VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Roland Dávidík

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

AUTOMATIZOVANÝ TESTER BEZPEČNOSTI CHYTRÝCH ZAŘÍZENÍ V ENERGETICE
AUTOMATED CYBER SECURITY TESTER FOR SMART DEVICES IN INDUSTRY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Roland Dávidík**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Roland Dávidík ***ID:*** 185924

Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Automatizovaný tester bezpečnosti chytrých zařízení v energetice

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je vytvoření automatizovaného testeru pro testování kybernetické bezpečnosti chytrých zařízení v elektrotechnice komunikujících pomocí protokolu DLMS (smart metrů, koncentrátorů, modemů, bran, ...). Výsledky budou poté prezentovány pomocí webové aplikace. Celé studentovo řešení musí být optimalizováno pro hardwarově omezené prostředí (Raspberry Pi). Cílem diplomové práce je vytvoření skeneru zranitelností, který bude mít zprovozněny následující funkce: sken sítě, nalezení aktivních zařízení, zjištění otevřených portů (aktivních služeb), pokus o útok na aktivní služby, ověření DLMS kompatibility a provedení alespoň 2 různých útoků na DLMS protokol.

DOPORUČENÁ LITERATURA:

[1] Gurux DLMS Server: https://www.gurux.fi/Gurux.DLMS.Server

[2] JIRKA, Bc Matěj. Framework DLMS/COSEM pro sběr dat v AMM systémech.

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

Vedoucí práce: Ing. Tomáš Lieskovan

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

ABSTRAKT

Táto diplomová práca vysvetluje princíp SCADA systémov a popisuje DLMS/COSEM protokol. Ďalej stručne popisuje Modbus protokol a popisuje výhody a nevýhody protokolu Modbus oproti DLMS/COSEM protokolu. Po popise protokolu je v diplomovej práci vytvorený open-source automatizovaný skener, ktorý detekuje zariadenia v laboratórnej sieti. Potom automatozovaný skener zistí otvorené porty a aktívne služby, ktoré bežia na týchto zariadeniach, a pokúsi sa zaútočiť na služby HTTP, SSH a Telnet. Potom program overí, či je zariadenie smart-meter kompatibilný s DLMS/COSEM protokolom. Ak áno, tak overí, či je zariadenie zraniteľné na útok DOS a na rozpojenie breakeru. Výsledky zistenia programu sú prezentované na vlastnej vytvorenej webovej aplikácii. V automatizovanom skenery sú použité open-source programy NMAP, Masscan a Metasploit. Celý automatizovaný skener je optimalizovaný pre hardwarove zariadenie Raspberry Pi s operačným systémom Raspbian Buster Lite. Ďalej je v diplomovej práci otestovaný tento skener na laboratórnom prostredí a je vyhodnotená správnosť jeho výsledkov.

KĽÚČOVÉ SLOVÁ

SCADA systémy, DLMS/COSEM protokol, Modbus protokol, automatizovaný skener zraniteľností, útok DOS, rozpojenie breakeru.

ABSTRACT

This diploma thesis explains the principle of SCADA systems and describes the DLMS/COSEM protocol. In the next part, it shortly describes the Modbus protocol and details the pros and cons of the Modbus protocol in comparison with the DLMS/COSEM protocol. In the next part, an open-source automated scanner was created. This scanner detects devices in a laboratory network. As the next step, the automated scanner finds out open ports and active services, which run on these devices, and tries to attack HTTP, SSH, and Telnet services. Next, the program checks, whether the found device is a smart-meter device and if it is compatible with DLMS/COSEM protocol. If yes, it checks, if the service is vulnerable to DOS attack and breaker disconnection. Scanner's findings are presented in a newly created web application. NMAP, Masscan, and Metasploit open-source programs are used in the automated scanner. The whole automated scanner is optimized for the HW device Raspberry Pi with the operating system Raspbian Buster Lite installed. This work also describes the testing of the scanner on the laboratory environment and the results are evaluated afterwards.

KEYWORDS

SCADA systems, DLMS/COSEM protocol, Modbus protocol, automatic vulnerability's scanner, DOS attack, disconnect breaker.

DÁVIDÍK, Roland. *Automatizovaný tester bezpečnosti chytrých zařízení v energetice*. Brno, Rok, 63 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Tomáš Lieskovan

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Automatizovaný tester bezpečnosti chytrých zařízení v energetice“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

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

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu diplomovej práce panu Ing.Tomášovi Lieskovanovi za odborné vedenie, konzultácie a podnetné návrhy k práci. Ďalej by som rád poďakoval každému, kto mi počas inžinierského štúdia pomohol alebo dal cennú radu.

Obsah

Úvod **10**

1 SCADA systémy **11**
1.1 Popis energetickej siete a zariadení v nej 12

2 Komunikačné protokoly v energetike **15**
2.1 DLMS/COSEM 15
2.1.1 Adresy a mená zariadení 15
2.1.2 Komunikácia 16
2.1.3 Bezpečnosť v DLMS 17
2.1.4 COSEM 19
2.1.5 OBIS 20
2.2 Modbus 21
2.3 Porovnanie DLMS/COSEM a Modbus 22

3 Laboratórne prostredie pre automatizovaný skener **23**

4 Popis automatizovaného skeneru **25**
4.1 TesterDLMS 25
4.1.1 Popis tried a funkcii 27
4.2 Skript 35
4.3 Webová aplikácia 38

5 Optimalizácia skeneru pre Raspberry Pi **42**
5.1 Inštalačný skript 43
5.2 Sprevádzkovanie automatizovaného skeneru 43
5.3 Použitie cez VPN 44
5.4 Ladiaca hodnota v scriptStart.sh 45

6 Test automatizovaného skeneru na laboratórnej sieti **47**

7 Postrehnuté problémy so smart metrami počas vývoja automatizovaného skeneru **51**

8 Porovnanie automatizovaného skeneru s už existujúcimi skenermi **53**

Záver **54**

Literatúra **55**

Zoznam symbolov, veličín a skratiek **59**

Zoznam príloh **61**

A Obsah priloženého CD **62**

Zoznam obrázkov

1.1 Schéma SCADA systému. 12
1.2 Približná predstava siete. 13
2.1 Umiestnenie DLMS/COSEM protokolu v referenčnom modele ISO/OSI. 16
2.2 Umiestnenie protokolu Modbus v referenčnom modele ISO/OSI. . . . 21
2.3 Modbus správa. 22
3.1 Popis laboratórneho prostredia. 23
4.1 Chybová hláška pri pôvodnej triede z knižnice Gurux.DLMS. 27
4.2 Štruktúra tried časti TesterDLMS. 28
4.3 Obsah súboru DLMSResults.txt. 29
4.4 Vývojový diagram funkcie `remoteCloseOpenBreaker`. 32
4.5 Preklad bajtov pomocou Gurux prekladača. 33
4.6 Výstup programu Masscan. 36
4.7 Obsah scanNetFinal.txt. 37
4.8 Výstup programu Nmap. 37
4.9 Obsah súboru scanPortsFinal.txt. 38
4.10 Výpis Metasploit Framework po príkaze `search telnet`. 39
4.11 Vstupná webová stránka nadefinovaná v súbore index.jsp. 40
4.12 Výstupná webová stránka nadefinovaná v súbore vysledok.jsp. 41
5.1 Raspberry Pi 4, 4 GHz RAM v špeciálnej hliníkovej krabičke. 42
5.2 Obsah priečinku `autoTester`. 44
5.3 Zakomentovaná časť v skripte scriptStart.sh. 44
5.4 Výsledky u `settingsValue` s hodnotou 3. 45
5.5 Výsledky u `settingsValue` s hodnotou 5. 45
6.1 Topológia siete z webovej aplikácie koncentrátora. 47
6.2 Prvá tabuľka vo výslednej webovej stránke. 48
6.3 Druhá tabuľka vo výslednej webovej stránke. 48
6.4 Webová stránka na smart metry. 49
6.5 Výsledné log súbory. 49
6.6 Posledná tabuľka vo výslednej webovej stránke. 49
6.7 Sken otvoreného portu pre DLMS/COSEM protokol pomocou programu Nmap. 50
7.1 Topológia na začiatku vývoja automatizovaného skeneru. 51
7.2 Topológia pri dokončovaní automatizovaného skeneru. 51

Zoznam tabuliek

2.1 Bezpečnostné sady [15]. 17
2.2 Typy zabezpečenia [10]. 18
2.3 Typy oprávnenia [10]. 19
4.1 Zachytený objekt s hexadecimálnymi hodnotami a ich prevod na decimálne hodnoty. 33

Úvod

Pri vzniku informačných sietí nikto nepredpokladal, že bude potrebné riešiť bezpečnosť. Preto aj prvé siete vznikali ako otvorené siete bez bezpečnostných prvkov. Pri rozvoji internetu sa ukázalo, že táto predstava bola mylná. Začali sa objavovať prvé útoky a bolo jasné, že bezpečnosť má veľmi dôležítú úlohu v týchto sieťach. V dnešnej dobe sa investuje velké množstvo peňazí na zabezpečenie, či už sietí alebo konkrétných staníc v sieti pripojených do internetu.

Rovnako tomu je aj pri priemyselných sieťach. Dôraz na bezpečnosť sa tu tiež nekládol, až do prvých velkých útokov, ktoré nastali ale oveľa neskôr ako v internetových sieťach. Bohužial sa ale v týchto sieťach ešte bezpečnosť nerieši v takej miere ako v internetových sieťach. Čo vidím ako veľkú chybu.

Najznámejší útok na priemyselné siete, ktorý ukázal že ani tieto siete sa nezaobídu bez bezpečnosti bol Stuxnet [1]. Stuxnet je virus (červ), ktorý sa zameriaval len na časť priemyselných sietí, konkrétne SCADA siete. Bol objavený v roku 2010 Bieloruskou firmou VirusBlokAda. Tento útok bol cielený na továrne na obohacovanie uránu.

Moja diplomová práca sa zameriava na bezpečnosť priemyselných sietí v energetike. Konkrétnejšie na bezpečnosť chytrých elektormerov teda „smart metrov“. Bezpečnosť v týchto sieťach je o to viac dôležitá, lebo siete v energetike spadajú poďla zákona č. 181/2014 Sb. o kybernetickej bezpečnosti do kritickej infraštruktúry [2]. Aj v týchto sieťach boli zaznamenané veľké útoky, ako napríklad útoky virusu BlackEnergy, ktorý spôsobil výpadky elektrickej energie v oblastiach Ukrajiny v decembri roku 2015 [3].

Cieľom mojej diplomovej práce je popísať a vysvetliť SCADA systémy a konkrétnejšie rozobrať a popísať DLMS/COSEM protokol. Ďalší cieľ je stručne popísať Modbus protokol a porovnať tento protokol s DLMS/COSEM protokolom. Ďalším cieľom mojej diplomovej práce je vytvoriť program pre laboratórne prostredie, ktorý detekuje zariadenia v laboratórnej sieti, zistí otvorené porty a aktívne služby, ktoré bežia na týchto zariadeniach, a pokúsi sa zaútočiť na služby HTTP, SSH a Telnet, ktoré sa často nachádzajú na týchto zariadeniach. Potom program overí, či je zariadenie smart meter kompatibilný s DLMS protokolom. Ak áno, tak overí či je zariadenie zraniteľné na útok DOS (Denial of service) a na rozpojenie ističa teda „breakeru“. Výsledky zistenia programu budú prezentované na vlastnej webovej aplikácii. Celý program bude optimalizovaný pre zariadenie Raspberry Pi s operačným systémom Raspbian Lite [4]. Ďalším cieľom je vyskúšať tento program na laboratórnom prostredí a porovnať výsledky z programu s poznatkami o laboratórnej sieti. Moje ciele vychádzajú zo zadania diplomovej práce.

1 SCADA systémy

Pre lepšie pochopenie jednotlivých prvkov energetickej siete, ktoré bude automatizovaný tester kybernetickej bezpečnosti testovať, stručne popíšem SCADA systémy.

Názov SCADA je skratka pre „Supervisory Control And Data Acquisition“ teda nadriadená kontrola a zber dát. Hlavnou úlohou tohto systému je interaktívne ovládanie technológie, zbieranie, spracovanie, ukladanie dát a následná historizácia týchto dát. Týmto zabezpečuje centrálne riadenie a plnú kontrolu a monitoring nad rozsiahlími technologiami.

SCADA systém sa skladá z týchto blokov:

- RTU - Remote Terminal Unit,
- komunikačná infraštruktúra,
- MTU - Master Terminal Unit,
- central control.

RTU (Remote Terminal Unit) je zariadenie, na ktoré sa napájajú všetky senzory (merače, teplomery ...) a aktuátory (ventily, klapky, pumpy ...), teda zariadenia fyzickej vrstvy. RTU z týchto zariadení odčítáva hodnoty a posiela ich cez komunikačné prostredie na MTU. RTU v podstate prepájá fyzickú vrstvu s MTU. Okrem odčítania hodnôt môže RTU vykonávať aj jednoduché operácie nad týmito dátami. Ďalej môže RTU vykonávať jednoduché operácie nad aktuátormi (otvorenie/zatvorenie ventila, spojenie/rozpojenie klapky ...).

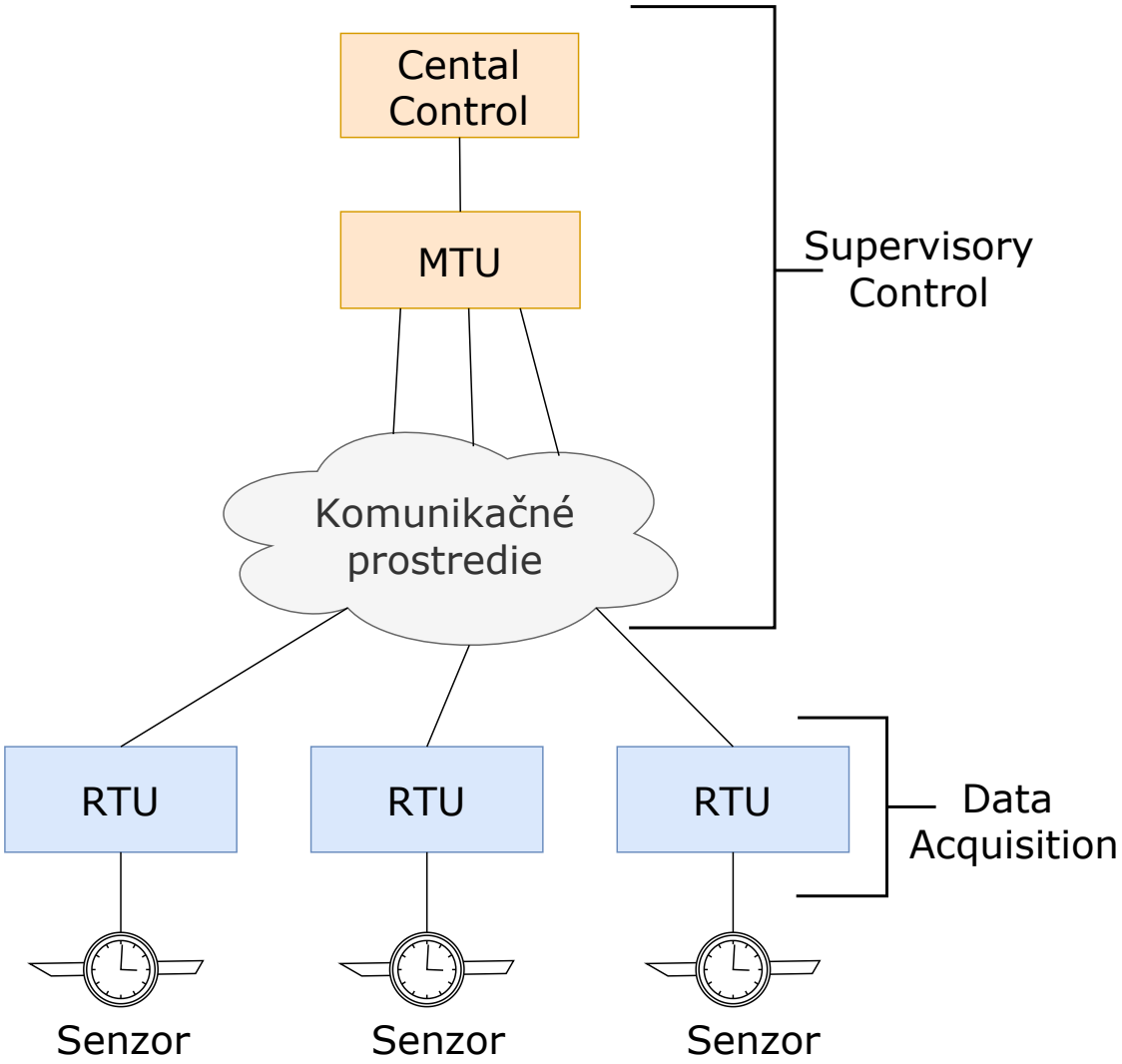
Komunikačné infraštruktúra je to prosteredie cez ktoré si RTU a MTU posielajú dáta. Môže byť bezdrôtové, drôtové alebo ich kombinácia. Komunikačná infraštruktúra može byť priama linka medzi RTU a MTU, čo sa ale často nestáva. Častejšia možnosť je, že sa dáta zašifrujú a pomocou niektorého komunikačného protokolu sú poslané cez internet na MTU.

MTU (Master Terminal Unit) je zariadenie, ktoré zadává príkazy RTU na zaslanie určitých hodnot. Tieto hodnoty potom MTU spracováva a ukladá. Výsledky zo spracovaní zobrazuje v podobe tabuliek, obrázkov a grafov v central control.

Komunikácia medzi MTU a RTU je na úrovni master-slave, kde MTU je master a RTU slave. Komunikáciu teda vždy iniciuje MTU s požiadavkom na určité dáta.

Central control je riadiace centrum systému. Sú to servery, zariadenia, monitory a klávesnice, za ktorými sedia ludia a vidia zobrazené dáta z MTU. V prípade nutnosti môžu zasiahnuť a poslať príkazy ku konkrétnym prvkom topológie. Tu nastavujú celé chovanie SCADA systému.

Bloky komunikačná infraštruktúra, MTU a control center spadajú pod kategóriu Supervisory Control z názvu SCADA. Do časti Data Acquisition spadá zasa RTU a senzory fyzickej vrstvy (Obr. 1.1) [5, 6, 7].



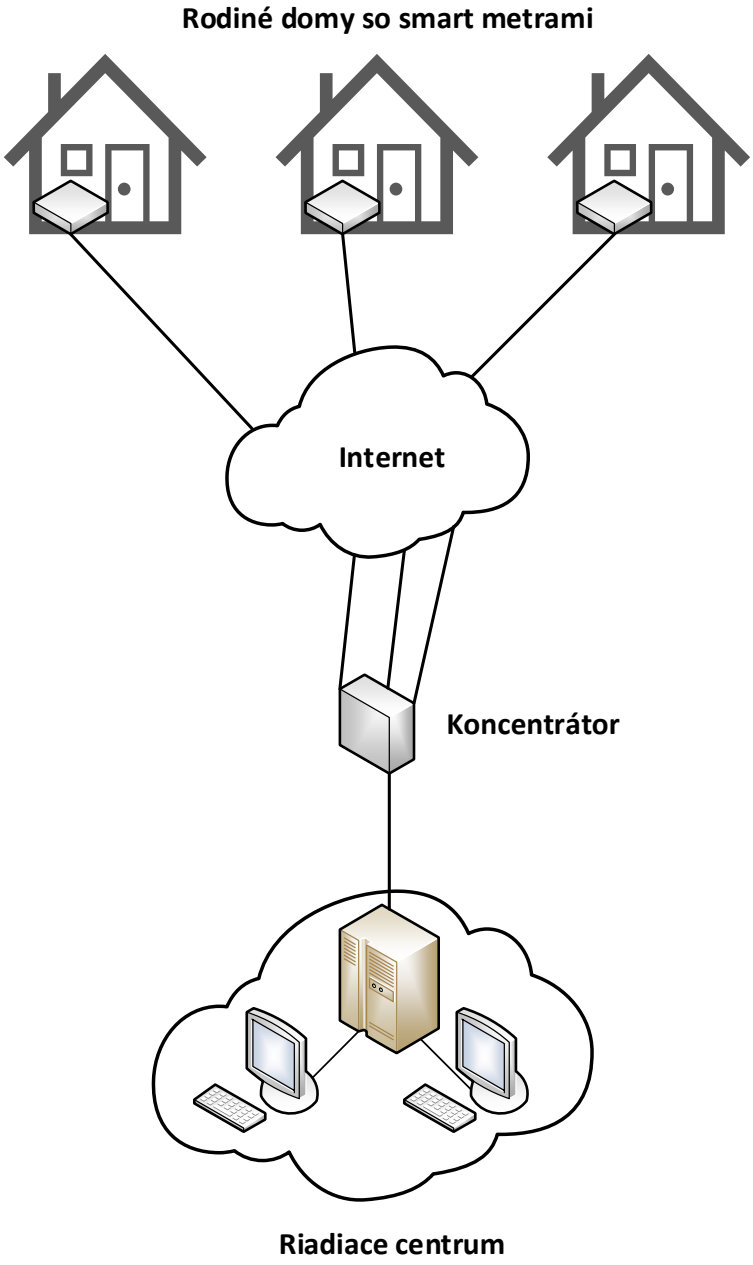
Obr. 1.1: Schéma SCADA systému.

1.1 Popis energetickej siete a zariadení v nej

Energetická sieť alebo smart grid sieť môže vyzereť ako na obrázku 1.2. Zariadenia, ktoré sa v tejto sieti vyskytujú sú smart meter a koncentrátor.

Smart meter je elektornické zariadenie, ktoré zaznamenáva spotrebu energie (elektrina, voda alebo plyn) z meriaceho senzoru a tieto dáta o spotrebe automaticky odosiela na riadiacu centrálu. Ďalej môže smart meter vykonávať aj jednoduché operácie ako napríklad rozpojenie/spojenie breakeru, ktoré odpojí poprípade pripojí určitú oblasť k vybranej energii. Smart meter umožnuje obojstranú komunikáciu medzi

riadiacou centrálou a meracím senzorom. V našom prípade zaznamenáva spotrebu elektrickej energie a dáta o nej posiela na koncentrátor.



Obr. 1.2: Približná predstava siete.

Koncentrátor je zariadenie, ktoré zdromažduje dáta z podriadených meriacích zariadení, archivuje tieto dáta a posiela ich na riadiace centrum. Zároveň zajisťuje

aj bezpečnostné oddelenie riadiaceho centra od jednotlivých častí siete (smart metrov). Hlavná nevýhoda siete by bola, keby sa každý smart meter pripájal na riadiace centrum, ktoré ani nemá toľko vstupov a zároveň by musel mať každý smart meter svoju komunikačnú linku (napr. drôt), čo by bolo veľmi finančne náročné. Zároveň z bezpečnostného hľadiska by musel byť každý komunikačný vstup do riadiaceho centra bezpečnostne zaistený, čo by dávalo veľkú šancu na chybu. A presne na vyriešenie týchto problémov slúži koncentrátor [9].

2 Komunikačné protokoly v energetike

Táto kapitola popisuje protokoly Modbus a DLMS/COSEM, ktoré sú najrozšírenejšie komunikačné protokoly v energetike. Protokol DLMS/COSEM bude popísaný podrobnejšie, kedže v praktickej časti zisťujem kompatibilitu tohto protokolu na zariadeniach. V závere kapitoly sú porovnané tieto dva protokoly medzi sebou.

Komunikácia v energetických sieťach je najčastejšie vedená formou klient-server. Takáto komunikácia sa nachádza medzi smart metrom a koncentrátorom. V tomto prípade je smart meter server a koncentrátor klient. Komunikácia funguje tak, že klient pošle požiadavku na server a ten pošle klientovi požadované dáta. Napríklad, koncentrátor pošle požiadavku na aktuálnu hodnotu spotreby elektrickej energie na smart meter a ten mu hodnotu spotreby pošle.

Aby mohli zariadenia v energetike medzi sebou komunikovať, musia používať rovnaký formát správ a komunikovať rovnakým spôsobom. Na to slúžia komunikačné protokoly.

2.1 DLMS/COSEM

DLMS/COSEM (Device Language Message Specification a Companion Specification for Energy Metering) [8] je skupina štandartov pre komunikáciu so smart metrami a inými meračmi energie (voda, plyn, ...), čím zajisťujú inteligentné meranie a riadenie energie. Tieto štandarty boli prijaté medzinárodnou elektrotechnickou komisiou IEC ako skupina štandartov IEC 62056. Štandarty boli vyvinuté spoločnosťou DLMS User Association, ktorá sa stará o ich rozširovanie a propagáciu.

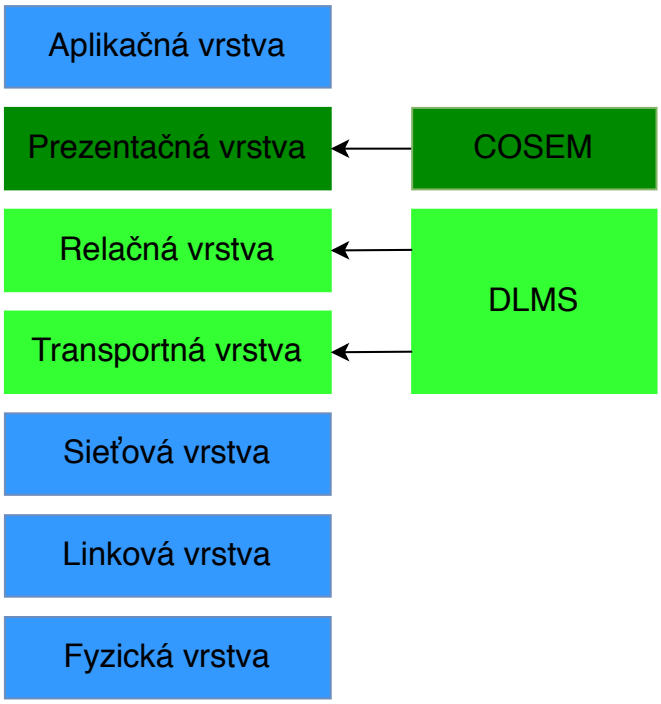
DLMS/COSEM zastáva v referenčnom modele ISO/OSI 4. až 6. vrstvu (Obr. 2.1).

Protokol COSEM sa stará o transformáciu alebo preklad dát do tvaru, ktorý používa aplikačná vrstva. Definuje dátové modely z aplikačnej vrstvy ako objekty pre DLMS a určuje čo sa má s jednotlivými objektami vykonávať. Protokol DLMS sa stará o prenos dát medzi koncovými uzlami. Ďalej sa stará o organizáciu a synchronizáciu tohto prenosu. Na tejto úrovni aj rieši bezpečnosť prenášaných dát.

Komunikácia na nižších vrstvách nie je do protokolu DLMS/COSEM zahrnutá, takže protokol nie je závislý na konkrétnom druhu siete [9, 10, 11, 12, 13, 14].

2.1.1 Adresy a mená zariadení

Adresácia zariadení v sieti je dôležitá, aby vedeli zariadenia kam majú posielať správy, keď chcú komunikovať s druhým zariadením.



Obr. 2.1: Umiestnenie DLMS/COSEM protokolu v referenčnom modele ISO/OSI.

Všetky DLMS/COSEM zariadenia (klienti, servery a tretie strany) sú pomenované pomocou názvu systému - system title (Sys-t). Sys-t musí byť staticky pridelený zariadeniu. Sys-t je 8 oktetov dlhý a musí byť jedinečný. Prvé 3 oktety sú ID výrobcu. Zvyšných 5 oktetov má vytvoriť jedinečné Sys-t. Fyzické zariadenia, môžu byť hostiteľom jedného alebo viacerých logických zariadení (LD - logic device). LD sa jednoznačne identifikujú pomocou LDN - logic device name. Takže na viecero LD môže pripadať jedna Sys-t.

Každé fyzické zariadenie by malo mať svoju adresu. Typ adresy závisí od použitého komunikačného profilu. Podľa profilu to môže byť MAC adresa, IP adresa, telefónne číslo alebo ich kombinácia. MAC adresa môže byť predkonfigurovaná alebo môže byť pridelená počas procesu registrácie [9, 10].

2.1.2 Komunikácia

Celá komunikácia pozostáva z naviazania spojenia medzi klientom a serverom na aplikačnej úrovni. Toto spojenie sa nazýva Application Association (AA). Je to logické spojenie medzi klientom a serverom. Spojenie vzniká tak, že klient posiela žiadosť na server. Ten spojenie potvrdí ak:

- pozná uživaťeľa a zariadenie,
- aplikačný kontext je prijateľný,

- autentizácia navrhnutá klientom odpovedá autentizácii servera.

Aplikačný kontext obsahuje:

- skupinu ASE (Application Service Elements),
- použité odkazovanie SN (Short Name) alebo LN (Logic Name),
- syntax prenosu,
- či sa bude šifrovať a aká bezpečnostná sada (Tab. 2.1) sa použije.

Bezp. sada ID	**Symetrická šifra**	**Dig. podpis**	**Kľúč pre dig. podpis**
0	AES-GCM-128	-	-
1	AES-GCM-128	ECDH s P-256	ECDH s P-256
2	AES-GCM-256	ECDH s P-384	ECDH s P-384

Bezp. sada ID	**Hash**	**Kľúč pre sym. šifrovanie**
0	-	AES-128
1	SHA-256	AES-128
2	SHA-384	AES-128

Tab. 2.1: Bezpečnostné sady [15].

Potom prichádza na rad výmena správ. Po výmene správ sa rozpojí AA. Zariadenie môže používať viac AA zároveň. AA môže byť potvrdzované alebo v prípade broadcastovej komunikácie nepotvrdzované.

Pri využití jednoduchších zariadení sa dá použiť aj jednosmerná multicastová alebo broadcastová komunikácia. Dochádza tu len k výmene správ. AA je tu už vopred prednastavené [9, 10].

2.1.3 Bezpečnosť v DLMS

Bezpečnosť Protokolu DLMS/COSEM sa rieši v časti DLMS. DLMS podporuje niekoľko bezpečnostných služieb. Medzi tieto služby patrí:

- autentizácia - proces overenia autora správy alebo dátového prenosu,
- autorizácia - proces overenia oprávnenia osôb alebo objektov, pre prácu s konkrétnymi dátami alebo objektami,
- dôvernosť - schopnosť utajenia informácie pred neoprávneným užívateľom.

Zabezpečenie týchto služieb sa v DLMS rozdeľuje na dve časti - prístupová a transportná časť.

Prístupová časť

Prístupová časť definuje práva klienta k určeným objektom z COSEM protokolu. Táto časť obsahuje autentizáciu, pretože klient musí najprv dokázať na základe nejakej znalosti, že je ten, za ktorého sa vydáva. Autentizácia môže prebiehať u klienta, serveru alebo u oboch, záleží na vybranej autentizácii.

DLMS definuje tri druhy zabezpečenia:

- Lowest Level Security - Bez autentizácie - slúži pre získanie základných informácii o serveri.
- Low Level Security (LLS) - V tomto prípade server vyžaduje, aby sa klient autentizoval pomocou zadania hesla. Heslo sa nachádza v aktuálnom objekte, takže ho server pozná. Klient zasiela heslo v správe COSEM-OPEN.request. Ak je heslo správne, server mu umožní prístupovať k danému objektu, ak nie, prístup zakáže. Odpoveď o správnosti hesla posiela server v správe COSEM-OPEN.response.
- High Level Security (HLS) - V tomto prípade sa musia klient aj server navzájom autentizovať. Proces autentizácie je 4 cestný typu výzva-odpoveď. Klient pošle serveru výzvu CtoS na jeho autentizáciu. Server správu obdrží a pošle klientovi taktiež výzvu StoC. Klient spracuje výzvu StoC podľa dohodnutých autentizačných mechanizmov (do týchto mechanizmo vstupuje klientov kľúč) a pošle serveru odpoveď r(StoC). Ak výsledok sedí, server akceptuje autentizáciu klienta. Server taktiež spracuje výzvu klienta podľa dohodnutých autentizačných mechanizmov (tu vstupuje zasa kľúč servera) a pošle klientovi odpoveď r(CtoS). Ak výsledok sedí, klient akceptuje autentizáciu servera.

Aký druh zabezpečenia sa vyberie určuje premenná mechanism_id (Tab. 2.2).

mechanism_id	**Autentizácia**
0	Bez zabezpečenia
1	LLS
2	HLS
3	HLS s použitím MD5
4	HLS s použitím SHA-1
5	HLS s použitím GMAC
6	HLS s použitím SHA-256
7	HLS s použitím EC-DSA

Tab. 2.2: Typy zabezpečenia [10].

Transportná časť

Transportná časť sa stará o dôvernosť komunikácie a autorizáciu. Dôvernosť je tu zabezpečená pomocou šifrovania dát. DLMS na šifrovanie určuje niekoľko Security Suites (bezpečnostných sád (Tab. 2.1)), pomocou ktorých sú prenášané dáta zabezpečené. Prvú správu posiela klient bez šifrovania. Na túto správu dostane klient od servera nešifrovanú odpoveď security_options (Tab. 2.1). Hodnota v správe určuje, aká šifrovacia sada sa bude používať. Ďalej správa obsahuje informácie pre vytvorenie kľúčov. Po tejto správe sa už ostatné správy posielajú šifrovane. Na šiforvanie sa používa AES v GCM mode. Ako iniciačný vektor doň vstupuje systémove meno zariadenia (Sys-t), ktoré sa upravuje pomocou náhodnej hodnoty (sčítanie, odčítanie a podobne, záleží na nastavení). Táto náhodná hodnota je potom pridaná k správe. Príjemca si potom z náhodnej hodnoty vyvorí inicializačný vektor, pretože pozná Sys-t zariadenia a matematickú úpravu nad ním.

Autorizácia je tu zabezpečená pomocou políčka unsigned8 (Tab. 2.3). Tu sa nastavuje, aké oprávnenie má daný klient.

unsigned8	**Oprávnenie**
0	Čítanie
1	Zápis
2	Autentizovaná žiadosť
3	Šifrovaná žiadosť
4	Digitálne podpísanážiadosť
5	Autentizovaná odpoveď
6	Šifrovaná odpoveď
7	Digitálne podpísaná odpoveď

Tab. 2.3: Typy oprávnenia [10].

Šifrovanie a autentizácia sú v základnom nastavení nastavené na najnižšiu úroveň, teda vypnuté. Bližšie informácie k celému zostavovaniu, kľúčom a šifrovacím mechanizmom nie sú verejne dostupné [9, 10].

2.1.4 COSEM

COSEM protokol vytvára rozhranie pre zberné systémy a meriace zariadenia aby si mohli vymieňať dáta. COSEM objektovo orientovane pristupuje k fyzickým zariadeniam. Server rozdeľuje na 3 úrovne:

- fyzické zariadenie,
- logické zariadenie (LD),

- COSEM objekty.

Fyzické zariadenie je čisto len zberný systém alebo meracie zariadenie. Na fyzickom zariadení sa nachádza jedno alebo viac logických zariadení. Fyzické zariadenie musí obsahovať aspoň riadiace logické zariadenie. To je dôležité pre komunikáciu a riadenie ostatných LD na zariadení. LD sú programy, ktoré bežia na fyzickom zariadení a plnia určitú úlohu. V každom LD sa nachádzajú COSEM objekty. Tie sa skladajú z metód a atribútov. Hodnoty atribútov charakterizujú objekt. Metódy sú, ako napovedá názov, metódy ktoré môže LD vykonávať. Dôležitým atribútom sú napríklad logic name (LN) a short name (SN). Sú to identifikátory objektu.

Pre odkazovanie na objekty sa používa buď SN alebo LN. SN sa používa u jednoduchších zariadení. Je to 13bitové číslo. LN je definované pomocou OBIS (kapitola 2.1.5). Skladá sa zo 6 hodnôt [9, 16].

2.1.5 OBIS

OBIS (Object Identification System) je to systém pre identifikáciu COSEM objektov a údajov prenášaných počas komunikácie. Identifikuje aj dáta na zariadení. Vytvorený identifikátor sa skladá z 6 stupín od A do F [9, 12, 16, 17].

Skupina A

Hodnota v skupine A určuje typ energie, s ktorou zariadenie pracuje, napríklad (voda, elektrina, plyn, ...).

Skupina B

Hodnota v skupine B určuje komunikačný kanál, z ktorého pochádzajú namerané dáta, pretože meracie zariadenie môže byť spojené so zberným zariadením viacerými komunikačnými linkami.

Skupina C

Hodnota v skupine C specifikuje druh meraných dát v danej energetickej skupine, napríklad (napätie, prúd, výkon, teplota).

Skupina D

Hodnota v skupine D určuje typy a výsledky zo spracovania hodnôt zo skupiny A až C.

Skupina E

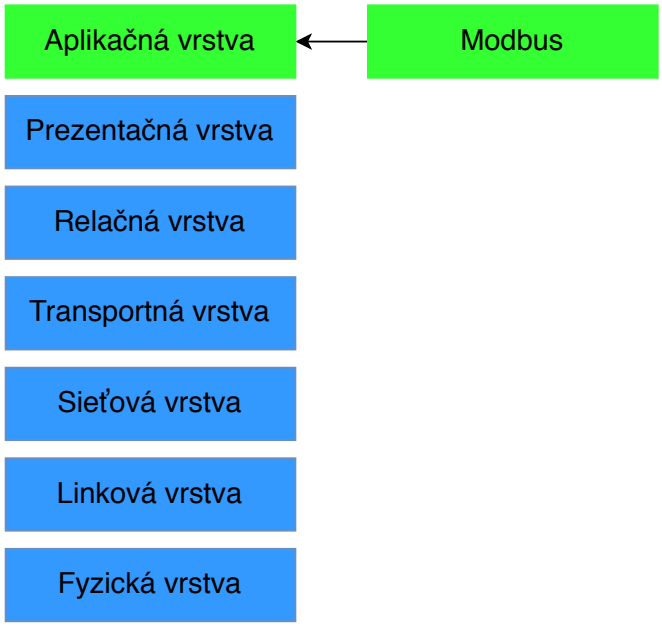
Hodnota v skupine E definuje spracovanie výsledkov meraní hodnôt z predchádzajúcich skupín.

Skupina F

Hodnota v skupine F definuje spôsob ukládania dát z blokov A až E. Môže sa použiť aj na načítanie histórie predchádzajúcich hodnôt.

2.2 Modbus

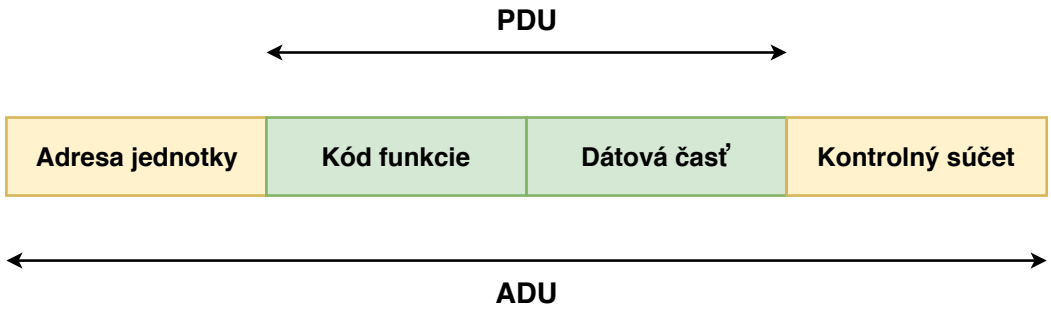
Modbus protokol [18] bol vytvorený v roku 1979 firmou MODICON. Je to protokol pre výmenu komunikácie medzi rôznymi zariadeniami v SDACA sieťach. Od roku 2000 je protokol verejný. V referenčnom modele ISO/OSI zastáva aplikačnú vrstvu (Obr. 2.2). Modbus zabezpečuje komunikáciu typu klient-server, kde klient zašle žiadosť na server a ten mu pošle odpoveď. Žiadosť je špecifikovaná kódom funkcie. Modbus umožňuje komunikáciu po mnohých typoch siete (napríklad TCP/IP Ethernet, sériove linky, vysokorýchlostné siete ...), pretože je nezávislý na nižších vrstvách.



Obr. 2.2: Umiestnenie protokolu Modbus v referenčnom modele ISO/OSI.

Modbus určuje štruktúru správy na úrovni PDU - Protokol Data Unit. Táto správa obsahuje kód funkcie a dátovú časť. V závislosti na typu siete potom Mod-

bus pridáva adresu jednotky a kontrolný súčet k PDU a vytvára tak správu na úrovni aplikačnej vrstvy ADU - Aplication Data Unit (Obr. 2.3).



Obr. 2.3: Modbus správa.

Kód funkcie udáva serveru aký typ operácie sa má uskutočniť. Dátová časť slúži na uskutočnenie operácie. Obsahuje napríklad, ktorého registru sa daná operácia týka alebo aká je adresa registru a podobne.

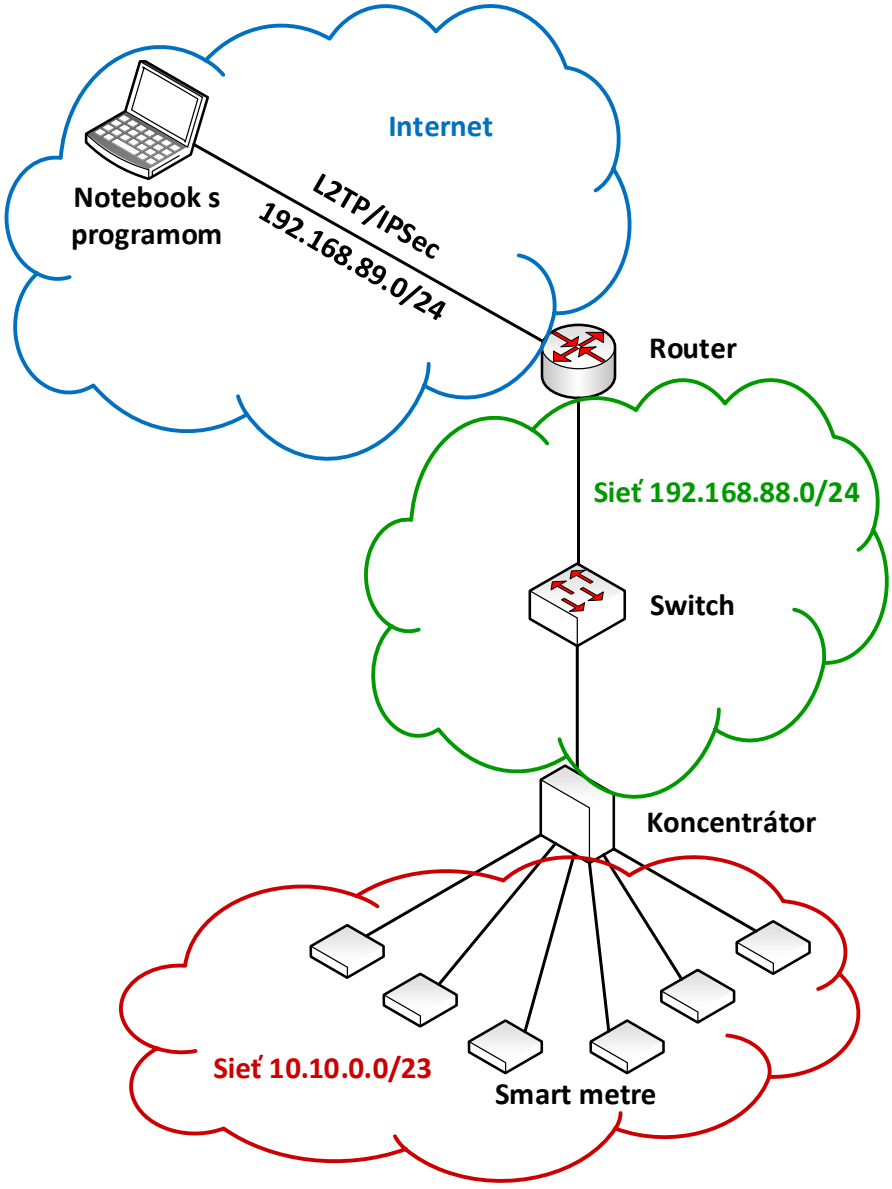
Keď server posiela odpoveď tak do kódu funkcie dá buď kód danej operácie, ktorá mu bola zadaná, alebo chybový kód. To záleží, či sa daná operácia podarila. Ak áno, nachádza sa tam kód operácie. Ak nie, nachádza sa tam chybový kód. V dátovej časti sa nachádzajú požadované dáta [19, 20, 21].

2.3 Porovnanie DLMS/COSEM a Modbus

Aj keď sú DLMS/COSEM a Modbus oba komunikačné protokoly, líšia sa vo viacerých veciach. DLMS/COSEM je určený pre komunikáciu v energetike, zatiaľ čo Modbus je pre komunikáciu medzi akýmikoľvek zariadeniami v SCADA systémoch. Ďalší rozdiel je, že pri Modbus protokole sa ako komunikačné prostredie najčastejšie používa sériová linka, zatiaľ čo u DLMS/COSEM siete TCP/IP. Oba protokoly ale podporujú rôzne druhy komunikačných prostredí. Ďalšia odlišnosť je, že Modbus zastáva len 7. vrstvu ISO/OSI oproti DLMS/COSEM, ktoré zastáva 4. až 6. vrstvu. To sa odzrkadluje tak, že Modbus musí mať svoju vlastnú aplikáciu na riadenia zatiaľ čo DLMS/COSEM sa dokáže prispôsobiť viacerým aplikáciam. Taktiež keďže obsahuje DLMS/COSEM 4. a 5. vrstvu riadi si prenos komunikačných správ sám, zatiaľ čo Modbus necháva všetko na ostatných protokoloch nižších vrstiev. Týmto je ale zasa zložitosť u Modbus protokolu menšia ako u DLMS/COSEM. Ďalší rozdiel je, že DLMS/COSEM si zabezpečuje bezpečnosť sám, pretože obsahuje v sebe protokoly na bezpečnosť. Modbus bezpečnosť v svojom protokole nerieši. Bezpečnosť u Modbus sa rieši tak, že v spodnejších vrstvách sa používa TLS protokol, to je ale len v prípade TCP/IP siete [21].

3 Laboratórne prostredie pre automatizovaný skener

Laboratórne prostredie bolo vytvorené v laboratórnej miestnosti na Vysokém učení technickém v Brně na Fakulte elektrotechniky a komunikačních technologií. Laboratórne prostredie je určené pre viacero študentov záverečných ročníkov a ich bakalárske a diplomové práce.



Obr. 3.1: Popis laboratórneho prostredia.

Zariadenia v Laboratórnej sieti spadajú pod Non-Disclosure Agreement [22]. Táto dohoda neumožnuje zverejňovať názvy, princípy fungovania a vizuálnu po-

dobu zariadení. Preto nebudem v popise laboratórneho prostredia uvádzať konkrétne značky zariadení a budem tieto zariadenia len obecne nazývať router, switch, smart meter a koncentrátor. Kvôli tejto dohode neuverejním ani fotografiu laboratórneho prostredia, ale len jeho topológiu v grafickom zobrazení.

Laboratórne prostredie sa skladá z týchto zariadení:

- router,
- switch,
- koncentrátor,
- 6 smart metrov.

K prostrediu nemám priamy prístup, preto sa budem k nemu vzdialene pripájať pomocou VPN spojenia (Obr. 3.1). Môj počítač sa musí najprv pripojiť k VPN serveru s verejnou adresou. Typ pripojenia je L2TP/IPSec. Po pripojení dostane počítač IP adresu z rozsahu 192.168.89.0/24. Laboratórna sieť je v rozsahu 192.168.88.0/24. Tu sa nachádza koncentrátor, ktorý má viac sieťových rozhraní. Ku koncentrátoru je pripojených 6 smart metrov s IP adresami v rozsahu 10.10.0.0/23.

4 Popis automatizovaného skeneru

Automatizovaný skener je aplikácia navrhnutá a vytvorená pre laboratórne prostredie popísane v kapitole 3. Automatizovaný skener slúži na detekovanie zariadení v rozsahu siete, ktorý je mu zadaný. Na zariadeniach, ktoré nájde, zistí otvorené porty. Ak tieto porty patria protokolom HTTP, SSH alebo Telnet, spustí na tieto porty predpripravené útoky. Potom skener overí, či sú detekované zariadenia smart metre kompatibilné s DLMS/COSEM protokolom, a ak áno, overí či sú zraniteľné na útok DOS a rozpojenie breakeru.

Automatizovaný skener sa skladá z troch častí:

- TesterDLMS,
- skript,
- webová aplikácia.

Každá časť skeneru funguje samostatne. Tento návrh automatizovaného skeneru som zvolil preto, aby bolo možné skener v budúcnosti rozšíriť o dalšie funkcie. Zároveň aj preto, ak by chcel uživateľ použit skener bez webovej aplikácie, alebo ak by chcel použiť vlastnú webovú aplikáciu pre tento skener.

Celý koncept automatizovaného skeneru pozostáva z toho, že klient zadá vo webovej aplikácii adresu siete, na ktorej má byť uskutočnený sken. Webová aplikácia spustí časť skript s touto adresou. Skript oskenuje sieť, zistí všetky zariadenia v nej, porty zariadení a pustí prípadné útoky na zariadenia. Potom skript spustí TesterDLMS s adresami zariadení, ktoré v rozsahu našiel. TesterDLMS zistí, či sa jedná o smart meter, ktorý je kompatibilný s DLMS/COSEM protokolom. Ak áno, spustí na smart meter útoky DOS a rozpojenie breakeru. Všetky výsledky skriptovej ale aj TesterDLMS časti sú uložené do textových súborov. Po dobehnutí skriptu a TesterDLMS časti, zoberie webová aplikácia výsledky z textových súborov a zobrazí ich v tabuľkách.

Tento automatizovaný skener je prisposobený a optimalizovaný pre zariadenie Raspberry Pi s operačným systémom Raspbian Buster Lite [4].

V následujúcich podkapitolách bližšie popíšem fungovanie jednotivých častí automatizovaného skeneru.

4.1 TesterDLMS

TeterDLMS je program napísaný v jazyku JAVA. Pre jazyk JAVA som použil open-source implementáciu OpenJDK 1.8.0_242 [23]. Jazyk JAVA som si zvolil preto, lebo mám v ňom najväčšie skúsenosti s programovaním. Ďalší dôvod je, že v tomto programe používam knižnice a zdrojové kódy, ktoré slúžia na zostavenie spojenia

a komunikáciu so smart metrom pomocou DLMS/COSEM protokolu. Tieto knižnice sú tiež napísané v jazyku JAVA.

V tomto programe používam nástroj Apache Maven [24], ktorý slúži na správu a zostavovanie aplikácií nad platformou JAVA. Nástroj využíva súbor pom.xlm, v ktorom si uživateľ nadefinuje, ktoré balíčky a knižnice sa majú do aplikácie stiahnuť.

V programe su použité:

- knižnice Gurux.DLMS [25] a Gurux.Net[26],
- časť zo zdrojového kódu bakalárskej práce „Zátěžový generátor zpráv DLMS/COSEM“[9].

Gurux.DLMS a Gurux.Net sú open-source knižnice od Fínskej spoločnosti Gurux. Gurux.Net slúži na zostavenie TCP a UDP spojenia medzi klientskou aplikáciou a smart metrom. Gurux.DLMS slúži na zostavenie DLMS/COSEM spojenia a následné zasielanie DLMS/COSEM správ medzi klientským programom a smart metrom komunikujúcim pomocou protokolu DLMS/COSEM. Knižnice sú stiahnuteľné pomocou nastroja Maven.

Zo zdrojového kódu z bakalárskej práce „Zátěžový generátor zpráv DLMS/COSEM“ využívam triedy:

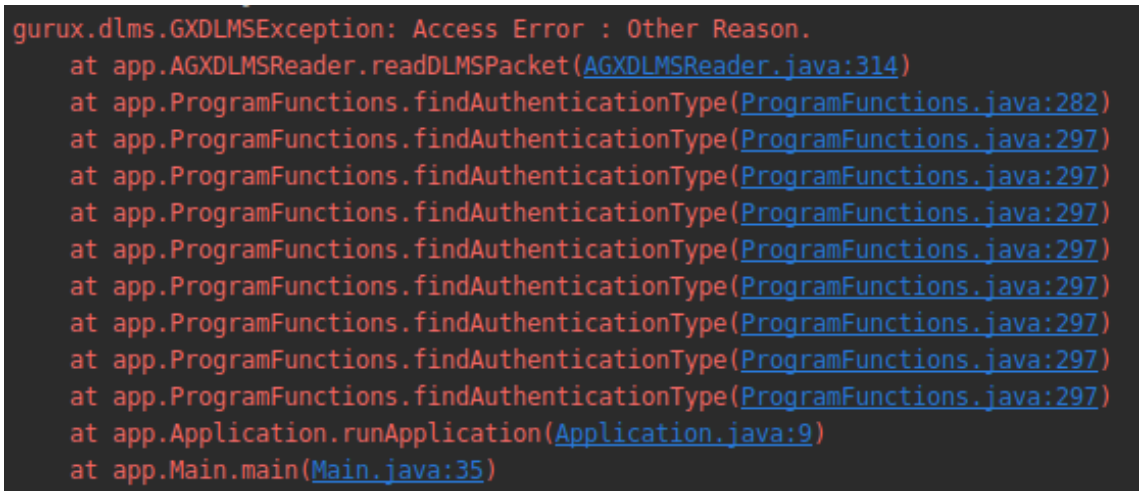
- `AGXCommon`,
- `AGXDLMSSecureClient`,
- `AGXDLMSClient`,
- `AGXDLMSReader`.

Prvé tri triedy sú upravené triedy z knižnice Gurux.DLMS. Posledná trieda je upravená časť zdrojového kódu z gurux.dlms.client.example.java[27], ktorý obsahuje základné funkcie na vytvorenie klienta a na následnú komunikáciu pomocou DLMS/COSEM protokolu. Zdrojový kód bakalárskej práce som obdržal od vedúceho mojej diplomovej práce.

Tieto triedy sú upravené preto, lebo smart metre, ktoré sa nachádzajú v laboratórnom prostredí nemajú implementované všetky časti DLMS/COSEM protokolu presne podľa špecifikácie štandardu. Smart metre používajú typ zabezpečenia HLS (popis v kapitole 2.1.3). Hlavná časť, ktorá u smart metrov neodpovedá špecifikácii je výpočet odpovedí r(CtoS) a r(StoC) na výzvu CtoS a StoC u autentizácie.

Preto, ak použijeme pôvodne triedy s knižnice Gurux.DLMS, autentizácia skončí chybovou hláškou ako na obrázku 4.1, pretože vypočítaná odpoveď na výzvu nesedí s odpoveďou, ktorú očakáva smart meter.

V upravených triedach sa nachádza zachytený záznam komunikácie tohto typu smart metra. Z tohto záznamu je zobraná odpoveď r(StoC) na výzvu StoC od smart



Obr. 4.1: Chybová hláška pri pôvodnej triede z knižnice Gurux.DLMS.

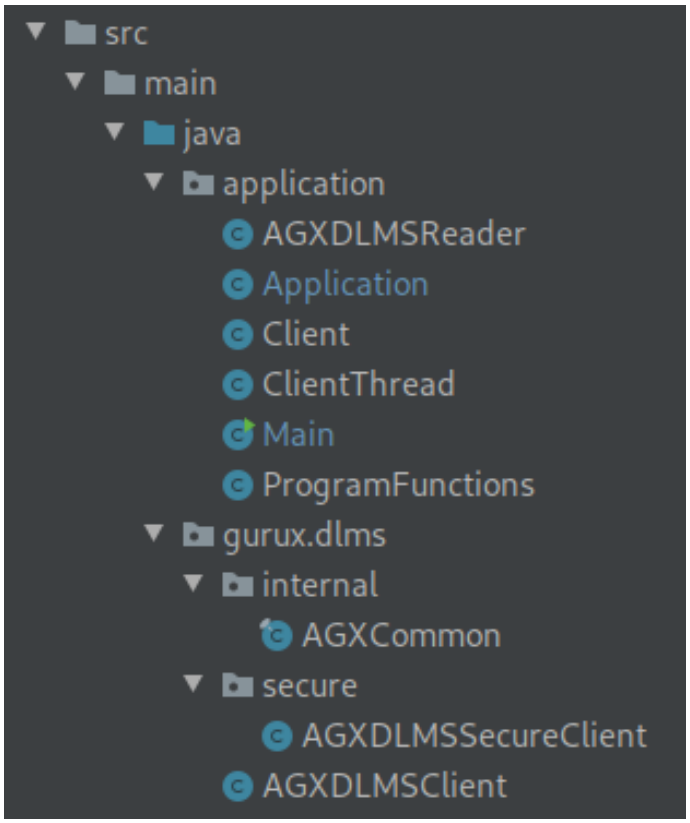
metra, ktorú smart meter akceptuje. Táto odpoveď je zaslaná smart metru u autentizácie, a tým sa vytvorený klient autentizuje voči smart metru. Odpoveď r(StoC) zo zachytenej komunikácie funguje preto, lebo obsah výzvy StoC, ktorú posiela smart meter, je vždy rovnaký. To vidím ako veľký bezpečnostný problém, pretože nám to dovoľuje spraviť tento postup, ktorý je v podstate útok prehraním. Odpoveď r(CtoS) na výzvu CtoS od klienta, ktorú pošle smart meter, upravená trieda ignoruje a vyhodnotí ju ako správnu. Týmto je odstránená autentizácia smart metra voči klientovi. Tento fakt nám neprekáža, pretože užívateľ, ktorý používa automatizovaný tester nepotrebuje mať overenú identitu smart metra, keďže tento smart meter testuje.

Kedže bakalárska práca pracovala s rovnakým typom smart metrov ako sa nachadzajú v laboratórnom prostredí, príde mi zbytočné aby som znova riešil tieto problémy so špecifikáciou. Preto používam tieto upravené triedy v programe.

Ako vývojové prostredie pre program používam intelliJ IDEA ULTIMATE [28]. Následujúca časť bližšie popisuje štruktúru a obsah programu TesterDLMS.

4.1.1 Popis tried a funkcii

Na obrázku 4.2 môžeme vidieť štruktúru tried programu TesterDLMS. Triedy, ktoré som nepopisoval v predchádzajúcej časti, budem popisavať v poradí v akom v programe bežia. V popise budem popisovať len hlavné funkcie v triedach. Podrobná dokumentácia ku všetkým funkciam sa nachádza v zdrojovom kóde.



Obr. 4.2: Štruktúra tried časti TesterDLMS.

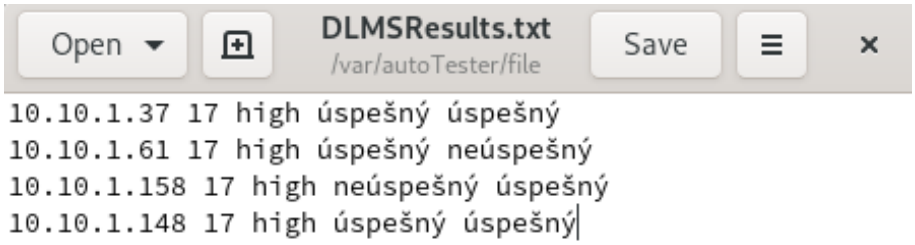
Trieda Main

`Main` je hlavná spustiteľná trieda programu. Trieda `Main` sa spúšťa s parametrami IP adresa zariadenia, u ktorého chceme zistiť či je smart meter s DLMS kompatibilitou a s ladiacou hodnotou. Doporučená ladiaca hodnota je 5. Ladiaca hodnota je počet sekúnd, koľko budú funkcie v programe čakať na odpoveď od smart metru. Ladiacu hodnotu budem ešte podrobnejšie vysvetlovať ďalej v triede `ProgramFunctions`.

Trieda Main v sebe spúšta funkciu `runApplication` z triedy `Application` do ktorej vstupujú parametre `ipAddress, password` a `time`. Do premennej `ipAddress` vstupuje IP adresa zariadenia. Do premennej `password` vstupuje heslo, ktoré sa využíva pri autentizácii na výpočet odpovede na výzvu a do `time` vstupuje ladiaca hodnota. Keďže by autentizácia aj napriek správnemu heslu neprešla, ako som popisoval v predchádzajúcej časti, heslo je zvolené ako náhodná hodnota. Heslo musí byť vždy zadané, lebo inak by neskôr nešiel vytvoriť klient pre komunikáciu so smart metrom.

Trieda Application

Trieda `Application` zabezpečuje celý beh programu. Je to hlavná trieda programu. Vo svojej funkcii `runApplication` najskôr vytvorí klienta pomocou funkcie `createClient` z triedy `Client` s parametrami `ipAddress` a `password`. Potom nastaví hodnotu `time` triede `ProgramFunctions`. Smart meter z laboratórneho prostredia, používa na komunikáciu logické mená, preto ďalej funkcia nastaví klientovi klientskú adresu a typ autentizácie, ktoré získa z funkcií `findClientAddress` a `findAuthenticationType` z triedy `ProgramFunctions`. Po nastavení týchto hodnôt sputia funkcie `DOS` a `remoteCloseOpenBreaker` z triedy `ProgramFunctions`, ktoré slúžia na zistenie, či je smart meter zraniteľný na útok DOS a rozpojenie breakeru. Potom funkcia vytvorí súbor DLMSResults.txt, ak ešte neexistuje, a pridá IP adresu, klientskú adresu, typ autentizácie, výsledok na zranitelnosť DOS a výsledok na zranitelnosť rozpojenie breakeru na koniec súboru. Medzi každú z týchto hodnôt je pridaná medzera. Obsah súboru vyzerá ako na obrázku 4.3.



Obr. 4.3: Obsah súboru DLMSResults.txt.

Trieda Client

Trieda `Client` slúži na vytvorenie klienta, ktorý bude s daným smart metrom komunikovať. Funkcie v triede ďalej nastavujú hodnoty danému klientovi. Hlavná funkcia v triede je `createClient`. Do tejto funkcie vstupujú parametry `ipAddress` a `password` ktoré funkcia nastavý vytvorenému klientovi.

Trieda ProgramFunctions

Trieda `ProgramFunctions` obsahuje všetky dôležité funkcie programu. Hlavné funkcie triedy su:

- `findClientAddress`,
- `findAuthenticationType`,
- `DOS`,
- `remoteCloseOpenBreaker`.

Funkcia `findClientAddress` slúži na zistenie či je zariadenie smart meter kompatibilný s DLMS/COSEM protokolom a k zisteniu klientskej adresy. Keďže smart metre v laboratórnom prostredí komunikujú pomocou logických mien tak klientskou adresou vlastne pomenovávam logické meno klienta v komunikácii medzi klientom a smart metrom. Do funkcie vstupujú parametre `client` a `start`. Parameter `client` je aktuálny klient pre ktorého zistujeme adresu, `start` je hodnota od ktorej začíname hladať klientskú adresu. Klientská adresa pre pripojenie k smart metru môže byť od 16 do 127 [29]. Takže začiatočná hodnota argumentu start je 16. Funkcia nastaví klientovi klientsku adresu na hodnotu 16. Potom klient vo vlákne z triedy `ClientThread` pošle prvú správu z Application Association smart metru. Ak je klientská adresa správna, smart meter na ňu odpovie. Ak nie, smart meter nepošle žiadnu odpoveď. Preto vlákno čaká počet sekund koľko udáva ladiaca hodnota. Ak nepríde odpoveď do tohto času vlákno sa ukončí a klientská adresa sa nastaví o 1 vyššie. Tento proces sa vo funkcii opakuje až dokiaľ smart meter neodpovie (vtedy sme zistili správnu hodnotu klientskej adresy) alebo až do hodnoty klientskej adresy 128 a vtedy program vypíše, že sa nejedná o smart meter s DLMS kompatibilitou a ukončí sa. U smart metrov v laboratórnom prostredí nastáva problém, že smart meter odpovie úspešne na klientskú adresu 16, ale nedokáže na tejto adrese naviazať úspešné spojenie. Toto je ošetrené vo funkcii `findAuthenticationType` kde funkcia overí úspešnosť spojenia a ak nie je úspešné spustí znova funkciu `findClientAddress` ale s parametrom `start` nastaveným na túto nefungujúcu klientskú adresu zvýšenú o 1.

Funkcia `findAuthenticationType` slúži na zistenie správneho autentizačného typu pre spojenie klienta a smart metru. Knižnica Gurux.DLMS podporuje všetky typy autentizácie z tabuľky 2.2 okrem typu HLS s použitím EC-DSA. Do funkcie vstupujú parametre `client` a `typeNumber`. Parameter `client` je klient z predchádzajúcej funkcie doplnený o klientskú adresu. Parameter `typeNumber` je číslo ktoré údáva funkcii ktorý typ autentizácie sa skúsi podľa tabuľky 2.2. Ak je autentizačný typ správny, odpovie smart meter chybovou hláškou: „Access Error : Other Reason.“ (dôvod tejto chybovej hlášky bol popísany v kapitole 4.1). Ak je autentizačný typ nesprávny, smart meter odpovie chybovou hláškou: „Connection is permanently rejected Authentication failure.“. Funkcia funguje tak, že má klient nastavený prvý autentizačný typ „Low“ a vymení si so smart metrom autentizačné správy. Potom funkcia porovná chybovú hlášku. Ak funkcia obdrží chybovú hlášku pre nesprávny typ, tak sa spustí znova s `typeNumber` o jedno vyšším. Ak obdrží správny typ chybovej hlášky, tak uloží autentizačný typ. Ak sa stane, že funkcia vystrieda všetky typy autentizácie a nedostane chybovú hlášku pre správny typ autentizácie, tak je nastavená zlá hodnota klientskej adresy. V tom prípade funkcia znova spustí funkciu

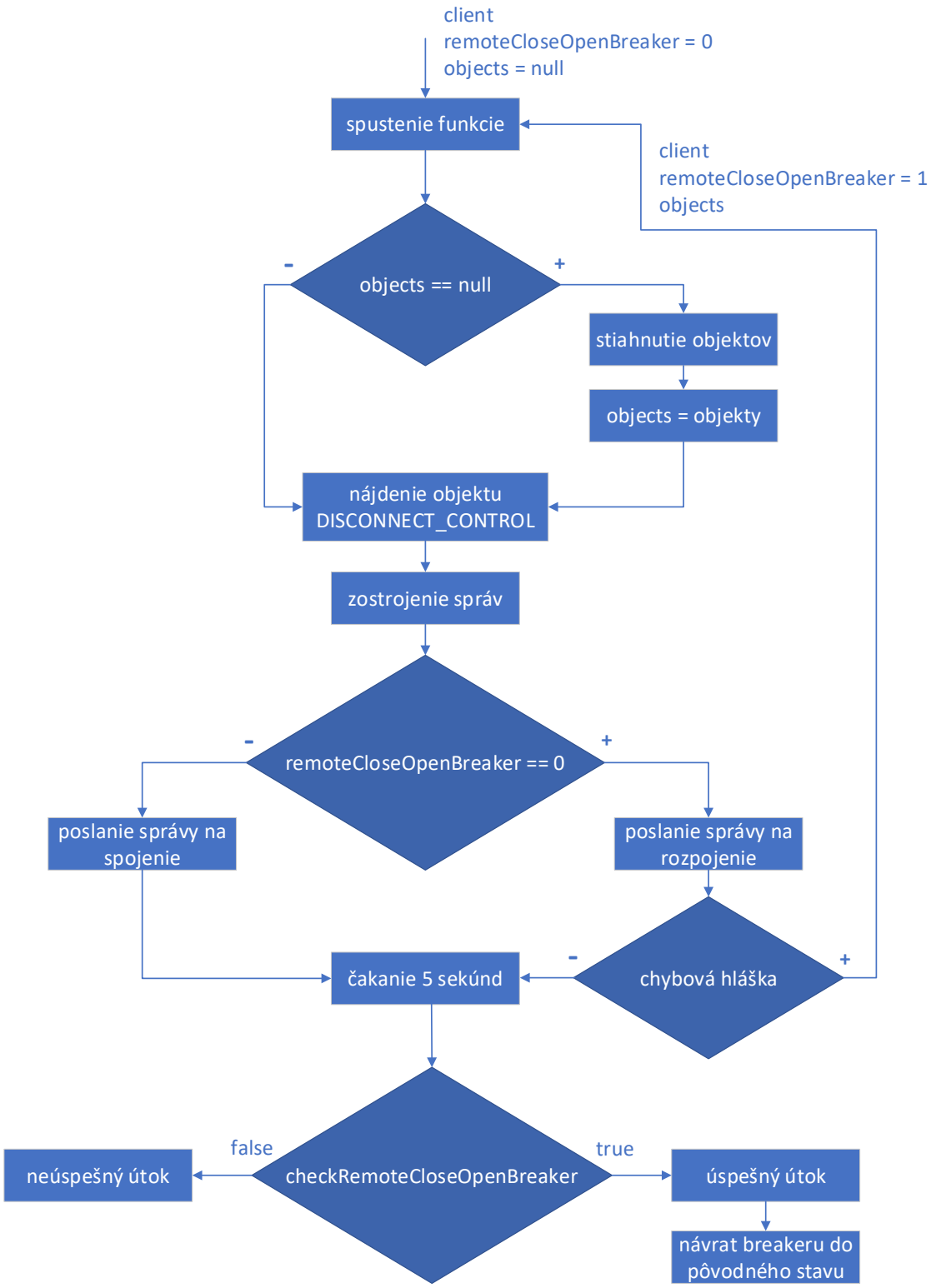
`findClientAddress` s parametrom `start` o jedna vyšším ako je aktuálna klientský adresa. Po získání novej klientskej adresy sa funkcia `findAuthenticationType` spúšta znova s počiatočnou hodnotou parametru `typeNumber`. Týmto je ošetrený problém, že smart meter odpovie úspešne na klientskú adresu 16 ale nedokáže ďalej s touto adresou naviazať úspešné spojenie.

Funkcia `DOS` slúži na otestovanie DOS zraniteľnosti u smart metru. Do funkcie vstupuje parameter `client`. Funkcia robí DOS útok na úrovni DLMS protokolu a to tak, že posiela na server cyklicky po dobu jednej minúty autentizačnú výzvu CtoS. Funkcia pritom nečaká na odpoveď smart metra. Smart meter po obdržaní výzvy, pošle svoju výzvu StoC a začne počítať odpoveď r(CtoS) na obdržanú výzvu. Odpoveď r(CtoS) potom smart meter pošle na klienta. Celý tento proces trvá smart metru dlhšie ako funkcii poslanie novej výzvy CtoS. Po celú dobu trvania funkcie `DOS` je teda smart meter zaneprázdnený zasielaním výzvy StoC a výpočtom odpovede r(CtoS). Funkcia `DOS` overuje nedostupnost smart metra tak, že v druhom vlákne po 20 sekundách spustí funkciu `checkDOS`. Táto funkcia vytvorí druhého klienta, ktorý taktiež pošle autentizačnú výzvu CtoS na smart meter. Ak do 8 sekúnd nedostane tento klient výzvu StoC a odpoveď r(CtoS) od smart metra, tak sa vyhodnotí útok ako úspešný a ukončí sa funkcia `checkDOS` a `DOS`. Hodnotu 8 sekúnd som odvodil testovaním na laboratórnom prostredí. Je to dvojnásobok času, do ktorého zvyčajne príde odpoveď.

Funkcia `remoteCloseOpenBreaker` je najzložitejšou funkciou v programe. Preto som pre vysvetlenie tejto funkcie vytvoril vývojový diagram (Obr. 4.4). Vývojový diagram nie je vytvorený presne podľa zdrojového kódu, ale je zjednodušený, aby išlo fungovanie funkcie jednoduchšie pochopiť. Vývojový diagram bude vysvetlený nižšie v popise funkcie. Funkcia slúži na overenie, či sa dá vzdialene spojiť a rozpojiť breaker. U smart metrov v laboratórnom prostredí je vzdialené rozpojenie a spojenie zakázané. Pri regulérnom pokuse poslať objekt s príkazom rozpojenia alebo spojenia breakeru dostane program naspäť chybovú hlášku: „Access Error : Device reports a temporary failure.“.

Vo funkcii teda využívam zachytenú komunikáciu z bakalárskej práce „Zátěžový generátor zpráv DLMS/COSEM“[9]. V tejto práci sa nachádza odchytená skupina bajtov (Tab. 4.1).

Je to zachytený objekt s príkazom rozpojenia breakeru od koncentrátoru, ktorý je priamo napojený na smart meter. Tento objekt obsahuje hexadecimálne hodnoty. Tieto hodnoty som previedol na decimalný tvar, ktorý je vidieť v druhom riadku. Tento objekt som analyzoval pomocou Gurux prekladača [30], ktorý je dostupný na ich webovej stránke. Útok spočíva v tom, že ak pošleme takto zostrojenú správu, tak



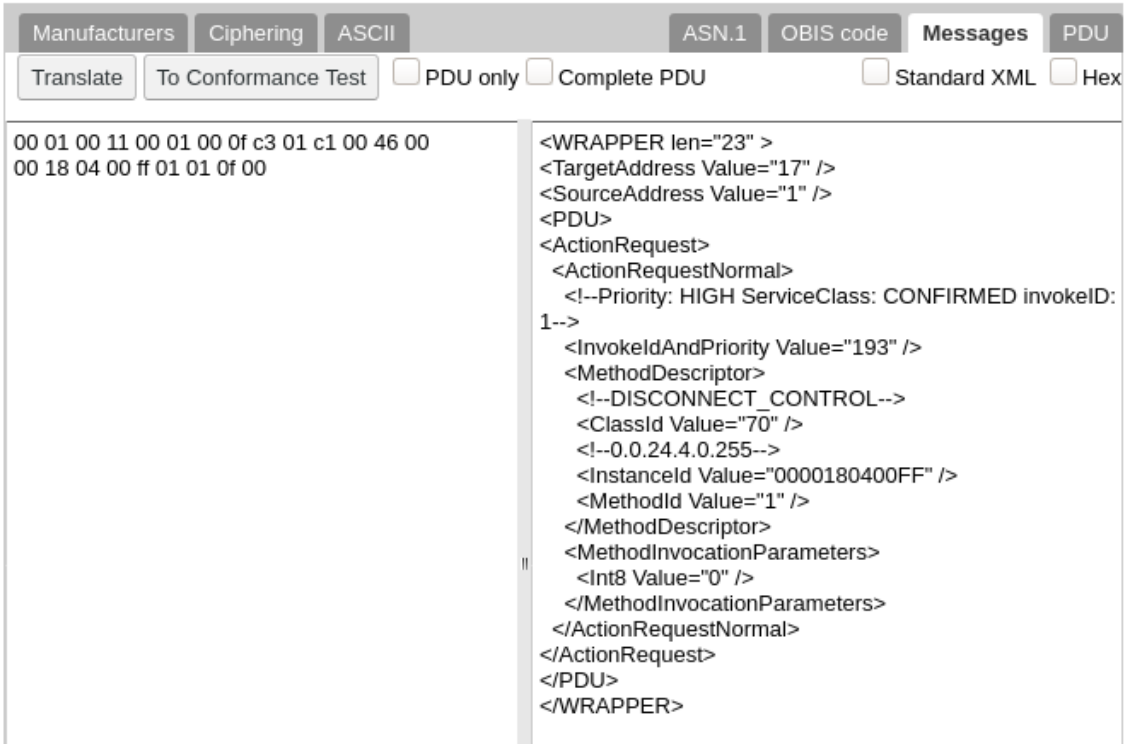
Obr. 4.4: Vývojový diagram funkcie `remoteCloseOpenBreaker`.

si smart meter myslí, že je klient koncentrátor, ktorý je priamo na neho napojený. Z tohto popisu vyplíva, že sa jedná o útok prehraním. Keby smart meter používal

00	01	00	11	00	01	00	0f	c3	01	c1	00
0	1	0	17	0	1	0	15	195	1	193	0

46	00	00	18	04	00	ff	01	01	0f	00
70	0	0	24	4	0	255	1	1	15	0

Tab. 4.1: Zachytený objekt s hexadecimálnymi hodnotami a ich prevod na decimálne hodnoty.



Obr. 4.5: Preklad bajtov pomocou Gurux prekladača.

vo svojich správach časové razítka alebo iný kryptografický prvok spojený s overením času, bol by chránený voči tomuto typu útoku. Ak by sme tento útok použili na reálnej sieti a rozpojili by sme breaker u smart metru, znamenalo by to, že by sme odpojili domácnosť alebo inú oblasť, v ktorej sa nachádza smart meter, od elektrickej energie. Preto je táto zraniteľnosť veľmi kritická.

Z obrázka 4.5 vieme určiť čo znamenajú žlto označené bajty z tabuľky 4.1. Breaker je v laboratórnych smart metroch označovaný pod názvom DISCONNECT_CONTROL.

- 17 - klientská adresa (logické meno klienta),
- 1 - adresa pre prístup k smart metru (logicke meno smart metru),
- 70 - čiselne označenie pre DISCONNECT_CONTROL,

- 0, 0, 24, 4, 0 a 255 - logické meno pro objekt DISCONNECT_CONTROL,
- 1 - hodnota pre rozpojenie breakeru (hodnota 2 by bola pre spojenie breakeru).

Ako môžeme vidieť vo vývojovom diagrame (Obr. 4.4) do funkcie `remoteCloseOpenBreaker` vstupujú parametre `client`, `closeOrOpenBreaker` a `objects`. Parameter `client` je klient, ktorý bol použitý v predchádzajúcej funkcii. Parameter `closeOrOpenBreaker` je číselná hodnota, ktorá určuje či predpokladáme, že je breaker spojený alebo rozpojený. 0 znamená spojený, 1 znamená rozpojený breaker. Na začiatku je nastavená 0. Parameter `objects` sú objekty zo serveru. Tento parameter slúži na to, aby sa pri opätovnom volaní funkcie nemuseli znovu sťahovať objekty zo smart metru. Funkcia `remoteCloseOpenBreaker` funguje tak, že si vyžiada od smart metra všetky objekty. V týchto objektoch nájde objekt typu DISCONNECT_CONTROL (tento objekt symbolizuje breaker) a zoberie z neho potrebné informácie pre zostavenie správ pre rozpojenie a spojenie breakeru. Potom funkcia vytvorí dve správy jednu na rozpojenie a druhú na spojenie breakeru. Správy sú vytvorené tak, že tie bajty, ktoré sa v tabuľke 4.1 nachádzajú v bielych políčkach, sa zopakujú a tie, čo sa nachádzajú v žltých políčkach sú nastavené podľa hodnôt z klienta a hodnôt získaných z objektu typu DISCONNECT_CONTROL. Ak je breaker spojený a pošle na neho funkcia správu pre rozpojenie, smart meter neodpovie žiadnou odpoveďou a rozpojí sa. Ak je ale smart meter rozpojený a pošleme na neho funkcia správu o rozpojenie, smart meter odpovie chybovou hláškou: „Access Error : Device reports a temporary failure.“. Tento fakt potom ďalej využíva funkcia na otestovanie zraniteľnosti. Po zostavení správ sa funkcia pozrie na parameter `closeOrOpenBreaker`. Keďže je 0, pošle funkcia správu na rozpojenie smart metru. Ak je breaker pred zaslaním správy rozpojený smart meter vráti chybovú hlášku. V tedy sa spustí funkcia znova ale s parametrom `closeOrOpenBreaker` rovnajúcim sa 1 a s parametrom `objects` v ktorom sa už nachádzaju objekty zo smart metra.

Ak je breaker pred odoslaním správy spojený, tak sa rozpojí a smart meter neodpovie. V tedy počká funkcia 5 sekúnd (do tohto času stihne smart meter rozpojiť breaker) a spustí funkciu `checkRemoteCloseOpenBreaker`, ktorá skontroluje či sa útok podaril. Funkcia pošle na smart meter požiadavok na rozpojenie breakeru, a ak je breaker naozaj rozpojený smart meter odpovie chybovou hláškou. Vtedy vráti funkcia hodnotu true. Ak sa navráti od smart metra chybová hláška, breaker sa nepodarilo rozpojiť a funkcia vráti hodnotu false. Po dobehnutí funkcie `checkRemoteCloseOpenBreaker` sa výsledok uloží, a ak sa útok podaril vráti funkcia `remoteCloseOpenBreaker` breaker do pôvodného stavu pred útokom.

Trieda ClientThread

Táto trieda reprezentuje vlákno. V tomto vlákne klient posiela autentizačnú výzvu smart metru a čaká na odpoveď. Táto trieda je vždy pustená exekútorom, ktorému nastavíme ladiacu hodnotu, ktorá reprezentuje čas ako dlho má vlákno bežať. Táto trieda sa používa v mnohých funkciách, kde nie je istota, že obdrží klient odpoveď od smart metru. Po uplynutí času, ktorý je nastavený exekútoru sa toto vlákno ukončí. Ak by sme nenastavili čas za ktorý sa vlákno ukončí, a neobdržal by klient odpoveď od smart metru, čakal by klient na odpoveď až do vypršania platnosti TCP spojenia.

4.2 Skript

Časť skript sa skladá z dvoch bash skriptov scriptStart.sh a scriptMetasploit.sh, a z troch textových súborov http.txt, ssh.txt a telnet.txt. Táto časť sa stará o oskenovanie rozsahu siete, ktorý je mu zadaný. Skriptová časť zistí aké stanice sa v rozsahu nachádzajú, a aké porty na týchto staniciach bežia. Ak sa nachádzajú na staniciach otvorené porty 80 (http), 22 (ssh) alebo 23 (telnet) spustí na tieto porty skriptová časť predpripravené útoky. Všetky výsledky z tejto časti sú uložené do textových súborov. Textové súbory vytvorené skriptom scriptStart.sh sa ukladajú do priečinku /var/autoTester/files/. Textové súbory vytvorené skriptom scriptMetasploit.sh sa ukladajú do priečinku /var/autoTester/log/. Toto ukladanie s absolútnou cestou je kvôli webovej aplikácii, ktorá ďalej s textovými súbormi pracuje.

Skript scriptStart.sh sa stará o celý chod tejto časti. Tento skript využíva na skenovanie siete open-source programy Nmap [31] a Masscan [32]. Tieto programy používam preto, lebo sú veľmi výkonné a spoľahlivé a nie som schopný v určenom čase napísať lepší skener ako sú tieto dva, pretože na ich vývoji sa podiela celá open-source komunita.

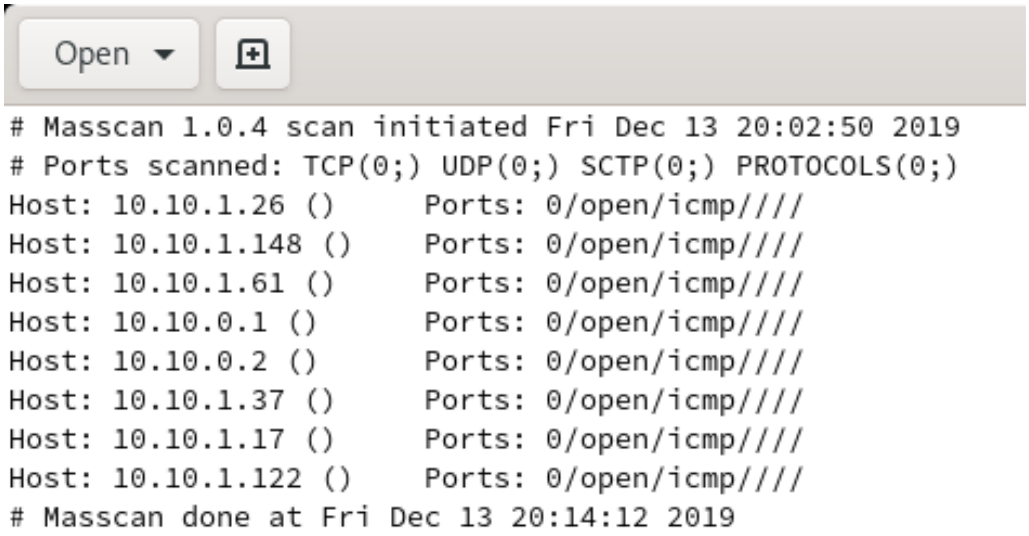
Nmap je open-source bezpečnostný skener portov. Bol vyvinutý Gordonom Lyonom. Na jeho vývoji sa ďalej podiela veľká open-source komunita, preto patrí medzi najlepšie bezpečnostné skenery. Nmap posiela špeciálne upravené pakety do siete. Výsledky z týchto paketov potom analyzuje. Vďaka tomu potom Nmap dokáže vyhľadávať zariadenia v sieti, zistiť ich operačný systém, verzie softwaru, otvorené porty, ale aj firewall a jeho nastavené pravidlá. Nmap je podporovaný na rôznych druhoch operačných systémov vrátane Windows, Linux a Mac OS [31].

Masscan je taktiež open-source bezpečnostný skener portov. Bol vyvinutý Robertom Davidom Grahamom. Na rozdiel od Nmap skenuje len aktívne zariadenia v sieti a ich aktívne porty. Jeho hlavnou výhodou je, že dokáže dobre koordinovať zasielanie paketov s dotazmi na skenované ciele a dosahuje tým velkú rýchlosť. Masscan

je taktiež podporovaný na rôznych operačných systémoch vrátane Windows, Linux a Mac OS [32].

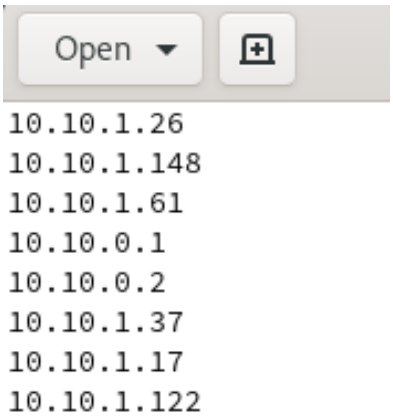
Nmap je oproti Masscanu oveľa presnejší a dokáže zistiť viac informácii o danom zariadení. Na druhú stranu, Masscan je zase pri skenovaní o mnoho násobne rýchlejší ako Nmap a jednoduchšie sa u neho nastavuje paketový tok. Tieto vlastnosti využívam v skripte scriptStart.sh. Na začiatku najprv zistím pomocou programu Masscan, ktoré zariadenia sa nachádzajú v sieti, a potom pomocou programu Nmap podrobne oskenujem už len tieto zariadenia. Toto by malo zaistiť väčšiu rýchlosť automatizovaného skeneru.

Skript scriptStart.sh sa spúšta s parametrom IP rozsahu, ktorý má byť oskenovaný. Na začiatku skript oskenuje zadaný rozsah pomocou programu Masscan, ktorý skúša TCP ping na jednotlivé stanice. Nájdené stanice uloží Masscan do textového súboru scanNet.txt. Obsah súboru vyzerá ako na obrázku 4.6. Potom skript pomocou príkazov `sed` a `awk` upraví súbor tak, aby sa v ňom nachádzali len výsledné stanice. Výsledok uloží do súboru scanNetFinal.txt (Obr. 4.7).



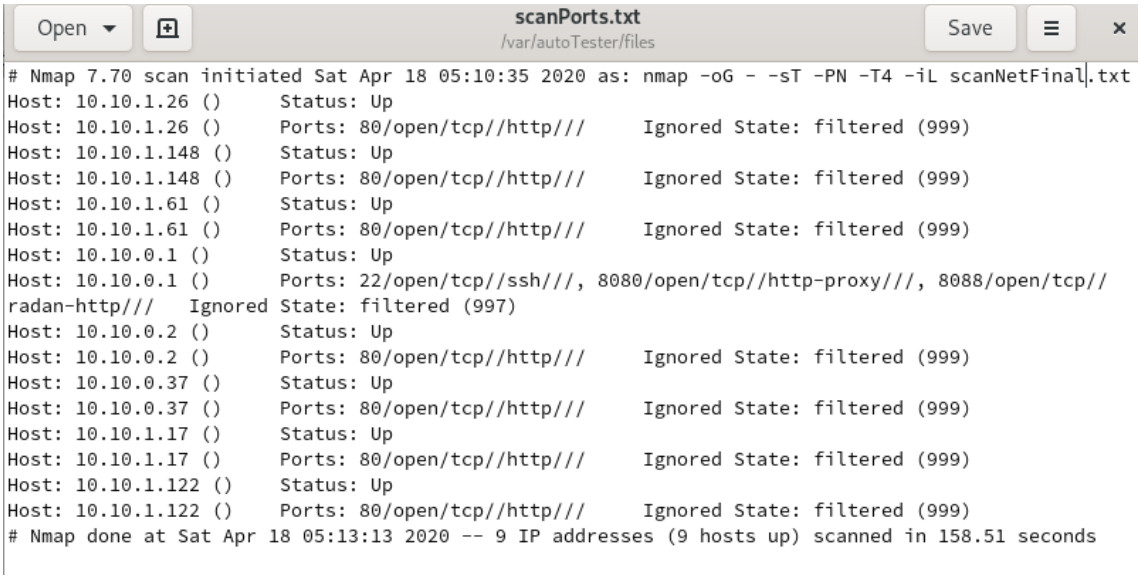
Obr. 4.6: Výstup programu Masscan.

Potom skript oskenuje aktívne porty na týchto staniciach pomocou programu Nmap a výsledok uloží do súboru scanPorts.txt (Obr. 4.8). Nmap má parametre nastavené tak, že vyhľadáva porty podľa zoznamu najpoužívanejších služieb u staníc. Všetky porty neskenujem preto, lebo by to zabralo veľké množstvo času a preto, lebo smart metre využívajú len niekoľko služieb. Nmap preto neoskenuje u zariadení port 4059, ktorý nie je tak často používaný u klasických staníc, a ktorý je port pre DLMS/COSEM protokol. To ale vôbec nevadí, pretože prítomnosť DLMS/COSEM protokolu sa testuje v časti TesterDLMS. Skript ďalej upraví súbor scanPorts.txt



Obr. 4.7: Obsah scanNetFinal.txt.

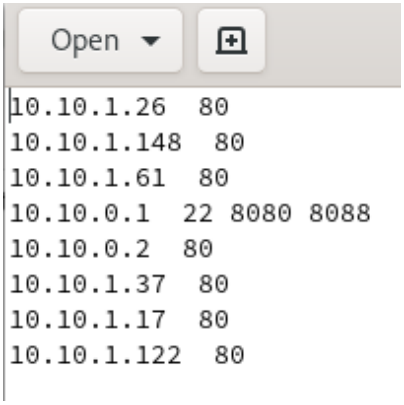
pomocou príkazov `sed` a `awk` tak, aby vo výslednom súbore boli len stanice a ich aktívne porty. Výsledok uloží do textového súboru scanPortsFinal.txt (Obr. 4.9).



Obr. 4.8: Výstup programu Nmap.

Potom skript scriptStart.sh spustí skript scriptMetasploit.sh s parametrom IP adresa a port zo súboru MetasploitAddress.txt (Obsah tohto súboru vyzerá rovnako ako obrázok 4.9 avšak riadky, kde je viac portov rozdelí na viac riadkov tak, aby bol vždy jeden stĺpec IP adresa a druhý stĺpec port).

Skript scriptMetasploit.sh používa v sebe program Metasploit Framework [33]. Metasploit Framework je open-source bezpečnostný program, ktorý slúži na bezpečnostnú analýzu, penetračné testovanie, vývoj a využitie exploitov. Program obsahuje veľa informácii o rôznych zraniteľnostiach služieb. Metasploit Framework je veľmi



Obr. 4.9: Obsah súboru scanPortsFinal.txt.

flexibilný a ľahko rozšíriteľný o mnoho bezpečnostných modulov. Program je podporovaný na rôznych druhoch operačných systémov vrátane Windows, Linux a Mac OS.

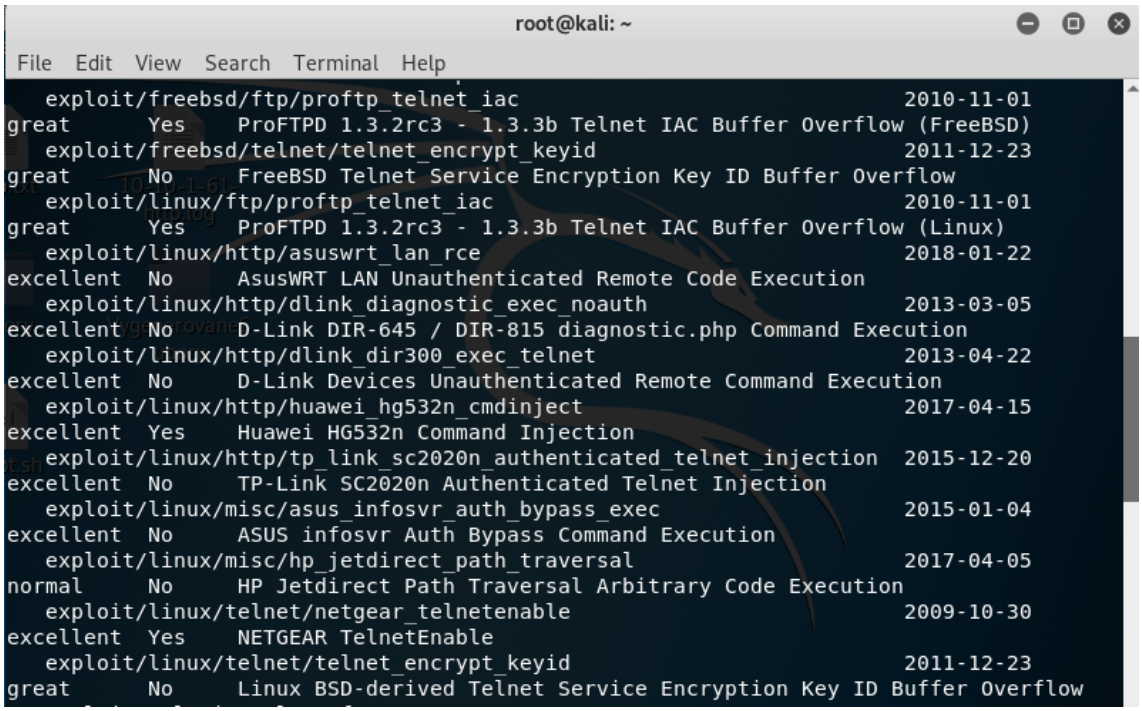
Každý z textových súborov http.txt, ssh.txt a telnet.txt obsahuje 20 názvov útokov z programu Metasploit Framework, ktoré potom scriptMetasploit.sh aplikuje na stanice u ktorých bežia tieto porty. Skript nie je omedzený na presný počet útokov, takže si môže uživateľ upraviť tieto súbory o ľubovolný počet názvov útokov, ktoré chce na tieto porty skúsiť, stačí len dopísať názvy do textových súborov. Názvy útokov môže užívateľ zistiť tak, že spustí Metasploit Framework a zadá príkaz `search <názov protokolu>` a do súbora pridá názvy začínajúce na exploit/ (Obr. 4.10).

Skript scriptMetasploit.sh porovná či je port u adresy 22, 23 alebo 80. Ak port spĺňa podmienku spustí tento skript Metasploit Framework a berie názvy útokov, ktoré majú byť na port použité z textového súboru (ssh.txt, telnet.txt alebo http.txt) pre daný port. Výsledok útokov z Metasploit Frameworku uloží skript do súboru s názvom <IP adresa stanice(namiesto bodiek sú pomlčky)>-<daný port>.log. Vo výsledku vyzerá názov súboru napríklad takto: 10-10-0-1-ssh.log.

Po dobehnutí skriptu scriptMetasploit.sh spustí skript scriptStart.sh skompilovaný program TesterDLMS so vstupnými parametrami IP adresa zo súboru scanNetFinal.txt a ladiaca hodnota, ktorá sa rovná 5 (hodnotu ladiacej hodnoty vysvetľujem v kapitole 5.4). Po dobehnutí programu TesterDLMS skript presunie jeho výsledky do priečinku /var/autoTester/files/ a ukončí sa.

4.3 Webová aplikácia

Ako webovú aplikáciu používam servlet Apache Tomcat 8 [34], pretože je určený pre vývojový jazyk JAVA. Apache Tomcat je open-source webový server a servlet



Obr. 4.10: Výpis Metasploit Framework po príkaze `search telnet`.

kontajner vyvýjaný firmou Apache Software Foundation. Vznikol v roku 1999 a je založený na jazyku JAVA. Servlet je trieda JAVA, ktorá reaguje na HTTP požiadavky. Najznámejší servlet je práve Apache Tomcat [35, 36].

Webovú aplikáciu som taktiež, ako časť TesterDLMS, vyvíjal vo vývojovom prostredí intelliJ IDEA ULTIMATE. Vo webovej aplikácii je použitý nástroj Maven pomocou, ktorého sa skompiluje webová aplikácia na súbor s koncovkou .war, ktorý potom stačí už len vložiť do servletu Apache Tomcat, ktorý túto webovú aplikáciu sprostredkuje pre webový prehliadač.

Webová aplikácia sa skladá z dvoch častí:

- programová časť,
- časť zodpovedná za vzhľad webovej aplikácie.

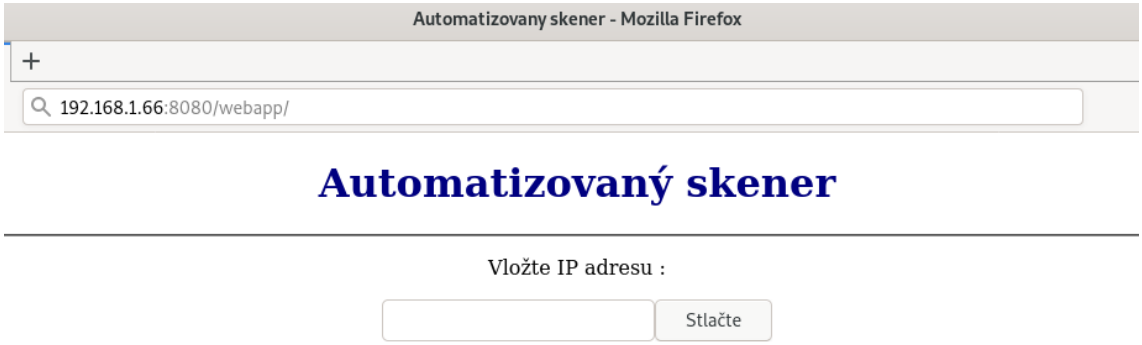
Programová časť sa skladá z dvoch tried `Servlet` a `FindProject`. Trieda `Servlet` sa stará o celú správu webovej aplikácie. Trieda `FindProject` obsahuje funkciu na spustenie skriptu scriptStart.sh v priečinku /var/autoTester/.

Časť zodpovedná za vzhľad webovej aplikácie, sa skladá z:

- index.jsp - definuje obsah vstupnej webowej stránky,
- script.js - obsahuje javascriptové funkcie pre naplnenie tabuliek z textových súborov,
- style.css - určuje vzhľad webových stránok,

- vysledok.jsp - definuje obsah výstupnej webovej stránky.

Súbory index.jsp, style.css a vysledok.jsp sú napísané v jazyku HTML. Súbor script.js je napísaný v javascripte a je v ňom použitá knižnica jQuery [37]. Knižnica jQuery je open-source javascriptová knižnica, ktorú podporuje väčšina webových prehliadačov. Táto knižnica bola vydaná v roku 2006 Johnom Resigom. Knižnica slúži na lepšie prepojenie jazyka javascript a HTML.

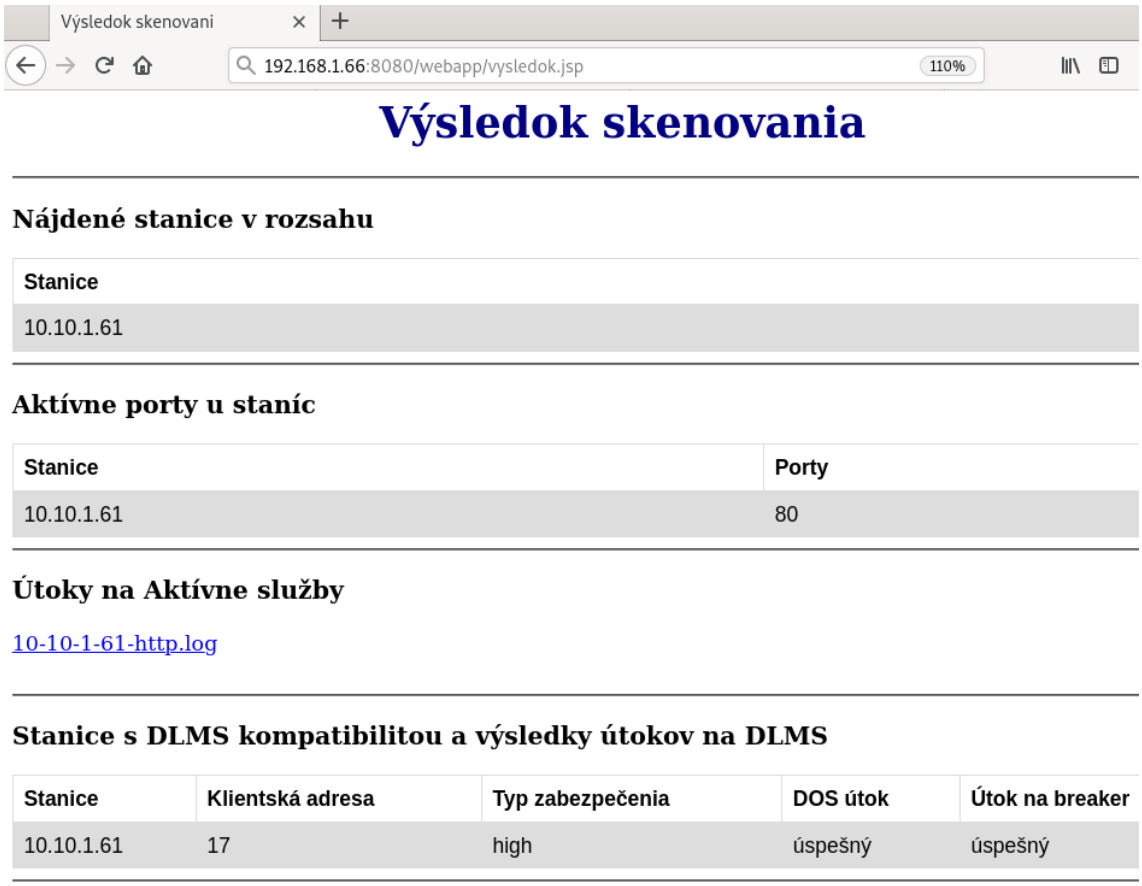


Obr. 4.11: Vstupná webová stránka nadefinovaná v súbore index.jsp.

Webová aplikácia funguje tak, že užívateľ zadá do webového prehliadača adresu http://<IP adresa stroja na ktorom webová aplikácia beží>:8080/webapp. Po načítaní v prehliadači sa zobrazí webová stránka, ktorá je definovaná súborom index.jsp (Obr. 4.11).

Užívateľ zadá do políčka IP adresu s rozsahom, ktorý chce oskenovať, napríklad 10.10.1.0/24, a stlačí tlačidlo stlačte (pre jednoduché zobrazenie ako stránka vyzerá, som zadal IP adresu len jednej konkrétnej stanice). Po stlačení tlačidla spustí funkcia z triedy `FindProject` skript scriptStart.sh s parametrom IP rozsahu. Po dobehnutí skriptu sa načíta v prehliadači výsledná webová stránka, ktorá je definovaná súborom vysledok.jsp (Obr. 4.12).

Vzhľad vstupnej a výslednej stránky určuje súbor style.css. Vo webovej stránke môžeme vidieť 4 časti. V prvej časti môzme vidieť obsah súboru scanNetFinal.txt v tabuľke, čo sú v podstate IP adresy nájdených zariadení v zadanom rozsahu. V druhej časti môžme vidieť tabuľku s obsahom súboru scanPortsFinal.txt, čo sú aktívne porty na týchto staniciach. V tretej časti môžeme vidieť odkazy na log súbory s výsledkami útokov na porty 22, 23 a 80, ktoré si môže užívateľ stiahnuť. Všetky log súbory su zobrazené a stiahnuteľné z priečinku /var/autoTester/log/. V poslednej časti je zobrazený v tabuľke výsledok zo súboru DLMSResults.txt, ktorý zobrazuje, ktoré zariadenia sú DLMS aktívne, aké sú ich nastavenia a výsledky útokov DOS



Obr. 4.12: Výstupná webová stránka nadefinovaná v súbore vysledok.jsp.

a rozpojenie breakeru. O všetko načítanie zo súborov do tabuľky sa stará súbor script.js.

5 Optimalizácia skeneru pre Raspberry Pi

Raspberry Pi [38] je jednodoskový mikropočítač s doskou plošných spojov veľkosti kreditnej karty. Bol vyvinutý nadáciou Raspberry Pi Foundantion v roku 2012. Odvtedy bola vyvinutá už 4. generácia tohto zariadenia.

Automatizovaný skener som optimalizoval pre moje vlastné zariadenia Raspberry Pi 4, 4 GHz RAM s operačným systémom Raspbian Buster Lite, ktoré je umiestnené v špeciálnej hliníkovej krabičke s lepším odvodom tepla z procesoru (Obr. 5.1). Automatizovaný skener by nemal byť problém sprevádzkovať aj na nižších verziách Raspberry Pi. Jediný problém by mohol byť v inej verzii operačného systému. Na automatizovaný skener nie je potrebný vysoký výkon.

Obr. 5.1: Raspberry Pi 4, 4 GHz RAM v špeciálnej hliníkovej krabičke.

Všetko potrebné pre inštaláciu služieb na Raspbian, nastavenie automatizovaného skeneru a sputenie skeneru som umiestnil do skomprimovaného súboru AutomatizovanySkenerDiplomovaPraca.zip. V tomto skomprimovanom súbore sa nachádza priečinok s rovnakým názvom. V priečinku AutomatizovanySkenerDiplomovaPraca je textový súbor navod.txt kde sa nachádzaja podrobný postup.

5.1 Inštalačný skript

V priečinku AutomatizovanySkenerDiplomovaPraca sa nachádza aj inštalačný skript installScript.sh. Keďže automatizovaný skener potrebuje pre svoj beh viacej služieb, tak tento skript všetky potrebné služby nainštaluje. Skript je testovaný na čistom operačnom systéme Raspbian Buster Lite. Služby, ktoré skript nainštaluje sú:

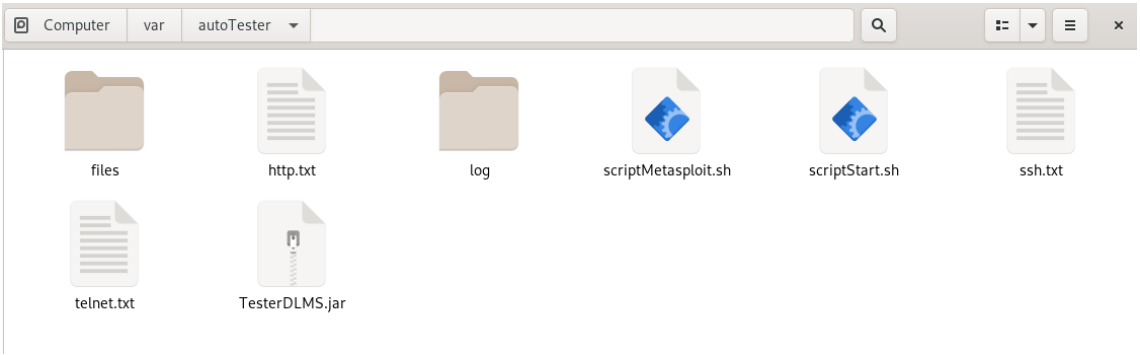
- Vim [39],
- Nmap,
- Apache Maven,
- Masscan,
- OpenJDK,
- Apache Tomcat 8,
- Metasploit Framework.

Pri spustení installScript.sh je treba mať zariadenie pripojené k internetu, kvôli stiahnutiu balíčkov na inštaláciu. Ďalej je treba spúšťať skript pod užívateľom root. Skript po nainštalovaní týchto služieb presunie súbor autotester.service do umiestnenia priečinku služby Tomcat, zastaví službu Tomcat, povolí v nej súbor autotester.service a spustí službu Tomcat s týmto súborom. Súbor autotester.service je upravené východzie nastavenie služby Tomcat. Súbor je upravený tak, že sa služba Tomcat spúšťa pod skupinou a uživateľom root. Inštalácia Metasploit Frameworku v skripte je napísaná podľa návodu na ich github stránke [40]. Po dokončení skriptu je treba v terminále operačného systému Raspbian spustiť Metasploit Framework príkazom `msfconsole` pod užívateľom pi a vytvoriť databázu. Program po prvotnom spustení sám napovedá ako vytvoriť databázu. Potom program ukončíme príkazom `exit`. Týmto sú všetky potrebné služby pre automatizovaný skener nainštalované.

5.2 Sprevádzkovanie automatizovaného skeneru

Po nainštalovaní všetkých potrebných služieb musíme priečinok autoTester z priečinku AutomatizovanySkenerDiplomovaPraca presunúť do priečinku /var (nová cesta k priečinku teda bude /var/autoTester). Priečinok sa v tejto ceste musí nachádzať preto, lebo túto cestu budeme definovať Tomcatu ako symbolický odkaz. V priečinku autoTester sa nachádza časť skript a časť TesterDLMS z automatizovaného skeneru. Ďalej sa v priečinku nachádzajú ďalšie dva priečinky files (priečinok pre výsledné textové súbory) a log (priečinok pre log súbory z Metasploit Framework) (Obr. 5.2).

Ďalej musíme premiestniť súbor webapp.war z priečinku AutomatizovanySkenerDiplomovaPraca do priečinku /var/lib/tomcat8/webapps. V tomto priečinku si Tomcat už automaticky sprevádzkuje webovú aplikáciu webapp.war. Ďalej musíme



Obr. 5.2: Obsah priečinku `autoTester`.

v terminále operačného systému Raspbian spustiť pod uživateľom root skript makeSymlink.sh, ktorý sa nachádza v priečinku AutomatizovanySkenerDiplomovaPraca. Tento skript pridá Tomcatu symbolické odkazy k priečinkom files a log. Potom už len pre istotu v príkazovom riadku reštartujeme nastavenie Tomcatu pod uživateľom root príkazom `systemctl restart autotester.service`. Po splnení týchto inštrukcií je automatizovaný skener už pripravený na použitie. Funkčnosť si môžeme overiť tak, že sa pripojíme iným zariadením na Raspberry Pi a zadáme do webového prehliadača adresu: <IP adresa Raspberry Pi>:8080/webapp. Po načítaní v prehliadači by sme mali vidieť vstupnú webovú stránku automatizovaného skeneru.

5.3 Použitie cez VPN

Súbor scriptStart.sh, teraz už v priečinku /var/autoTester), v sebe obsahuje zakomentovanú časť (Obr. 5.3).

```
#####################################
## Skenovanie zariadeni v sieti
#####################################
sudo nmap -sP -PE -oG $fileDir/scanNet.txt $1
sudo chown $thisUser:$thisGroup $fileDir/scanNet.txt
sed '1d; $d' $fileDir/scanNet.txt > $fileDir/scanNet2.txt
awk '{print $2}' $fileDir/scanNet2.txt > $fileDir/scanNetFinal.txt
#masscan $1 --ping -oG $fileDir/scanNet.txt
#sed '1d; 2d; $d' $fileDir/scanNet.txt > $fileDir/scanNet2.txt
#awk '{print $2}' $fileDir/scanNet2.txt > $fileDir/scanNetFinal.txt
#####################################
## Skenovanie aktivnych portov u zariadeni
#####################################
```

Obr. 5.3: Zakomentovaná časť v skripte scriptStart.sh.

Je to z toho dôvodu, aby sa dal automatizovaný skener použiť aj cez VPN a nie len napriamo pripojený do laboratórneho prostredia. V danej časti je zakomentovaný program Masscan, ktorý má bohužiaľ problém pracovať cez VPN a nenájde tak žiadne zariadenia v sieti. Po dlhom hľadaní som nenarazil na spôsob ako prenastaviť program Masscan tak, aby ho bolo možné použiť cez VPN. Program Masscan je teda v tejto časti nahradený programom Nmap, ktorý vie tiež vyhľadať zariadenia v sieti pomocou TCP pingu, bohužiaľ je ale v tomto oveľa pomalší ako Masscan. Preto, ak sa nachádza Raspberry Pi pripojené v laboratórnej sieti, odporúčam zakomentovať prvé 4 a odkomentovať následujúce 3 príkazy. Automatizovaný skener tak bude fungovať omnoho rýchlejšie.

5.4 Ladiaca hodnota v scriptStart.sh

V súbore scriptStart.sh sa nachádza premenná `settingsValue`. Táto premenná je ladiaca hodnota. Jej hodnota je 5, čo symbolizuje, že na odpoveď od smart metru na výzvu zaslanú časťou TesterDLMS, bude program čakať 5 sekúnd. Tento čas som odvodil testovaním na laboratórnom prostredí.

Stanice s DLMS kompatibilitou a výsledky útokov na DLMS

Stanice	Klientská adresa	Typ zabezpečenia	DOS útok	Útok na breaker
10.10.1.26	17	high	úspešný	úspešný
10.10.1.37	17	high	úspešný	úspešný
10.10.1.61	19	highSha1	úspešný	neúspešný

Obr. 5.4: Výsledky u `settingsValue` s hodnotou 3.

Stanice s DLMS kompatibilitou a výsledky útokov na DLMS

Stanice	Klientská adresa	Typ zabezpečenia	DOS útok	Útok na breaker
10.10.1.26	17	high	úspešný	úspešný
10.10.1.37	17	high	úspešný	úspešný
10.10.1.61	17	high	úspešný	úspešný

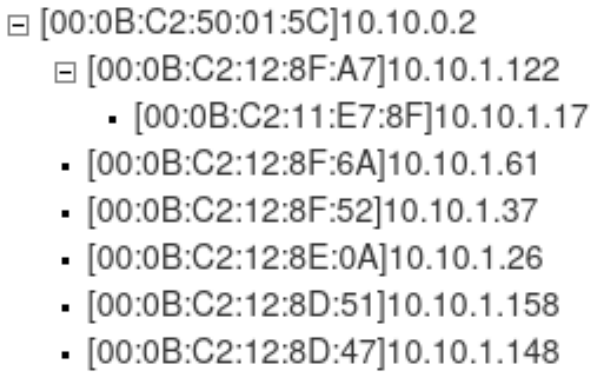
Obr. 5.5: Výsledky u `settingsValue` s hodnotou 5.

Čas odpovede sa môže aj meniť, záleží od záťaže a premávky v sieti. Čím väčší čas sa nastaví premennej, tým dlhšie bude trvať dokončenie časti TesterDLMS. Ak je premenná `settingsValue` nastavená na príliš nízku hodnotu, smart meter nestihne poslať odpoveď a TesterDLMS potom zle vyhodnotí výsledky u tohoto smart metru (Obr. 5.4 a 5.5). Keďže pri nastavení `settingsValue` na hodnotu 3 vyjde u stanice 10.10.1.61 klientská adresa 19 a mala vysť 17, tak smart meter potom odpovie aj zlým typom zabezpečenia. To spôsobí, že útok na rozpojenie breakeru je neúspešný, pretože na jeho realizáciu musí mať TesterDLMS a smart meter úspešne naviazané DLMS spojenie.

6 Test automatizovaného skeneru na laboratórnej sieti

Pre overenie správnosti automatizovaného skeneru oskenujem laboratórnu sieť, a výsledky zo skenu porovnám so znalosťami, ktoré o laboratórnej sieti mám. Pri teste mám pripojené zariadenie Raspberry Pi cez VPN do laboratórnej siete.

Po pripojení na koncentrátor v laboratórnej sieti a pripojení sa na jeho webové rozhranie (https://10.10.0.1:8080) som získal z jeho webovej aplikácie túto topológiu (Obr. 6.1). V topológii môžeme vidieť jeden koncentrátor 10.10.0.2 a sedem smart metrov 10.10.1.17-158. Tieto zariadenia by mal automatizovaný skener odhaliť. Ako môžeme vidieť topológia sa nám zmenila a je v nej o jeden smart meter viac oproti popisu laboratórneho prostredia (Obr. 3.1). Je to preto, lebo laboratórne prostredie používa viacej študentov na svoje diplomové a bakalárske práce, takže sa topológia pravidelne mení.



Obr. 6.1: Topológia siete z webovej aplikácie koncentrátora.

Po zadaní IP adresy 10.10.1.0/24 do vstupnej webovej stránky automatizovaného skeneru a stlačení tlačidla Stlačte, nám skener vyhodil tieto hodnoty (Obr. 6.2, 6.3, 6.5 a 6.6).

V prvej tabuľke na obrázku 6.2 môžeme vidieť, že nám skener detegoval všetkých 7 zariadení. (7 zariadení je v rozsahu 10.10.1.0/24 aj na obrázku 6.1).

V druhej tabuľke na obrázku 6.3 môžeme vidieť, že všetky zariadenia používajú port 80, teda HTTP protokol. Toto tvrdenie môžeme overiť zadaním konkrétnej adresy do webového prehliadača. Ako môžeme vidieť na obrázku 6.4 na smart metry 10.10.1.17 sa nachádza webová služba.

Na obrázku 6.5 môžeme vidieť log súbory, ktoré si môže užívateľ stiahnuť, a signalizujú, na ktorých zariadeniach a portoch testoval Metasploit Framework útoky. Ako môžeme vidieť Metasploit testoval útoky na port 80 u všetkých zariadení.

Nájdené stanice v rozsahu

Stanice
10.10.1.17
10.10.1.26
10.10.1.37
10.10.1.61
10.10.1.122
10.10.1.148
10.10.1.158

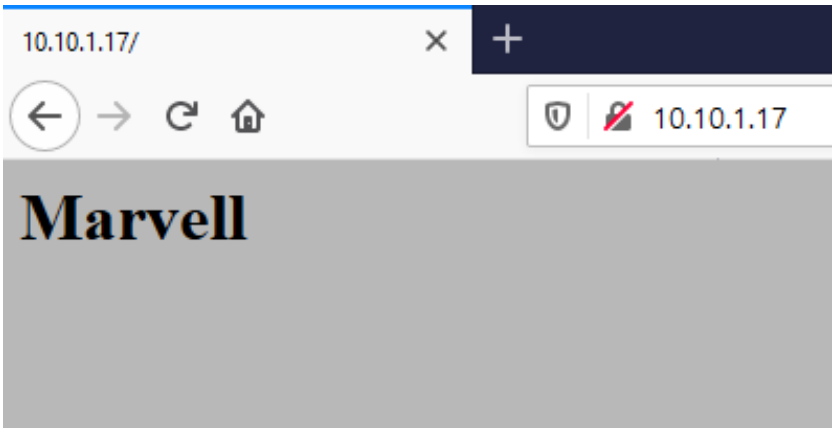
Obr. 6.2: Prvá tabuľka vo výslednej webovej stránke.

Aktívne porty u staníc

Stanice	Porty
10.10.1.17	80
10.10.1.26	80
10.10.1.37	80
10.10.1.61	80
10.10.1.122	80
10.10.1.148	80
10.10.1.158	80

Obr. 6.3: Druhá tabuľka vo výslednej webovej stránke.

V poslednej tabuľke na obrázku 6.6 môžeme vidieť, že všetky zariadenia okrem 10.10.1.17 sú smart metre s DLMS kompatibilitou. Pre dôkaz správnosti som uskutočnil test oskenovania portu 4059 (port pre DLMS/COSEM protokol) pomocou programu Nmap na zariadení 10.10.1.148 a 10.10.1.17 (Obr. 6.7). Program Nmap potvrdzuje správnosť automatizovaného skeneru.



Obr. 6.4: Webová stránka na smart metry.

Útoky na Aktívne služby

10-10-1-122-http.log
10-10-1-148-http.log
10-10-1-158-http.log
10-10-1-17-http.log
10-10-1-26-http.log
10-10-1-37-http.log
10-10-1-61-http.log

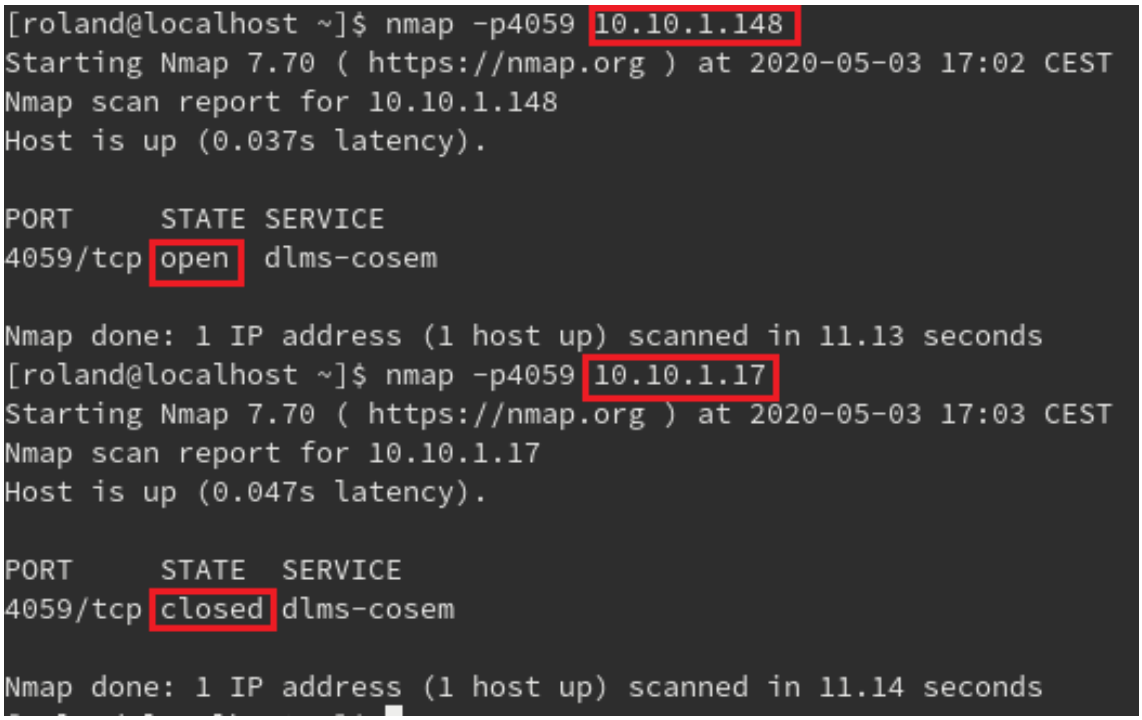
Obr. 6.5: Výsledné log súbory.

Stanice s DLMS kompatibilitou a výsledky útokov na DLMS

Stanice	Klientská adresa	Typ zabezpečenia	DOS útok	Útok na breaker
10.10.1.26	17	high	úspešný	úspešný
10.10.1.37	17	high	úspešný	úspešný
10.10.1.61	17	high	úspešný	úspešný
10.10.1.122	17	high	úspešný	úspešný
10.10.1.148	17	high	úspešný	úspešný
10.10.1.158	17	high	úspešný	úspešný

Obr. 6.6: Posledná tabuľka vo výslednej webovej stránke.

Klientská adresa u všetkých smart metrov je 17 a typ zabezpečenia je „High“. Tieto hodnoty sú správne. Všetky tieto zariadenia sú zraniteľné na útoky DOS a rozpojenie breakeru.



Obr. 6.7: Sken otvoreného portu pre DLMS/COSEM protokol pomocou programu Nmap.

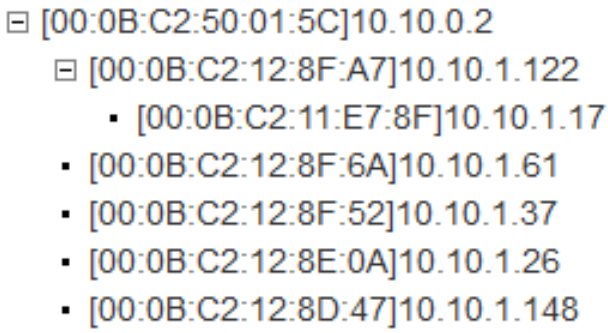
Týmto môžem konštatovať, že sa výsledky automatizovaného skeneru zhodujú s reálnym nastavením smart metrov a so znalosťami, ktoré o laboratórnej sieti mám.

7 Postrehnuté problémy so smart metrami počas vývoja automatizovaného skeneru

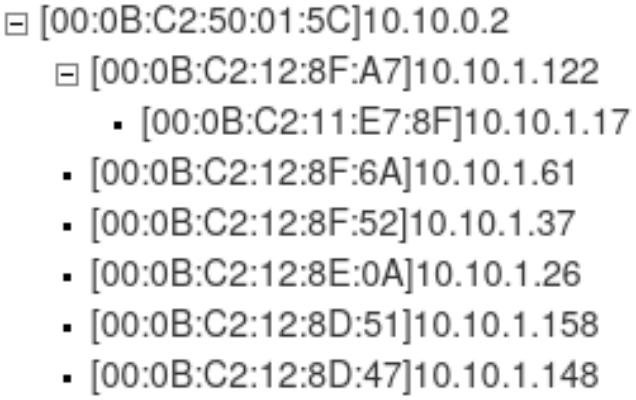
Počas vypracovávania diplomovej práce som narazil na niekoľko problémov u smart metrov. Tieto problémy sa môžu objaviť aj pri použití automatizovaného skeneru na laboratórnej sieti.

Jeden z problémov, ktorý znepríjemňoval vývoj automatizovaného skeneru bol, že smart metre v laboratórnom prostredí niekedy pri zasielaní vyššieho počtu správ prestanú komunikovať a neposielajú odpovede na dotazy. Automatizovaný skener teda vyhodnotí, že tieto zariadenia nie sú kompatibilné s DLMS/COSEM protokolom aj keď opak je pravdou.

Ďalší problém, ktorý sa pri vývoji vyskytol bolo, že smart metre úspešne odpovedajú na klientskú adresu 16, aj keď nie sú schopné komunikovať na tejto adrese. Toto je ošetrené v časti TesterDLMS.



Obr. 7.1: Topológia na začiatku vývoja automatizovaného skeneru.



Obr. 7.2: Topológia pri dokončovaní automatizovaného skeneru.

Posledný problém, na ktorý som u smart metrov v laboratórnom prostredí narazil bol výrazne sa meniaci čas odpovedí od smart metrov. Tento problém som vyriešil ladiacou hodnotou, ktorú som umiestnil do skriptu scriptStart.sh, ktorá potom vstupuje do časti TesterDLMS. Tento problém je spôsobený možno aj tým, že laboratórne prostredie je určené pre viac diplomových a bakalárskych prác a niektorá z týchto prác možno viac zaťažuje sieť svojím testovaním. Ďalšie možné vysvetlenie môže byť aj meniaca sa topológia prostredia, ktorú som postrehol počas vývoja automatizovaného skeneru (Obr. 7.1 a 7.2).

8 Porovnanie automatizovaného skeneru s už existujúcimi skenermi

Z automatizovaného skeneru budem porovnávať s už existujúcimi skenermi len časť TesterDLMS, pretože v skriptovej časti používam programy Nmap, Masscan a Metasploit Framework, ktoré sú sami o sebe skenery, a časť webová aplikácia je len grafická nadstavba nad fungujúci skener pre lepšiu užívateľskú prívetivosť automatizovaného skeneru. Program TesterDLMS by mohol byť ako samostatný skener, ktorý zistí či je zariadenie smart meter s DLMS kompatibilitou a či je zariadenie zraniteľné na útoky DOS a rozpojenie breakeru.

DLMS kompatibilitu vie zistiť mnoho skenerov napríklad už spomínaný Nmap a Masscan. Čo sa však týka zraniteľností u smart metrov používajúcich DLMS/COSEM protokol, tak na túto problematiku je problém nájsť nejaký program. Napríklad Metasploit Framework, ktorý obsahuje veľa modulov na skenovanie zraniteľností u rôznych zariadení, nemá žiadny modul pre DLMS/COSEM protokol.

Jediný open-source skener zraniteľností u DLMS/COSEM protokolu, ktorý som našiel po dlhom hľadaní, je ValiDLMS Framework [41]. ValiDLMS Framework je nadstavba nad program Wireshark [42] a slúži na bezpečnostný audit. Tento program skúma komunikáciu zariadení používajúcich DLMS/COSEM protokol. Ďalej ValiDLMS Framework posiela vlastné upravené správy na tieto zariadenia a skúma z tejto kumunikácie použité bezpečnostné sady a autentizáciu. Tento program je navrhnutý pre určitý typ smart metrov. O ktorý typ smart metru sa ale jedná nie je v práci spomenuté.

ValiDLMS Framework je lepší oproti programu TesterDLMS v tom, že skúma typ autentizácie a použité bezpečnostné sady a hovorí aký typ autentizácie a bezpečnostné sady by sa mali použiť. Na druhú stranu dokument o programe ValiDLMS Framework nepopisuje žiadne testovanie konkrétnych útokov, takže je možné, že program netestuje útok DOS a rozpojenie breakeru na danom type smart metra, ktoré program TesterDLMS testuje. Toto zasa vidím ako výhodu programu TesterDLMS.

Keďže v dokumente o ValiDLMS Framework nie je spomenutý presný typ smart metra pre ktorý je zostrojený, je možné, že by tento program nefungoval správne u smart metrov z laboratórneho prostredia, u ktorých funguje program TesterDLMS. Problém s funkčnosťou skenerov je spôsobený tým, že si výrobcovia niektoré časti DLMS/COSEM protokolu implementujú podľa seba a nie presne podľa špecifikácie. Preto väčšinou nevedia spolu komunikovať dve zariadenia od dvoch rôznych výrobcov. Preto je nemožné vyrobiť skener, ktorý by bol kompatibilný so všetkými typmi smart metrov.

Záver

Vo svojej diplomovej práci som vysvetlil SCADA systémy a popísal som siete v energetike. Konkrétnejšie som potom popísal DLMS/COSEM protokol. Ďalej som stručne popísal Modbus protokol a porovnal tento protokol s DLMS/COSEM protokolom a popísal som ich vzájomné výhody a nevýhody. Ďalej som vo svojej diplomovej práci vytvoril a popísal návrh automatizovaného skeneru, ktorý dokáže oskenovať zadaný rozsah siete, nájsť v tomto rozsahu zariadenia, zistiť aktívne porty na týchto zariadeniach, zaútočiť na aktívne porty HTTP, SSH a Telnet, dokáže zistiť či sú zariadenia smart metre kompatibilné s DLMS/COSEM protokolom a na takýchto zariadeniach dokáže otestovať, či sú zraniteľné na útoky DOS a rozpojenie breakeru. Automatizovaný skener je obsluhovaný vlastnou webovou aplikáciou, na ktorej sa zobrazujú aj výsledky skenovania. Potom som podľa tohto návrhu zostrojil automatizovaný skener, ktorý spĺňa všetky tieto funkcie, ktoré má podľa zadania diplomovej práce spĺňať. Ďalej som automatizovaný skener prispôsobil pre hardwarové zariadenie Raspberry Pi 4, 4 GHz RAM s operačným systémom Raspbian Buster Lite. Pre tento operačný systém som vytvoril návod a inštalačné skripty, ktoré automaticky na stanicu nainštalujú všetky potrebné programy a služby pre správny beh automatizovaného skeneru. Ďalej som v návode popísal ako sprevádzkovať automatizovaný skener na operačnom systéme Raspbian tak, aby správne fungoval.

Ďalej som automatizovaný skener otestoval na laboratórnom prostredí a vyhodnotil som výsledky zo skenu laboratórnej siete s poznatkami o laboratórnom prostredí, ktoré mi boli poskytnuté.

Na záver práce som popísal problémy, ktoré mi sťažovali vývoj automatizovaného skeneru a porovnal som tento skener s ďalšími dostupnými skenermi pre smart metre kompatibilné s DLMS/COSEM protokolom.

Na základe vyššie uvedeného som presvedčený, že som splnil všetky ciele diplomovej práce.

Literatúra

[1] Príchod hackerů: příběh Stuxnetu. *Root.cz* [online]. 29. 4. 2014 [cit. 2019-12-19]. Dostupné z: <https://www.root.cz/clanky/prichod-hackeru-pribeh-stuxnetu/

[2] Zákon č. 181/2014 Sb. *Zakonyprolidi.cz* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.zakonyprolidi.cz/cs/2014-181

[3] Malware vypnul elektřinu na Ukrajině. *Dvojklik.cz* [online]. 6. 1. 2016 [cit. 2019-12-19]. Dostupné z: <https://www.dvojklik.cz/malware-vypnul-elektrinu-na-ukrajine/

[4] Raspbian. *Raspberrypi.org* [online]. [cit. 2020-04-25]. Dostupné z: <https://www.raspberrypi.org/downloads/raspbian/

[5] Bc.VÁŇA, Martin. *KYBERNETICKÉ PROSTŘEDÍ PRO SYSTÉMY TYPU ICS/SCADA* [online]. BRNO, 2019 [cit. 2019-12-19]. Dostupné z: <https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/177591/DP_final.pdf ?sequence=1&isAllowed=y. DIPLOMOVÁ PRÁCA. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedúcí práce Ing. Radek Fujdiak, Ph.D.

[6] PRISTAŠ, JÁN. *GENEROVANIE PREVÁDZKY IOT SIETÍ A DETEKCIA BEZPEČNOSTNÝCH INCIDENTOV* [online]. BRNO, 2018 [cit. 2019-12-19]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=181859. BAKALÁRSKA PRÁCA. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedúcí práce Ing. PETR MATOUŠEK, Ph.D., M.A.

[7] Bc.RAK, Jan. *SCADA SYSTÉM STANIC PRO KONTROLU TĚSNOSTI* [online]. BRNO, 2016 [cit. 2019-12-19]. Dostupné z: <https://dspace.vutbr.cz/bitstream/handle/11012/65228/xrakja00.pdf?sequence=1&isAllowed=y. DIPLOMOVÁ PRÁCA. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedúcí práce Ing. Radek Štohl, Ph.D.

[8] DLMS. *Dlms.com* [online]. [cit. 2019-12-17]. Dostupné z: <https://www.dlms.com/dlms-cosem/overview

[9] KOHOUT, David. *ZÁTĚŽOVÝ GENERÁTOR ZPRÁV DLMS/COSEM* [online]. BRNO, 2019 [cit. 2019-12-19]. Dostupné z: <https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/173566/Zatezovy_generator_DLMSKohout.pdf?sequence=2&isAllowed=y. BAKALÁRSKA PRÁCA. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedúcí práce Ing. Tomáš Lieskovan.

[10] Green Book Edition 8.3. *Dlms.com* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.dlms.com/files/Green-Book-Ed-83-Excerpt.pdf

[11] Acklio enables DLMS Smart Meter communication over LoRaWAN®. *Ackl.io* [online]. [cit. 2019-12-17]. Dostupné z: <https://www.ackl.io/blog/acklio-innovation-enabling-dlms-smart-meters-communication-over-lorawan

[12] Komunikácia - DLMS/COSEM protokol. *Ipesoft.com* [online]. [cit. 2019-12-17]. Dostupné z: <https://www.ipesoft.com/sk/blog/komunikacia-dlms/cosem-protokol

[13] DLMS/COSEM. *Opensmartgridplatform.org* [online]. [cit. 2019-12-19]. Dostupné z: <http://documentation.opensmartgridplatform.org/Protocols/DLMS/index.html

[14] KMETHY, Gyozo. IEC 62056 DLMS/COSEM workshopPart 4: Main concepts. *Read.pudn.com* [online]. 2010 [cit. 2019-12-19]. Dostupné z: <http://read.pudn.com/downloads527/ebook/2184822/IEC%2062056%20DLMS%20COSEM%20workshop%20Part%204%20Main%20Concept.pdf

[15] Design Key Management System for DLMS/COSEM Standard-based Smart Metering. *Researchgate.net* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.researchgate.net/publication/328074183_Design_key_management_system_for_DLMSCOSEM_standardbased_smart_metering

[16] Blue Book Edition 12.2. *Dlms.com* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.dlms.com/files/Blue-Book-Ed-122-Excerpt.pdf

[17] Bc.KURINEC, Matej. *Implementácia vybraných objektov DLMS protokolu pre produkt Antik eMeter* [online]. Košice, 2017 [cit. 2019-12-19]. Dostupné z: <https://opac.crzp.sk/?fn=detailBiblioForm&sid=998D604438B59F55B28156FC0D22&seo=CRZP-detail-kniha. Diplomová práca. Technická univerzita v Košiciach. Vedúcí práce Ing. Peter Feciľák, PhD.

[18] Modbus. *Modbus.org* [online]. [cit. 2020-05-09]. Dostupné z: <http://www.modbus.org/

[19] Ing.RONEŠOVÁ, Andrea. *Přehled protokolu MODBUS* [online]. 2005, , 20 [cit. 2019-12-19]. Dostupné z: <http://home.zcu.cz/~ronesova/bastl/files/modbuspdf.

[20] NOŽKA, TOMÁŠ. *MODUL DIGITÁLNÍCH VSTUPŮ S ROZHRANÍM MODBUS* [online]. BRNO, 2009 [cit. 2019-12-19]. Dostupné z: <https://dspace.vutbr.cz/xmlui/bitstream/handle/11012/259/Nozka_Tomas_Modul_dig_vstupu.pdf?sequence=2&isAllowed=y. Bakalárska práca. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedúcí práce Doc. Ing. ZDENĚK BRADÁČ, Ph.D.

[21] MODBUS/TCP Security. *Modbus.org* [online]. 27s [cit. 2019-12-19]. Dostupné z: <http://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf

[22] Non-Disclosure Agreement (NDA) Template — Sample. *Nondisclosureagreement.com* [online]. [cit. 2020-05-07]. Dostupné z: <https://nondisclosureagreement.com/

[23] OpenJDK. *Openjdk.java.net* [online]. [cit. 2020-04-28]. Dostupné z: <https://openjdk.java.net/

[24] Apache Maven Project. *Maven.apache.org* [online]. [cit. 2020-04-27]. Dostupné z: <http://maven.apache.org/

[25] Gurux.DLMS. *Gurux.fi* [online]. [cit. 2020-04-27]. Dostupné z: <http://www.gurux.fi/Gurux.DLMS

[26] Gurux.Net. *Gurux.fi* [online]. [cit. 2020-04-27]. Dostupné z: <http://www.gurux.fi/index.php?q=Gurux.Net

[27] Gurux / gurux.dlms.java. *Github.com* [online]. [cit. 2020-04-27]. Dostupné z: <https://github.com/Gurux/gurux.dlms.java/tree/master/gurux.dlms.client.example.java/src/gurux/dlms/client

[28] IntelliJ IDEA. *Jetbrains.com* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.jetbrains.com/idea/

[29] DLMS Client and Server addressing. *Gurux.fi* [online]. [cit. 2020-04-28]. Dostupné z: <https://www.gurux.fi/dlmsAddress

[30] Gurux DLMS Translator. *Gurux.fi* [online]. [cit. 2020-04-30]. Dostupné z: <https://www.gurux.fi/GuruxDLMSTranslator

[31] Nmap. *Nmap.org* [online]. [cit. 2019-12-19]. Dostupné z: <https://nmap.org/

[32] GRAHAM, Robert David. MASSCAN: Mass IP port scanner. *Github.com* [online]. [cit. 2019-12-19]. Dostupné z: <https://github.com/robertdavidgraham/masscan

[33] Metasploit. *Metasploit.com* [online]. [cit. 2020-05-01]. Dostupné z: <https://www.metasploit.com/

[34] Apache Tomcat. *Tomcat.apache.org* [online]. [cit. 2019-12-19]. Dostupné z: <http://tomcat.apache.org/

[35] Servlet Tutorial. *Javatpoint.com https://www.javatpoint.com* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.javatpoint.com/servlet-tutorial

[36] Java Servlets - predstavenie technológie. *Interval.cz* [online]. [cit. 2019-12-19]. Dostupné z: <https://www.interval.cz/clanky/java-servlets-predstavenie-technologie/

[37] jQuery. *Jquery.com* [online]. [cit. 2020-05-01]. Dostupné z: <https://jquery.com/

[38] Raspberry Pi. *Raspberrypi.org* [online]. [cit. 2020-05-02]. Dostupné z: <https://www.raspberrypi.org/

[39] Vim. *Vim.org/* [online]. [cit. 2020-05-02]. Dostupné z: <https://www.vim.org/

[40] Rapid7/metasploit-framework Nightly Installers. *Github.com* [online]. [cit. 2020-05-02]. Dostupné z: <https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers

[41] MENDES, Henrique, Ibéria MEDEIROS a Nuno NEVES. *Validating and Securing DLMS/COSEM Implementations with the ValiDLMS Framework* [online]. Lisabon, Portugalsko, 2018 [cit. 2020-05-21]. Dostupné z: <http://www.di.fc.ul.pt/~imedeiros/papers/DSN2018_CERTS_Validlms.pdf. Universidade de Lisboa.

[42] Wireshark. *Wireshark.org* [online]. [cit. 2020-05-21]. Dostupné z: <https://www.wireshark.org/

Zoznam symbolov, veličín a skratiek

AA	Application Association
ADU	Aplication Data Unit
AES	Advanced Encryption Standard
AES-128	Advanced Encryption Standard s vekosťou bloku 128 bitov
ASE	Application Service Elements
Bezp.	Bezpečnostná
COSEM	Companion Specification For Energy Metering
CToS	Výzva od klienta k serveru
Dig.	Digitálny
DLMS	Device Language Message Specification
ECDH	Elliptic Curve Diffie Hellman
GCM-128	mód Galois/Counter Mode, velkosť bloku 128 bitov
GCM-256	Mód Galois/Counter Mode, velkosť bloku 256 bitov
GHz	Giga-hertz
HLS	High Level Security
ID	Identifikačné číslo
IEC	International Electrotechnical Commission
IP	Internet Protocol
IPSec	IP Security
ISO/OSI	International Organization For Standardization/Open Systems Interconnection
LD	Logic Device
LDN	Logic Device Name
LLS	Low Level Security
LN	Logic Name
L2TP	Layer 2 Tunneling Protocol
MAC	Media Access Control
MTU	Master Terminal Unit
OBIS	Object Identification System
OS	Operation System
PDU	Protokol Data Unit
P-256	Kryvky P-256
P-384	Kryvky P-384
RAM	Random-Access Memory
RTU	Remote Terminal Unit
r(StoC)	Odpoveď na StoC
r(CtoS)	Odpoveď CtoS

SCADA	Supervisory Control And Data Acquisition
SHA-256	Secure Hash Algorithm s velkosťou výstupu 256 bitov
SHA-384	Secure Hash Algorithm s velkosťou výstupu 384 bitov
SN	Short Name
StoC	Výzva od serveru ku klientovi
sym.	symetrické
Sys-t	System Title
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
VPN	Virtual Private Network

Zoznam príloh

A Obsah priloženého CD **62**

A Obsah priloženého CD

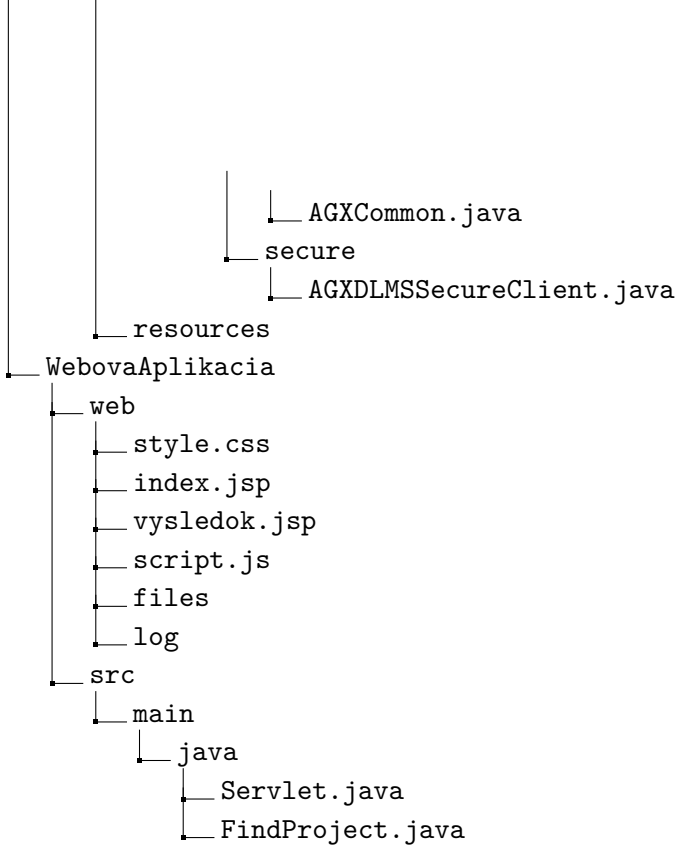
Priložené CD obsahuje hlavné súbory tejto práce. Všetky súbory, na priloženom CD, boli nahrané do informačného systému.

Súbor AutomatizovanySkenerDiplomovaPraca.zip obsahuje automatizovaný skener, inštalačné skripty a návod ako sprevádzkovať automatizovaný skener na zariadení Raspberry Pi.

Súbor Automatizovaný_Skener_Diplomová_Práca_Roland_Dávidík.pdf je elektronická verzia diplomovej práce.

Súbor ZdrojovyKod.zip obsahuje zdrojové kódy automatizovaného skeneru a to konkrétne časti TesterDLMS a webová aplikácia.

```
/ ..... koreňový adresár priloženého CD
├── AutomatizovanySkenerDiplomovaPraca.zip
│   ├── webapp.war
│   ├── installScript.sh
│   ├── makeSymlink.sh
│   ├── navod.txt
│   ├── autotester.service
│   └── autoTester
│       ├── http.txt
│       ├── TesterDLMS.jar
│       ├── telnet.txt
│       ├── scriptStart.sh
│       ├── scriptMetasploit.sh
│       ├── ssh.txt
│       ├── files
│       └── log
├── Automatizovaný_Skener_Diplomová_Práca_Roland_Dávidík.pdf
└── ZdrojovyKod.zip
    └── TesterDLMS
        └── main
            └── java
                ├── application
                │   ├── Main.java
                │   ├── ProgramFunctions.java
                │   ├── AGXDLMSReader.java
                │   ├── Application.java
                │   ├── ClientThread.java
                │   └── Client.java
                └── gurux
                    └── dlms
                        ├── AGXDLMSClient.java
                        └── internal
```


AGXCommon.java
secure
AGXDLMSSecureClient.java
resources
WebovaAplikacia
web
style.css
index.jsp
vysledok.jsp
script.js
files
log
src
main
java
Servlet.java
FindProject.java