica[4].

## 4.2 Typy útokov

Ked' sa raz útočník dostane do systému a identifikuje svoj cieľ, existuje mnoho spôsobov, ako ho poškodiť. Medzi najbežnejšie útoky patria:

- Man-in-the-middle
- Denial-of-Servise (DoS)
- Replay útoky

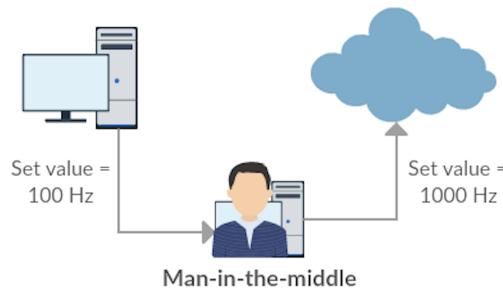
Hlavným dôvodom je kombinácia rôznych protokolov a slabá úroveň autentizácie, napr. prenos nezašifrovaných kľúčov po sieti, ktoré sa dajú jednoducho odchytiať a použiť na získanie neoprávneného prístupu do systému. V systémoch, ktoré tvoria spojitosť medzi niekolkými systémami, ako napr. SCADA server, môže byť trvalá prítomnosť útočníka v riadiacej časti použitá ako základ sekundárneho útoku proti iným časťiam siete. Pri takomto útoku je veľmi dôležitý rozdiel medzi ovládnutím ciela a útokom samotným. Ovládnutie ciela znamená prístup alebo schopnosť vykonávať s objektom neznámu činnosť. Napríklad spustenie určitého procesu s parametrami nad prípustné medze a tým zariadenie fyzicky poškodiť, prípadne poškodiť viaceré zariadenia, ktoré sú na ovládanom závislé. Útok je na druhej strane schopnosť útočníka vykonávať s cieľovým objektom nežiadúce akcie. V tomto prípade môže zariadenie pracovať, tak ako má, avšak schopnosť napadnúť zariadenie a spôsobiť, že vykoná akciu, ktorú obsluhujúci personál nepožadoval, môže mať negatívne následky. Takéto útoky bývajú spojované s využívaním funkčnosti a zraniteľných miest zariadení. To znamená, že zaslanie riadiacemu zariadeniu príkaz na vypnutie nepredstavuje slabosť zariadenia ako

takého, avšak nedostatok overovania a autentizácie umožňuje útočníkovi zaslať nežiadúci príkaz, a tým na istý čas ochromiť systém. Obdobné útoky sú tzv. replay útoky[4].

#### 4.2.1 Man-in-the-middle útok

Vo svojej práci sa budem najviac zaoberať tzv. útokom man-in-the-middle, ktorý pozostáva z odchytenej a následne zmenej komunikácie medzi dvoma zariadeniami.

Pri útoku typu man-in-the-middle sa hovorí o útoku kedy útočník sleduje komunikáciu medzi zariadeniami, odchytáva zasielané správy, mení ich a opäťovne zasiela pôvodnému adresátovi. Zariadeniam sa zdá, že medzi sebou komunikujú priamo, avšak v skutočnosti komunikujú cez tretie zariadenie, ktoré ich odpočúva a môže interagovať (zasahovať, meniť zasielané správy). Na obrázku 4.2 je úkažka typickej komunikácie pri útoku man-in-the-middle, spolu so zmenou v zasielanom príkaze.



Obr. 4.2: Man-in-the-middle útok

Ak chce útočník vykonávať man-in-the-middle útok, musí byť schopný zachytiť komunikáciu medzi oboma cieľovými systémami a vložiť do nej nové (upravené) správy. Ak spojenie nemá šifrovanie alebo dokonca ani overovanie autentizáciou, ako je častý prípad pri priemyselnej prevádzke, je man-in-the-middle útok veľmi jednoduchý proces. Avšak aj v komunikáciach, kde sa používa autentizácia, alebo šifrovanie, môže byť stále man-in-the-middle útok úspešný, napr. odpočúvaním kľúčových výmien a prenesením útočníkovho kľúča namiesto legitímneho. Tento spôsob útoku je ale v priemyselných sieťach trochu komplikovaný, nakoľko jednotlivé strany môžu spolu komunikovať dlhú dobu a útočník by musel zneužiť existujúce komunikačné spojenie. Najobtiažnejšou časťou úspešného man-in-the-middle útoku je úspešne prepojiť dátový tok a presvedčiť obe strany, že útočník je skutočne určeným príjemcom jednotlivých správ. Proti tomu je možné sa brániť autentizačnými kontrolami, avšak mnohé priemyselné komunikačné protokoly zasielajú autentizačné údaje v otvorenej podobe, čo ulahčuje útočníkovi prácu, nakoľko mu stačí odsledovať komunikáciu, prečítať autentizačný kľúč a úspešne sa pripojiť do systému[4].

#### 4.2.2 Denial-of-Service útok

Denial-of-Service, čiže útok typu odoprenia služby, nastáva, keď sa nejaký útočník snaží odoprieť službu systému pre bežných užívateľov tým, že ho robí nedostupným (neschopným bežnej prevádzky). DoS útoky sú veľmi široká kategória útokov a môžе zahŕňať čokolvek od straty komunikácie so zariadením po narušenie alebo zablokovanie určitých služieb v rámci daného zariadenia (ukladanie, spracovanie I/O). DoS útoky v bežných systémoch väčšinou nemajú veľké následky, ak sú riešené včas. Avšak ak je DoS útok dobre cielený, môže prepnuť dôležité systémy do offline režimu, alebo ich dokonca vypnúť.

Automatizované systémy sa nasadzujú na monitorovanie a riadenie rôznych procesov. Tento proces môže ovplyvňovať tok ropy v potrubí, premenú vodnej pary na elektrinu alebo kontrolovať časovač zapalovania v motore. Strata schopnosti správcu tieto procesy riadiť sa nazýva strata kontroly (Loss of Control - LOC) a zvyčajne vedie k prepnutiu fyzického procesu do "bezpečného" stavu vypnutia. To znamená, že aj jednoduché prerušenie kontrolných funkcií môže viesť k fyzickým poruchám na systéme, ktoré môžu pokračovať v odstavení systému, mechanickému zlyhaniu alebo iným katastrofickým scenárom.

Oproti DoS útokom na rôzne webové stránky môžu mať priemyselne cielené DoS útoky za následok napr. únik ropy, neplatné šarže výrobkov alebo dokonca explózie. DoS útok v priemyselnom prostredí je oveľa viac ako iba nepríjemnosť a môže viesť k veľmi nepríjemným následkom, ak nebude včas riešený[4].

### 4.3 Zaznamenané útoky

Na rozdiel od IT sietí boli OT siete a komunikácia v nich väčšinou izolované a mali iba minimálnu komunikáciu s inými sieťami. História útokov na priemyselné systémy je preto oveľa menšia ako na IT siete. Avšak útoky na priemyselné systémy môžu mať veľké následky, ktoré môžu spôsobiť stratu na životoch, fyzické poškodenie systému alebo znečistenie okolitého ekosystému.

Napríklad na jar roku 2000 sa bývalý zamestnanec istej austráliskej softvérovej spoločnosti uchádzal o zamestnanie v miestnej samospráve, avšak jeho žiadosť bola zamietnutá. Následne sa nespokojnému uchádzca podarilo pomocou rádiového vysielača vzdialene získať prístup do kontrolného systému čističky odpadných vôd a zmeniť elektronické údaje pre konkrétnu kanalizačné čerpacie stanice. To malo za následok poruchy v prevádzke systému a následné vypustenie viac ako 950 000 litrov odpadných vôd do blízkych riek a parkov, čo malo veľký efekt na miestny ekosystém[9, p. 3-20].

V tejto podkapitole bude uvedený prehľad rôznych typov zdokumentovaných útokov na priemyselné IoT siete.

#### 4.3.1 Vlakový signalizačný systém CSX (2003)

V auguste 2003 bol počítačový vírus Sobig viniený za vypnutie vlakových signalizačných systémov na celom východnom pobreží USA. Vírus infikoval počítačový systém v centrále spoločnosti CSX Corp. v Jacksonville v štáte Florida a vypol signalizáciu, dispečing a iné systémy. Podľa Dana Stessela, hovorca spoločnosti Amtrak, bolo útokom postihnutých desať vlakov. Vlaky medzi Pittsburgom a Florenciou v Južnej Karolíne boli zastavené kvôli tmavým signálom a jeden regionálny vlak z Richmondu vo Virginii do Washingtonu a New Yorku bol odložený o viac ako dve hodiny. Diaľkové vlaky sa oneskorili o štyri až šesť hodín[9, p. 3-20].

#### 4.3.2 Výpadok prúdu v elektrárni v Northeast (2003)

V auguste 2003 zlyhanie procesoru alarmu v SCADA systéme spoločnosti First Energy zabránilo operátorom riadiacich staníc byť dostatočne informovaní o kritických prevádzkových zmenách v elektrickej sieti. Navyše neúplné informácie o zmenách topológií v systéme zabránili efektívemu dohľadu nad jeho spoľahlivosťou.

Niekoľko klúčových prenosových 345kV vedení v severnom Ohiu sa zastavilo kvôli kontaktu so stromami. To spustilo kaskádové preťaženie ďalších 345 kV a 138 kV vedení, čo

viedlo k nekontrolovanému zlyhaniu celej siete. Celkovo bolo 61 800 MW stratených po zastavení 508 generátorov v 265 elektrárňach[9, p. 3-20].

#### 4.3.3 Stuxnet červ (2010)

Stuxnet bol počítačový červ pre OS Microsoft Windows objavený v júli 2010, ktorý špecificky útočil na priemyselný software a zariadenia[9, p. 3-20]. Bol schopný infikovať počítače využívajúce Microsoft Windows od Win2000 po Windows 7 a Windows Server 2008 R2.

Ak infikovaný hostiteľ neboli ciel útoku, ale iba "medzistanica", počiatočná infekcia mohla nahrať rootkit, ktorý automaticky načíta malware pri bootovaní zariadenia a ponechá ho neodhalený kým sa útočník nerozhodne zaútočiť na svoj ciel z infikovaného zariadenia.

Ak infikované zariadenie obsahovalo software Siemens Simatic, existovali metódy využívajúce predvolené poverenia SQL servera, ktoré umožňovali inštaláciu malware do SQL (WinCC). Tiež mal schopnosť prepísať ovládač používaný na komunikáciu s PLC S7 (Programmable Logic Controller) a účinne vytvoriť man-in-the-middle útok, ktorý by umožnil zmenu kódu bežiaceho v PLC bez toho, aby to detekovali používateelia systému. Stuxnet bol využívaný na prenos užitočnej záťaže nielen pre konkrétnie riadiace systémy, ale tiež pre konkrétné konfigurácie riadiacich systémov zahŕňajúc jedinečné čísla modelov PLC a dodávateľov pripojených zariadení. Hľadal konfiguráciu cieľového systému (PLC S7-315-2 / S7-417) a keď bola nájdená, vložil bloky kódu do cieleného PLC, ktorý mohol prerušiť alebo zmeniť prebiehajúce procesy. Stuxnet používal infikované PLC na sledovanie špecifického správania monitorovaním zbernice.

Stuxnet mal dva hlavné ciele svojho útoku na užitočnú záťaž:

- Zvyšovanie a znižovanie rýchlosťi centrifugy nad normálne hodnoty.

Normálne centrifugy rotujú pri rýchlosti 807-1210 Hz. Po nainštalovaní Stuxnetu, začal monitorovať rýchlosť a veľmi priležitosne zmenil rýchlosť na veľmi rýchlu (1410 Hz), hned na to na veľmi pomalú (2 Hz) a opäť na rýchle (1064 Hz). To malo za následok fyzické poškodenie súčasti systému.

- Zvyšovanie tlaku vo vnútri odstrediviek nad normálne hodnoty.

Stuxnet taktiež ovplyvňoval tlak vo vnútri odstrediviek. Tento proces zahrňal uzatváranie tlakových spätných ventilov, ktoré boli počas prevádzky normálne otvorené, čo spôsobovalo nebezpečné zvyšovanie tlaku vo vnútri zariadení[4][5].

#### 4.3.4 Útok na ukrajinskú elektráreň (2016)

Týždeň pred Vianocami v roku 2016 sa podarilo skupine útočníkov napadnúť elektrickú prenosovú stanicu Pivnichna severne od mesta Kyjev a zablokovať dodávku elektrickej energie do časti Kyjeva a príslahlého okolia na jednu hodinu. Útok začal chvíľu pred poľnocou 17. decembra a dodávka elektriny bola obnovená po jednej hodine ráno.

Vedúci výzkumu bezpečnosti informačných systémov na Ukrajine Oleksii Yasynskyi povedal, že útočníci zjavne testovali novú techniku útoku a ich cielom bolo sabotovať systém.

Pri útoku bol použitý nový nástroj na infikovanie a narušenie prevádzky SCADA systémov - Win32/Industroyer[5].

## Win32/Industroyer malware

Win32/Industroyer je sofistikovaný malware navrhnutý na narušenie prebiehajúceho procesu v zariadeniach SCADA systémov.

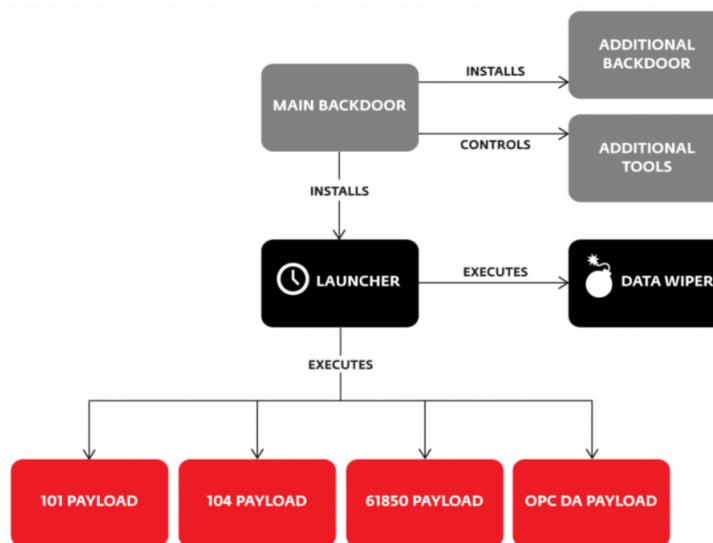
Industroyed je v súčasnosti veľmi nebezpečná hrozba, nakoľko je schopný priamo ovládať spínače a ističe na elektrických stanicach. Na tento účel používa priemyselné komunikačné protokoly, ktoré sa bežne využívajú v SCADA systémoch. Win32/Industroyer zahŕňa štyri komunikačné protokoly:

- IEC 60870-5-101
- IEC 60870-5-104
- IEC 61850 - Goose
- OLE na riadenie procesov (Process Control Data Access - OPC DA)

Problém v týchto protokoloch je, že boli navrhnuté ešte v časoch, keď bola priemyselná IoT prevádzka izolovaná od bežnej a nebolo možné na ňu zaútočiť z verejnej IP siete. Protokoly preto neboli navrhnuté na podporu bezpečnosti a autentizácie, čo znamená, že útočník v súčasnosti nemusí hľadať zraniteľnosti protokolu, stačí mu vytvoriť malware, ktorý bude komunikovať v danom štandarde. Win32/Industroyer je modulárny malware. Jeho hlavnou zložkou je "zadný vchod" (backdoor) do systému, ktorý útočníci využívajú na riadenie útoku: inštaluje a riadi ostatné komponenty a pripája sa k vzdialenému serveru na prijímanie príkazov a reportovanie útočníkovi.

Okrem toho implementuje DDoS útok proti špecifickým skupinám ochranných relé, konkrétnie sa zameriava na sériu Siemens SIPROTEC.

Všeobecne platí, že užitočné zataženie funguje v etapách, ktorých cieľom je mapovanie siete a následné prijímanie, a vydávanie príkazov pre konkrétnie riadiace komponenty systému, viď obrázok 4.3.



Obr. 4.3: Etapy užitočného zataženia[5]

Zadný vchod do systémov celkom priamočiaro spojuje malware so vzdialeným serverom využívajúc HTTPS protokol na zasielanie príkazov od útočníka. Zaujímavé je, že útočníci si

môžu nadefinovať špecifickú hodinu, kedy bude tento vchod do systému aktívny, napríklad mimo pracovnej doby. Keď sa vytvorí spojenie so serverom, v POST žiadosti sú zaslané jednotlivé údaje[5].

Je podporovaných niekoľko základných príkazov:

- 0: spustenie procesu
- 1: spustenie procesu pod špecifickým užívateľom s patričnými autentizačnými údajmi
- 2: stiahnutie súboru zo servera
- 3: zkopírovanie súboru
- 4: spustenie `shell` príkazu
- 5: spustenie `shell` príkazu pod špecifickým užívateľom s patričnými autentizačnými údajmi
- 6: zastavenie
- 7: zastavenie služby
- 8: zastavenie služby pod špecifickým užívateľom s patričnými autentizačnými údajmi
- 9: spustenie služby pod špecifickým užívateľom s patričnými autentizačnými údajmi
- 10: nahradenie hodnoty registra `image path` pre danú službu

Pri užitočnom zaťažení komponent protokolu IEC 104 obsahuje konfiguračný súbor pre jednotlivé komponenty IP adresu stanice, cieľový port, ASDU adresu, hodnotu prepínača (zap./vyp.), hodnotu zmeny (0/1) a operáciu (iteračný typ pre IOA: rozsah, sekvenciu alebo posunutie).

Po spustení vytvorí vlákno pre každú sekciu staníc definovaných v konfiguračnom súbore. V každom vlákne komunikuje s konkrétnou adresou využívajúc protokol IEC 104. Po vytvorení spojenia s daným zariadením začne zasielať príkazy s ASDU adresou definovanou v súbore kde interaguje s IOA využívajúc SCO (Single command) typ[5].

Iteračné typy IOA:

- Rozsah - využívajúc mód rozsahu môže útočník odhaliť všetky dostupné IOA v cieľom zariadení. Protokol IEC 104 štandardne neposkytuje žiadnu obdobnú metódu na získanie takých údajov.

Keď útočník získa potrebný rozsah IOA, môže cez ne začať iterovať a zasielať príkazy `select and execute`, aby mohol zmeniť typ daného IOA a potvrdiť, že patrí do typu SCO.

Keď je preiterované cez všetky potrebné IOA, malware začne rozposielat príkazy `select and execute` v nekonečnom cykle, kde postupne zapína a vypína zariadenia.

- Sekvencia - tento mód môže byť použitý jedine ak útočník vie hodnoty všetkých IOA, ktoré využívajú SCO typ. Malware môže hneď začať nekonečný cyklus v ktorom bude posielat `select and execute` príkazy IOA definovaným v konfiguračnom súbore.
- Posunutie - je veľmi podobné módu rozsahu. Iteruje sa cez celý rozsah IOA, kde nový rozsah je vypočítaný pridaním hodnoty posunutia k pôvodnej hodnote rozsahu[5].

## **4.4 Zhrnutie**

SCADA systémy v súčasnosti čelia mnohým rizikám. Je to veľký problém najmä z dôvodu, že systémy sú väčšinou súčasťou kritickej infraštruktúry a útoky na ne môžu mať veľký dopad aj na bežných obyvateľov alebo ekosystém ako taký. Je preto potrebné vedieť útoky včas detektovať a vedieť sa im efektívne brániť.

V tejto kapitole bol uvedený základný prehľad bezpečnostných rizík na SCADA systémy spolu s popisom jednotlivých typov útokov. Taktiež bolo uvedených niekoľko príkladov zaznamenaných útokov spolu s dôsledkami, ktoré spôsobili.

V ďalšej časti mojej práce budem vychádzať z popisu jednotlivých rizík, ktorým SCADA systémy čelia a vytvorím simulačné prostredie na ich testovanie a sledovanie reakcií systému na ne.

# Kapitola 5

## Implementácia

V kapitole 3 boli uvedené rôzne premyselne využívané programy a nástroje na simuláciu a riadenie prevádzky SCADA systémov. V kapitole 4 bola popísaná bezpečnosť týchto systémov spolu s rôznymi typmi útokov, ktoré sú pre ne hrozobou. Z nadobudnutých informácií bolo navrhnuté a implementované simulačné prostredie pre protokoly DLMS/COSEM a IEC 104 na testovanie rôznych typov útokov a sledovanie reakcií systému na ne. V nasledujúcej časti bude podrobne popísaná implementácia prostredia spolu s testovaním.

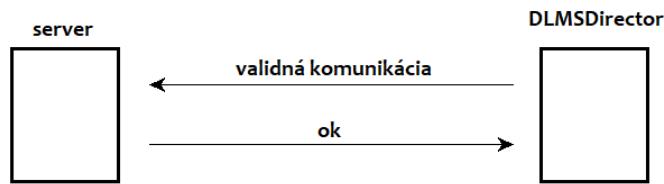
### 5.1 Návrh prostredia

Prvý návrh prostredia pre protokol DLMS/COSEM spočíval vo vytvorení vlastného objektu za pomoci C++ knižnice od spoločnosti GuruX, ktorý sa pripojí do existujúcej siete a bude slúžiť ako zadný vchod (backdoor) do systému. Útoky tohto typu boli popísané v kapitole 4. Avšak s novo-vytvoreným objektom nechcel program DLMS Director spolupracovať a nebolo možné vykonávať testy. Po komunikácií so spoločnosťou GuruX som zistil, že program nepodporuje komunikáciu so špecifickými objektami a v čase písania tejto práce nebol program aktualizovaný o túto funkcionality.

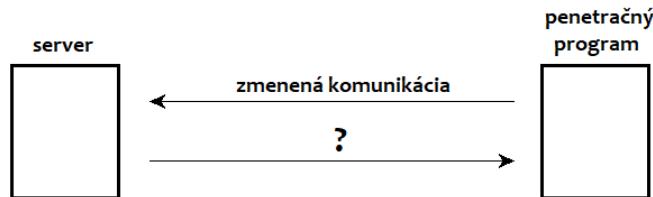
Druhý návrh, ktorý bol spoločný pre oba skúmané protokoly (DLMS/COSEM a IEC 104) pozostával z vytvorenia programu, ktorý bude simulovať stranu klienta, a bude serveru zasielať nevalidné alebo zmenené príkazy, a budú sa sledovať reakcie systémov. Komunikáciu program nevytvára, iba dalej zasiela príkazy z odchytenej a následne zmenenej komunikácie. Výhodou je, že pracuje na štrvtej (transportnej) vrstve TCP/IP modelu a vďaka tomu nemusí rozlišovať, s akým protokolom komunikuje. Iba posielá ďalej príkazy ktoré dostane na vstupe, a čaká na odpoveď. Na obrázku 5.1 je ukážka validného prostredia, z ktorého sa vychádzalo pri testovaní zmien v príkazoch, ako je možné vidieť na obrázku 5.2. Validácia vzorových systémov preukázala, že programy sú schopné simulovať komunikáciu odpovedajúcu štandardom protokolov a je preto možné ich použiť na testovacie účely a očakávať obdobné reakcie aj od reálnych zariadení.

### 5.2 Implementácia simulačného prostredia

Ako bolo spomenuté vyššie, pre účely simulácie útokov bol vytvorený program klienta, ktorý je schopný zasielať strane servera rôzne zmenené alebo nevalidné príkazy. Program je implementovaný v jazyku C++ a funguje ako TCP klient, ktorý sa pripojí na server. Na vstup dostáva binárny súbor obsahujúci jednotlivé príkazy, ktoré sa budú serveru odosielat.



Obr. 5.1: Vzorový systém



Obr. 5.2: Testovací systém

Príkazy boli získané analýzou zachytenej komunikácie medzi reálnymi stanicami klient a server. Analýza prebiehalo pomocou nástroja **Wireshark**<sup>1</sup>. V nástroji bol zobrazený TCP stream zachytenej komunikácie a následne odfiltrovaný iba na správy zasielané stranou klienta. Analýza však nemusí prebiehať výhradne pomocou nástroja Wireshark. Je možné použiť akýkoľvek obdobný nástroj. Napríklad pre analýzu priamo z príkazovej riadky je možné použiť nástroj **tshark**<sup>2</sup>, príkazom `tshark -r input.pcap -q -z follow,tcp,raw,0 > output`. Príkaz zobrazí do zadанého súboru binárnu podobu zachytenej komunikácie, ktorú je možné ďalej upravovať.

Odfiltrované príkazy boli uložené do binárneho súboru, ktorý sa ďalej posielal ako vstup simulačnému programu. Vďaka tomu, že program pracuje priamo s binárnou podobou príkazov, je možné ich ľubovoľne upraviť, a vytvárať so serverom neštandardnú komunikáciu. Odchytenu komunikáciu je možné ďalej analyzovať a zistovať reakcie systému na rôzne typy útokov.

Pri načítavaní jednotlivých príkazov zo súboru je potrebné, aby boli od seba oddelené špecifickým znakom, aby program rozoznal koniec jedného a začiatok druhého. Pôvodne bola zvolená hexadecimálna hodnota `0x0a`, čo je hodnota pre nový riadok. Túto hodnotu však bolo nutné zmeniť, pretože `0x0a` sa v jednotlivých príkazoch často vyskytuje aj ako hodnota 10 označujúca objekt alebo namerané dátá. Pri načítavaní to spôsobovalo rozdelenie jedného príkazu na dva. Po tomto zistení som sa pokúšal nájsť inú hexadecimálnu hodnotu ktorá sa nevyskytuje v príkazoch ani jedného z protokolov. Spoločnú hodnotu sa mi nepodarilo nájsť a preto za príkazy protokolu DLMS bola pridaná hodnota `0x3f` a za príkazy protokolu IEC 104 hodnota `0x7e`. Príkazy boli upravované v nástroji **Sublime Text 3**<sup>3</sup> po zobrazení hexadecimálneho kódovania.

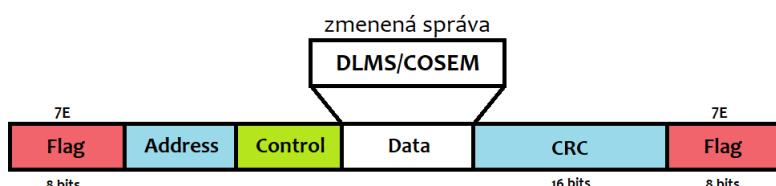
<sup>1</sup>Wireshark <https://www.wireshark.org>

<sup>2</sup>TShark <https://www.wireshark.org/docs/man-pages/tshark.html>

<sup>3</sup>Sublime Text <https://www.sublimetext.com>

### 5.2.1 Popis behu programu

Program pri spustení načíta vstupné argumenty, skontroluje správnosť ich kombinácie a validitu zadaných hodnôt. Keď je všetko v poriadku, zo vstuпу sa načítajú jednotlivé príkazy, ktoré sa budú odosielat. Pri spustení programu je potrebné zadať, s akým protokolom má program pracovať, DLMS alebo IEC 104. Je to nutné z dôvodu, že program rozlišuje podľa akého znaku má jednotlivé príkazy od seba odlišiť, ako už bolo spomínané vyššie. Protokol DLMS/COSEM taktiež využíva na prenos príkazov protokol HDLC, ktorý slúži na spoľahlivý prenos dát po sieti. Súčasťou dátového rámca protokolu sú dva byty vyhradené na tzv. kontrolný súčet CCITT-CRC16. Ukážka rámca je na obrázku 5.3. Príkazy protokolu DLMS obsahujú prvý a posledný byte režijný, označujúci začiatok a koniec príkazu, hodnota je 0x7E. Následuje adresa, typ príkazu, samotné data a dva byty s hodnotou spomínaného kontrolného súčtu. Pre každý zmenený príkaz je potrebné nanovo vypočítať jeho kontrolný



Obr. 5.3: HDLC zapúzdrenie

súčet. Ten sa počíta z celého rámca, okrem prvého a posledného (režijného bytu). Ak by totiž súčet nesúhlasil s dátami, strana príjemcu (servera) by správu zahodila a prestala by ďalej komunikovať. Implementácia funkcií na výpočet súčtu bola prevzatá zo stránok stackoverflow.com<sup>4</sup> a www.zorc.breitbandkatze.de<sup>5</sup>. Program načítava jednotlivé príkazy do pomocnej premennej `message` typu `std::vector<std::string>`. Do premennej `answer`, ktorá je rovnakého typu sa ukladajú odpovede od servera.

Po načítaní jednotlivých príkazov sa vytvorí TCP spojenie a začína komunikácia. Príkazy sa odosielajú v cykle. Pre protokol DLMS sa vždy odošle jeden príkaz a očakáva sa jedna odpoveď, napäťko protokol komunikuje synchronne. Protokol IEC104 je o niečo komplikovanejší. Oproti protokolu DLMS, kde ide o synchronnu komunikáciu 1:1 (jedna správa : jedna odpoveď), to tak pri protokole IEC 104 nefunguje, komunikácia prebieha asynchronne. Niekedy je potrebné odoslať niekoľko správ bez odpovedi, alebo očakávať niekoľko odpovedí po jednej správe. Problém bol vyriešený využitím funkcie `select()` spolu s `timeoutom`. Funkcia je volaná v cykle. Ak sa podarí prijať správu, cyklus prijímania sa opakuje, ak vyprší timeout, z cyklu sa vyskočí a odosielajú sa nová správa. Komunikácia je implementovaná pomocou bsd schránek. Odosielanie a prijímanie správ zabezpečujú funkcie `send()` a `recv()`.

### 5.2.2 Návod na použitie

#### Preloženie a spustenie programu

Program je nutné pred prvým spustením preložiť príkazom `make` v koreňovom adresári projektu. Keď je program preložený, môžeme ho spustiť. Spúšťa sa nasledovným príkazom:

<sup>4</sup>Stackoverflow <https://stackoverflow.com/questions/7983862/calculating-fcscrc-for-hdln-frame> [Online: Marec 2018]

<sup>5</sup>CRC <http://www.zorc.breitbandkatze.de/crc.html> [Online: Marec 2018]

```
./client [-h] [a <adresa serveru>] [-p port] [-i vstupný súbor] [-o výstupný súbor] [-dlms] [-iec]
```

### Parametre programu

Program `client` je možné spustiť s niekoľkými parametrami:

- h --help - zobrazí nápovedu
- a --address - IP adresa serveru, s ktorým bude klient komunikovať
- p --port - port, na ktorom komunikuje server
- i --input - vstupný binárny súbor s jednotlivými príkazmi
- o --output - súbor, kam sa budú ukladať odpovede od serveru
- 1 --iec - komunikácia s iec 104 serverom
- 2 --dlms - komunikácia s dlms serverom

Vždy musí byť zadaný aspoň jeden z parametrov `-h --help`, `-a --address` | `--dlms`, `-o --output`. Nemôžu byť zadané oba naraz, ani z jednej kombinácie. Navyše parameter `-h --help` nemôže byť kombinovaný so žiadnym ďalším parametrom. Pri absencií parametra `-p --port` sa nastaví predvolená hodnota podľa zadaného protokolu. 2404 pre IEC 104 a 4059 pre DLMS. V prípade nezadanie parametrov `-i --input` a `-o --output` sa využije štandardný vstup a výstup.

### Príklady použitia

```
./client -h - vypíše nápovedu
```

```
./client -a 192.168.137.189 -p 4060 --dlms -i data1 -o output1 - program bude komunikovať s DLMS serverom na porte 4060 pomocou protokolu TCP. Príkazy si načíta zo súboru data1 a odpovede od servera zapíše do súboru output1. Ak takýto súbor neexistuje, vytvorí ho.
```

## 5.3 Testovanie

Pri validácii programu klienta a simulačného prostredia ako takého boli využívané zachytené komunikácie medzi jednotlivými simulátormi SCADA prevádzky. Zachytená komunikácia bola v simulačnom prostredí opäťovne vytvorená a zachytená. Po overení, že všetky dvojice komunikácií sú totožné, bol program prehlásený za validný.

Na obrázku 5.4 je ukážka časti zachytenej originálnej komunikácie medzi simulačnými nástrojmi pre klienta a server, konkrétnie ide o protokol DLMS/COSEM. Na obrázku 5.5 je možné vidieť tú istú komunikáciu vygenerovanú pomocou simulačného programu klienta. Obe komunikácie boli následne porovnané programom `diffchecker`<sup>6</sup> a bolo overené, že komunikácie sú totožné.

Pri testovaní protokolov bola vždy vytvorená testovacia topológia so vzorovou komunikáciou. Komunikácia bola pri každom teste čiastočne zmenená a boli sledované reakcie systému, ako je možné vidieť na obrázku 5.6.

---

<sup>6</sup>Diffchecker <https://www.diffchecker.com> [Online: Marec 2018]

```

7ea007032393bf327e

7ea01e2303737bcf8180120501800601800704000000108040000001533b7e
7ea04403231056b7e6e6006036a1090607608574050801018a0207808b0760857405080201ac0a
800870617373776f7264be10040e0100000065f1f0400001e1dfffb86b07e
7ea037230330d4c9e6e7006129a109060760857405080101a203020100a305a103020100be1004
0e0800065f1f0400001e1d04000007e0a67e
7ea019032332dfebe6e600c001c1000f0000280000ff020091537e
7ea87e230352343ce6e700c402c1000000001008203f10116020412000f110209060000280000
ff0202010b02030f0116030002030f0216030002030f0316030002030f0416030002030f051603
0002030f0616030002030f0716030002030f0816030002030f0916030002030f0a16030002030f
0b160300010602028c4a7e
7ea007032351a1d77e
7ea87e23037400780f01160102020f02160102020f03160102020f04160102020f05160102020f
0616010204120017110009060000160000ff0202010902030f0116030002030f0216030002030f
0316030002030f0416030002030f0516030002030f0616030002030f0716030002030f08160300
02030f091603000179337e
7ea007032371a3f67e
7ea87e2303961cbc0002041200011100090600002a0000ff0202010202030f0116030002030f02
16030001000204120001110009060100000200ff0202010202030f0116030002030f0216030001
00020412000111000906010100000ff0202010202030f0116030002030f021603000100020412
0001110009060101b2d47e

```

Obr. 5.4: Originálna komunikácia medzi klientom a serverom

```

7ea007032393bf327e

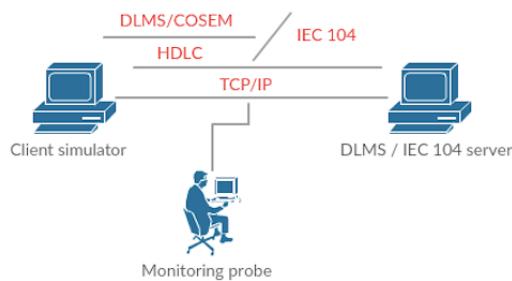
7ea01e2303737bcf8180120501800601800704000000108040000001533b7e
7ea04403231056b7e6e6006036a1090607608574050801018a0207808b0760857405080201ac0a
800870617373776f7264be10040e0100000065f1f0400001e1dfffb86b07e
7ea037230330d4c9e6e7006129a109060760857405080101a203020100a305a103020100be1004
0e0800065f1f0400001e1d04000007e0a67e
7ea019032332dfebe6e600c001c1000f0000280000ff020091537e
7ea87e230352343ce6e700c402c1000000001008203f10116020412000f110209060000280000
ff0202010b02030f0116030002030f0216030002030f0316030002030f0416030002030f051603
0002030f0616030002030f0716030002030f0816030002030f0916030002030f0a16030002030f
0b160300010602028c4a7e
7ea007032351a1d77e
7ea87e23037400780f01160102020f02160102020f03160102020f04160102020f05160102020f
0616010204120017110009060000160000ff0202010902030f0116030002030f0216030002030f
0316030002030f0416030002030f0516030002030f0616030002030f0716030002030f08160300
02030f091603000179337e
7ea007032371a3f67e
7ea87e2303961cbc0002041200011100090600002a0000ff0202010202030f0116030002030f02
16030001000204120001110009060100000200ff0202010202030f0116030002030f0216030001
00020412000111000906010100000ff0202010202030f0116030002030f021603000100020412
0001110009060101b2d47e

```

Obr. 5.5: Opäťovne vygenerovaná komunikácia pomocou programu klienta

Pri testovaní bezpečnostných incidentov na systémy komunikujúce pomocou protokolov DLMS/COSEM a IEC 104 som postupoval v niekoľkých krokoch:

1. Vytvorenie simulačného prostredia, ktoré sa chová ako reálny systém.
2. Vygenerovanie validnej komunikácie, ktorá bude slúžiť ako vzorová.
3. Pozmenenie príkazov z ochytenej vzorovej komunikácie.
4. Opäťovné vygenerovanie s patričnými zmenami.
5. Sledovanie reakcií systému.



Obr. 5.6: Sledovanie reakcií systému zo zachytenej komunikácie

### 5.3.1 DLMS/COSEM

Pri testovaní protokolu DLMS bola využívaná open source knižnica pre jazyk C++ poskytovaná spoločnosťou GuruX. Knižnica je voľne dostupná v Github repozitári<sup>7</sup>. Podrobnej inštalácia knižnice je popísaná medzi prílohami v časti B. Knižnica obsahuje vzorový program pre server, ktorý bol následné upravený pre potreby testovania. Pôvodná implementácia zahŕňa štyry rôzne typy DLMS serverov, pričom každý komunikuje na inom porte.

Pri testovaní bol ponechaný iba jeden server využívajúci logické mená (Logical Names), komunikujúci na porte 4060. Taktiež bola povolená možnosť hodnoty v atribútoch jednotlivých objektov aj nastavovať. Pôvodne ich bolo možné iba vzdialene čítať. Čo sa týka knižnice samotnej, implementácia dátových objektov nepodporovala dynamickú zmenu nameraných hodnôt za behu programu. Aby to viac odpovedalo reálnemu systému, v súbore `GXDLMSSData.cpp` bola zmenená implementácia `CGXDLMSSData::GetValue` metódy, ktorá vračala nameranú hodnotu. Do metódy bola pridaná podmienka, ktorá vráti novú hodnotu ak od predošlého dotazu uplynulo aspoň 10 sekúnd. Uplynutý čas od posledného dotazu sa počítava pomocou premennej typu `std::time_t`, pričom sa počítava rozdiel jej hodnoty (od posledného dotazu) k aktuálnemu času `std::time(nullptr)`. Nové hodnoty sa získavajú dvoma spôsobmi. Sú generované náhodne funkciou `rand()` alebo sú načítavané zo súboru. Voľba konkrétneho spôsobu je na základe konfiguračného súboru, ktorý je uložený v `Gurux.DLMS.cpp-master/conf`. Obsah konfiguračného súboru môže byť napríklad:

- `file = ../../values` - nastavenie čítania zo súboru `values` spolu s cestou k nemu
- `rand = 1000` - nastavenie náhodného generovania hodnôt z rozsahu 0..1000

Pri analýze zachytených .pcap súborov bol použitý dissector pre program Wireshark vytvorený pánom Ing. Petrom Matouškom, Ph.D., M.A.<sup>8</sup> Dissector z načítanej binárnej postupnosti vyberie časti pre HDLC zapúzdrenie a pre samotné dátá protokolu DLMS/COSEM. Ukážku je možné vidieť na obrázku 5.7.

### Testy

#### Validná komunikácia:

Na začiatku testovania bol vytvorený vzorový .pcap súbor obsahujúci validnú (vzorovú) komunikáciu protokolu DLMS/COSEM. Zasielané príkazy boli následne systematicky menené a boli sledované reakcie systému.

<sup>7</sup>DLMS knižnica - Github <https://github.com/Gurux/Gurux.DLMS.cpp> [Online: Marec 2018]

<sup>8</sup>DLMS Dissector <https://github.com/matousp/dlms-analysis> [Online: Apríl 2018]

No.	Time	Source	Destination	Protocol	Length	Info												
2	0.000076915	192.168.137.189	192.168.137.1	TCP	66	4060 → 54409												
3	0.013350501	192.168.137.1	192.168.137.189	TCP	54	54409 → 4060												
4	0.502330050	192.168.137.1	192.168.137.189	TCP	63	54409 → 4060												
5	0.502451551	192.168.137.189	192.168.137.1	TCP	54	4060 → 54409												
6	0.502668253	192.168.137.189	192.168.137.1	TCP	86	4060 → 54409												
7	0.519981367	192.168.137.1	192.168.137.189	DLMS	99	DLMS AARQ Association Request												
8	0.520267045	192.168.137.189	192.168.137.1	DLMS	111	DLMS AARE Association Response												
9	0.561860199	192.168.137.1	192.168.137.189	TCP	54	54409 → 4060												
10	1.759392893	192.168.137.1	192.168.137.189	DLMS	81	GetRequestNormal												
11	1.760490138	192.168.137.189	192.168.137.1	DLMS	182	GetResponseWithDBlock												
12	1.762724104	192.168.137.1	192.168.137.189	HDLC over TCP	63	S-Frame, From												
	13	1.762838017	192.168.137.189	192.168.137.1	HDLC over TCP	182	I-Frame, From											
► Internet Protocol Version 4, Src: 192.168.137.189, Dst: 192.168.137.1																		
► Transmission Control Protocol, Src Port: 4060, Dst Port: 54409, Seq: 90, Ack: 82, Len: 128																		
▼ HDLC over TCP																		
Flag: 0x7e																		
► Frame Format: 0xa87e																		
Destination Address: 33																		
Source Address: 3																		
► Control Field: 0x52																		
Header Checksum: 0x8c89																		
► Information																		
Frame Checksum: 0x8c4a																		
Flag: 0x7e																		
▼ DLMS/COSEM																		
Type: GetResponse (0xc4)																		
GetResponse: GetResponseWithDBlock (0x02)																		
Invoke ID and Priority: 0xc1																		
► DataBlock-G																		
0000	f6	06	69	81	3a	03	94	39	e5	ae	7c	91	08	00	45	00	..i...9	... ...E.
0010	00	a8	91	21	40	00	40	06	15	1f	c0	a8	89	bd	c0	a8	....!@.	.....
0020	89	01	0f	dc	d4	89	34	37	51	e3	bf	dd	76	4f	50	18	.....47	Q...vOP.
0030	00	e5	b4	41	00	00	7e	a8	7e	21	03	52	8c	89	e6	e7	...A..~.	~!R....
0040	00	c4	02	c1	00	00	00	00	01	00	82	03	f1	01	16	02	.....	.....
0050	04	12	00	0f	11	02	09	06	00	00	28	00	00	ff	02	02	.....	.....
0060	01	0b	02	03	0f	01	16	03	00	02	03	0f	02	16	03	00	.....	.....
0070	02	03	0f	03	16	03	00	02	03	0f	04	16	03	00	02	03	.....	.....
0080	0f	05	16	03	00	02	03	0f	06	16	03	00	02	03	0f	07	.....	.....
0090	16	03	00	02	03	0f	08	16	03	00	02	03	0f	09	16	03	.....	.....
00a0	00	02	03	0f	0a	16	03	00	02	03	0f	0b	16	03	00	01	.....	.....

Obr. 5.7: Analýza zachytenej komunikácie

**Zachytenie komunikácie:** dlms.pcap

**Test č. 1:**

**Zmena:** Data obsahovali zlú velkosť pri otváraní asociácie v príkaze AARQ – Association Request. 11 bytov namiesto 9 bytov: 0x09 → 0x0b.

**Reakcia:** Server si zmenu vôbec nevšimal a zaslal späťne potvrzovaciu správu o zistení deviatich pripojených zariadení.

**Zachytenie komunikácie:** data1.pcap

**Test č. 2:**

**Zmena:** Dĺžka OID prvého požiadavku bola zle zadaná v príkaze AARQ – Association Request. 9 bytov namiesto 7 bytov: 0x07 → 0x09.

**Reakcia:** Server si opäť zmenu nevšimol. Žiadost o asociáciu prijal a odpoveď obsahovala OID zmenené na správne.

**Zachytenie komunikácie:** data2.pcap

**Test č. 3:**

**Zmena:** Dĺžka OID prvého požiadavku bola zle zadaná v príkaze AARQ – Association Request. 3 byty namiesto 7 bytov: 0x07 → 0x03.

**Reakcia:** Reakcia bola rovnaká ako pri predchádzajúcom teste.

**Zachytenie komunikácie:** data3.pcap

#### **Test č. 4:**

**Zmena:** Zlý typ paketu. Pôvodná správa bola typu **Get-Request** a bola následne zmenená na **Set-Request**:  $0xc0 \rightarrow 0xc1$ . Zmena bola vykonaná iba v poli obsahujúcom typ. Zvyšok paketu ostal pôvodný pre **Get-Request**.

**Reakcia:** Server prijal žiadosť na nastavenie hodnoty, vykonal zmenu a odpovedal potvrzovacou správou o úspešnom nastavení.

**Zachytenie komunikácie:** data4.pcap

#### **Test č. 5:**

**Zmena:** Zlý typ paketu. Namiesto **Get-Request** sa poslalo **Set-Response**:  $0xc0 \rightarrow 0xc5$ .

**Reakcia:** Server nevedel, ako má zareagovať a na správu neodpovedal.

**Zachytenie komunikácie:** data5.pcap

#### **Test č. 6:**

**Zmena:** Chybný OBIS kód požadovaného objektu. Požaduje sa  $0.0.40.0.0.1$  namiesto  $0.0.40.0.0.255$ :  $0xff \rightarrow 0x01$ .

**Reakcia:** Server žiadosť prijal a pokúsil sa vrátiť požadovanú odpoveď. Odosielal ale prázdnú správu, napokoľko požadovaný objekt nepoznal.

**Zachytenie komunikácie:** data6.pcap

#### **Test č. 7:**

**Zmena:** Zmena požadovaného atribútu objektu. Požaduje sa atribút 10 namiesto 1:  $0x02 \rightarrow 0x0a$ .

**Reakcia:** Server na zmenu reagoval štandardnou odpoveďou a vrátil hodnotu požadovaného atribútu. Tento test bol zameraný skôr na typ útoku, kedy útočník zachytí prebiehajúcu komunikáciu, čiastočne zmení obsah príkazu, ale ponechá ho validný a pošle pôvodnému adresátovi. Server správu príjme a odpovie, ale riadiacej stanici príde iná odpoveď akú požadovala. Ak nie je odpoveď dostatočne overená, že je to naozaj tá, ktorá bola očakávaná, môže to znamenať napríklad zlú informovanosť o nameraných hodnotách.

**Zachytenie komunikácie:** data7.pcap

#### **Test č. 8:**

**Zmena:** Zasielané správy odpovedali správam z testovacej komunikácie pre protokol DLMS-Director (DLMSDirector.pcap). Test sa zameriaval na sledovanie otvorenia spojenia a testovanie autentizácie. Komunikácia bola analyzovaná v nástroji **Wireshark**, kde bolo sledované v akej podobe sa prenášajú autentizačné údaje po sieti.

**Reakcia:** Testovanie autentizácie bolo zamerané hlavne na sledovanie prebiehajúcej komunikácie. Bolo overené, že autentizačné klúče sa prenášajú v otvorennej podobe. Útočník si tak môže jednoducho ochytiť komunikáciu, prečítať si klúč a vytvoriť svoje vlastné spojenie so systémom.

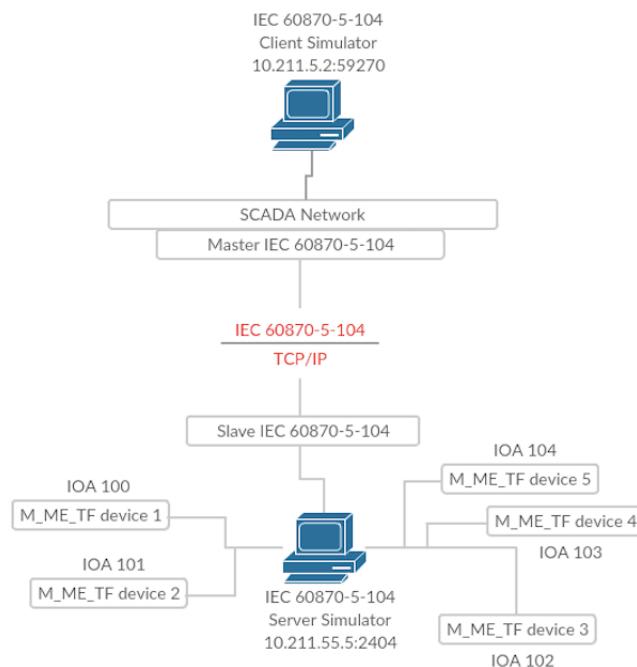
**Zachytenie komunikácie:** auth.pcap

Testovanie preukázalo, že server sa pokúša na príkaz vždy odpovedať nehľadiac na to, či požadovaný objekt alebo atribút pozná. Spojenie ukončuje iba v prípade nevalidného

príkazu. Veľmi nebezpečný je tiež nešifrovaný prenos hesla po sieti, čo umožňuje útočníkom jednoduchý prístup k autentizačným údajom systému.

### 5.3.2 IEC 104

Pri testovaní protokolu IEC 104 bol využívaný program QTester 104 a IEC 60870-5-104 Server Simulator. Testovacia topológia pozostávala z jedného klienta a jedného servera, ku ktorému bolo pripojených päť objektov typu **Measured Short Float**. Objekty slúžia na ukladanie nameraných hodnôt s pohyblivou rádovou čiarkou. Testovaciu topológiu je možné vidieť na obrázku 5.8. Testovanie protokolu IEC 104 bolo o niečo náročnejšie ako pri



Obr. 5.8: Testovacia topológia pre protokol IEC 104

protokole DLMS/COSEM z niekoľkých dôvodov. Simulačné programy neumožňovali pridať automatizáciu zmien nameraných hodnôt, a tak bolo potrebné všetky zmeny vykonávať ručne. Z rovnakého dôvodu simulačné prostredie neodpovedalo reálnemu systému do takej miery ako pri DLMS/COSEM. Asynchronny charakter protokolu tiež trochu skomplikoval opäťovné generovanie komunikácie z už odchytenej, ako bolo spomínané vyššie. Posledný problém, ktorý som mal oproti protokolu DLMS/COSEM, bol, že nebolo možné pridať autentizáciu na server a preto ju nebolo možné ani testovať.

### Testy

#### Validná komunikácia:

Na začiatku testovania bol vytvorený vzorový .pcap súbor obsahujúci validnú (vzorovú) komunikáciu protokolu IEC 104. Zasielané príkazy boli následne systematicky menené a boli sledované reakcie systému.

**Zachytenie komunikácie:** iec104.pcap

### **Test č. 1:**

**Zmena:** Test na buffer overflow. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli `Value` zmenená hodnota z 2 na NaN: 0x40 → 0xff ffff fff.

**Reakcia:** Hodnota bola nad rámec povoleného maxima a spôsobila, že dĺžka príkazu bola väčšia ako je štandard, čo spôsobilo, že príkaz bol nevalidný. Server odpovedal hláškou `ERR prefix 1 bytes` spolu s informáciou, že hodnota bola nastavovaná na `nan - Not-A-Number` hodnotu.

**Zachytenie komunikácie:** data\_buffer\_overflow.pcap

### **Test č. 2:**

**Zmena:** Test na zmenu adresy objektu. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli `Addr` zmenená hodnota adresy cieleného objektu z 1 na 65281: 0x0100 → 0x01ff.

**Reakcia:** Server si zmenu všimol a nevedel nájsť dotazovaný objekt. Odpovedal správou `UkComAddrASDU_NEGA`.

**Zachytenie komunikácie:** data\_wrong\_addr.pcap

### **Test č. 3:**

**Zmena:** Test na zmenu dôvodu prenosu. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli `CauseTx` zmenená hodnota z 6 na 10: 0x06 → 0x0a (Act (6) na ActTerm (10)).

**Reakcia:** Server si zmenu všimol a odpovedal správou `UkCauseTx_NEGA`.

**Zachytenie komunikácie:** data\_wrong\_cause.pcap

### **Test č. 4:**

**Zmena:** Test na zmenu IOA adresy. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli `IOA` zmenená hodnota z 11 na 15: 0x0b → 0x0f.

**Reakcia:** Server si zmenu všimol a dotazovaný objekt nepoznal. Odpovedal správou `UkIOA_NEGA`.

**Zachytenie komunikácie:** data\_wrong\_ioa15.pcap

### **Test č. 5:**

**Zmena:** Test na zmenu OA adresy. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli `OA` zmenená hodnota z 2 na 5: 0x02 → 0x05.

**Reakcia:** Server si zmenu nevšímal a odpovedal potvrzovacou správou o úspešnom nastavení hodnoty.

**Zachytenie komunikácie:** data\_wrong\_oa.pcap

### **Test č. 6:**

**Zmena:** Test na zmenu typu id príkazu. Pôvodný príkaz na nastavenie hodnoty - `Set point command` typu `C_SE_NC_1` bol zmenený na typ `C_BO_NA_1` - `Bitstring of 32 bits`. Zmena bola vykonaná v poli `TypeId` z hodnoty 50 na hodnotu 51: 0x32 → 0x33. Zvyšok príkazu zostal pôvodný pre príkaz typu `C_SE_NC_1`.

**Reakcia:** Server si zmenu všimol a odpovedal správou `UkIOA_NEGA`.

**Zachytenie komunikácie:** data\_wrong\_type\_id.pcap

### **Test č. 7:**

**Zmena:** Posledný test bol opäť nameraný na zmenu hodnoty IOA adresy. Pri nastavovaní hodnoty príkazom `Set point command` bola v poli IOA zmenená hodnota z 11 na 12: 0x0b → 0x0c.

**Reakcia:** Server zmenu prijal, cielený objekt našiel medzi pripojenými a odpovedal potvrzovacou správou o úspešnom nastavení hodnoty.

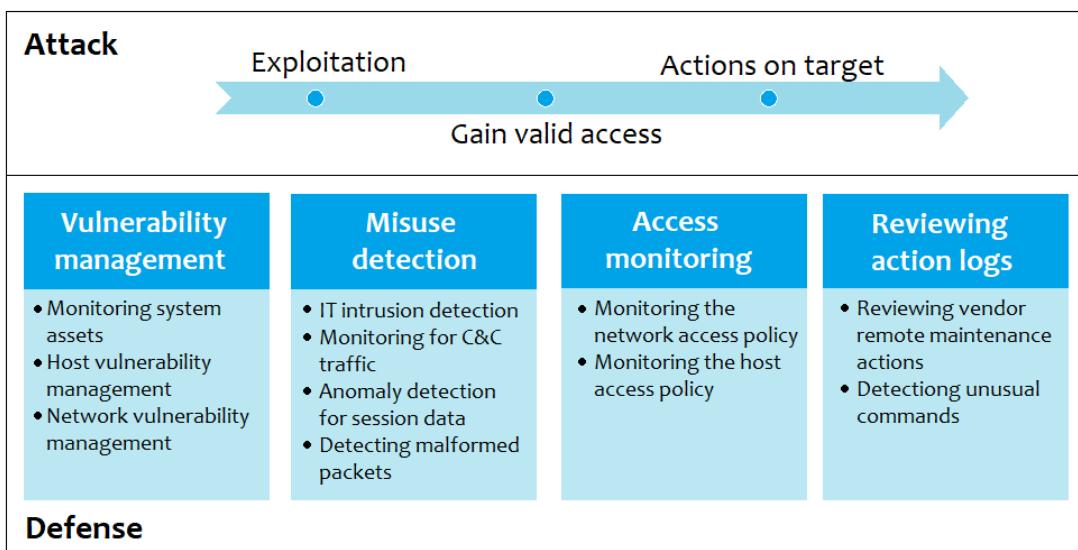
**Zachytanie komunikácie:** different\_iaa.pcap

## Kapitola 6

# Detekcia útokov na SCADA systémy

V nasledujúcej kapitole sa zameriam na možnosti detektie rôznych typov útokov popísaných v kapitole 4 spolu s popisom ochrany systémov a sietí pred nimi.

Na obrázku 6.1 je možné vidieť model jednoduchého kybernetického útoku. Myšlienka



Obr. 6.1: Model jednoduchého kyber-útoku

za týmto modelom je, že väčšina útokov začína s využitím zraniteľností danej siete kvôli získaniu prístupu do systému. Keď sa raz útočník dostane "dnu", snaží sa získať oprávnenia na získanie platného prístupu k jednotlivým prvkom systému. Akonále má čo potreboval, vykoná kroky potrebné na dosiahnutie svojho cieľa, napríklad na prepnutie alebo vypnutie rôznych častí siete.

Stratégia detektie pozostáva zo štyroch časťí:

- Kontrola a riadenie zraniteľností systému - nájdenie a odstránenie zraniteľných miest skôr ako môžu byť využité útočníkom.
- Detekcia zneužitia - detekovanie napadnutia systému.

- Monitorovanie prístupu - detekovanie neoprávnených alebo škodlivých akcií s platnými povereniami.
- Kontrola logovacích záznamov - zistovanie akcií s platnými povereniami, ktoré sú škodlivé alebo proti bezpečnostným pravidlám.

Model na obrázku 6.1 je veľmi zjednodušený. Skutočný útok nemusí byť takto priamočiarý. Útočníci môžu napríklad potrebovať nájsť niekoľko zraniteľných miest v systéme, kým sa im ich podarí využiť a získať platný prístup. Môžu tiež vykonávať rôzne akcie na preskúmanie vlastností danej siete alebo môžu úplne preskočiť fázu hľadania zraniteľných miest, ak už majú k dispozícii platne autentizačné údaje[4][5].

## 6.1 Kontrola a riadenie zraniteľností systému

Aktivity kontroly a riadenia zraniteľností sú iba preventívne opatrenia pre systém. Po kúšajú sa nájsť a opraviť zraniteľné miesta a zadné vchody (backdoor) do systému, skôr ako sú odhalené útočníkmi. Ako také zmierňujú väčšinu hrozieb s výnimkou vnútorných, ktoré sa využívajú iba s platnými povereniami.

Sú zamerané na hľadanie chýb v systéme a na ich opravu. V OT sieťach je zabezpečenie veľmi závislé od architektúry daného systému. Ak sa v sieti využívajú staršie zariadenia, je potrebné hľadať zraniteľné miesta priamo v architektúre, nakoľko nie je možné sa spoliehať iba na zabezpečenie daného zariadenia.

Na efektívnu kontrolu je potrebné mať podrobny prehľad o pripojených zariadeniach a o ich konfigurácií. Tieto informácie uľahčujú nájsť a opraviť slabé miesta a zadné vchody do systému. Základné aktivity na kontrolu systému sú:

- Sledovať/reagovať na novo-pripojené zariadenia do siete
- Monitorovať prístupy k portom a MAC adresy na jednotlivých prepínačoch
- Použiť alarm pri pripojení nového zariadenia
- Odstrániť neautorizovaných hostov

Alarmy na nové zariadenia môžu byť využívané za pomoc aktívneho alebo pasívneho skenovania siete. Aktívne skenovanie môže byť vykonávané za pomoc nástrojov ako `nmap` a pasívne sledovaním sietovej prevádzky.

Avšak vykonávanie takejto aktivity vyžaduje zoznam povolených zariadení, ktoré majú oprávnenie sa pripojiť do SCADA systému, aby mohli byť neoprávnené jednoduchšie detektovateľné. Na začiatku monitorovania ale väčšinou obdobný zoznam nie je plne k dispozícii a tak je potrebné aby sa postupne vytváral spolu s monitorovacou aktivitou.

Okrem zoznamu je tiež potrebné dodržiavať isté pravidlá pri pripájaní nového zariadenia. To môže byť riešené podrobnným regisračným procesom, ktorý pomože analytikom overiť, že zariadenie má oprávnenie na vstup do systému[4].

## 6.2 Detekcia zneužitia

Aktivity na detekovanie zneužitia hľadajú v komunikácií znaky známych útokov. V IT sfére bolo za týmto účelom vyvinutých veľa senzorov a sônd. Môžu byť naviazané na hosťiteľa, ako napríklad rôzne antivírové programy, alebo priamo na sieť, ako napríklad rôzne systémy na detekciu narušenia siete.

Pôvodne boli senzory využívané na detekciu správnosti autentizačných údajov a podpisov. Používané pravidlá boli spisované priamo výrobcom daného senzoru na základe poznatkov o zaznamenaných útokoch. Na dosiahnutie spoľahlivej úrovne detekcie boli potrebné desaťtisíce pravidiel, ktoré museli byť pravidelne aktualizované. Moderné systémy na detekciu narušení doplňujú tento prístup o využitie inteligentných algoritmov, pomocou ktorých je možné odhaliť niekoľko rôznych tried útokov. Ide o IDS (Intrusion Detection System)/IPS (Intrusion Prevention System) zariadenia, ktoré sa bežne využívajú na detekciu podozrivých aktivít v systéme.

Aktivity detekcie zneužitia aktívne hľadajú znaky známych útokov alebo porušenia pravidiel systému. Toto môže byť vykonávané pomocou detekčných systémov naviazaných na hostiteľa alebo na sieť samotnú. Tradične to bolo riešené pomocou znakov útoku (užitočné zataženie, porušenie pravidiel), ktoré spustili určitú preddefinovanú udalosť detekčného systému. Moderné systémy využívajú o niečo sofistikovanejšie postupy. Štandardné monitorovanie ich komunikácie sa tiež považuje za časť detekčného postupu, napäťko sa priamo hľadajú útoky, namiesto odchýlok od bežnej komunikácie, ako to robia systémy na detekciu anomalií.

Za účelom detekcie znakov útokov v IT sfére bolo vyvinutých mnoho rôznych senzorov a zariadení, ako napríklad rôzne antivírové programy, systémy na detekciu prieniku ap. Kedže sú OT systémy stále viac a viac založené na IT technológiach ako je TCP/IP prenos, využívanie operačných systémov na báze UNIXu, využívanie webových rozhraní ap., tak sa stali tiež zraniteľnými voči útokom vyvinutým pre IT a je možné pre ne využívať pôvodné detekčné IT senzory.

Pri samotnom detekovaní dostávajú analytici rôzne alarmy od jednotlivých senzorov, ktoré musia postupne overovať čo daný alarm spôsobilo a či bola daná aktivita relevantná útoku.

Detekčné systémy neustále hľadajú špecifické znaky útokov a tak musia byť stále údržiavané aktuálne kvôli neustále sa meniacej inteligencií útokov. To môže byť vykonávané pomocou automatických aktualizácií a rozširovaní systému o nové znaky útokov[4].

### 6.3 Monitorovanie prístupu

Väčšina systémov, ktoré implementujú kontrolu prístupu taktiež uchovávajú logovacie záznamy o úspešných a neúspešných pokusoch o vstup (prihlásenie) do systému. Monitorovanie takéhoto typu informácií môže pomôcť detektovať veľa rôznych útokov. Monitorovanie kontroly prístupu je v OT systémoch pravdepodobne oveľa dôležitejšie ako v IT systémoch, napäťko sú OT systémy relatívne otvorené. Ak sa útočníkovi podarí získať platné autentizačné údaje, môže sa voľne pohybovať po systéme a vykonávať validné zmeny. Napríklad kybernetické útoky na Ukrajine využívali iba zraniteľnosti IT sietí, v OT sietach bolo všetko vykonané s platnými povereniami. Toto je pravdepodobne najjednoduchšia cesta do SCADA systémov, kým nebude bezpečnosť a overovanie autentizácie výrazne zlepšené.

Monitorovanie je však efektívne iba vtedy, ak je v systéme jasne daná a dodržiavaná prístupová politika. Jednotlivé procesy by mali mať nedefinované, ktorí užívatelia majú právo na vstup do systému a spúštanie operácií. Navyše by sa mali presne dodržiavať rôzne rozvrhy na údržbu a aktualizáciu systému, aby analytici vedeli, kto môže kedy pristupovať do systému.

Bez takýchto pravidiel a rozvrhov môžu byť analýzy vykonávané iba na základe hrubých heuristik, ako je napríklad hľadanie prístupu do systému v neobvyklých hodinách alebo

mnoho po sebe neúspešných pokusov o prihlásenie. Analýza možného incidentu by tak mohla trvať oveľa dlhšie.

V SCADA systémoch by mala byť komunikácia a prístupy do siete obzvlášť dôkladne monitorované. Taktiež by mali byť vytvorené pravidlá o povolenom prístupe k jednotlivým sieťovým službám a používaným komunikačným protokolom. Monitorovanie prístupu je jedna z variánt ako tieto pravidlá presadzovať. Je to obzvlášť dôležité pri službách, ktoré nepoužívaniu overovanie autentizáciou.

Monitorovanie prístupu a dodržiavanie jednotlivých pravidiel nie je užitočné iba pre bezpečnosť. Môže tiež zlepšiť robustnosť siete tým, že zachytí jej nesprávne konfigurácie. Napríklad ak je nesprávne nakonfigurovaná ip adresa pre sietové servery ako DNS, SNMP alebo NTP.

Najefektívnejší spôsob ako monitorovať prístup do siete je s využitím firewallov, napäťko sa pomocou nich najlepšie presadzujú prístupové pravidlá. Firewally by mali byť monitorované tak, že sa pravidelne kontrolujú a aktualizujú použité pravidlá, a sledujú sa zachytené upozornenia. Pri monitorovaní by sa mali bráť do úvahy všetky firewally v danej doméne:

- firewally použité na IT/OT rozhrania
- firewally použité na poskytovanie vzdialeného prístupu na údržbu
- firewally použité na spojenie riadiacej stanice s WAN (Wide Area Network) sieťou
- firewally na koncových staniciach[4]

## 6.4 Kontrola logovacích záznamov

Kontrola logovacích záznamov znamená analýzu toho, čo všetko bolo v systéme (a so systémom) vykonané. Aplikačný software ako aplikácie pre SCADA systémy udržujú záznamy o operáciach vykonaných jednotlivými užívateľmi. Taktiež môže byť prevádzka v SCADA systéme rozdelená na extrakciu iba špecifických príkazov (záleží od analýzy). Všetky tieto informácie môžu byť použité na nájdenie rôznych príznakov útoku na systém.

Aktivity na kontrolu záznamov monitorujú, čo bolo kým v systéme vykonané. Sú definované podľa rôznych skupín užívateľov: dodávatelia, ktorí vykonávajú údržbu koncových zariadení na diaľku, zamestnanci, ktorí vykonávajú údržbu v rámci spoločnosti a operátori SCADA systému, ktorí vysielajú príkazy do koncových staníc. Samozrejme, keď útočník získa platné autentizačné údaje z jednej z týchto skupín, má prístup ku všetkým operáciám, ktoré môže daná skupina vykonávať.

Pri kontrole vzdialeného prístupu do OT systému by mali byť zavedené jednoznačné pravidlá a na rôzne technické opatrenia (údržba ap.) by mal byť stanovený a dodržiavaný presný harmonogram aby analytici mohli rozpoznať prienik do systému od bežnej (plánovanej) údržby. Avšak stále je tu riziko povoliť externým pracovníkom vstup do kritického systému. Aby bolo toto riziko zredukované, je možné pravidelne sledovať a zhodnocovať ich úkony v systéme.

Na detekciu útokov zvnútra je užitočné analyzovať logovacie záznamy obsluhy systému. Vykonané akcie môžu byť zaznamenané operačným systémom, ktorý ukladá kritické príkazy, alebo príkazy spustené s právami administrátora. Od tohto bodu môžu byť detektované zmeny v systémových konfiguráciach[4].

Pri monitorovaní a analýze komunikácie v sieti sa taktiež často využíva flow monitorovanie. S tým súvisí aj pojmom IPFIX (IP Flow Information Export), ktorý označuje export

informácií z flow záznamov. IPFIX je veľmi podobný Netflow v tom zmysle, že umožňuje sieťovým administrátorom zbierať informácie o komunikácii zo smerovačov, prepínačov a z akéhokoľvek zariadenia podporujúceho daný protokol. IPFIX má však oproti Netflow niekoľko výhod. Za prvé, informácie, ktoré by za normálnych okolností boli zasielané nástrojmi Syslog alebo SNMP má IPFIX schopnosť integrovať a exportovať priamo v IPFIX pake-toch. Tým sa eliminuje potreba týchto dodatočných služieb na zber dát. To v podstate výrobcom hardvéru umožňuje špecifikovať ID výrobcu a vložiť do flow záznamu akúkolvek proprietárnu informáciu a vyexportovať ho z kolektoru na ďalšiu analýzu a monitorovanie. IPFIX taktiež povoluje polia o premenlivej dĺžke, čo umožňuje uložiť informácie ako URL adresy, správy, HTTP stanice a pod. Pracuje na porte 4739.

# Kapitola 7

## Záver

### Zhodnotenie práce

SCADA systémy sú v dnešnej dobe veľmi rozšírené a stále pribúdajú nové spoločnosti, ktoré ich využívajú. Avšak rovnako narastá aj počet útočníkov a hrozba, ktorej musia čeliť. Je preto veľmi potrebné a dôležité vedieť jednotlivé útoky včas detektovať a systém pred nimi chrániť. Táto práca sa zameriava na komunikačné protokoly DLMS/COSEM a IEC 104, ktoré sú využívané najmä v energetických odvetviach priemyslu.

Práca môže pomôcť pri skúmaní reakcií systému na jednotlivé typy útokov a tiež pri vývoji rôznych typov monitorovacích zariadení a sónd, ktoré budú schopné útoky včas odhaliť pri sledovaní prebiehajúcej komunikácie. Súčasťou práce je zhodnotenie rôznych simulačných nástrojov na prevádzku SCADA systémov využívajúcich komunikačný protokol DLMS/COSEM a IEC 104. Pomocou nástrojov a novo-vytvoreného simulačného programu klienta pre jazyk C++ bolo vytvorené simulačné prostredie, umožňujúce testovať jednotlivé útoky a sledovať reakcie systému na ne.

Pri jednotlivých testovaniach bola vždy vytvorená jedna vzorová komunikácia a v nej následne niekoľko zmien (jedna v rámci testu). Výsledky testov a zaznamenaná komunikácia bola porovávaná so vzorovou a následne bolo vyhodnotené chovanie systému na zmeny.

Posledná kapitola práce je venovaná popisu rôznych spôsobov detekcie útokov a prevenčie pred nimi.

### Výstup práce

Výstupom práce sú programy a postup vytvorenia simulačného prostredia na testovanie rôznych typov útokov. Vďaka simulačnému programu klienta, ktorý pracuje na štvrtnej (transportnej) vrstve TCP/IP modelu je postup aplikovateľný na širokú škálu komunikačných protokolov pracujúcich nad TCP/IP. Výstupom jednotlivých testovaní je datová sada vo formáte .pcap obsahujúca jednotlivé útoky spolu s popisom reakcií systému na ne. Všetky zdrojové kódy a zachytené komunikácie sú k dispozícii na priloženom médiu a v github rezozitári<sup>1</sup>.

Práca bola taktiež prezentovaná na študentskej konferencií Excel@FIT a je súčasťou projektu IRONSTONE<sup>2</sup> vo fakultnej výzkumnej skupine NES@FIT.

---

<sup>1</sup>Github <https://github.com/janpristas/bakalarska-praca>

<sup>2</sup>Projekt Ironstone <http://www.fit.vutbr.cz/units/UIFS/grants/index.php.cs?id=1101>

# Literatúra

- [1] Clarke, G.; Reynders, D.: *Practical Modern SCADA Protocols*. ELSEVIER SCIENCE TECHNOLOGY, United Kingdom, 2004.
- [2] Hanes, D.; Salqueiro, G.; Grossete, P.; aj.: *IoT Fundamentals. Networking Technologies, Protocol and Use Cases for the Internet of Things*. Cisco Press, 2017.
- [3] Horych, V.: *Analýza řídicích protokolů využívaných v průmyslových aplikacích*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009, diplomová práca.
- [4] Knapp, E. D.; Langill, J. T.: *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2011.
- [5] Matoušek, P.: *Security Issues in Industrial IoT Systems*. Vysoké učení technické v Brně, projekt IRONSTONE, technická správa č. FIT-TR-2018-xx.
- [6] Matoušek, P.: *Analysis of DLMS Protocol*. Vysoké učení technické v Brně, 2017, projekt IRONSTONE, technická správa č. FIT-TR-2017-13.
- [7] Matoušek, P.: *Description of IEC 60870-104 Protocol*. Vysoké učení technické v Brně, 2017, projekt IRONSTONE, technická správa č. FIT-TR-2017-12.
- [8] Pekárek, D.: *Popis a testování komunikačních protokolů normy IEC 60870-5-103 a 60870-5-104*. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017, diplomová práca.
- [9] Stouffer, K.; Falco, J.; Scarfone, K.: Guide to Industrial Control Systems (ICS) Security. 2011.  
URL  
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82.pdf>

## Príloha A

# Inštalácia monitorovacích nástrojov

V tejto kapitole je uvedený popis práce s jednotlivými monitorovacími nástrojmi popísanými v kapitole 3. Jedná sa o popis, kde jednotlivé nástroje nájst na internete, postup inštalácie a konfiguráciu koncových zariadení na simuláciu.

### A.1 DLMS Director

Kedže je program DLMS Director typu open source, je možné ho zdarma stiahnuť priamo zo stránky výrobcu<sup>1</sup>. Na stránke je rovnako odkaz na GitHub repozitár v ktorom je možné nájst zdrojové kódy programu. Po stiahnutí inštalačného súboru, ho stačí spustiť a program sa automaticky nainštaluje. K programu je od výrobcu poskytovaný online manuál<sup>2</sup>, ktorý veľmi prehľadne popisuje prácu s programom. Práca s nástrojom je ale aj bez návodu veľmi jednoduchá. Pri pridávaní nového zariadenia postupujeme v niekolkých krokoch:

1. Spustiť DLMS Director
2. V DLMS Directore nakonfigurovať server s ktorým sa ide program spojiť:
  - (a) File → Add Device
  - (b) Zvoliť meno pre server
  - (c) Zvoliť výrobcu → Gurux,
  - (d) Zvoliť typ autentizácie a heslo (ak server využíva autentizáciu)
  - (e) Nastaviť IP adresu servera a port
3. Po úspešnom nakonfigurovaní vytvoriť spojenie so serverom. Spojenie inicializuje program DLMS Director. V ľavom paneli je potrebné rozkliknúť zoznam Devices a kliknúť pravým tlačítkom na server, následne na Connect.
4. Po vytvorení spojenia sa zobrazí zoznam pripojených zariadení, z ktorých je možné čítať a zapisovať hodnoty, príkazy Read a Write.

Na stránke programu je podrobnejší postup ako nakonfigurovať jednotlivé zariadenia, ktoré k programu pripájame<sup>3</sup>.

---

<sup>1</sup>DLMS Director <http://www.gurux.fi/Download> [Online: Október 2017]

<sup>2</sup>DLMS Director manuál <https://www.gurux.fi/GXDLMSSDirectorHelp> [Online: Október 2017]

<sup>3</sup>DLMS Director konfigurácia <http://www.gurux.fi/GXDLMSSDirectorExample> [Online: Október 2017]

## A.2 XmlDemo

XmlDemo je zdarma poskytovaný nástroj spoločnosti iCube. Je možné ho opäť stiahnuť na stránke výrobcu<sup>4</sup>. Jednotlivé vývojové balíky v ktorých je program napísaný však nie sú voľne k dispozícii. Je ale možné napísať priamo spoločnosti so žiadostou o ne. Odkaz na inštalačný súbor je na konci vyššie odkazovanej stránky, predtým je celkom podrobny návod ako s programom pracovať, ako pripájať jednotlivé zariadenia, čítať z nich hodnoty a pod. Inštalačia prebieha v niekoľkých krokoch:

1. Spustiť stiahnutý súbor a vyskočí nám sprievodca inštalačiou
2. **Next** → Vybrať zložku, kam sa program nainštaluje → **Next** → **Install**
3. Program sa nainštaluje a môžme ho spustiť

Pri pridávaní nového zariadenia postupujeme nasledovane:

1. Spustiť XmlDemo
  - (a) **Show** → **Settings**
  - (b) **Select profile** (TCP/HDLC)
  - (c) Nastaviť IP adresu servera a port
  - (d) Nastaviť referenčný model a prípadne heslo
  - (e) Nastaviť timeout, obe hodnoty **Connect** a **Response** na prijateľné hodnoty v milisekundách
  - (f) Nastaviť adresy pre klienta aj server
2. Po nakonfigurovaní je možné vytvoriť spojenie. **Do** → **Connect** a **Do** → **Read object-list**.

## A.3 WinPP104

WinPP104 sa dá stiahnuť na stránke spoločnosti, ktorá program vytvorila<sup>5</sup>. Inštalačia prebieha v niekoľkých krokoch:

1. Spustiť stiahnutý súbor a vyskočí nám sprievodca inštalačiou
2. Vybrať jazyk inštalačie (angličtina/nemčina) → **OK** → **Next**
3. Vybrať zložku, kam sa program nainštaluje → **Next**
4. Vybrať názov zložky, ktorý sa vytvorí v Štart menu → **Next**
5. Zvoliť, či chceme zástupcu na ploche → **Next** → **Install**

Po inštalačii je možné program spustiť. Výrobca k programu tiež poskytuje podrobny manuál ako s programom pracovať<sup>6</sup>.

---

<sup>4</sup>XmlDemo <https://icube.ch/xmldemo/xmldemo.html> [Online: Október 2017]

<sup>5</sup>WinPP104 <http://www.ppfink.de/> [Online: Október 2017]

<sup>6</sup>WinPP104 - manuál <http://www.ppfink.de//downloads/Bed104Usa.pdf> [Online: Október 2017]

## A.4 Knižnica OpenMUC - j60870

Knižnica j60870 je na stiahnutie priamo na stránke výrobcu<sup>7</sup>. Priamo v stiahnutej zložke sú už dva preložené binárne súbory, v zložke `run-scripts`. Konkrétnie sa jedná o jednoduchú aplikáciu klienta a servera. Na aplikáciach je pekne vidieť čo sa dá pomocou knižnice naimplementovať. Ja osobne som ich spúšťal cez terminál. Na spustenie je ale nutné mať nainštalované Java JDK. Čo sa týka samotnej knižnice, stačí ju vložiť do projektu v jazyku Java a ľubovoľne s ňou pracovať. Súčasťou je aj rozsiahla dokumentácia popisujúca jednotlivé funkcie, ktoré knižnica ponúka<sup>8</sup>.

## A.5 QTester 104

Výrobca programu QTester 104 nemá vlastnú stránku produktu. Pre stiahnutie programu je potrebné navštíviť stránku [sourceforge.net](https://sourceforge.net)<sup>9</sup>. Na stránke nájdete zložku súboru obsahujúcu zdrojové kódy programu (v zložke `src`) a samotný spustiteľný súbor (v zložke `bin`). Program nie je nutné inštalovať, stačí ho iba spustiť. Po spustení je možné pripojiť ľubovoľné meracie zariadenia komunikujúce cez protokol IEC 60870-5-104. Bohužiaľ výrobca neposkytuje nijaký návod na prácu s programom. Samotné GUI je ale veľmi intuitívne a návod nie je ani potrebný. Pri príprave nového servera sa postupuje v niekoľkých krokoch:

1. Spustiť program QTester 104: `qtester104 → bin → QTester104`
2. Zadať ip adresu servera
3. Po úspešnom spojení, kliknúť na **GI (General Interrogation)** aby sa načítali objekty pripojené k serveru
4. Zakliknúť **Log Messages**, prípadne **AutoScroll** na zobrazenie logovacej správy

## A.6 IEC 60870-5-104 Client/Server Simulator

Programy IEC 60870-5-104 Client a Server Simulator súčasťou súboru na demoverzie na nej nenájdeme. Aspoň nie priamo. Kvôli inštalacnému súboru je nutné vyplniť určité osobné informácie. Späť nám potom spoločnosť pošle na emailovú adresu, ktorú sme zadali, odkaz na stiahnutie daného programu. Súbory sa dajú stiahnuť aj nepriamo na stránke [sourceforge.net](https://sourceforge.net)<sup>10</sup>. Inštalácia oboch programov prebieha rovnako:

1. Spustiť stiahnutý súbor a vyskočí nám sprievodca inštaláciou
2. **Next** Zobrazia sa licenčné podmienky k danému programu. Po prečítaní klikneme na **I Agree**. Prípadne, ak s podmienkami nesúhlasíme, môžeme inštaláciu zrušiť tlačidlom **Cancel**
3. Po odsúhlasení vyberieme kam sa program nainštaluje → **Next**

<sup>7</sup>j60870 <https://www.openmuc.org/iec-60870-5-104/download/> [Online: Október 2017]

<sup>8</sup>Javadoc <https://www.openmuc.org/iec-60870-5-104/javadoc/> [Online: Október 2017]

<sup>9</sup>QTester 104 <https://sourceforge.net/projects/qtester104/> [Online: Október 2017]

<sup>10</sup>IEC 60870-5-104 Client/Server Simulator <https://sourceforge.net/u/freyrscada/profile/> [Online: November 2017]

4. Vybrať názov zložky, ktorý sa vytvorí v Štart menu → **Next**
5. Zvoliť, či chceme zástupcu na ploche → **Next** → **Install**

Po úspešnom nainštalovaní môžeme program spustiť a pracovať s ním. Pridávanie nových uzlov (klient, server) prebieha v oboch programoch obdobne:

1. Spustiť programu klienta/servera
2. Kliknutie na **Add Client/Server**
3. Nastaviť ip adresu a port
4. Pre server je potrebné nastaviť pripojené objekty
  - (a) **Configuration** → **Add Row** → pridať požadované objekty
  - (b) **Load Configuration**
5. **Data\_Objects** → **Start Communication**
6. Klient sa pripojí k serveru a zobrazí pripojené objekty

Uzly sa dajú nastaviť celkom komplexne, vyžaduje to však väčšiu znalosť problematiky. Spoločnosť ale poskytuje veľmi dobre spracované inštruktážne videá na vytvorenie koncových staníc a prácu s nimi. Videá sú štyri a je možné ich nájsť na stránke spoločnosti<sup>11</sup>. So samotnými programami je možné pracovať maximálne 15 minút, nakoľko ide o demoverzie. Po opäťovnom spustení je potrebné opäť nastaviť všetky predošlé konfigurácie.

---

<sup>11</sup>Tutoriály <http://freyrscada.com/iec-60870-5-104-Client-Simulator.php> [Online: November 2017]

## Príloha B

# Inštalácia C++ knižnice pre protokol DLMS

Open source C++ knižnicu pre protokol DLMS poskytovanú spoločnosťou GuruX je možné nájsť na stránke Github<sup>1</sup>. Koreňová zložka obsahuje časť `development` obsahujúcu všetky komponenty samotnej knižnice. Ďalej sú tam vzorové programy pre klienta a server. Oba v samostatnej zložke. Pre nainštalovanie knižnice je potrebné íst do zložky `development` a vytvoriť zložky `lib` a `obj`. Potom spustiť priložený `Makefile`. Prekladač ale zahlási niekoľko chýb. Všetky sa týkajú súborov `GXDLMSSetupLogicalName.h/.cpp`. Súbory obsahujú znaky » a prekladač si pravdepodobne myslí, že má íst o bitovú operáciu na mieste, kde by byť nemala. Je tam preto potrebné pridať medzeru a prepísať to na > >. Teraz keď spustíme `make`, preklad by mal prejst bez problémov. Pre inštaláciu vzorového klienta a servera je tiež vytvorený `Makefile`. Samostatný pre každý program, vždy v jeho zložke. Pred spustením je opäť potrebné vytvoriť dve nové zložky, tentokrát `bin` a `obj`. Po vytvorení môžeme zadáť príkaz `make`.

---

<sup>1</sup>DLMS knižnica - Github <https://github.com/Gurux/Gurux.DLMS.cpp> [Online: Marec 2018]