VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Karel Kuchař

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

VHODNÁ STRATEGIE PRO DETEKCI BEZPEČNOSTNÍCH INCIDENTŮ V PRŮMYSLOVÝCH SÍTÍCH

APPROPRIATE STRATEGY FOR SECURITY INCIDENT DETECTION IN INDUSTRIAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Karel Kuchař**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Radek Fujdiak, Ph.D.**
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. Karel Kuchař ***ID:*** 185931
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Vhodná strategie pro detekci bezpečnostních incidentů v průmyslových sítích

POKYNY PRO VYPRACOVÁNÍ:

Student provede analýzu dnešních industriálních komunikačních protokolů. Detailně se pak zaměří na jeden specifický protokol (i jeho infrastrukturu) a popíše bezpečnostní hrozby jednotlivých částí vybraného protokolu. V rámci praktické části si student vytvoří vlastní experimentální síť, kde otestuje jednotlivé hrozby formou navržených scénářů (bezpečnostních incidentů) zahrnující identifikované hrozby. Z nasimulovaných bezpečnostních incidentů na základě logů a získaných informací navrhne, jak je možné detekovat jednotlivé hrozby na základě anomálií, datového přenosu, komunikace a dalších specifických metod pro industriální sítě. Z dostatečného množství dat bude následně vytvořena statistika a navržené metody detekce budou graficky prezentovány. Výsledkem tak bude nejen implementace vybraného protokolu, ověřovacích scénářů bezpečnostních incidentů, ale také samotná implementace detekčních metod a následně jejich ověření i praktická evaluace.

DOPORUČENÁ LITERATURA:

[1] DING, Derui, et al. A survey on security control and attack detection for industrial cyber-physical systems. Neurocomputing, 2018, 275: 1674-1683.

[2] ZENG, Pu; ZHOU, Peng. Intrusion Detection in SCADA System: A Survey. In: Intelligent Computing and Internet of Things. Springer, Singapore, 2018. p. 342-351.

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

Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

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

ABSTRAKT

Tato diplomová práce se zaměřuje na problematiku průmyslových sítí a poskytované bezpečnosti průmyslových protokolů. Cílem této práce je vytvořit specifické metody pro detekci bezpečnostních incidentů. Práce je nejvíce zaměřena na protokol Modbus/TCP a protokol DNP3. V teoretické části jsou průmyslové protokoly popsány, jsou definovány vektory útoku a je popsána bezpečnost jednotlivých protokolů. Praktická část práce se zaměřuje na popis a simulaci vybraných bezpečnostních incidentů. Na základě dat, získaných ze simulací, jsou detekovány hrozby pomocí navržených detekčních metod. Tyto metody využívají k rozpoznání bezpečnostního incidentu detekci anomálií v síťovém provozu pomocí vytvořených vzorců nebo strojového učení. Navržené metody jsou implementovány do IDS (Intrusion Detection System) systému Zeek. Pomocí navržených metod lze detekovat vybrané bezpečnostní incidenty v rámci cílové stanice.

KLÍČOVÁ SLOVA

Detekce anomálií, DNP3, ICS, IDS, Modbus, SCADA, Strojové učení, Zeek

ABSTRACT

This diploma thesis is focused on problematics of the industrial networks and offered security by the industrial protocols. The goal of this thesis is to create specific methods for detection of security incidents. This thesis is mainly focused on protocols Modbus/TCP and DNP3. In the theoretical part, the industrial protocols are described, there are defined vectors of attacks and is described security of each protocol. The practical part is focused on the description and simulation of security incidents. Based on the data gathered from the simulations, there are identified threats by the introduced detection methods. These methods are using for detecting the security incident an abnormality in the network traffic by created formulas or machine learning. Designed methods are implemented to IDS (Intrusion Detection System) of the system Zeek. With the designed methods, it is possible to detect selected security incidents in the destination workstation.

KEYWORDS

Anomaly detection, DNP3, ICS, IDS, Modbus, SCADA, Machine learning, Zeek

KUCHAŘ, Karel. *Vhodná strategie pro detekci bezpečnostních incidentů v průmyslových sítích*. Brno, 2020, 68 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Radek Fujdiak, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vhodná strategie pro detekci bezpečnostních incidentů v průmyslových sítích“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radkovi Fujdiakovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Infrastruktury, které používají protokoly pro monitorování a řízení procesů, stále více spolupracují s jinými systémy, které jsou dostupné prostřednictvím internetu. Poslední dobou je velmi časté, že jsou průmyslové infrastruktury vystaveny hrozbám. Navíc není možné vytvořit jeden typ zabezpečení a použít jej na více místech, zabezpečení musí být vždy provedeno specificky pro jednotlivé prostředí.

Při návrhu ICS (Industrial Control System) je zásadní porozumění aplikovaným protokolům k nalezení potencionálních slabých míst, resp. porozumět potencionálním vektorům útoku a adekvátním zabezpečením systému. K zajištění bezpečnosti je nutné již v návrhu síťové infrastruktury uvažovat o segmentaci sítě. Rozdělení do různých sekcí dle funkce a účelu napomáhá k lepšímu porozumění, odlišení toku dat a tím pádem je umožněno efektivní zabezpečení pro každou vzniklou sekci. I proto je doporučeno kritickou infrastrukturu oddělit od prostředí internetu [1].

Pod pojmem OT (Operation Technology) rozumíme systémy, které jsou používány k řízení průmyslových operací. Prostředí v OT je deterministické, má přesně stanovené chování, na rozdíl od IT (Information Technology), kde je prostředí dynamické. Oblast IT se zaměřuje zejména na data, OT se zaměřuje převážně na procesy. IT prostředí má pevně stanovené priority CIA (důvěrnost, integrita, dostupnost). OT prostředí má stanovené priority odlišně, na prvním místě stojí řízení, poté dostupnost, integrita a důvěrnost. Tyto systémy se také výrazně liší v aplikování bezpečnosti a provádění záplat, OT prostředí výrazně zaostává.

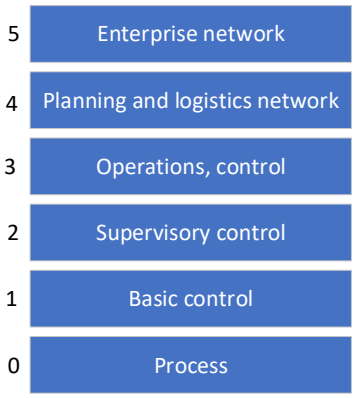
Hlavním segmentem OT systému jsou průmyslové řídicí systémy (ICS). Ty zahrnují systémy pro sledování a řízení průmyslových procesů. ICS jsou obvykle aplikace s požadavkem na vysokou dostupnost. Prostředí ICS je běžně spravováno pomocí PLC (Programmable Logic Controller), RTU (Remote Terminal Unit), nebo DPCS (Discrete Process Control Systems), ty mohou využívat PLC, nebo jiná zařízení.

Průmyslové řídicí systémy jsou často řízeny pomocí SCADA (Supervisory Control and Data Acquisition) systémů, poskytující grafické rozhraní HMI (Human Machine Interface), které zobrazuje stav systému, poplachy apod. Jednotlivé kontroléry (PLC, RTU) komunikují se senzory a aktivními prvky.

Tato práce se nejprve zaměřuje na analýzu vybraných průmyslových protokolů. U těchto protokolů je popsána bezpečnost, identifikovány hrozby a popsány jednotlivé vektory útoku. Praktická část je následně zaměřena na návrh síťového zapojení a implementaci průmyslových protokolů. Na základě bezpečnostních scénářů jsou simulovány bezpečnostní incidenty s navrženým a implementovaným opatřením.

1 Průmyslové komunikační protokoly

V rámci protokolů ICS se využívá tzv. Purdue model. Tento model rozděluje zařízení do šesti vrstev dle jejich funkce, viz obr. 1.1



Obr. 1.1: Purdue model [1].

Nultá vrstva obsahuje senzory a aktivní prvky (actuators). První vrstva obsahuje PLC a RTU. Prvky další vrstvy tvoří ovladače a HMI pro interakci se systémem. Třetí vrstvu tvoří zařízení poskytující například služby DHCP (Dynamic Host Configuration Protocol), DNS (Domain Name System), LDAP (Lightweight Directory Access Protocol), NTP (Network Time Protocol), databázové a souborové servery. Čtvrtou vrstvu tvoří systémy plánování, systémy řídící výrobu a místní IT služby (např. e-mail a tisk). Poslední vrstva je tvořena podnikovými aplikacemi (ERP, správa dokumentů) [1].

Dále bude provedeno porovnání běžně používaných průmyslových protokolů. Na tyto modely je pohlíženo skrze model TCP/IP (Transmission Control Protocol/Internet Protocol).

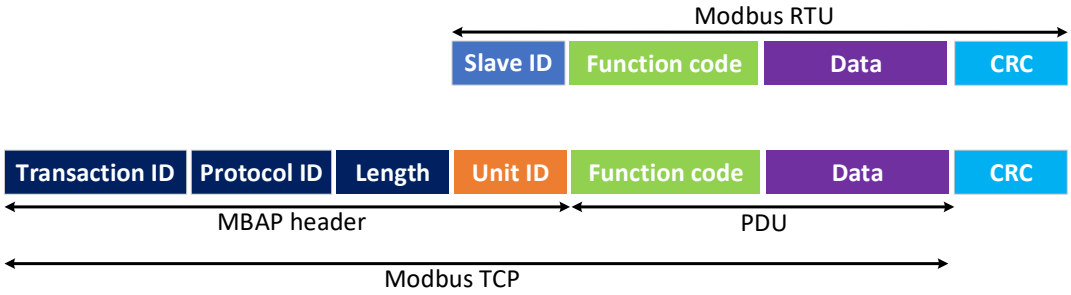
1.1 Modbus

Protokol Modbus je jeden z nejstarších průmyslových protokolů. Poprvé byl představen v roce 1979, kdy komunikoval s jednotlivými PLC sériově. K zajištění větší integrace s moderními systémy, tedy podporou TCP/IP, vznikl Modbus/TCP. Jedná se o jeden z nejmasivněji rozšířeného protokolu pro průmyslové řízení. Protokol pracuje na aplikační vrstvě. Komunikace je typu master-slave (klient-server pro ethernet), z důvodu využití TCP/IP nezáleží na použité přenosové technologii. Komunikace je typu výzva-odpověď. Protokol lze rozdělit podle implementace:

- Serial Modbus – je použita přenosová technologie HDLC (High-level Data Link Control), komunikace master-slave.
- Modbus/TCP – pro přenos dat je použit TCP/IP protokol.

Existují různé verze protokolu Modbus, například Modbus ASCII, Modbus Plus a Modbus RTU. Slave zařízení odpovídají na výzvu Master zařízení. Struktura přenášených zpráv (Modbus RTU) je složena z 1 B adresy (Slave/broadcast adresa), kódu funkce (1 B pole k definici prováděné akce), bloku dat s dalšími informacemi a kontroly chyb vzniklé přenosem dat [2], struktura přenášených zpráv viz obr. 1.2.

V případě Modbus/TCP je přidána 7B hlavička. Hlavička je složena z identifikátoru transakce (2 B), identifikátoru protokolu (2 B), délky (2 B), identifikátoru jednotky (1 B) je nahrazeno slave ID a je odebráno CRC (2 B).



Obr. 1.2: Srovnání Modbus RTU a Modbus TCP.

Kód funkce se dělí na tři typy. Public Function Codes jsou přesně definovány, je garantována jejich jedinečnost a využívají rozsahy 1–64, 73–99 a 111–127. Dalším typem jsou User-Defined Function Codes, není garantována jejich jedinečnost a obsahuje dva rozsahy 65–72, 100–110. Uživatel má možnost definovat a implementovat kód funkce, který specifikace nepodporuje. Reserved Function Codes, jsou využívány pro starší produkty, nejsou k dispozici pro veřejnost. Rozsah 128–522 je vymezen pro oznámení chyb. Některé kódy umožňují definování podkódů. Běžně využívané kódy funkce, provedená akce a popis, viz tab. 1.1.

Tab. 1.1: Tabulka základních kódu funkce.

Kód funkce	Akce	Popis
01	čtení	Discrete output coils
05	jednotlivý zápis	Discrete output coil
15	vícenásobný zápis	Discrete output coils
02	čtení	Discrete output contacts
04	čtení	Analog input contacts
03	čtení	Analog output holding registers
06	jednotlivý zápis	Analog output holding register
16	vícenásobný zápis	Analog output holding registers

Následuje datová jednotka (PDU) určená ke specifikování akce, která se má provést a přenos dat. To je využito ke čtení/zápisu z/do paměťového registru jednotlivých zařízení. Každé takové zařízení typicky obsahuje *register map*, z/do kterého probíhá čtení/zápis dat. Tento registr je rozdělen do 4 skupin. Discrete Inputs (jednobitový fyzický vstup), Coils Outputs (jednobitový fyzický výstup), Input Register a Holding Register, kde každá skupina je odlišena pomocí prefixu [2]. Kód funkce definuje, která skupina se použije, prefix registru s následnou akcí a popisem, viz tab. 1.2.

Tab. 1.2: Registr koncového zařízení.

Prefix registru	Akce	Popis
0xxx	čtení/zápis	Coils outputs
1xxx	čtení	Discrete inputs
3xxx	čtení	Input registers
4xxx	čtení/zápis	Holding Registers

Samotné datové pole tak definuje *Starting Address HI*, *Starting Address LO*, *Count of Register HI* a *Count of Register LO*. Slouží tedy k definování adres v paměťovém registru odkud začne akce čtení/zápisu a k definování počtu registrů, které budou použity ve slave zařízení a dále data, která budou zapsána, viz tab. 1.3. Informace čerpány z [3, 4].

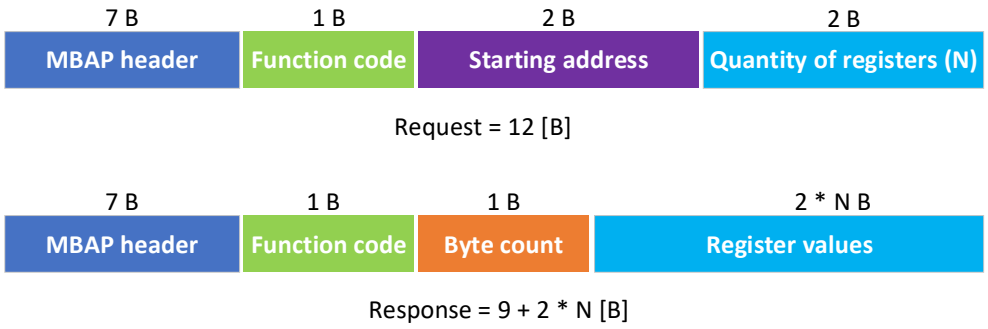
Tab. 1.3: Definice pozice v registru, Modbus RTU.

Název pole	Modbus RTU hodnota
starting address HI	00
starting address LO	00
count of register HI	00
count of register LO	02

Pokud je použit Modbus/TCP, jsou do zprávy (oproti Modbus RTU) přidána další pole (MBAP header). Transaction ID, 2 B pole k identifikaci žádosti, může být

libovolné. Při odpovědi je toto pole kopírováno. Protocol ID, 2 B pole nastavené na „00 00“, odpovídající protokolu Modbus. Length, 2 B pole identifikující počet bajtů ve zprávě. Počítá se od Unit ID do konce zprávy. Unit ID, 1 B pole k identifikaci Slave zařízení. Slave toto pole v odpovědi opět kopíruje [5].

Pokud proběhl požadavek na zápis dat, tak PLC odpovídá stejným paketem, jaký byl obdržen. Tím Master zařízení ví, že přenos proběhl v pořádku. Totožně jako v případě požadavku pro čtení, jen jsou vrácena požadovaná data. Pokud přenos neproběhl v pořádku je upraveno pole pro kód funkce. Logická hodnota nejvýznamnějšího bitu je invertována a zbytek je ponechán beze změny. Pole *data* pak nese identifikaci, o který typ chyby se jedná (rozlišováno 11 druhů chyb) [6]. Struktura ADU (Application Data Unit), Modbus/TCP, viz obr. 1.3.



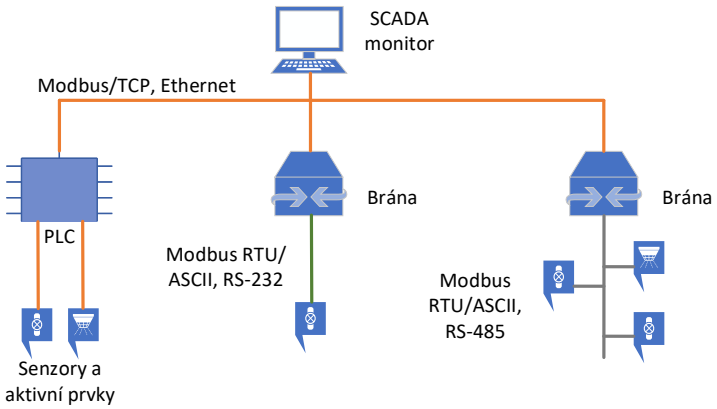
Obr. 1.3: Struktura ADU u protokolu Modbus/TCP [5].

Na obr. 1.4 je zobrazeno schéma základní sítě využívající protokol Modbus. V případě využití Modbus/TCP je třeba nadřazený prvek (SCADA monitor), dále je připojeno PLC s připojenými senzory a aktivními prvky. Komunikace může probíhat pomocí Ethernetu, nebo pomocí sériové linky. Pokud je využito Modbus/RTU nebo Modbus/ASCII, tak je využita brána, která provádí překlad z Modbus/TCP na Modbus/RTU, popřípadě Modbus/ASCII [3, 4].

1.1.1 Bezpečnost

Implementace serial Modbus používá standardy RS232 a RS485. Tyto standardy nezajišťují bezpečnost, jedná se pouze o techniky pro přímý přenos. Veškeré bezpečnostní mechanismy jsou na vyšších vrstvách.

Modbus byl vytvořen pro použití ve vysoce kontrolovaném (bezpečném) prostředí. Neprovádí se autentizace, pro vytvoření Modbus spojení je třeba pouze adresa a kód funkce. Jedná se ovšem o informace, které mohou být odposlechnuty. Také nepodporuje šifrování přenášených informací. Tyto funkcionality nebyly přidány, dříve



Obr. 1.4: Ukázková síť využívající protokol Modbus.

se tvrdilo, že bude postačovat použití IDS, nebo firewallu. Ale tohoto lze dosáhnout jen v implementaci na technologii Ethernet, ne v případě sériové implementace.

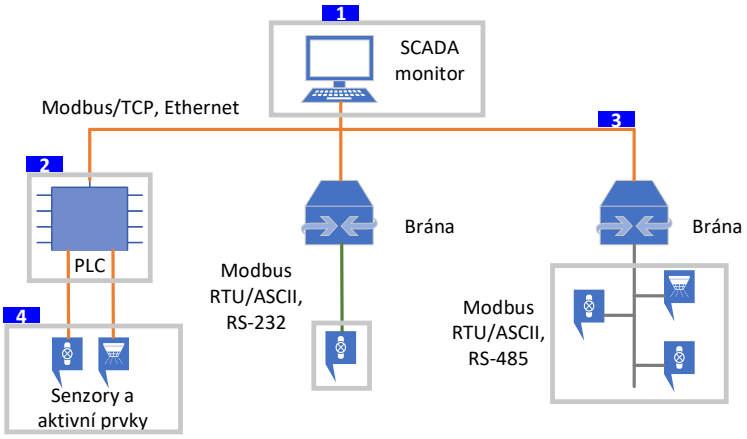
Sériová implementace Modbus šíří příkazy pomocí broadcastu, všechny zařízení tak mohou být vystaveny DoS (Denial of Service). Tyto nedostatky mohou být zesíleny tím, že Modbus protokol je také určen pro programování PLC, tím pádem útočník může zavést nežádoucí kód.

Z tohoto vyplývá, že veškerá komunikace pomocí tohoto protokolu musí být kontrolována. Je nutné kontrolovat, zda Modbus síťový provoz je povolen pouze z definovaných zařízení a pouze s povolenými funkcemi. Je vhodné také kontrolovat port TCP 502 (port Modbusu), zda jsou pakety správně formátovány.

Mezi další opatření by měla být zařazena kontrola funkcí, které nutí podřízené jednotky přejít do režimu „listen-only“, funkce sloužící k reinicializaci komunikace, funkce pro vymazání nebo podání diagnostických informací (např. informace z čítačů). Je doporučeno používat Snort, či podobné IDS/IPS (Intrusion Detection System/Intrusion Prevention System) systémy, které jsou upraveny přímo pro Modbus.

1.1.2 Vektory útoku

Obr. 1.5 zobrazuje nejčastější vektory útoku na infrastrukturu využívající protokol Modbus. Nejčastěji je útok veden na Master zařízení (SCADA monitor) (1), popřípadě na PLC (2). Útočník je schopen dále cílit útok na jednotlivé senzory a aktivní prvky (4), provádět monitoring sítě, do které může vniknout například pomocí existujícího přenosového média (3). Také lze využít existujících chyb v hardwarovém vybavení. Popřípadě může být útok zaměřen na další prvky sítě, např. servery.



Obr. 1.5: Vektory útoku u protokolu Modbus.

1.1.3 Zranitelnost

Implementace protokolu Modbus obsahuje zranitelnosti, mezi které lze zahrnout například:

Překročení maximální přípustné délky paketu [7]: Protokol má nastavenou velikost PDU (Protocol Data Unit) na maximální velikost 253 B, aby mohl probíhat přenos přes sériovou linku. Modbus TCP připojí 7 B hlavičku protokolu (MBAP), ta je zapouzdřena do TCP paketu, čímž je dosažena maximální přípustná velikost paketu. Útočník vytvoří paket přesahující 260 B, který odešle klientovi a serveru. V případě, že není zkontrolována velikost příchozího paketu, dochází k přetečení bufferu vyrovnávací paměti, popřípadě DoS útoku.

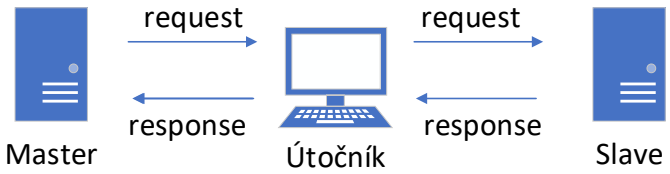
Authentication bypass by capture replay CWE-294 [8–10]: Zranitelnost je způsobena tím, že síťový přenos probíhá v otevřené podobě, tím může docházet k obcházení autentizace a získání schopnosti k útoku opakováním. Lze přehrávat příkazy „run“, „stop“, „upload“ a „download“. Zranitelnost je popsána v CVE-2017-6034.

Man In The Middle [11–13]: Jedná se o formu aktivního odposlechu, kdy útočník předává komunikaci mezi obětmi, mezi kterými je prováděn odposlech. Celá komunikace je řízena útočníkem a oběti netuší, že komunikace je odposlouchávána. Útočník (označován jako sniffer) je schopen zachytit komunikaci mezi Master a Slave, po zachycení lze provádět libovolné změny. Schéma útoku s pozicí útočníka je znázorněno na obr. 1.6.

V práci [13], byl proveden útok MITM (Man In The Middle). Síť je složena ze tří prvků. Master zařízení, PLC a útočníka. Master je tvořen počítačem s OS Windows 7 využívající Modbus Master 0.4.8 pro komunikaci se slave zařízením. PLC je představováno zařízením s OS Debian se SW „ModbusPal 1.6“. Útočník je předsta-

vován Kali Linuxem vyskytujícím se v síti s cílem zachytit příkaz odeslaný z Master zařízení, pozměnit a odeslat na PLC. Útočník využil nástroje Ettercap, který umožňuje aktivace sniffing módu. Poté na základě skenu sítě lze identifikovat jednotlivá zařízení v síti s přiřazenými MAC adresami. Následuje ARP (Address Resolution Protocol) poisoning, pomocí něhož je přesměrován provoz z Master zařízení na útočníka a poté na PLC.

Pomocí filtru na straně útočníka tak lze měnit libovolné údaje v paketu. V tomto konkrétním případě dochází k vyfiltrování paketů obsahující adresu PLC zařízení a pozměnění hodnoty (coil), která značí zapnutí/vypnutí. Tento filtr je nahrán do nástroje Ettercap a za jeho pomoci je pozměňován provoz jdoucí z Master zařízení na PLC. Útok využívá nedokonalosti tohoto protokolu v absenci autentizace, popřípadě využívání MACsec.



Obr. 1.6: Znázornění Man in the middle útoku na protokol Modbus

Útok na serial Modbus, Modbus/TCP: V případě, že je získán přístup do sítě a následně je provedeno sledování provozu, je umožněno generovat Modbus paket.

Tab. 1.4 zobrazuje, jak lze narušit bezpečnost systému, pokud je používán serial Modbus. Tab. 1.5 zobrazuje, jak lze narušit bezpečnost systému, pokud je používán Modbus/TCP. K útoku postačuje zařízení umístěné v síti, které je schopno provádět sledování provozu a vytvářet Modbus pakety. Ke generování TCP SYN flood, TCP RST flood lze využít nástroj „hping3“.

Tab. 1.4: Některé útoky na serial Modbus.

Název útoku	Kód funkce	Sub-Kód funkce	Dopad
Diagnostic register reset	08	0A	Smazání všech čítačů a diagnostického registru
Remote restart	08	01	Restart adresního pole, spuštění testu napájení
Slave reconnaissance	17	-	Získání obsahu adresního pole, útok na důvěrnost

Tab. 1.5: Některé útoky na Modbus/TCP.

Název útoku	Provedení
TCP RST flood	Generování TCP paketu s příznakem RST za účelem resetování spojení a přerušení komunikace
TCP FIN flood	Generování TCP paketu s příznakem RST za účelem uzavření spojení a přerušení komunikace
Irregular TCP framing	Vytvoření zprávy s nesprávným formátováním za účelem uzavření spojení

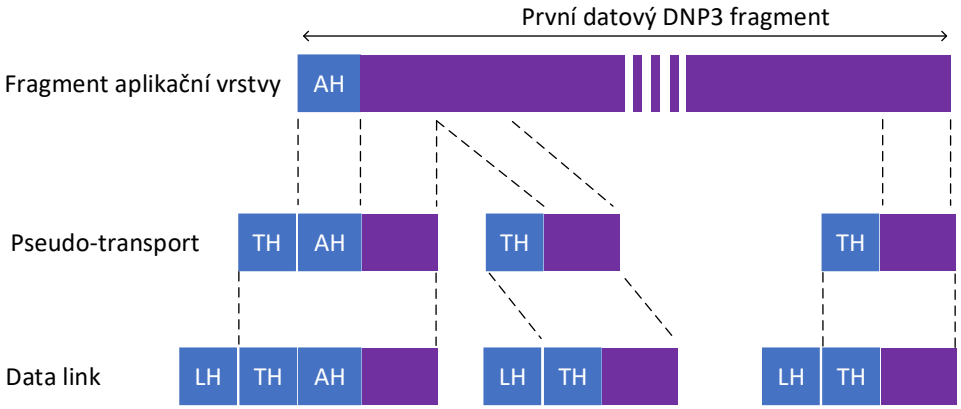
1.2 DNP3

Jedná se o protokol, který je široce rozšířen v prostředí energetického sektoru (převážně USA, Kanada). DNP3 je třívrstvý protokol pracující na linkové, transportní a aplikační vrstvě. Používá se pro přenos naměřených dat od klienta (outstation) k nadřazené stanici (master/server). Pracuje na principu výzva-odpověď [1].

Protokol je vytvořen pro maximální systémovou dostupnost. Na zajištění důvěrnosti a integrity dat není kladen velký důraz. Na linkové vrstvě je typicky aplikováno CRC (Cyclic Redundancy Check) k detekci chyb vzniklých při přenosu. Nejedná se však o prvek zajišťující bezpečnost, pouze slouží k zajištění bezchybového přenosu. Na aplikační vrstvě byly snahy o zajištění bezpečnostního standardu [1]. Tento standard řeší problémy:

- Krádež identity.
- Modifikace zprávy, za účelem pozměnění funkce systému.
- Re-injection provozu nežádoucím, nebo podvodným obsahem.
- Odposlech linky.

DNP3 zprávy lze rozdělit do tří vrstev [14]. Application Layer, Pseudo-Transport Layer, Data Link Layer. Kde každá vrstva má své záhlaví, AH (Application Header), TH (Transport Header) a LH (data Link Header). Pokud bude datový tok odeslán přes LAN/WAN, bude zpráva sestavena ze tří vrstev (AH, TH, LH) a následně zapouzdřena do TCP pomocí transportní vrstvy, viz obr. 1.7.

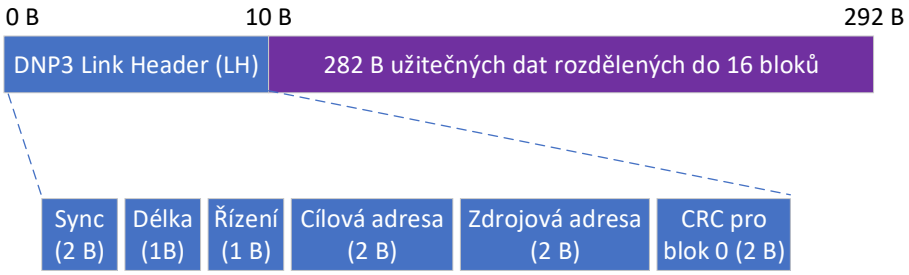


Obr. 1.7: Vrstvy protokolu DNP3 [14].

Na aplikační vrstvě je datová jednotka APDU (Application Protocol Data Unit), tvořena ASDU (Application Service Data Unit) a APCI (Application Protocol Control Info), což představuje AH. Následně Pseudo-Transport Layer rozdělí fragmenty z vyšší vrstvy na segmenty označované TPDU (Transport Protocol Data Unit) a přidá k nim jednorázově hlavičku TH o velikosti 8 b. Maximální velikost TPDU

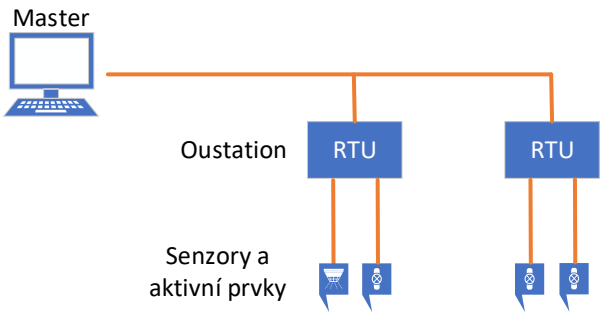
segmentu je 250 B. Poté Data Link Layer převezme TPDU a ke každému z nich přidá hlavičku LH a CRC pro detekci a opravu chyb. Takto vzniklé jednotky jsou nazývány LPDU (Link Protocol Data Unit), popřípadě DNP3 pakety. Velikost takových paketů je omezena na 292 B. Hlavička v Data Link Layer (LH) je tvořena 10B blokem záhlaví označeným jako „blok 0“, následuje 282 B datovou částí. Tato část je rozdělena do 16 bloků. Vzniká tak blok 1 až blok 16. Kde každý blok obsahuje 2 B pro CRC. Pro CRC je tak celkem rezervováno 32 B [14].

LH je rozděleno na 2 B sloužící pro synchronizaci příjemce a odesilatele, následně 1 B pro určení délky následujících položek kromě CRC, 1 B pro řízení. Dále dvě pole po 2 B pro určení adresy příjemce a odesilatele. Poté 2 B CRC pro blok 0, struktura rozdělení link header viz obr. 1.8.



Obr. 1.8: Rozdělení link header u protokolu DNP3 [14].

Na obr. 1.9 je znázorněno zapojení sítě tzv. multi-drop. DNP3 Master je typicky specifické zařízení pro protokol DNP3, může být také připojeno HMI. Dále jsou v síti outstation stanice (RTU), které připojují jednotlivé senzory a aktivní prvky.

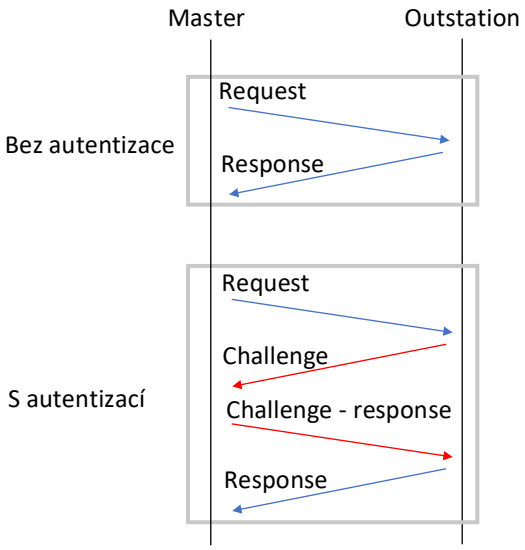


Obr. 1.9: Ukázková síť využívající protokol DNP3.

1.2.1 Bezpečnost

Je doporučeno používat pouze Secure DNP3, zároveň je vhodné tento protokol kombinovat například s protokolem TLS. Nevýhodou Secure DNP3 autentizace (ver. 5) je, že implementace autentizace neslouží k zajištění důvěrnosti (data nejsou při přenosu šifrována), ale k identifikaci stanic. Cílem je tak předejít modifikace transakce a útoku opakováním. Samotná autentizace tak nepřináší dostatečné zabezpečení (DNP3-SA, ver. 5) přenášených zpráv [15]. Dále je nutné monitorovat DNP3 port (TCP/UDP 20000) jestli neprobíhá jiná komunikace nesouvisející s protokolem.

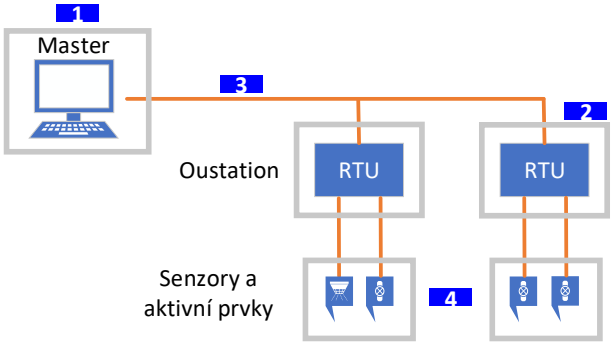
Standard umožňuje stanovit, zda je nutné před započetím komunikace provést autentizaci. Vynucení autentizace může způsobovat přetížení sítě, popřípadě způsobovat zpoždění. Při použití „agresivního“ módu může být prvotní požadavek a Authentication Response odeslán v jedné zprávě, viz obr. 1.10 [14].



Obr. 1.10: Autentizace u protokolu DNP3.

1.2.2 Vektory útoku

Obr. 1.11 zobrazuje vektory útoku zaměřené na protokol DNP3. Útok je typicky zaměřen na Master stanici (1), nebo Outstation (2). Útočník může vstoupit do komunikace pomocí útoku na přenosové médium (3), nebo na senzory a aktivní prvky (4). Útok může využívat i zranitelnosti dalších připojených zařízení, například serveru.



Obr. 1.11: Vektory útoku u protokolu DNP3.

1.2.3 Zranitelnost

V protokolu DNP3 byly objeveny zranitelnosti mezi které patří:

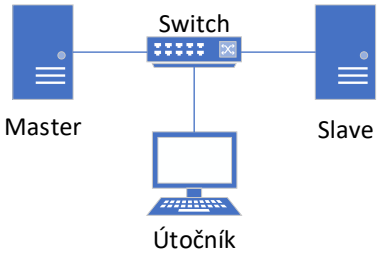
Remote denial of service vulnerability [16, 17]: Tato zranitelnost je způsobena nesprávným zpracováním určitých datových paketů. Úpravou formátování paketů (úmyslného zavedení chyb) je útočník schopen při zpracování vytvořeného paketu vyvolat selhání procesu, vedoucí ke stavu DoS. Tato zranitelnost nese označení CVE-2014-5429.

Improper input validation – serial-based [18]: Tato zranitelnost je založena na nesprávném ověření vstupu. Útočník s přístupem k fyzickému zařízení vytvoří specifický vstup, který odešle na Master zařízení přes sériovou linku. Tento vstup způsobí, že zařízení začne pracovat v nekonečné smyčce a přestane reagovat, což vyvolává DoS stav. Zranitelnost je označována CVE-2013-2798.

DNP3 Data Set Manipulation Attack [14]: Jedná se o útok zaměřený na pozměnění obsahu paketu. Tento útok je zaměřen na nezabezpečenou verzi protokolu DNP3 přenášeného pomocí TCP/IP, které nevyužívá žádného šifrování na transportní vrstvě. Útočník je tak schopen pozměnit obsah přenášených zpráv, popřípadě obsah kompletně měnit i s úpravou TCP/IP hlaviček. To umožňuje manipulovat, řídit, nebo přesměrovat probíhající síťový provoz mezi Master a Outstation.

Tento typ útoku se často používá k shromažďování informací o Master zařízení. Toho je docíleno pozměněním zprávy od Master zařízení, kde je pozměněna cílová adresa na vlastní DNP3 Outstation. V práci [14] byla provedena simulace ve virtuálním prostředí sestávající se z Master zařízení, Outstation (Slave) a útočníka. Veškeré stroje používají operační systém Ubuntu, pro komunikaci pomocí protokolu DNP3 byla využita aplikace „OpenDNP3“. Útočník využívá nástrojů TCPdump a Ettercap. Virtuální zapojení prvků, viz obr. 1.12.

Na začátku útoku je proveden ARP poisoning za pomoci nástroje Ettercap. Ná-



Obr. 1.12: Zapojení stanic při útoku Data Set Manipulation, DNP3.

sledně je pomocí skriptu v programovacím jazyce Python provedena filtrace provozu, viz výpis 1.1. Pokud zdrojová IP adresa odpovídá Master IP adrese a je detekováno DNP3 záhlaví je provedena změna paketu (v jiném případě je paket ponechán beze změny). Následně je paket odeslán zpět do sítě a směřuje k Outstation. Pozměněním tohoto útoku je provedena změna cílové adresy na adresu vlastní DNP3 Outstation určený ke shromažďování informací.

Výpis 1.1: Část kódu filtrující DNP3 provoz.

```
1 if (ip.src=='192.168.1.10'):
2   if (search(DATA.data, "\x05\x64\")):
3     print "DNP3 header seen.\n"
4     DATA.data="\x05\x64\x05\xc0\x64\x00\x00\x01"
5     print "Master packet replaced.\n"
```

Tento útok je proveditelný z důvodu absence zabezpečení na transportní vrstvě. Ochrana proti tomuto útoku může být testování, zda útočník do sítě neodeslal testovací paket, aby otestoval bezpečnost.

DNP3 unsolicited messages attack: Unsolicited zprávy slouží ke kontaktování Master stanice bez toho, aby Outstation byl vyzván. Například, když je nutné provést změnu nastavení hodnot. Tento útok využívá zprávy „disable unsolicited message“, který je zaslán z master stanice na Outstation. Tato zpráva využívá kód funkce 21, který způsobí, že Outstation přestane takové zprávy zasílat. V práci [14] byl vytvořen skript, který detekuje příkazový paket Operate s funkčním kódem 04. Pokud se na síti takový paket objeví, skript zkopíruje informace v paketu včetně pořadového čísla ACK (Acknowledgement) a dalších TCP informací k vytvoření disable unsolicited zprávy. Skript poté skenuje pakety a porovnává sekvenční čísla paketů s pořadovým číslem ACK, pokud se shodují, tak je tento paket použit. Skript aktualizuje pořadové číslo, pole délky IP a přepočítává DNP3 CRC. Nejprve je přerušena zpráva jdoucí od Master stanice a je nahrazena upraveným paketem. Outstation tak přestane informovat Master stanici o kritických situacích. Jednou z metod, jak mitigovat tento útok je měření průměrného RTT Delay (Round-Trip Time Delay) a pokud je překročena určitá hranice, je detekováno možné napadení.

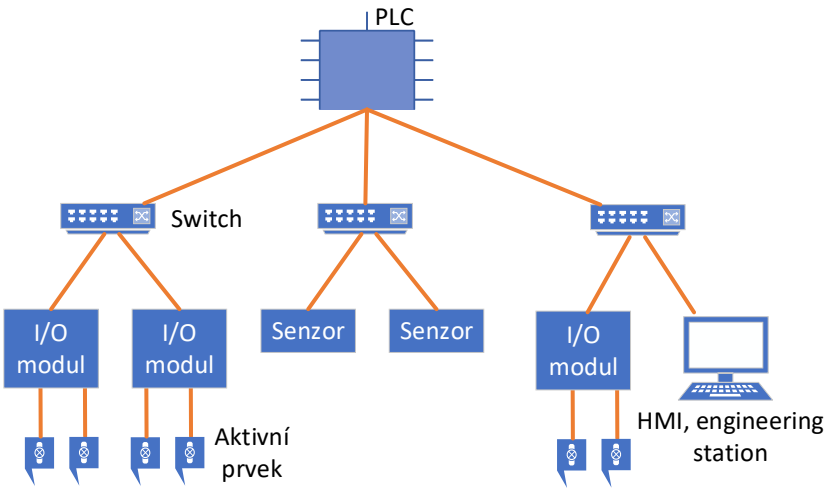
1.3 Common Industrial Protocol (CIP)

Protokol CIP je určen pro automatizaci průmyslových procesů. Obsahuje množinu služeb a zpráv pro řízení, zajištění bezpečnosti, konfiguraci, poskytnutí informací a synchronizaci. Vyskytuje se v řadě adaptací protokolu k zajištění interkomunikace a integrace pro odlišné druhy počítačových sítí, čerpáno z [19]. Mezi adaptace patří:

- EtherNet/IP – adaptace CIP do TCP/IP.
- ControlNet – adaptace pro technologie vícenásobného přístupu (CTDMA).
- DeviceNet – adaptace CIP s CAN (Controller Area Network).
- CompoNet – adaptace pro vícenásobný přístup (TDMA).

Obr. 1.13 znázorňuje základní síťové zapojení využívající protokol EtherNet/IP. V tomto případě disponuje PLC třemi síťovými kartami. Jednotlivé I/O moduly umožňují připojení velkého počtu zařízení, vstup je typicky binárního charakteru. Dále mohou být zapojeny senzory, nebo zobrazovací zařízení. V neposlední řadě je v síti engineering station.

Síť by dále mohla obsahovat další prvky jako například server, kde se budou ukládat jednotlivé hodnoty, popřípadě připojení k vyšším vrstvám (managementu), nebo pomocí bran připojit jiný průmyslový protokol.



Obr. 1.13: Ukázková síť využívající protokol EtherNet/IP.

1.3.1 Bezpečnost – implementace EtherNet/IP

EtherNet/IP je náchylný na zranitelnosti tak jako Ethernet. Může se například jednat o krádež identity nebo zachycení komunikace. V případě využívání UDP (u implicitních zpráv) je možné zavedení nežádoucího síťového provozu.

Protože EtherNet/IP je protokol založen na Ethernetu, který využívá protokoly UDP a IGMP, je nezbytné vhodně definovat perimetr a v rámci něj zajistit bezpečnostní mechanismy založené na Ethernetu a IP. Je také vhodné zajistit pasivní monitoring sítě k dohledu, že je síťový provoz asociován jen s definovaným zařízením a nepochází z prostředí internetu.

I přes využívání definování modelů pro objekty, není definován žádný mechanismus pro přímé zajištění bezpečnosti. Z důvodu definování povinných objektů pro definování zařízení, je snazší pro útočníka nalézt zařízení v síti, na které bude útok zaměřen. Používají se také společné aplikační objekty pro výměnu informací mezi zařízeními. Útočník je tak schopen manipulovat s velkým rozsahem průmyslových zařízení pomocí úpravy a odeslání tohoto typu objektu. Charakteristiky některých zpráv (komunikace v reálném čase, multicast) nejsou kompatibilní se zajištěním šifrování komunikace [1, 19].

1.4 Profibus

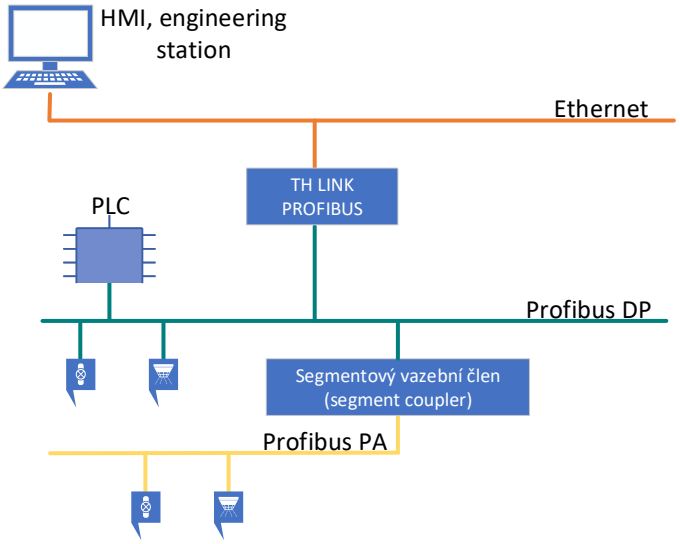
Jedná se o standard pro komunikaci skrze Fieldbus. Jako přenosové médium se používá sériová linka RS-485, popřípadě optické vlákno. Existují dvě varianty:

- Profibus DP – pro decentralizované periferie; používaný k ovládání senzorů a aktivních prvků prostřednictvím centrálního ovladače.
- Profibus PA – pro automatizaci procesů; pro monitorování měřicích zařízení prostřednictvím systému řízení procesů.

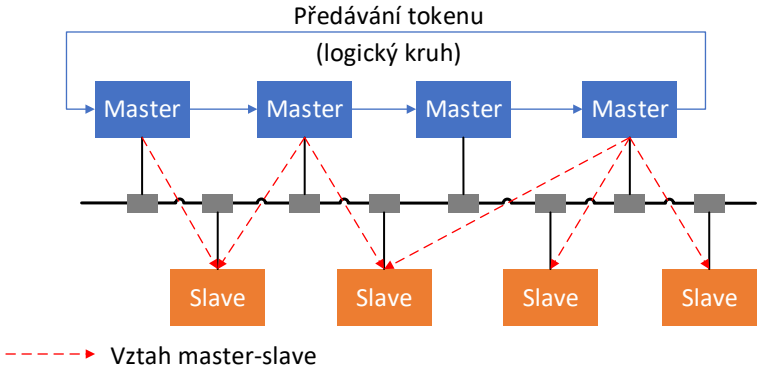
Na obr. 1.14 je znázorněno typické zapojení pro využití protokolu Profibus DP a Profibus PA. K propojení Ethernetu a Profibusu je využito zařízení TH link Profibus pro převod mezi standardy.

1.4.1 Bezpečnost

Pro řízení přístupu na linkovou vrstvu je použit FDL (Fieldbus Data Link), který používá hybridní metodu přístupu, možné schéma viz obr. 1.15, kombinující Master-Slave komunikaci a předávání tokenu. Token definuje, jaká stanice může obsadit sběrnici. Dále zajišťuje, že nedojde ke stavu, kdy komunikují dvě stanice současně. Tímto ovšem není zajištěna bezpečnost, může dojít k Traffic Injection, popřípadě DoS [1].



Obr. 1.14: Ukázková síť využívající protokol Profibus.



Obr. 1.15: Hybridní metoda přístupu, Profibus.

Na aplikační vrstvě jsou definovány tři vrstvy přístupu:

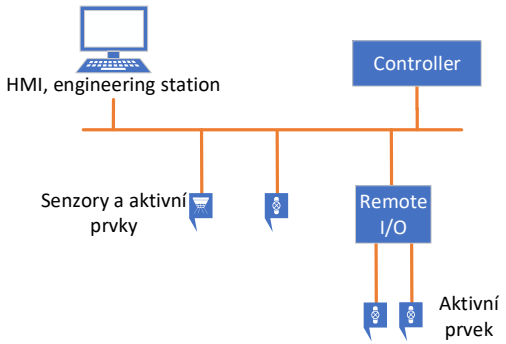
- DP-V0 – výměna periodických dat.
- DP-V1 – výměna komunikace bez pevné periodicity.
- DP-V2 – asynchronní komunikace prostřednictvím vysílání zpráv.

Dokumentace protokolu neuvádí na této vrstvě žádné přídavné zabezpečení, je nutné využít jiné mechanismy [20]. Pouze lze pro počáteční fázi přiřazení zařízení využít TCP jako transportní protokol. Pokud tím nebude dotčena činnost systému, lze použít přídavné prvky zabezpečení. Z důvodu absence autentizace a dostatečného zabezpečení protokolu je nutné, aby byl síťový provoz izolován od zbytku sítě.

1.5 Profinet

Profinet je standard založený na Profibusu, který používá pro komunikaci fyzické rozhraní Ethernet. Má opakovací systém založený na předávání tokenů. Pro přenos dat nabízí stejné funkcionality jako TCP/IP, tím umožňuje využívat bezdrátové aplikace a vysokorychlostní přenos dat. Zařízení podporující tento protokol jsou orientována na spolehlivost a komunikaci v reálném čase [1].

Na obr. 1.16 je zobrazeno možné zapojení sítě využívající protokol Profinet. Oblast je řízena controllerem, což může být PLC.



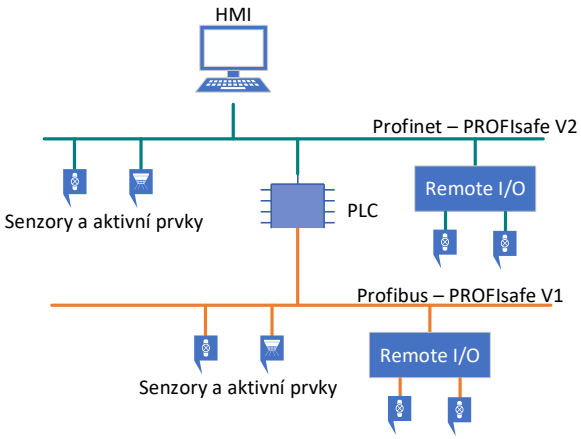
Obr. 1.16: Ukázková síť využívající protokol Profinet.

1.5.1 Bezpečnost

Absence autentizace vyžaduje, aby veškerý provoz byl izolován od zbytku sítě. Dále je doporučeno provádět autentizaci veškerých zařízení v síti a využívání šifrované komunikace. Je doporučeno nastavit zabezpečení perimetru na nejstriktnější možné, za účelem zabránění neautorizovaného přístupu do sítě nebo podezřelého provozu.

1.6 PROFIsafe

PROFIsafe je bezpečnostní protokol popsán v normě IEC 61784-3. PROFIsafe lze označit za funkční bezpečnostní komunikační profil [21]. Existují dvě verze tohoto protokolu. První verze (V1) byla zaměřena pro provoz s protokolem Profibus. Druhá verze (V2) je rozšířená o funkcionality Profinet. Na obr. 1.17 je zobrazeno typické zapojení při využití Profisafe. Z důvodu, že se jedná o protokol pracující nad komunikačním protokolem Profibus/Profinet, tak je zapojení totožné v porovnání s těmito protokoly.



Obr. 1.17: Ukázková síť využívající protokol PROFIsafe.

1.6.1 Bezpečnost

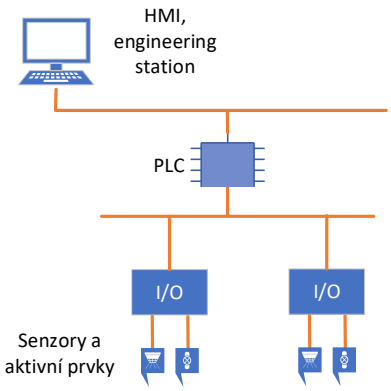
K přenosu datových jednotek jsou využívány kontejnery. Jeden kontejner je složen z dat o maximální délce 12/123 B, kontrolního bajtu a CRC2 o délce 3 B odpovídajících 12B datové části, resp. o délce 4 B odpovídajících 123B datové části. Maximální délka závisí na vybraném operačním módu. Jeden Profinet rámec může obsahovat i více PROFIsafe kontejnerů. Cílem tohoto protokolu je především detekovat chyby v komunikaci. Je rozlišováno několik chyb (Corruption, Unintended repetition, Unacceptable delay, Incorrect sequence, Loss, Insertion, Masquerade, Addressing). CRC2 je generováno na základě CRC1, dat, statusu/kontrolního bajtu a pořadového čísla (Corresponding Consecutive Number). Pořadové číslo je využíváno jako měřítko pro řešení některých druhů chyb, je například používáno pro sledování zpoždění mezi přenosem a příjmem dat.

1.7 Powerlink Ethernet

Powerlink over Ethernet je komunikační protokol vytvořen pro komunikaci v reálném čase. Jedná se o rozšíření Ethernetu v souladu s IEEE 802.3, mechanismy pro přenos dat s přesnou synchronizací a předvídatelnými intervaly [1]. Powerlink Ethernet obsahuje mechanismy, které slouží k zajištění:

- Přenosu informací s kritickou oblastí času v asynchronních cyklech.
- Synchronizace uzlů v síti s velkou přesností.
- Přenos informací s dobou, která není na vyžádání tak kritická. Asynchronní komunikace může využívat sady TCP/IP, nebo z protokolů vyšších vrstev, jako HTTP, FTP, atd.

Přenos dat v síti probíhá pomocí vyhrazených časových intervalů pro synchronní a asynchronní komunikaci. K přenosovému médiu má přístup vždy jedno zařízení v síti. Pomocí přidělování časových intervalů je zajištěno, že synchronní a asynchronní komunikace do sebe vzájemně nezasahují. Tento mechanismus je znám jako SCNM (Slot Communication Network Management), ten je řízen pomocí řídícího uzlu MN (Management Node). MN se může v síti vyskytovat pouze jednou. Ostatní uzly v síti jsou označovány jako CN (Controlled Node) a jsou řízeny MN. CN využívají pouze ty časové intervaly, které jim byly přiděleny MN. MN používá pro šíření zpráv broadcastového vysílání [22, 23]. Na obr. 1.18 je zobrazena jedna z možných variant zapojení sítě Powerlink Ethernet. MN představuje PLC (controller) a CN jednotlivé senzory a aktivní prvky.

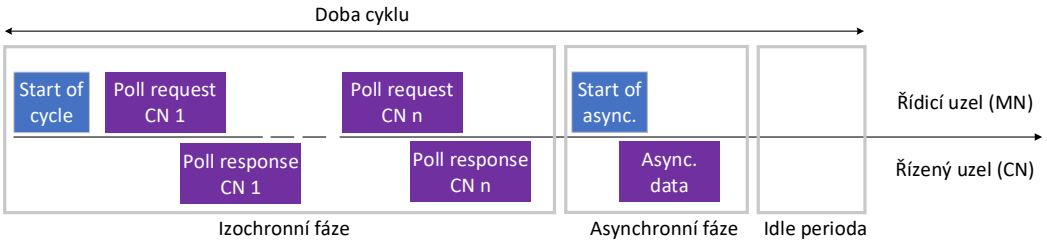


Obr. 1.18: Ukázková síť využívající protokol Powerlink Ethernet.

1.7.1 Bezpečnost

Mechanismy pro kontrolu autenticity zpráv a jednotlivých uzlů naprosto chybí. Každý prvek sice odpovídá ve stanoveném intervalu, který byl přidělen MN, ale není možnost, jak ověřit, že stanice, která odpověděla je skutečně ta, za kterou se označuje. Je zde možnost šířit nelegitimní provoz do sítě, popřípadě vyvolat DoS. Používání broadcastového šíření zpráv od CN umožňuje útočníkovi zachytit veškerou komunikaci, kterou zaslal [1, 22, 23].

SCNM, viz obr. 1.19, je náchylný na zpoždění, je tedy třeba jej oddělit od ostatních sítí, které využívají Ethernet. Je doporučeno nastavit bezpečnostní opatření perimetru tak, aby byla síť izolována a předcházelo se nežádoucímu provozu.



Obr. 1.19: Postup komunikace, Powerlink Ethernet.

1.8 OLE for Process Control (OPC)

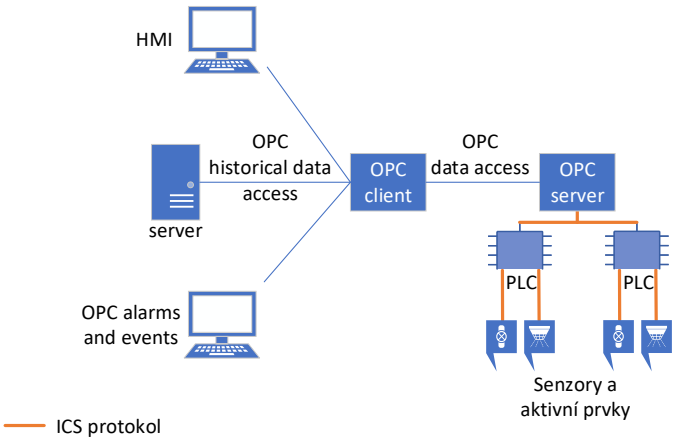
OPC (OLE for Process Control), nejedná se o průmyslový protokol, ale o operační rámec pro komunikaci v systémech pro řízení procesů založených na Windows, které používají propojování a vkládání objektů (OLE). OLE jsou využívány komunikačními protokoly jako RPC (Remote Procedure Call). Jde tedy o sadu protokolů umožňující komunikaci systémů řízení procesů. Jednotlivé systémy založené na operačním systému Windows jsou spojeny prostřednictvím sady TCP/IP. OPC byl původně založen na DCOM (Distributed Component Object Model), který je stále využíván i přes existenci aktualizované verze OPC-UA (OPC Unified Architecture), která umožňuje použití SOAP (Simple Object Access Protocol) přes HTTPS [24, 25].

OPC specifikace popisuje OPC COM (Component Object Model) objekty a jejich rozhraní implementované OPC serverem. OPC klient je schopen spojení s OPC serverem poskytovaný jedním, nebo více poskytovateli. Na nejnižší úrovni mohou OPC klienti získat nezpracovaná data z fyzických zařízení do SCADA, nebo ze systému SCADA do aplikace. Obr. 1.20 znázorňuje možné zapojení využívající operační rámec OPC. Vzhledem k tomu, že se nejedná o skutečný průmyslový protokol, tak musí dojít k jeho kombinaci s průmyslovým protokolem.

1.8.1 Bezpečnost

Díky použití DCOM a RPC je OPC velice citlivý na útoky. Může být ovlivněn všemi zranitelnostmi vyskytujícími se v OLE. OPC je využíváno pouze na systémech Windows, tudíž mohou být využity všechny zranitelnosti i tohoto operačního systému. Kvůli obtížnému provádění záplat v průmyslových řídících systémech je stále mnoho objevených zranitelných míst bez záplaty. Pro dosažení větší úrovně bezpečnosti je doporučeno využívat OPC-UA [1, 24, 25].

OPC servery by měly mít zesílené zabezpečení, je doporučeno veškeré nepoužívané porty a služby vypnout. Měly by být sledovány všechny porty a služby, které přímo nenáleží OPC, ale jsou OPC serverem iniciovány. Také by měl být monito-



Obr. 1.20: Ukázková síť využívající protokol OPC.

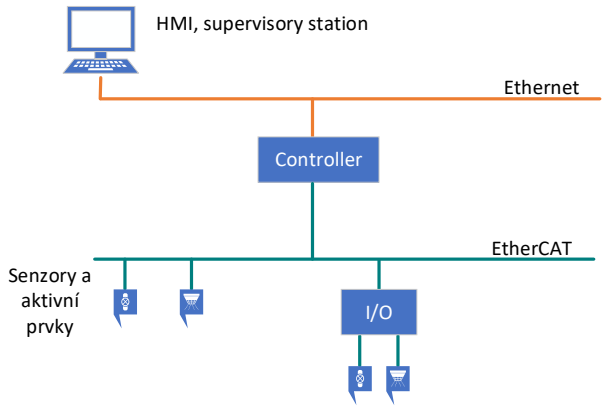
rován výskyt zranitelností spojených se systémy Windows, OPC, OLE RPC nebo DCOM. Ke zvýšení bezpečnosti by všechny OPC služby iniciované z neznámého OPC serveru měly být sledovány, také veškeré neúspěšné pokusy o autentizaci.

1.9 Ethernet for Control Automation Technology

EtherCAT (Ethernet for Control Automation Technology) je komunikační protokol s otevřeným kódem. Protokol byl standardizován v normě IEC 6115817 v rámci standardizace Fieldbus. Slouží k začlenění Ethernetu do průmyslového prostředí. Je používán v automatizačních aplikacích s velmi nízkými aktualizačními cykly a malým jitterem. Ethernetový paket je zpracováván za chodu a v každém slave uzlu dojde k aktualizaci informací a následně je paket odeslán na další zařízení [26]. Obr. 1.21 zobrazuje ukázkovou síť využívající protokol EtherCAT.

1.9.1 Bezpečnost

Protože je EtherCAT odvozený od Ethernetu, je náchylný na všechny zranitelnosti jako Ethernet. Je tak ohrožen velkou skupinou útoků zaměřených na DoS. Služby tohoto protokolu mohou být narušeny vložením ethernetových paketů takovým způsobem, že dochází k narušování synchronizace. Jsou také náchylné na útoky typu Man in the middle z důvodu nedostatečné autentizace. Jako opatření je tak doporučeno provoz izolovat od ostatních sítí založených na Ethernetu. Zavést pasivní monitoring provozu k zajištění integrity a kontrolovat, zda provoz pochází pouze z těch zdrojů, které jsou explicitně povoleny.



Obr. 1.21: Ukázková síť využívající protokol EtherCAT.

1.10 Obecné vektory útoku

Z důvodu výstavby sítí kritické infrastruktury před mnoha lety, kdy nebyl kladen takový důraz na bezpečnost, nejsou tyto sítě schopny odolat dnešním typům útoku. Při propojování distribuovaných systémů a umožnění komunikace z internetu se škála možných útoků [27] dále zvětšuje. Vzniká totiž další prostor pro malware, exploit, manipulaci konfigurace a ohrožení tak provozu ICS sítí. Je nutné, aby se bezpečnostní týmy IT a OT více přiblížily a začaly více spolupracovat. Většina bezpečnostních problémů, kterým čelí IT, čelí i OT. Mezi nejvíce časté vektory patří útok na dohledovou kontrolu a sběr dat, dále na pracovní stanice operátorů, HMI a controllery (PLC, RTU, DCS). Závěrečná fáze útoku typicky nahradí logiku controlleru vedoucí k poruše, fyzickému poškození zařízení apod. Vektory útoku se rozšiřují i na samotné senzory a aktivní prvky (field devices). Útoky mohou být mířeny na vyvolání DoS stavu, získání vzdáleného přístupu k PLC, kde je pozměněn kód a dochází k vzdálenému ovládání celého systému atd. Tento typ útoku může být dále rozšířen o napadení softwaru HMI, tak aby operátorům hlásil běžný stav systému a pravé chování skryl. IT odborníci vidí největší hrozbu v útoku na HMI. Jedná se o kritické místo zobrazující aktuální chování systému. Shromažďuje veškerá data, vytváří zprávy, provádí alarmování, generuje oznámení atd. Navíc v praxi jsou tyto systémy dostupné z internetu, i když mnohdy nedopatřením nebo chybným bezpečnostním nastavením.

Útoky lze rozdělit do tří kategorií. *IT* útoky, *IT to OT* útoky a *OT* útoky. IT útoky pocházejí z prostředí IT a jejich cílem je také IT prostředí. Do této skupiny spadá velké množství útoků, malware, spam apod. IT to OT je útok pocházející z IT prostředí zaměřený na OT zařízení/prostředí. Účelem tohoto druhu útoku je například pozměnit chování systému, přerušit provoz, krádež dat apod. Do této katego-

rie lze zařadit například malware Industroyer (také známý jako Crashoverride), který je schopen šíření po OT systémech prostřednictvím speciálních protokolů s cílem krádeže informací ze systému a následně jeho zničení. Dalším zástupcem je malware NotPetya využívající zranitelnosti OS Windows EternalBlue. Malware je schopen získat přístup na úrovni jádra, následně extrahovat privilegovaná pověření z paměti. Poté získá práva správce celé domény. Poslední možností jsou OT útoky. Tyto útoky pocházejí i cílí do OT prostředí.

1.11 Obecná protiopatření

Aby bylo možné provést protiopatření [28, 29] před možným napadením je nutné mít přehled o tom, z jakých komponent se síť skládá. Dalším kritickým bodem je včasné varování, že probíhá neobvyklé chování (monitorování sítě a detekce anomálií). Například pokud některé zařízení bude vyžadovat připojení na adresu mimo ICS síť, je to varování, že zařízení bylo zřejmě napadeno. Rychlost případné reakce je kritická. Dále je třeba identifikovat provoz, který se v síti nemá vyskytovat (například přítomnost malwaru). To vše musí být prováděno v jádru sítě. Sítě kritické infrastruktury typicky neumožňují zajistit bezpečnost koncových bodů. Pokud musí být průmyslová síť napojena do internetu je nutné zvolit co nejsilnější restrikce s přísnou kontrolou přístupu.

Dále omezit externí připojení a komunikaci izolovat od zbytku sítě. Stanovit požadavky na autentizaci a integritu dat. Dále kontrolovat přístup operátorů z intranetu do procesní sítě, protože i počítače operátorů mohou být infikovány a mohou sloužit jako brána pro vzdálené vykonání útoku. Také je nutno kontrolovat přístup z pracoviště operátora na SCADA servery. Možným řešením je využití brány firewall s podporou připojení VPN (Virtual Private Network) a autentizace pomocí Radius serveru. Typicky se provádí kombinace firewallu a IDS. Výrobci zařízení musejí klást větší důraz na zapracování bezpečnosti do samotných zařízení.

1.12 Výběr průmyslového protokolu

Mezi nejrozšířenější protokoly v rámci průmyslových sítí patří protokol Modbus a protokol DNP3 [30, 31]. Patří také mezi protokoly, na které jsou cíleny útoky (protokol Modbus 2. nejčastější, DNP3 3. nejčastější) [32]. Proto se bude dále tato práce zabývat jen průmyslovými protokoly Modbus/TCP a DNP3.

Tab. 1.6 zobrazuje porovnání protokolu Modbus a protokolu DNP3 z pohledu poskytované bezpečnosti [33, 34].

Tab. 1.6: Porovnání implementované bezpečnosti u protokolu Modbus/TCP, DNP3.

Protokol	Zabezpečení	Popis
Modbus/TCP	Žádné	Postačuje znalost IP adresy a kódu funkce, port 502
Modbus/TCP Security	Minimální požadavky: **Klíčová výměna**: RSA **Šifrování**: AES 128 CBC **Integrita**: SHA256	Využívá TLS, port 802 x.509v3 certifikát
DNP3	Žádné	Postačuje znalost IP adresy a kódu funkce, port 20000
DNP3 Secure (ver. 6)	**Klíčová výměna**: RSA **Šifrování**: AEAD-AES-256-GCM (jako TLS1.2) **Integrita**: Message Authentication Codes (BLAKE2, SHA-3, eliptické křivky)	Využívání certifikátů

Modbus: Aby bylo možné provést komunikaci mezi Master (klient) a Slave (server) je třeba implementovat knihovnu. Tab. 1.7 zobrazuje některé knihovny, které umožňují realizovat komunikaci pomocí protokolu Modbus.

Tab. 1.7: Dostupné knihovny pro implementaci Modbus komunikace.

Název	Serial Modbus	Modbus/TCP	Verze	Programovací jazyk	Dokumentace
Arduino Modbus	ANO	ANO	Master, slave	Arduino	Online
EasyModbus	ANO	ANO	Master, slave	.NET, Java, Python	Online
PyModbus	ANO	ANO	Master, slave	Python	Online
JLibModbus	ANO	ANO	Master, slave	Java	Online
Libmodbus	ANO	ANO	Master, slave	C	Online – částečná
Minimalmodbus	ANO	NE	Master, slave	Python	Online
Modbus4J	ANO	ANO	Master, slave	Java	–
Modbusdriver	ANO	ANO	Slave	C#, VB.net, Java	Online
Modbus-tk	ANO	ANO	Master, slave	Python	Online – částečná
SuperCom	ANO	ANO	Master, slave	C++, C#, Delphi, Java	Online

Existují i další nástroje, které umožňují provést komunikaci ve formě aplikací. Pro simulaci Modbus lze využít například *Modbus tool* (realizující Master, Slave), *Modbus PLC simulator* (Slave), *Modbus Examiner* (Master), *XmasterSlave* (Master, Slave) nebo *Vinci software* (Master, Slave).

DNP3: Protokol DNP3 nabízí knihovnu *openDNP3*, popřípadě aplikace pro simulaci, například DNP3 *Protocol Client Master Simulator* (Master), *DNP3 RTU Outstation Server Simulator* (Slave), *DNP3 protocol* (Master, Slave) nebo *FreyrSCADA/DNP3* (Master, Slave). Také je možné využít nástroj *Axon Test*, který je určen pro simulaci v rámci jedné aplikace.

K simulaci protokolu Modbus/TCP byla vybrána *Pymodbus* [35] z důvodu implementace v programovacím jazyce Python a také z důvodu podpory jak Master, tak Slave verze. Tato knihovna obsahuje i podrobnou dokumentaci. K simulaci protokolu DNP3 byla využita knihovna *openDNP3* [36], python vazbu na tuto knihovnu poskytuje open source implementace *PyDNP3*.

1.13 Vybrané bezpečnostní incidenty

Tab. 1.8 zobrazuje vektory útoků zaměřené na Master, Slave zařízení a na přenosové médium. Ve sloupci protokol je uveden protokol, na kterém bude daný útok simulován/detekován. Útoky na Master a Slave zařízení mohou být velice podobné a některé z detekčních metod z vytvořených pro detekci na Slave zařízení mohou být nasazeny i na Master zařízení. Útoky mohou být různého charakteru, ale pro účel detekování bezpečnostních incidentů nezáleží, zda bylo Master zařízení napadeno a je ovládáno útočníkem, nebo útok pochází z jiného nevalidního Master zařízení.

Útok na médium je spojen s ustanovením Man in the middle, tuto akci lze detekovat pomocí odlišného *RTT delay*. V tabulce nejsou uvedeny útoky zaměřené na samotné senzory a aktivní prvky. Tyto útoky jsou často fyzického charakteru spojeného s poškozením zařízení.

Tab. 1.8: Detekce bezpečnostních incidentů.

Vektor útoku	Protokol	Útok	Událost	Nástroj	Detekování
Slave	Modbus	Změna IP adresy, viz 3.1	Signatura	Snort	Porovnání IP adresu s white-listem, viz 3.1
		Generování zpráv neobvyklých velikostí, viz 3.1	Signatura	Snort	Detekování velikosti paketu a porovnání s normální velikostí, viz 3.1
		Skenování UID, viz 3.2.1	Anomálie	Zeek	Detekování prováděné metody, viz 3.2.3
		DoS, viz 3.2.1	Anomálie	Zeek	Detekování změny rozestupů zpráv, viz 3.2.3
		Skenování FC, viz 3.2.1	Anomálie	Zeek	Detekování posloupných operací, viz 3.2.3
	DP3	Nmap, viz 3.3.1	Anomálie	Zeek	Detekování nárustu TCP spojení/operace Freeze at time/nevalidní zprávy, viz 3.3.3
		Operace mimo interval, viz 3.3.1	Anomálie	Zeek	Detekování operace s odlišným odstupem, viz 3.3.3
			Anomálie	ML, viz 3.4.2	Detekování anomálií, viz 3.4.2
Master	-	DoS	Anomálie	Zeek	Detekování změny rozestupů zpráv
		Podvržení Slave	Anomálie	Snort, Zeek	Porovnání IP adresu s white-listem, detekce odlišného chování
Médium	Modbus	MITM, viz 3.3.1	Anomálie	ML	Detekování odlišného RTT delay, viz 3.4.1
	DNP3	MITM	Anomálie	Zeek	Detekování odlišného RTT delay, viz 3.3.3
			Anomálie	ML	Detekování odlišného RTT delay, viz 3.4.2

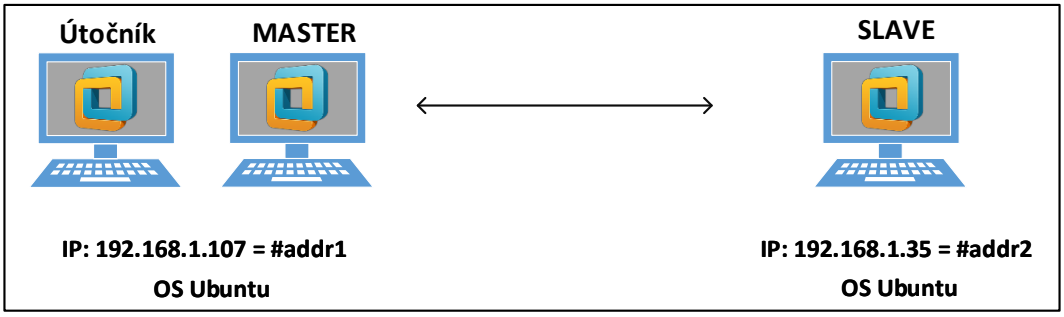
Mezi hlavní zástupce bezpečnostních incidentů v síti lze zařadit *manipulace s Master stanicí*, *skenování sítě*, *vyčerpání výpočetních kapacit*, *manipulace s pravidelností komunikace* a *manipulace toku dat*.

Vybrané bezpečnostní incidenty lze detekovat pomocí specifických metod na Slave zařízení, obdobně lze tyto bezpečnostní incidenty detekovat i na Master zařízení. Jako manipulaci s Master stanicí lze považovat změnu IP adresy, nebo generování zpráv s pozměněnou velikostí. Dále skenování sítě na přítomnost určitého zařízení (jako např. skenování UID, Nmap, FC). Útok za účelem vyčerpání výpočetních kapacit cílového zařízení (DoS). Dále odlišnosti druhu komunikace z pohledu změny časového intervalu, po kterých jsou pravidelně zasílané zprávy (operace mimo interval). V neposlední řadě ustanovení MITM, princip detekce toho útoku pomocí RTT delay je schopen detekovat i další útoky, jako například podvržení jiné stanice v síti. Popřípadě vytvoření další Master stanice, která běžně nekomunikuje s cílovým Slave zařízením. K detekování těchto vybraných incidentů v síti je nutné využívat detekci signatur a anomálií v provozu, popřípadě využívat metodu strojového učení.

2 Příprava experimentálního prostředí

Experimentální testování

K experimentálnímu testování bezpečnostních incidentů bylo navrženo síťové zapojení, viz obr. 2.1. Jednotlivé operační systémy jsou virtualizovány ve virtualizačním nástroji VMware Workstation 15 Player, stroje mají přiděleny 2 GB operační paměti. Hostované OS využívají OS Ubuntu 18.04. Master stanice (Útočník) používá IP adresu *192.168.1.107*, která bude v rámci výpisů zkrácena na *#addr1*. Podobně Slave stanice využívá IP adresu *192.168.1.35*, která bude zkrácena na *#addr2*. Síťové propojení jednotlivých stanic bylo nastaveno na síťový most, aby byla zajištěna přímá viditelnost stanic. Master stanice generuje legitimní provoz a útočník je vybaven nástrojem pro provádění penetračního testování. Experimentální testování bude zaměřeno na průmyslový protokol Modbus a protokol DNP3. K realizaci protokolu Modbus bylo využito knihovny PyModbus a k realizaci protokolu DNP3 bylo využito knihovny PyDNP3.



Obr. 2.1: Síťové zapojení ve virtualizačním nástroji VMware.

2.1 Instalace knihovny pro protokol Modbus

PyModbus

Jedná se o plnou implementaci protokolu Modbus a umožňuje provést jakoukoli operaci, kterou protokol podporuje. Instalaci knihovny lze provést pomocí příkazu viz výpis 2.1.

Výpis 2.1: Instalace knihovny PyModbus.

```
1 pip3 install pymodbus
```

Čistá instalace Ubuntu neobsahuje nástroj Pip (správa balíčků pro moduly programovacího jazyka Python). Tento nástroj lze nainstalovat pomocí příkazu viz výpis 2.2.

Výpis 2.2: Instalace správce balíčků Pip.

```
apt install python3-pip
```

K zajištění komunikace mezi Master a Slave zařízením byly využity vzorové skripty, které nejsou součásti balíčku PyModbus. K realizaci Master zařízení byl použit Python skript Synchronous Client [37] (master.py) a k realizaci Slave zařízení Updating Server [38] (slave.py).

Nejprve je nutné provést konfiguraci Master a Slave skriptů. Zde je nutné upravit IP adresy obou zařízení. Na straně Slave zařízení je nutné doinstalovat nástroj Twisted, aby bylo možné aktualizovat kontext při běhu skriptu, viz výpis 2.3.

Výpis 2.3: Instalace nástroje Twisted.

```
pip3 install Twisted
```

Spuštění skriptů lze provést viz výpis 2.4. Nejprve je nutné spustit skript slave.py na Slave zařízení a následně spustit skript master.py na Master zařízení.

Výpis 2.4: Spuštění skriptů.

```
python3 slave.py
python3 master.py
```

Po spuštění skriptu na straně Master je nejprve sestaveno TCP spojení, poté je provedena operace write single register, která je potvrzena a zopakována ze strany Slave zařízení. Tato zpráva je opět potvrzena. Následně dochází k ukončení spojení.

2.2 Instalace knihovny pro protokol DNP3

PyDNP3

K realizaci protokolu DNP3 bylo využito vazby knihovny DNP3 pro Python, dostupné na [39]. Následně je třeba instalovat podporu protokolu DNP3 pomocí příkazu pip, viz výpis 2.5.

Výpis 2.5: Zachycený útok pomocí druhé metody.

```
sudo pip3 install pydnp3
```

K zajištění základní komunikace bylo využito nástrojů, které jsou součástí [39]. V souboru *master.py* je třeba pozměnit adresu Outstation zařízení. Tato adresa je zapsána do proměnné HOST, tedy *HOST="#addr2"*.

2.3 Instalace IDS Snort

Pro detekci bezpečnostních incidentů ve formě signatur byl vybrán systém IDS/IPS Snort [40]. Tento systém slouží pro detekci anomálií a signatur v síťovém provozu. V případě, že je detekována anomálie, popřípadě rozpoznána signatura, je vyhlášen poplach. Příkaz pro instalaci IDS/IPS Snort viz výpis 2.6.

Výpis 2.6: Instalace IDS/IPS Snort.

```
1 sudo apt install snort
```

Po instalaci je třeba provést konfiguraci. V konfiguračním souboru je třeba definovat síť a rozhraní, na kterém má systém naslouchat a vybrat balíčky s předpřipravenými pravidly. Následně je třeba provést ověření konfigurace. Nově vzniklá pravidla je vhodné přidávat do souboru „local.rules“. Jedná se o soubor vyhrazený pro pravidla vytvořená uživatelem, popřípadě vytvořit nový soubor a definovat jej v konfiguračním souboru.

Příkaz, pomocí kterého je provedena kontrola síťového provozu v režimu IDS na rozhraní *ens33* viz výpis 2.7.

Výpis 2.7: Spuštění IDS/IPS Snort.

```
1 sudo snort -A console -q -u snort -c /etc/snort/snort.conf -i ens33
```

2.4 Instalace IDS ZEEK

Pro detekci anomálií byl vybrán IDS systém ZEEK. Tento systém umožňuje práci s protokolem Modbus i protokolem DNP3. Kontrolu síťového provozu provádí na základě vytvořených skriptů. Před instalací samotného systému je třeba provést instalaci závislostí, viz výpis 2.8. Následně je třeba získat git repozitář a provést instalaci, viz výpis 2.9. Poté je nutné upravit konfigurační soubory *node.cfg, networks.cfg* a *zeekctl.cfg*. Využití základních skriptů pro podporu Modbus protokolu není ve výchozím stavu definována. Proto je nutné upravit soubor *local.zeek* umístěný v */usr/local/zeek/share/zeek/site*. V tomto souboru je nutné definovat, který skript bude využíván. V tomto případě byl využit skript *track-memmap.zeek*, který byl dále doplňován. Pro jeho načtení je nutné uvést cestu ke skriptu *@load protocols/modbus/track-memmap*. Pro podporu DNP3 není nutné provádět žádné změny.

Výpis 2.8: Zeek, instalace závislostí.

```
1 sudo apt-get install cmake make gcc g++ flex bison libpcap-dev libssl-dev
      python-dev swig zlib1g-dev
```

Výpis 2.9: Zeek instalace.

```
git clone --recursive https://github.com/zeek/zeek
./configure
make
make install
```

2.5 Scénáře pro testování

Testování bezpečnostních incidentů bude založeno na vybraných bezpečnostních incidentech, viz tab 1.8, zaměřené na průmyslový protokol Modbus/TCP a protokol DNP3. Testování bude založeno na výskytu útočníka v síti schopného generovat validní zprávy průmyslového protokolu. Síťové zapojení bude odpovídat obr. 2.1.

K zachytávání síťového provozu bude využita aplikace Wireshark. V případě ustanovení MITM bude využíván stroj s operačním systémem Kali Linux, který bude využívat integrovaných nástrojů, jako Ettercap k vytvoření ARP spoofingu. V případě MITM útoku bude použit v síťovém zapojení další virtualizovaný stroj.

Experimentální testování zaměřené na protokol Modbus/TCP bude využívat nástroj penetračního testování smod1 a upravené skripty Master stanice *master.py*. Testování zaměřené na protokol DNP3 bude využívat nmap nástroj na skenování DNP3 protokolu a upravené skripty Master stanice. Skripty sloužící ke komunikaci pomocí zvoleného průmyslového protokolu budou v obou případech upraveny a budou sloužit k simulaci útoků zmíněných v tab. 1.8.

Realizace strojového učení bude využívat open source nástroj Weka a Python implementace strojového učení scikit-learn. K detekci anomálií bude využito obou přístupů, jak *strojové učení s učitelem*, tak *strojové učení bez učitele*. Strojové učení bez učitele bude využito k identifikaci operací provedených mimo interval a strojové učení s učitelem bude využito k rozpoznávání ustanovení MITM.

Tab. 2.1 zobrazuje vybrané průmyslové protokoly a nástroje, které budou využívány během experimentálního testování.

Tab. 2.1: Protokoly a vybrané nástroje.

Protokol	Nástroj
Modbus	smod1
	upravené skripty
DNP3	nmap
	upravené skripty
	Ettercap

3 Experimentální testování

3.1 Detekce signatury

Pro ověření schopnosti detekovat bezpečnostní incident byl proveden pokus s detekováním žádosti a pozměnění hodnot registru Slave zařízení z neautorizovaného Master zařízení. Z důvodu, že protokol Modbus neumožňuje provést autentizaci, lze filtrovat Master zařízení dle jejich známé IP adresy. Výpis 3.1 zobrazuje pravidlo, které filtruje provoz a v případě, že je navázáno spojení na IP adresu Slave zařízení na port Modbus protokolu (v případě experimentálního testování pomocí signatury na port 5020) z jiné, než definované IP adresy je provedeno pravidlo alert (upozornění). V případě, že se definované chování v síti vyskytne, dochází k vygenerování upozornění. Provedené upozornění, viz výpis 3.2.

Výpis 3.1: Pravidlo pro detekci neautorizovaného přístupu.

```
1 alert tcp !$MODBUS_CLIENT any -> $MODBUS_SERVER 5020 (content:"|00 00|";
      msg:"SCADA_IDS: Modbus/TCP: Unauthorized request"; sid:102; priority:1;)
```

Výpis 3.2: Alert vygenerovaný pravidlem při pokusu i změnu hodnoty.

```
1 12/02-11:59:20.782511 [**] [1:102:0] SCADA_IDS: Modbus/TCP: Unauthorized
      request [**] [Priority: 1] {TCP} #addr1:58669 -> #addr2:5020
```

Dále bylo vytvořeno pravidlo, které detekuje komunikaci pomocí Modbus/TCP a v případě, že je překročen stanovený limit maximální délky paketu, je vygenerován alert, viz výpis 3.3. Vygenerovaný alert viz výpis 3.4.

Výpis 3.3: Pravidlo pro detekci příliš velkého paketu.

```
1 alert tcp any any <> $MODBUS_SERVER 5020 (content:"|00 00|"; dsize:>300;
      msg:"SCADA_IDS: Modbus/TCP: Illegal Packet Size, Possible DoS"; sid:104;
      priority:1;)
```

Výpis 3.4: Alert při překročení maximálního limitu délky paketu.

```
1 12/03-06:43:17.499610  [**] [1:104:0] SCADA_IDS: Modbus/TCP: Illegal Packet
      Size, Possible DoS [**] [Priority: 1] {TCP} #addr1:58669 -> #addr2:5020
```

Veškeré alerty, které poukazují na nestandardní chování je nutné prověřit. Popřípadě na ně přímo reagovat stanoveným opatřením.

3.2 Modbus

3.2.1 Simulace bezpečnostních incidentů

UID

K realizaci testování bezpečnostních incidentů bylo využito Master stanice. Tato stanice k provádění penetračního testování využívá nástroj smod1 (MODBUS Penetration Testing Framework) [41]. Tento nástroj umožňuje zaměřit penetrační testování na různé operace, což je vhodné k jejich detekci. Aby bylo možné provádět penetrační testování, je nejprve nutné detekovat UID slave zařízení. K spuštění skriptu postačuje znalost IP adresy Slave zařízení. Detekce UID, viz výpis 3.5.

Výpis 3.5: Detekce UID na zařízení Slave.

```
SMOD >use modbus/scanner/uid
SMOD modbus(uid) >set RHOSTS #addr2
SMOD modbus(uid) >exploit
[+] Module Brute Force UID Start
[+] Start Brute Force UID on : #addr2
[+] UID on 192.168.1.35 is : 10
```

V době spuštění skriptu bylo spuštěno zaznamenávání síťového provozu na straně Slave zařízení. Skript pracuje na principu navázání TCP spojení se Slave zařízením a následné využití operace Read Coils (kód funkce 1). Pomocí této operace je z paměti zařízení vyčteno UID.

DoS

K provedení DoS byla vybrána operace WSR (Write Single Register). Protokol Modbus je založen tak, že tato operace musí být Slave zařízením ihned potvrzena a to tak, že je zaslána totožná zpráva zpět k odesilateli zprávy. Slave zařízení typicky není příliš výkonné a může tak dojít k vyčerpání výpočetních kapacit. Postačuje, když útočník bude disponovat vyšším výpočetním výkonem než Slave zařízení. V případě, že operace WSR budou zasílány z více Master zařízení dochází k distribuovanému útoku (DDoS). Výpis 3.6 zobrazuje postup ke spuštění DoS útoku na Slave zařízení.

Výpis 3.6: DoS na slave zařízení.

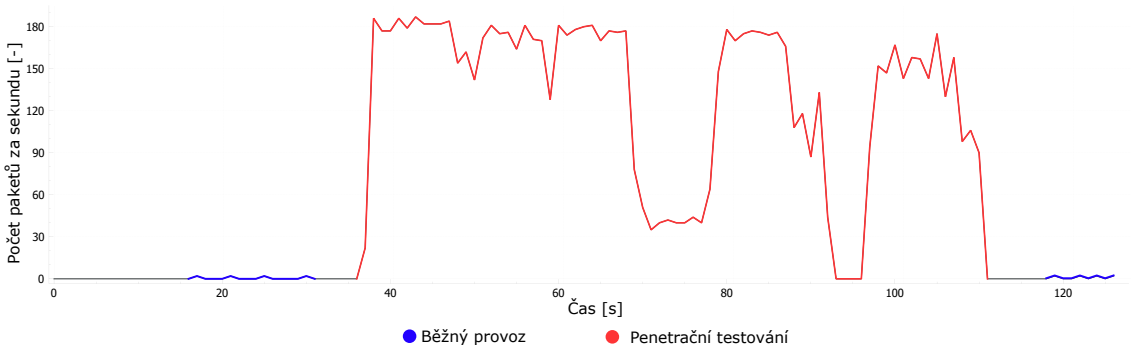
```
SMOD >use modbus/dos/writeSingleRegister
SMOD modbus(writeSingleRegister) >set RHOST #addr2
SMOD modbus(writeSingleRegister) >set UID 10
SMOD modbus(writeSingleRegister) >exploit
[+] Module DOS Write Single Register Start
```

Probíhající DoS útok byl zaznamenáván. Během probíhajícího penetračního testování (2 minuty a 6 sekund) bylo zaznamenáno celkem 49 284 paketů. Počet TCP paketů byl 49 227, v kterých bylo 9 852 Modbus paketů k operaci WSR (kód

funkce 6). Tedy 20 % komunikace bylo tvořeno samotnými zprávami Modbus protokolu a 80 % tvořilo navazování a ukončování TCP relace a další komunikace. Průměrná velikost zpráv byla 69,70 B, resp. 70 B. Nejmenší přenesená zpráva dosahovala velikosti 60 B a největší velikosti 78 B. Z Master zařízení byl proveden zápis na 4 927 registrů ve Slave zařízení.

Pokud je zobrazen průběh testování v závislosti na zpracovaných paketech za 1 s, získaný graf, viz obr. 3.1. Běžná komunikace (reprezentovaná odesláním jedné hodnoty na první registr) je zobrazena modře. Po spuštění nástroje na penetrační testování počet zpracovaných paketů stoupá, na grafu zobrazeno červeně. V případě prováděného penetračního testování jsou zřetelné výkyvy, ty lze vysvětlit vyčerpáním paměti potřebné ke zpracování komunikace z důvodu virtualizace obou systémů.

Princip útoku DoS je založen na vygenerování velkého množství paketů směrem k cíli tak, aby došlo k odepření služby. Totožná situace je zřetelná z vizualizovaného penetračního testování. Jednotlivé pakety jsou generovány s velmi malými rozestupy, tím dochází k hromadění zpráv u cíle, který postupně vyčerpává své výpočetní kapacity na uložení a zpracování příchozích zpráv. Modře zvýrazněné zprávy dosahují rozestupu jednotek sekund, červeně zvýrazněné zprávy jednotek setin sekundy.



Obr. 3.1: DoS na operaci Write Single Register.

Detekce Function Code

K provedení simulace bezpečnostního incidentu byl použit nástroj *smod1* [41]. Pomocí tohoto nástroje lze provést skenování podporovaných kódů funkce na straně Slave zařízení. Spuštění skenování viz výpis 3.7. Během probíhajícího skenování byl zaznamenáván síťový provoz na straně Slave zařízení. Dochází k navázaní spojení a testování jednotlivých kódů funkce začínající kódem funkce 0 (nedefinovaný kód funkce) až do kódu funkce 15 (Write Multiple Coils).

Výpis 3.7: Detekce podporovaných funkčních kódů.

```
SMOD >use modbus/scanner/getfunc
SMOD modbus(getfunc) >set RHOSTS #addr2
SMOD modbus(getfunc) >set UID 10
SMOD modbus(getfunc) >exploit
[+] Module Get Function Start
[+] Looking for supported function codes on #addr2
[+] Function Code 1(Read Coils) is supported.
```

3.2.2 Navržená opatření

UID

Proti možnosti vyčtení hodnoty UID z paměti Slave zařízení je možné detekovat pokus o přístup k vyčtení hodnoty pomocí metody Read Coils. V případě, že je takový pokus detekován je provedeno upozornění. Další možností je detekovat útok hrubou silou. Pokud je navazováno spojení s nevalidním UID, je nutné vygenerovat upozornění.

DoS

DoS útok lze detekovat pomocí hodnoty mezi-rámcové mezery δ. Tuto hodnotu ale není možné přesně stanovit bez znalosti provozu a běžného využívání metody WSR, vůči které je nutné anomálie detekovat. K detekci anomálie je tak nutné stanovit prahovou hodnotu Δ. Tato hodnota je vyhodnocena na základě několika (x) mezi-rámcových mezer δ předchozích WSR operací, které jsou následně zprůměrovány. Je nutné nejprve provést fázi *učení*, pomocí které je prahová hodnota vypočtena dle vzorce 3.1.

$$\Delta = \frac{\sum_{n=1}^{x} \delta}{x} \quad [\text{s}] \qquad (3.1)$$

Po fázi *učení* následuje fáze *nasazení*. V této fázi je vždy porovnávána jednotlivá mezi-rámcová mezera δ s prahovou hodnotu Δ, viz rovnice 3.2. V případě, že momentální mezi-rámcová mezera dosahuje menší hodnoty než prahová hodnota, může se jednat o nežádoucí provoz. Následně je vhodné provést upozornění na nestandardní provoz. Tyto upozornění je dále vhodné prověřit, zda byl tento provoz skutečně zamýšlen nebo je tento provoz projevem výskytu útočníka v síti. Tuto podmínku lze také obrátit, tedy kontrolovat, zda není δ příliš vzdálena od prahové hodnoty. Toto závisí na typickém používání WSR operace v síti. Přísnost rovnice lze korigovat pomocí parametru k.

$$\delta < k \cdot \Delta \quad [–] \qquad (3.2)$$

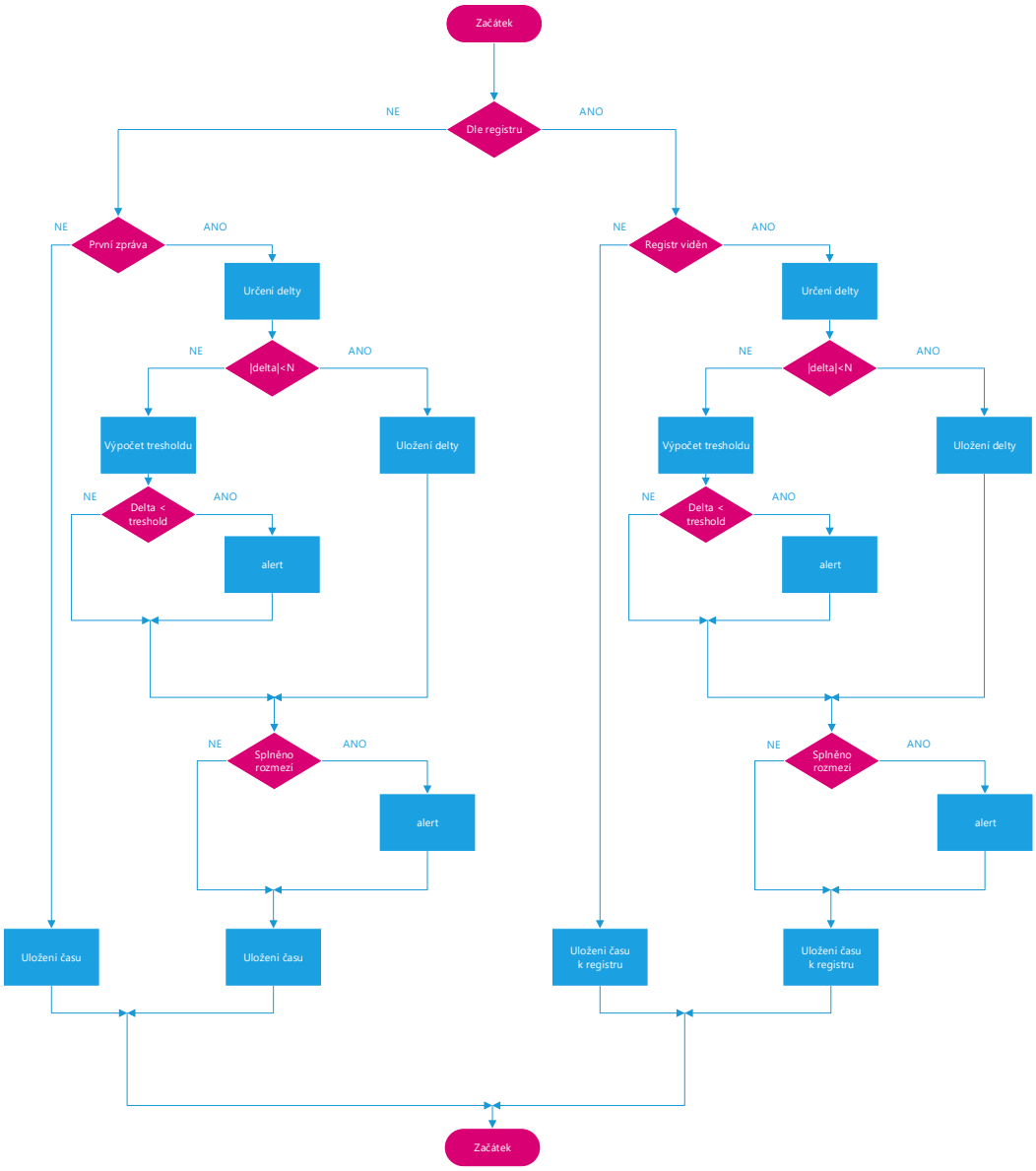
K DoS útoku zaměřeného na WSR operaci lze přistupovat z více pohledů. Jeden z pohledů je takový, že zprávy, které způsobí DoS mohou mít nastavený registr a cílit na jeden jediný. V praxi je možné, že se v běžném stavu bude ke každému registru přistupovat v jiných intervalech (některý registr bude vícekrát čten, než jiný). Pro tento případ je možné vzorec 3.1 a 3.2 definovat pro každý registr odlišně.

Útok DoS lze také provádět cyklicky. Zapisované registry se cyklicky opakují a mezi-rámcová mezera, po kterém dojde k opětovnému zápisu na vybraný registr, je velmi podobná s předchozí hodnotou. Pro tento případ lze využít vzorec 3.3. V tomto případě musí dojít k určení hodnoty r *[%]*, která určuje proměnlivost od předchozí hodnoty tak, aby při splnění došlo k vyhlášení poplachu. Je nutné zmínit, že pokud je operace WSR v infrastruktuře prováděna cyklicky, tak tento přístup bude způsobovat falešné poplachy a je nutné k rovnici přistupovat z opačného pohledu. V případě jejího splnění bude provoz validní a pokud splněna nebude, se jedná o anomálii.

$$\delta_{k-1} \cdot (1 - r) \leq \delta_k \leq \delta_{k-1} \cdot (1 + r) \qquad [-] \qquad (3.3)$$

Z analýzy provedeného penetračního testování je ale zjevné, že hodnoty registrů, do kterého bude proveden zápis, jsou určeny náhodně. Tím pádem nelze uplatnit přístup, vztahování hodnot ke každému registru odlišně. Kdyby systém byl nastaven tímto způsobem, docházelo by k vyhlášení poplachu jen minimálně, z důvodu velké proměnlivosti zápisu na totožný registr. Vytvoření prahové hodnoty Δ proto musí být nezávislé na vybraných registrech, stejně jako uplatňování zmíněných podmínek.

Zjednodušený diagram zobrazený na obr. 3.2 zobrazuje návrh, jakým způsobem přistupovat k detekci anomálií v WSR operaci protokolu Modbus. Oba dva přístupy (s použitím registru, nebo bez něj) jsou téměř totožné. Jediná odlišnost je v případě využití odlišování více registrů vytvoření více prahových hodnot (tresholdů) pro každý registr odlišně. Bez využití vazby na jednotlivý registr vzniká jednotná prahová hodnota, která je porovnávána. Detekce běžného útoku (jednorázová změna) je tak těžší, ale útok DoS, popřípadě DDoS lze touto metodou detekovat snadno. Jednodušší detekce DoS je zajištěna z důvodu neodlišování přístupů pro každý registr odlišně. Varianta, kdy nedochází k rozlišování registrů je vhodnější pro útok, kdy jsou jednotlivé registry k zápisu voleny náhodně.



Obr. 3.2: Vývojový diagram postupu detekce anomálií.

Detekce Function Code

Skenování portů může být prováděno několika způsoby. Lze využít například toho, že skenování je typicky prováděno v iterativní smyčce, tedy jednotlivé kódy funkce jsou iterativně navyšovány. Dále je vždy vybrán jeden kód funkce a k otestování jeho podpory ze strany Slave zařízení je použit pouze jeden dotaz a na základě příchozí odpovědi je rozhodnuto, zda je podporován. Dochází tak ke střídání kódů funkce po jedné zprávě. Stačí proto detekovat iterativní chování a určení hodnoty, kolik iterativně navyšovaných provedených kódů funkce má být považováno za neobvyklé.

Detekce také závisí na běžném chování sítě, kde tato komunikace probíhá. Pokud nejsou v síti některé kódy funkce využívány, může zpráva vyskytující se v síti s dotazem na tento nevyužívaný kód funkce upozorňovat na možnost přítomnosti útočníka nebo na neobvyklého chování.

3.2.3 Implementovaná opatření

UID

V IDS systému ZEEK byla ve skriptu *track-memmap.zeek* vytvořena metoda *modbus-read-coils-request*. Tato událost vytvoří záznam při pokusu o přečtení hodnoty registru. Z logu, viz výpis 3.8, je zřetelný pokus o získání hodnoty UID. Tato hodnota je uložena na nulté pozici (ve výpisu modře vyznačena) v registru o celkové délce jeden registr (červeně vyznačeno).

Výpis 3.8: Zeek UID, detekce.

```
1 #path modbus_register_change
2 #open 2020-03-16-08-39-16
3 #fields ts id.orig_h  id.orig_p id.resp_h id.resp_p text  start_addr
      start_addr_quant
4 #types  time    addr  port  addr  port  string  count count
5 1584373156.723518 #addr1  58698 #addr2  502 DETEKCE 0  1
```

DoS

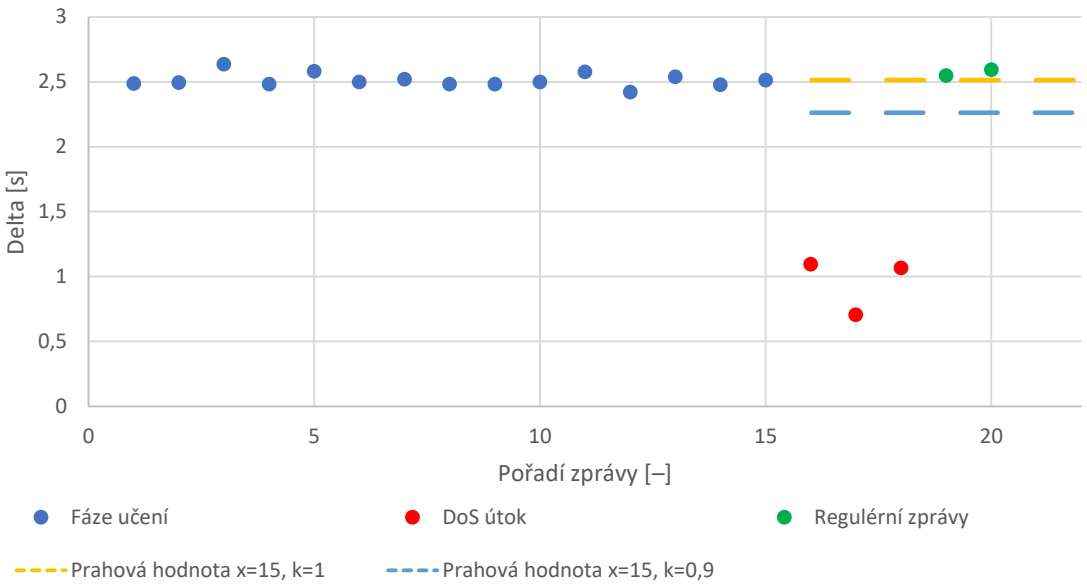
V IDS systému Zeek byla ve skriptu vytvořena metoda *modbus-write-single-register-request*, která detekuje veškeré příchozí WSR operace. Tento skript je vytvořený tak, že umožňuje pracovat dle vývojového diagramu, obr. 3.2. V tomto skriptu byly využity vzorce 3.1, 3.2 a 3.3. V rámci experimentálního testování bylo x nastaveno na hodnotu 15 a r na hodnotu 7 %. Z důvodu, že je odstup jednotlivých zpráv konstantního charakteru, je parametr x dostačující pro vytvoření prahové hodnoty a je možné jej i zkrátit. Nastavení parametru r vychází z experimentálního testování, kde je tato hodnota schopna zachytit útok, ale nevyvolává ani falešné poplachy, ani nedochází k situaci, že by útok nebyl detekován. Prvních x = 15 zpráv slouží k vytvoření prahové hodnoty Δ. Tato fáze vyžaduje, aby bylo zajištěno, že se v síti nevyskytuje útočník. Okamžitě po naučení hodnoty dochází k detekování útoku (zobrazeno červeně).

Výpis 3.9: Zachycený útok DoS.

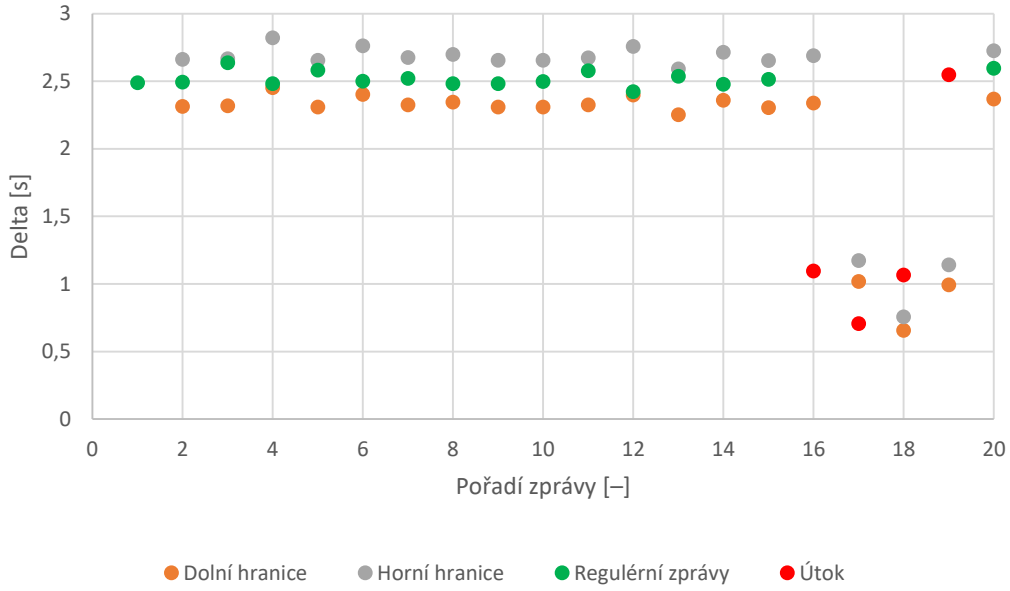
```
#path modbus_register_change
#open 2020-04-28-08-28-09
#fields id,orig_p id,resp_p delta src_addr  dst_addr  text  treshold
#types  port  port  interval  addr  addr  string  interval
  44667 502 -         #addr1  #addr2  Navazano spojeni          -
  33775 502 2,488247  #addr1  #addr2  Ucim se                   0
  41443 502 2,493375  #addr1  #addr2  Ucim se                   0
  50385 502 2,636994  #addr1  #addr2  Ucim se                   0
  53723 502 2,482671  #addr1  #addr2  Ucim se                   0
  39281 502 2,582492  #addr1  #addr2  Ucim se                   0
  54969 502 2,500161  #addr1  #addr2  Ucim se                   0
  42703 502 2,521266  #addr1  #addr2  Ucim se                   0
  57003 502 2,482172  #addr1  #addr2  Ucim se                   0
  40683 502 2,482638  #addr1  #addr2  Ucim se                   0
  39729 502 2,499380  #addr1  #addr2  Ucim se                   0
  42875 502 2,577473  #addr1  #addr2  Ucim se                   0
  55365 502 2,422150  #addr1  #addr2  Ucim se                   0
  41127 502 2,538051  #addr1  #addr2  Ucim se                   0
  34571 502 2,478753  #addr1  #addr2  Ucim se                   0
  40553 502 2,513979  #addr1  #addr2  Ucim se                   0
  50279 502 1,095429  #addr1  #addr2  TRESHOLD PREKROCEN, DoS!  2,51332
  33729 502 0,706493  #addr1  #addr2  TRESHOLD PREKROCEN, DoS!  2,51332
  48083 502 1,067000  #addr1  #addr2  TRESHOLD PREKROCEN, DoS!  2,51332
  35173 502 2,547971  #addr1  #addr2  Zprava prijata            2,51332
  45167 502 2,594318  #addr1  #addr2  Zprava prijata            2,51332
```

Vizualizovaný log 3.9 s využitím rovnice 3.3 je zobrazen na obr. 3.3. Po zaslání x = 15 zpráv dochází k „naučení“ prahové hodnoty. Při zvolení nejpřísnější prahové hodnoty (parametr k nastaven na hodnotu 1, zobrazeno žlutě) dochází ke generování falešných poplachů. Jako vhodnější nastavení tohoto parametru se jeví k = 0,9 (zobrazeno modře). Po nastavení prahové hodnoty dochází ke stanovení, zda se jedná o DoS útok (zobrazeno červeně) nebo se jedná o legitimní provoz (zobrazeno zeleně).

Obr. 3.4 zobrazuje nasazení rovnice 3.3 na totožná data. Z důvodu, že zprávy jsou zde cyklického charakteru, je při splnění rovnice provoz považován za legitimní, v opačném případě za útok. Parametr r zde byl nastaven na hodnotu 7 %. Nedochází tak k falešným poplachům. Tato rovnice nepotřebuje fázi „učení“, vždy je vycházeno z předchozí zprávy, u které jsou nastaveny hranice, ve kterých by se měla následující zpráva vyskytovat. Pokud tyto hranice nejsou dodrženy, je vyhlášen poplach. Z důvodu, že se vždy využívá předchozího paketu, dochází k označování legitimního provozu za útok, a to v případě, že dochází k ukončení útoku. Totožně v případě, že nastalý útok má konstantní charakter, je dále označován za legitimní provoz.



Obr. 3.3: Vizualizace logu pomocí první metody.



Obr. 3.4: Vizualizace logu pomocí druhé metody.

Detekce Function Code

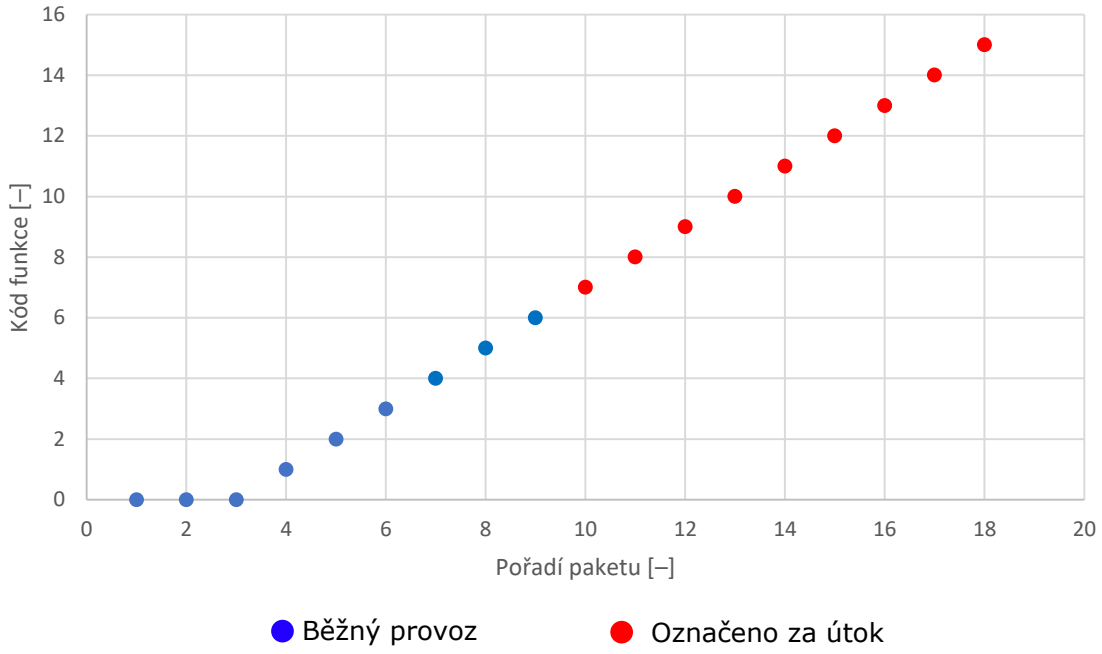
Na výpisu 3.10 je zobrazen log vygenerovaný pomocí IDS systému ZEEK. K detekování skenování podporovaných kódu funkce byl využit skript *known-masters-slaves*. Po spuštění experimentálního testování a provedení skenování kódu funkce je provedeno upozornění ve sloupci *popisek2* textem *SKEN* v případě, že je iterativně využit

kód funkce, a to sedmkrát a více krát po sobě. Iterativní chování je reprezentováno textem *iterace* ve sloupci *popisek*. Řádek 6 a 7 ve výpisu vyznačuje uložení adresy Master a Slave zařízení.

Výpis 3.10: Detekce prováděného skenování kódů funkce.

```
1  #path known_modbus
2  #open 2020-04-01-09-41-25
3  #fields ts         host    popisek fc    popisek2
4  #types  time       addr    string  count string
5  1585759285.629042 #addr1  OK      0     (empty)
6  1585759285.629042 #addr1  MASTER  -     -
7  1585759285.629042 #addr2  SLAVE   -     -
8  1585759285.660939 #addr1  ITERACE 1     (empty)
9  1585759285.682069 #addr1  ITERACE 2     (empty)
10 1585759285.725588 #addr1  ITERACE 3     (empty)
11 1585759285.768793 #addr1  ITERACE 4     (empty)
12 1585759285.792416 #addr1  ITERACE 5     (empty)
13 1585759285.819813 #addr1  ITERACE 6     (empty)
14 1585759285.862380 #addr1  ITERACE 7     SKEN
15 1585759285.868764 #addr1  ITERACE 8     SKEN
16 1585759285.906584 #addr1  ITERACE 9     SKEN
17 1585759285.927641 #addr1  ITERACE 10    SKEN
18 1585759285.952356 #addr1  ITERACE 11    SKEN
19 1585759285.973768 #addr1  ITERACE 12    SKEN
20 1585759286.012793 #addr1  ITERACE 13    SKEN
21 1585759286.030020 #addr1  ITERACE 14    SKEN
22 1585759286.060853 #addr1  ITERACE 15    SKEN
```

Na obr. 3.5 je zobrazen záznam logu na výpisu 3.10. Červeně vyznačené body reprezentují text *sken* na výpisu 3.10.



Obr. 3.5: Vizualizace logu skenování kódu funkce.

3.3 DNP3

3.3.1 Simulace bezpečnostních incidentů

Nmap

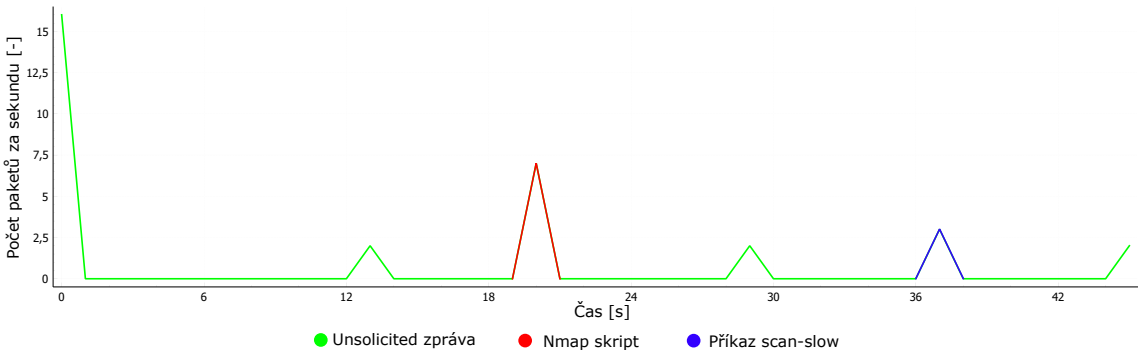
Chod protokolu může být ohrožen pomocí skriptu pro nástroj *nmap*. Skript [42] se dotáže Outstation zařízení na podporu DNP3 protokolu. Poté vysílá požadavek na prvních 100 adres v registru. Ke spuštění skriptu postačuje znalost IP adresy Outstation zařízení viz výpis 3.11.

Výpis 3.11: Zachycený útok pomocí druhé metody.

```
1 nmap --script dnp3-script -p 20000 #addr2
```

Na obr. 3.6 je zobrazena závislost počtu paketů za sekundu na čase protokolu DNP3 při provedené simulaci bezpečnostního incidentu. První vrchol zobrazuje spuštění Master stanice. Zelené vrcholy znázorňují komunikaci způsobenou změnou hodnoty v Outstation zařízení (zpráva unsolicited). Modře zvýrazněný vrchol znázorňuje čtení hodnot z Outstation zařízení pomocí příkazu *scan-slow*. Červený vrchol zobrazuje DNP3 komunikaci způsobenou spuštěním nmap skriptu. Dochází ke čtení velké části paměti. Samotné čtení je předcházeno testováním na dostupnost různých služeb, dochází tak k výraznému zatížení zařízení skrze TCP komunikaci (na grafu není tato komunikace znázorněna).

Během experimentálního testování bylo přeneseno 1 997 TCP paketů a 94 DNP3 paketů. Z důvodu spuštěného nmap příkazu byl proveden pokus o navázání 1 006 spojení. Průměrná velikost 98,20 % zpráv byla 64,07 B, resp. 65 B. Při provedení totožné komunikace bez spuštění nmap byla průměrná velikost 52,54 % zpráv 64,58 B, resp. 65 B a 34 % komunikace tvořily pakety o průměrné velikosti 83,30 B, resp. 84 B. Počet TCP spojení bylo pouze jedno pro DNP3.

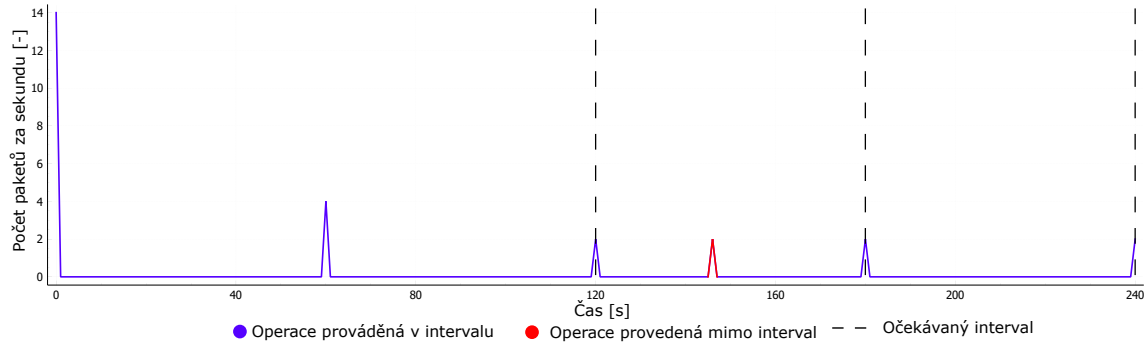


Obr. 3.6: Nmap na protokol DNP3.

Operace mimo interval

Dalším typem útoku, je odstranění přenášené zprávy z Master na Slave zařízení, popřípadě opačným směrem. Tato zpráva může být ze sítě úplně odstraněna (zachycena MITM a nepřeposlána dále), popřípadě může být zpožděna. Případně může být vygenerován dotaz na stav a hodnoty registru Slave zařízení mimo interval komunikace, který běžně využívá legitimní Master zařízení.

Simulaci tohoto bezpečnostního incidentu lze provádět spuštěním další instance Master skriptu *master.py*. Simulovaný bezpečnostní incident je zobrazen na obr. 3.7. Komunikace začíná spuštěním Master stanice a od času 120 s se v intervalu každých 60 s provádí pravidelná komunikace. V čase 146 s (červeně zvýrazněno) proběhla komunikace mimo pravidelně prováděný interval.



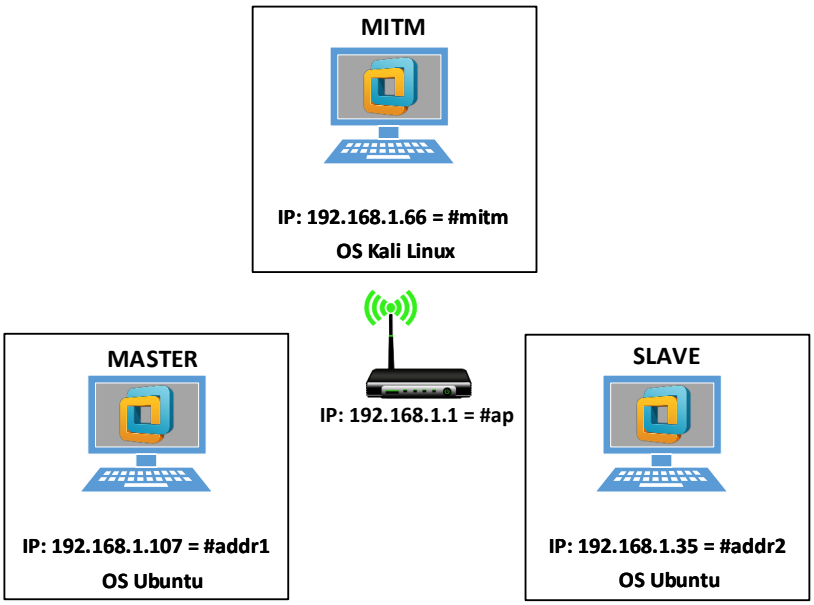
Obr. 3.7: Provedená operace mimo interval.

Man in the middle

Velmi častým útokem je ustanovení *Man in the middle*. Při ustanovení není komunikace mezi Master a Outstation přenášena přímo, ale přenášena přes prostředníka, který je schopen komunikaci číst, pozměnit nebo ze sítě úplně odstranit.

Aby bylo možné útok MITM realizovat, bylo využito zapojení, viz obr 3.8. Do dosavadní komunikace je přímo zapojen útočník, přes kterého je směrována komunikace. Ke vzájemné komunikaci bylo použito bezdrátové připojení IEEE 802.11n.

Provedení útoku MITM bylo provedeno pomocí předinstalované aplikace Ettercap, která provede ARP spoofing. Jsou tak pozměněny ARP záznamy jak na straně Master zařízení, tak na straně Outstation zařízení. Pozměněný ARP záznam, získaný z Outstation viz výpis 3.12.



Obr. 3.8: Pozměněné síťové zapojení.

Výpis 3.12: Změna v ARP záznamech.

```
1 student@ubuntu:~$ ip neigh
2 #ap dev ens33 lladdr 04:d9:f5:ed:7b:a8 STALE
3 #mitm dev ens33 lladdr c4:17:fe:22:c5:a2 STALE
4 #addr1 dev ens33 lladdr 28:c6:3f:40:af:c4 STALE
5 student@ubuntu:~$ ip neigh
6 #ap dev ens33 lladdr 04:d9:f5:ed:7b:a8 STALE
7 #mitm dev ens33 lladdr c4:17:fe:22:c5:a2 STALE
8 #addr1 dev ens33 lladdr c4:17:fe:22:c5:a2 REACHABLE
```

3.3.2 Navržená opatření

Nmap

Možnost detekce použití nmap je možné provést pomocí detekce velkého počtu pokusů o navázání spojení na různé porty. Případně lze provádět detekce zprávy, které bylo využito pomocí nmap skriptu. Tato zpráva je označována kódem funkce *0x0c HEX, 12 DEC*, tedy funkce *Freeze at time – no response*. Tato funkce způsobí, že jsou data uložena do oddělené paměti zařízení a potvrzení této operace není provedeno. Outstation následně zašle tyto informace, protože je tím vygenerována změna v paměti zařízení. Jakákoli změna v paměti Outstation zařízení způsobí, že jsou tyto informace odeslány do Master zařízení pomocí zprávy *Unsolicited Response*.

Operace mimo interval

K zachycení zápisu mimo běžný interval je nutné využít mezi-rámcovou mezeru δ. Za účelem vytvoření průměrné mezi-rámcové mezery Δ je zapotřebí přijetí několika (x) zpráv, viz rovnice 3.4. Po vytvoření průměrné mezi-rámcové mezery Δ ji lze porovnat s momentální mezi-rámcovou mezerou δ. V případě, že momentální mezera je menší než průměrná (v určitém intervalu definovaném pomocí parametru t $\in$ <0, 1>), dojde ke splnění rovnice 3.5. Splnění znamená, že komunikace nastala dříve, než je pro tuto komunikaci běžné. Obdobně rovnice 3.6 definuje stav, kdy zpráva dorazila později, než je běžné. Interval je upraven pomocí hodnoty u $\in$ <0, 1>.

$$\Delta = \frac{\sum_{n=1}^{x} \delta}{x} \quad [\text{s}] \qquad (3.4)$$

$$\delta \leq \Delta \cdot (1 - t) \quad [–] \qquad (3.5)$$

$$\delta \geq \Delta \cdot (1 + u) \quad [–] \qquad (3.6)$$

Man in the middle

K zachycení man in the middle útoku lze využít například RTT delay. V případě, že RTT bude navýšeno, může se jednat o ukazatel na případný výskyt útočníka v síti. K vytvoření prahové hodnoty je využito n RTT. Z těchto n zpráv je vybráno maximum. Aby detekce nebyla příliš striktní, je možné upravit prahovou hodnotu pomocí parametru k [%], viz rovnice 3.7. Pokud bude tato prahová hodnota překročena je vyhlášen poplach.

$$\Delta = k \cdot Max(rtt_1, rtt_2 \ldots rtt_n) \quad [\text{s}] \qquad (3.7)$$

3.3.3 Realizace opatření

V případě protokolu DNP3 jsou již ve výchozím stavu využívány základní skripty v IDS Zeek. Jsou uloženy v adresáři */base/protocols/dnp3* a není proto nutno cestu k těmto souborům definovat.

Nmap

K detekci funkce *Freeze at time* byl využit skript *main*, který prováděl detekci využití této metody. Další metodou, pomocí které lze detekovat tento útok je určení počtu

registrů, které jsou běžně vyčítány a v případě překročení limitu vyhlásit poplach. V tomto případě jde převážně o způsobení odepření služby než o data samotná, proto není nutné, aby se skripty, které provádí tuto činnost chovaly korektně. Na výpisu 3.13 je toto neadekvátní chování zachyceno pomocí systému Zeek. Tento provoz byl zachycen pomocí skriptu k zachytávání tohoto nestandardního provozu a výsledek uložen v logu *weird*.

Výpis 3.13: Detekce nmap zaměřeného na protokol DNP3.

```
1 #path weird
2 #open 2020-03-25-05-33-09
3 #fields ts      id.orig_h id.orig_p id.resp_h id.resp_p name    peer
4 #types  time    addr      port      addr      port      string  string
5 1585139589.354844 #addr1  40815 #addr2  20000 bad_TCP_checksum        zeek
6 1585139619.227238 #addr1  45156 #addr2  20000 data_before_established zeek
7 1585139619.228618 #addr1  45156 #addr2  20000 inappropriate_FIN       zeek
```

Operace mimo interval

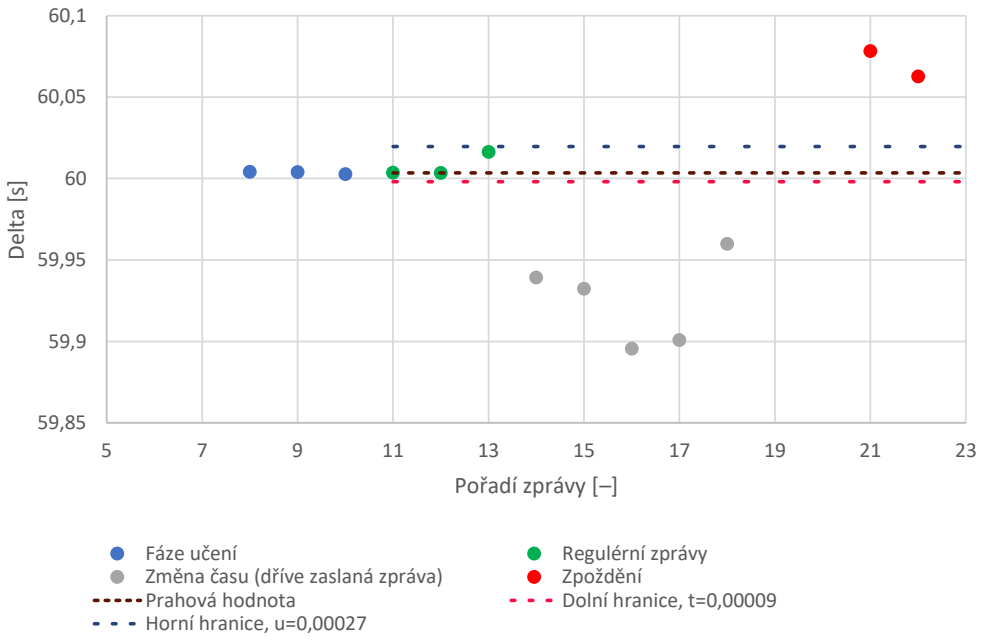
K aplikovaní vytvořených rovnic byl vybrán skript *main*, který byl rozšířen o metodu *dnp3-application-request-header*. V této metodě dochází k zaznamenávání jednotlivých zpráv. Parametr t v rovnici 3.5 byl nastaven na hodnotu 0,00009. Parametr u v rovnici 3.6 byl nastaven na hodnotu 0,00027. Nastavení těchto parametrů vyplynulo z experimentálního testování, kde byly tyto parametry nastaveny tak, aby oblast přijetí zpráv byla co nejužší, ale zároveň nedocházelo k falešným poplachům. Nejprve je proveden proces učení, parametr x byl v případě experimentálního testování nastaven na hodnotu x = 3. Z důvodu konstantní doby, za kterou je generována další zpráva je velikost tohoto parametru postačující. K vytváření průměrné prahové hodnoty Δ jsou využívány jen operace *READ* (kód funkce 1).

Aby nedošlo k chybnému nastavení průměrné prahové hodnoty jsou vynechány první 4 zprávy operace *READ*. Tato hodnota vychází z počtu zpráv provedených při spuštění skriptu na straně Master zařízení a intervalem δ se výrazně liší od běžného intervalu. Výsledná prahová hodnota je tedy naučena po zaslaní 7 zpráv *READ*. Výpis 3.14 zobrazuje zachycené zprávy zaslané pod vytvořenou prahovou hodnotou (červeně zvýrazněno) a zachycení zpráv přesahující prahovou hodnotu (oranžově zvýrazněno). Naučená prahová hodnota, viz řádek 15, sloupec *prumerny odstup*.

Výpis 3.14: Detekce simulovaného útoku, DNP3, využita rovnice 3.5.

```
#path dnp3
#open 2020-04-29-11-04-36
#fields odstup  prumerny_odstup popisek ip_zdroj  ip_cil  func  fc_request
#types  interval  interval  string  addr  addr  count string
0          0          Prijato     #addr1  #addr2  21  DISABLE_UNSOLICITED
0          0          Prijato     #addr1  #addr2  0   CONFIRM
0          0          Prijato     #addr1  #addr2  1   READ
0          0          Prijato     #addr1  #addr2  20  ENABLE_UNSOLICITED
0,052331   0          Prijato     #addr1  #addr2  1   READ
0,054173   0          Prijato     #addr1  #addr2  1   READ
60,016949  0          Prijato     #addr1  #addr2  1   READ
60,004151  0          Prijato     #addr1  #addr2  1   READ
60,003936  0          Prijato     #addr1  #addr2  1   READ
60,002816  0          Prijato     #addr1  #addr2  1   READ
60,00363   60,003461  Prijato     #addr1  #addr2  1   READ
60,003463  60,003461  Prijato     #addr1  #addr2  1   READ
60,016407  60,003461  Prijato     #addr1  #addr2  1   READ
59,939231  60,003461  Zmena casu!  #addr1  #addr2  1   READ
59,932317  60,003461  Zmena casu!  #addr1  #addr2  1   READ
59,895530  60,003461  Zmena casu!  #addr1  #addr2  1   READ
59,900808  60,003461  Zmena casu!  #addr1  #addr2  1   READ
59,959835  60,003461  Zmena casu!  #addr1  #addr2  1   READ
60,078300  60,003461  Zpozdeni!    #addr1  #addr2  1   READ
60,062700  60,003461  Zpozdeni!    #addr1  #addr2  1   READ
```

Na obr. 3.9 je zobrazen log na výpisu 3.14. Proces „učení“ je znázorněn modrými body. Po zaslání x = 3 zpráv dochází k vypočtení horní a dolní hranice, které budou uplatňovány pro posouzení, zda příchozí paket není přijat dříve nebo později, než je obvyklé. Červeně vynesené body hranici překračují (zprávy jsou zaslány poději), šedě vynesené jsou zaslány pod vytvořenou hranicí.



Obr. 3.9: Vizualizace logu, DNP3.

Man in the middle

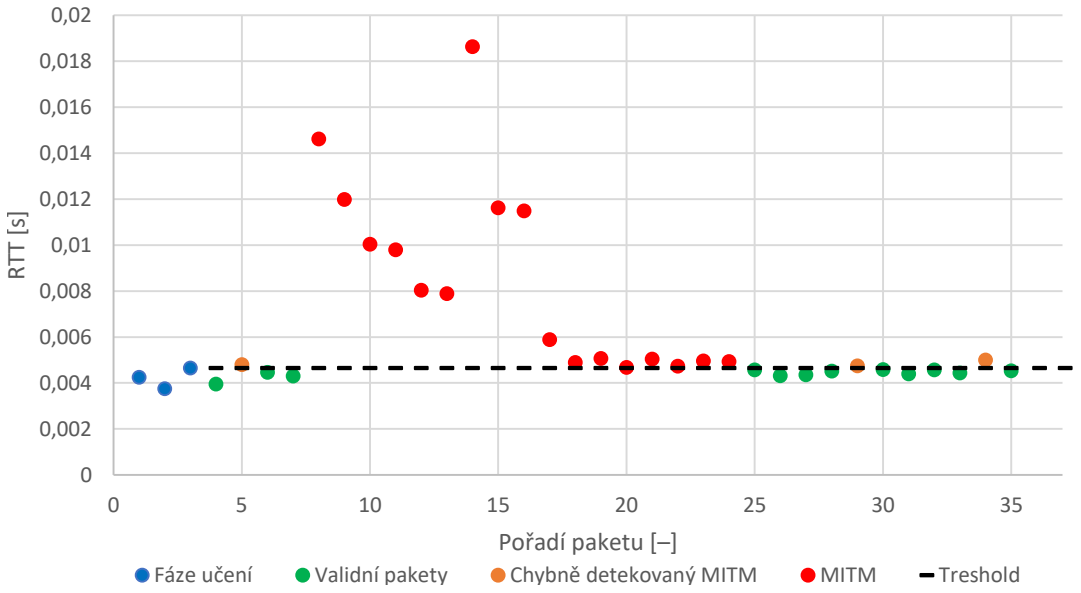
Pro detekci zpoždění byla vybrána funkce *Direct Operate*. Tato funkce byla vybrána z důvodu, že na ni Outstation zařízení reaguje okamžitě a velikost zprávy je konstantní. U lišících se velikostí zpráv by se RTT mohlo „proměnlivě“ měnit na základě operací, které je třeba vykonat, než bude provedena odpověď. U této operace se využívá potvrzení přijetí odpovědi pomocí TCP protokolu (TCP-ACK). Master stanice zašle příkaz *Direct operate*, od této doby probíhá počítání RTT. Poté Outstation odpoví, po příchodu DNP3 reply na toto reaguje Master stanice potvrzením příchodu DNP3 reply pomocí TCP-ACK. Po příjmu se výpočet RTT delay zastaví.

Získaný log viz výpis 3.15. Parametr n v rovnici 3.7 je nastaven na hodnotu 3, viz řádek 7 sloupec *RTT-treshold*. Velikost tohoto parametru se jeví jako dostačující z důvodu nízkého kolísání mezi jednotlivými RTT. Po spuštění útoku je zřetelné navýšení doby RTT. Pro testovací účely byl parametr k nastaven na hodnotu 1. Jedná se o nejpřísnější nastavení, ale dochází k falešným poplachům, viz řádek 9.

Výpis 3.15: Detekce rozdílného RTT.

```
1  dnp3
2  2020-04-30-10-53-13
3  ts          text     RTT       RTT-treshold  func    func_text
4  time        string   interval  interval      count   string
5  1588269194  Ucim se  0,004246  -           5 "DIRECT_OPERATE "
6  1588269194  Ucim se  0,003745  -           5 "DIRECT_OPERATE "
7  1588269195  prijato  0,004649  0,004649    5 "DIRECT_OPERATE "
8  1588269196  prijato  0,003952  0,004649    5 "DIRECT_OPERATE "
9  1588269196  mitm     0,004794  0,004649    5 "DIRECT_OPERATE "
10 1588269197  prijato  0,004468  0,004649    5 "DIRECT_OPERATE "
11 1588269198  prijato  0,004303  0,004649    5 "DIRECT_OPERATE "
12 1588269237  mitm     0,014615  0,004649    5 "DIRECT_OPERATE "
13 1588269237  mitm     0,011986  0,004649    5 "DIRECT_OPERATE "
14 1588269238  mitm     0,010034  0,004649    5 "DIRECT_OPERATE "
15 1588269238  mitm     0,009800  0,004649    5 "DIRECT_OPERATE "
16 1588269239  mitm     0,008037  0,004649    5 "DIRECT_OPERATE "
17 1588269239  mitm     0,007887  0,004649    5 "DIRECT_OPERATE "
18 1588269252  mitm     0,018633  0,004649    5 "DIRECT_OPERATE "
19 1588269262  mitm     0,011619  0,004649    5 "DIRECT_OPERATE "
20 1588269372  mitm     0,011486  0,004649    5 "DIRECT_OPERATE "
21 1588269409  mitm     0,005883  0,004649    5 "DIRECT_OPERATE "
22 1588269417  mitm     0,004889  0,004649    5 "DIRECT_OPERATE "
23 1588269418  prijato  0,004067  0,004649    5 "DIRECT_OPERATE "
```

Obr. 3.10 zobrazuje vizualizovaný log. Nejprve je provedena fáze učení, kde se využije prvních n = 3 zpráv (modře zvýrazněno). Z důvodu jen velmi nízkého kolísání RTT je tato hodnota postačující. Všechny RTT následujících paketů jsou porovnávány s prahovou hodnotou. Při vyšší hodnotě je vyvolán poplach. Z důvodu, že byl parametr k = 1, dochází k falešným poplachům (oranžově zvýrazněno). Hodnota tohoto parametru by mohla být navýšena na hodnotu 1,03, tak by k falešným poplachům nedošlo. Vytvořená pravidla je možné upravit na vyhlášení poplachu až po několikátém překročení RTT.



Obr. 3.10: Vizualizace logu, využita rovnice 3.7.

3.4 Využití strojového učení

Využití strojového učení v oblasti detekce bezpečnostních incidentů narůstá [43, 44]. Strojové učení (ML – Machine Learning) lze rozdělit na několik hlavních přístupů:

- Strojové učení s učitelem – data opatřena *labelem*.
  - Regrese
  - Klasifikace
- Strojové učení bez učitele – systém založen na *shlukování*.
  - Clustering
  - Dimension Reduction
- Zpětnovazební učení – systém založen na *odměnách*.
  - Model Free
  - Model Based

K vyhledávání anomálií je vhodné využít strojové učení bez učitele, protože není třeba označovat provoz za anomálii předem (label) a mohou tak být „nalezeny“ jakékoli anomálie. V případě, že má strojové učení „nalézt“ definovanou anomálii je vhodné využít strojové učení s učitelem. Data jsou tak předem označena pomocí *labelu*. Jednotlivé přístupy se dále dělí na hlavní kategorie. Jako zástupce strojového učení s učitelem pracující s klasifikací lze uvést například Open source nástroj *Weka*. Zástupce strojového učení bez učitele pracující se shluky dat (Clustering) lze uvést například python implementaci strojového učení *scikit-learn*.

Nástroj *Weka* podporuje několik modelů [45] k využití strojového učení s učitelem. Rozhodovací strom využívá k predikci binární strom určen ke klasifikaci. V každém uzlu je provedeno testování vstupu, jehož výsledkem je výběr dílčí větve

podstromu [46]. Reprezentace rozhodovacího stromu jsou klasifikační pravidla. Rozhodovací tabulku tvoří jednotlivá rozhodovací pravidla a může být vytvořena na základě rozhodovacího stromu [47]. Klasifikace založená na entropii je v nástroji Weka reprezentována jednoúrovňovým rozhodovacím stromem. [48]. Model *Knn* využívá k rozhodnutí o zařazení prvku do třídy členství okolních bodů [48].

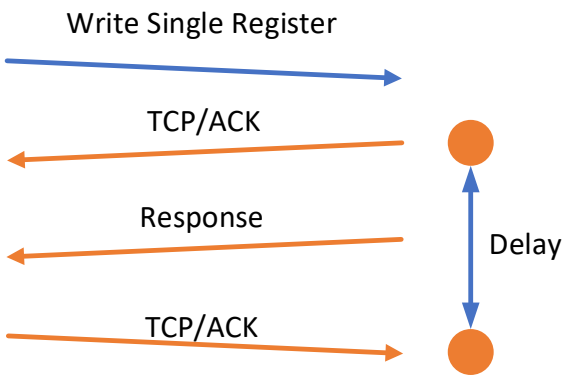
Strojové učení bez učitele za použití *scikit-learn* s využitím implementace *OneClassSVM* umožňuje proložení dat pomocí různých jader (kernelů) [49]. Kernel definuje, jakým způsobem bude použito proložení dat. Například u kernelu „poly“ jsou data proložena pomocí polynomu.

3.4.1 Modbus

Weka, Man in The Middle

K práci se strojovým učením byl vybrán open source nástroj pro strojové učení *Weka*. Pomocí tohoto nástroje bylo vytvořeno několik predikčních modelů založených na strojovém učení s učitelem. Jednotlivá data tak bylo nutné nejprve označit identifikátorem (label), který definuje, zda jednotlivý záznam reprezentuje běžné chování nebo útok. Takto označená data byla pomocí cross validace rozdělena na trénovací a testovací množiny.

Model strojového učení byl vytvořen k rozpoznávání útoku MITM v síti. K vygenerování dat bylo využito skriptu v rámci IDS Zeek. Vytvořený log pracoval s operací Write Single Register, kde bylo zaznamenáváno zpoždění. Konkrétně byla měřena doba mezi jednotlivými ACK zprávami. Zpoždění je počítáno od přijetí WSR operace, následně je odesláno potvrzení o přijetí zprávy (TCP/ACK) následované Modbus odpovědí. Jakmile je tato zpráva přijata Mater stanicí, vygeneruje se potvrzení o přijetí (TCP/ACK). Po přijetí tohoto potvrzení Slave zařízením je výpočet zpoždění ukončen, viz obr. 3.11. Dále bylo využito mezi-rámcové mezery. Pro vytvoření modelu bylo využito 732 záznamů, kde polovinu (tj. 366) záznamů tvořila data označena labelem „0“ představující legitimní provoz. Druhou polovinu záznamů tvořila data označena labelem „1“, tedy provoz, při kterém došlo k ustanovení MITM. Průměrný RTT delay byl v případě legitimních dat 0,13750 s a v případě MITM 0,16715 s. Rozdíl mezi těmito zprávami tak tvoří přibližně 0,03 s. K vytvoření MITM bylo využito síťového zapojení na obr. 3.8, ale jednotlivé stroje byly propojeny pomocí přepínače. Zpoždění je tak ještě více zřetelné. K ustanovení MITM byla použita aplikace Ettercap.



Obr. 3.11: Počítání zpoždění v komunikaci u Modbus/TCP protokolu.

Tab. 3.1 zobrazuje porovnávání vybraných modelů z hlediska výsledné úspěšnosti a doby učení. Největší úspěšnosti dosáhl model *Random Forest* s úspěšností 95,77 %.

Tab. 3.1: Detekce bezpečnostních incidentů.

Model	Popis	Úspěšnost modelu	Doba učení
J48	Rozhodovací strom	95,08 %	304 ms
DecisionTable	Rozhodovací tabulka	95,22 %	130 ms
DecisionStump	Klasifikace založená na entropii	92,76 %	38 ms
NaiveBayes	Analýza trénovacích dat	89,62 %	48 ms
RandomForest	Rozhodovací strom	95,77 %	313 ms
RandomTree	Rozhodovací strom, zvažuje K atributů v každém uzlu	94,95 %	70 ms
IBk	Knn	95,49 %	826 ms
REPTree	Rychlý rozhodovací strom	95,63 %	85 ms

3.4.2 DNP3

OneClassSVM, operace mimo interval

Základem pro detekci anomálií byl projektu [50], který byl následně upraven. Pro detekci anomálií v síťovém provozu bylo využito detekce operace mimo interval. Strojové učení je využito pomocí Python implementace OneClassSVM, jedná se o detekci anomálií bez učitele. OneClassSVM využívá python implementaci strojového učení *scikit-learn*. K vytvoření modelu byl využit kernel *poly* s nastavením hodnoty *nu* na 0,53 a hodnota *gamma* na 0,1 a hodnota „tolerance“ na 0,0001 zobrazeno na výpisu 3.16.

Výpis 3.16: Detekce anomálií, OneClassSVM.

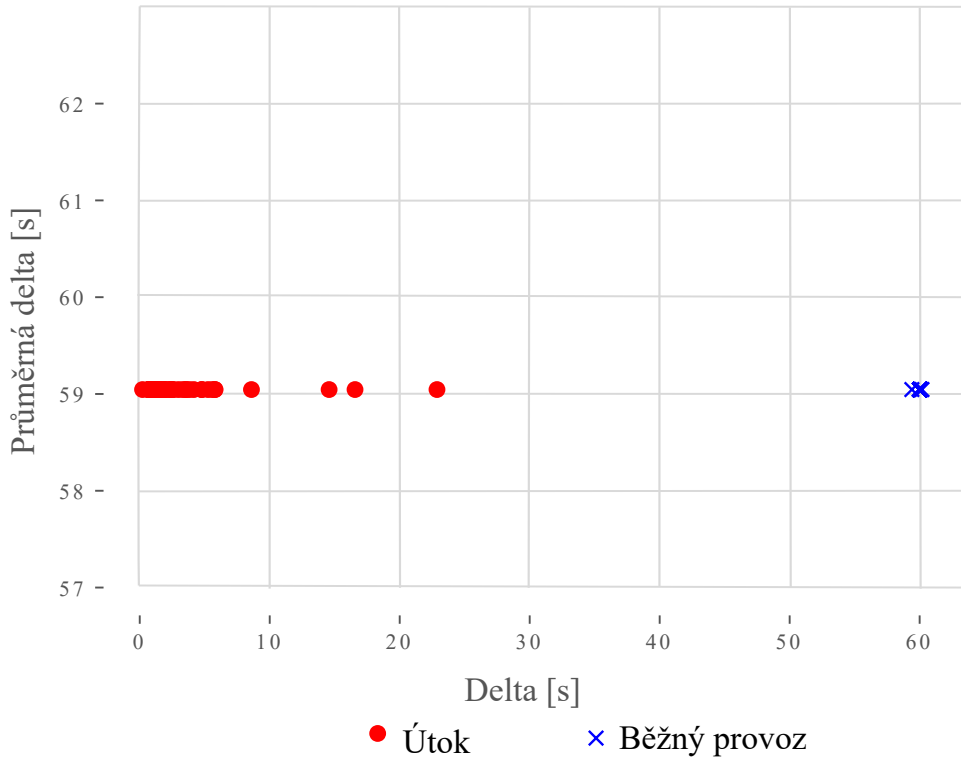
```
1 clf = svm.OneClassSVM(nu=0.53, kernel="poly", gamma=0.1, tol=0.0001)
2 clf.fit(tr_data)
```

K vytvoření modelu a detekování anomálií byly využity data získaná během testování detekce operace mimo interval. K provedenému testování byl použit malý

vzorek dat, 55 vzorků splňovalo požadavek a pakety byly přijaty v rámci prahové hodnoty. Dalších 64 vzorků naopak prahovou hodnotu překračovalo, resp. nedosahovalo (anomálie). Použitá data byla zbavena počáteční fáze, kdy dochází ke prvotnímu spuštění a postupnému vytvoření prahové hodnoty.

K vytvoření modelu byla nejprve využita dvojice momentálního zpoždění (δ) a „application control“, ale model nesplňoval požadavky. Nebylo možné rozpoznat anomálie, aniž by byly některá z relevantních dat označena jako anomálie. Bylo tedy nutné vybrat jiný z provozních parametrů, které byly získány pomocí IDS systému Zeek.

Na obr. 3.12 je zobrazen výstup po aplikování vytvořeného modelu na poskytnutá data. V grafu je na ose „x“ vynesena hodnota delta (δ) a na ose „y“ vytvořená průměrná delta (Δ) získaná po fázi „učení“. Červeně vynesené body, jsou považovány za anomálie síťového provozu, modré body za „normální“ data. V testovaném vzorku dat odpovídala δ hodnotě 60,013187 s. Vytvořený model byl díky velkému rozdílu mezi-rámcových mezer „běžných“ paketů a paketů zaslaných mimo interval, schopen tyto dvě skupiny přesně odlišit.



Obr. 3.12: Detekování anomálií pomocí strojového učení.

Weka, Man in The Middle

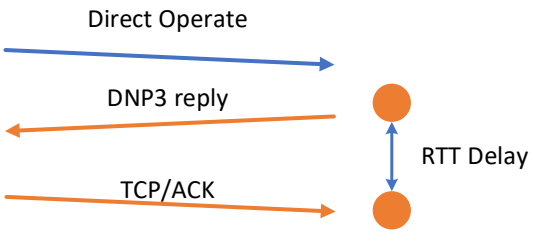
V rámci protokolu DNP3 bylo také využito open source nástroj pro strojové učení *Weka*, který byl také vybrán pro detekci MITM v síti. Zacházení je totožné jako v případě Modbus protokolu. Je vytvořeno několik predikčních modelů s učitelem. Vstupní data bylo proto třeba nejprve opatřit labelem. Trénovací a testovací data byla vygenerována pomocí skriptu v rámci IDS Zeek. Vytvořený log zaznamenal jednotlivé operace. RTT delay byl zpracováván u operace *Direct Operate*. Po přijetí této zprávy se začíná počítat RTT delay, poté dochází k vygenerování DNP3 reply zprávy. Výpočet RTT delay je ukončen po přijetí TCP/ACK zprávy, viz obr. 3.13.

První přístup pracoval pouze s RTT hodnotami, které byly označeny *labelem*. Během testování bylo nasbíráno 1 468 dat. Polovinu z těchto dat (tj. 734 záznamů) bylo označeno labelem „0“ představující legitimní provoz a druhá polovina označena labelem „1“ označující probíhající MITM útok. Data označena „0“ měla průměrnou hodnotu 0,00224 s a data označena „1“ 0,02388 s. Rozdíl mezi těmito skupinami je přibližně 0,022 s. Tab. 3.2 zobrazuje porovnávání vybraných modelů z hlediska výsledné úspěšnosti a doby učení. Model *J48* dosáhl úspěšnosti 99,93 %, zbytek modelů dosáhl úspěšnosti 100 % doba učení je brána jako doba trénování bez testování.

Tab. 3.2: Detekce bezpečnostních incidentů.

Model	Popis	Úspěšnost modelu	Doba učení
J48	Rozhodovací strom	99,93 %	138 ms
DecisionTable	Rozhodovací tabulka	100,00 %	303 ms
DecisionStump	Klasifikace založená na entropii	100,00 %	47 ms
NaiveBayes	Analýza trénovacích dat	100,00 %	70 ms
RandomForest	Rozhodovací strom	100,00 %	285 ms
RandomTree	Rozhodovací strom, zvažuje K atributů v každém uzlu	100,00 %	45 ms
IBk	Knn	100,00 %	351 ms
REPTree	Rychlý rozhodovací strom	100,00 %	62 ms

Druhý přístup byl upraven tak, aby pracoval i s mezi-rámcovou mezerou mezi jednotlivými zprávami. Byly tak za využití vytvořeného skriptu implementovaného v IDS Zeek vygenerována nová data. K vytvoření modelu bylo využito RTT delay u operace *Direct Operate* a odpovídající mezi-rámcová mezera. Pro vytvoření modelu bylo využito 1 136 záznamů, kde polovinu (tj. 568) záznamů tvořila data označena labelem „0“ představující legitimní provoz. Druhou polovinu záznamů tvořila data označena labelem „1“, tedy provoz, při kterém došlo k ustanovení MITM. Rozdělení na trénovací a testovací data bylo provedeno pomocí metody *cross validace*. Průměrný RTT delay byl v případě legitimních dat 0,00705 s a v případě MITM 0,02545 s. Rozdíl mezi těmito zprávami tak tvoří přibližně 0,0184 s. K vytvoření MITM bylo využito síťového zapojení na obr. 3.8, ale jednotlivé stroje byly propojeny pomocí přepínače. Zpoždění je tak ještě více zřetelné. K ustanovení MITM byla použita aplikace Ettercap.



Obr. 3.13: Počítání zpoždění v komunikaci pomocí DNP3 protokolu.

Tab. 3.3 zobrazuje porovnávání vybraných modelů z hlediska výsledné úspěšnosti a doby učení. Největší úspěšnosti dosáhl model *REPTree* a model *J48* shodně s úspěšností 97,89 %. Doba učení je brána jako doba trénování bez následného testování.

Tab. 3.3: Detekce bezpečnostních incidentů pomocí strojového učení.

Model	Popis	Úspěšnost modelu	Doba učení
J48	Rozhodovací strom	97,89 %	221 ms
DecisionTable	Rozhodovací tabulka	97,27 %	333 ms
DecisionStump	Klasifikace založená na entropii	97,80 %	40 ms
NaiveBayes	Analýza trénovacích dat	96,13 %	129 ms
RandomForest	Rozhodovací strom	96,65 %	159 ms
RandomTree	Rozhodovací strom, zvažuje K atributů v každém uzlu	97,10 %	64 ms
IBk	Knn	96,21 %	249 ms
REPTree	Rychlý rozhodovací strom	97,89 %	202 ms

Z důvodu použití metalického vedení dosahuje RTT delay více konstantního zpoždění v případě, že dochází k ustanovení MITM, než v případě bezdrátového přenosu. Metoda strojového učení se tak jeví k odhalení ustanovení MITM jako vhodná, naučené modely dosahují vysoké úspěšnosti. Použití druhého přístupu s použitím mezi-rámcové mezery dosahuje nižší úspěšnosti v porovnání s prvním přístupem, kde je využito jen RTT delay, druhý přístup se ale zaměřuje na více parametrů, které mohou pomoci identifikovat útočníka v síti.

Závěr

Tato diplomová práce byla zaměřena na problematiku bezpečnosti průmyslových protokolů, zejména na průmyslový protokol Modbus/TCP a protokol DNP3. V první kapitole byly nejprve popsány vybrané průmyslové komunikační protokoly. Následně byla popsána bezpečnost jednotlivých průmyslových protokolů. U protokolu Modbus/TCP a protokolu DNP3 byly identifikovány vektory útoku. Následně jsou popsány vektory útoku a protiopatření z obecného pohledu. Na základě vektorů útoku byly vybrány bezpečnostní incidenty s navrženými metodami detekce.

Druhá kapitola je zaměřena na přípravu experimentálního prostředí, instalaci potřebných nástrojů a knihoven k experimentálnímu testování. Jako IDS nástroj určený k detekci signatur byl vybrán nástroj Snort a k detekci anomálií nástroj Zeek. Kapitola se také zaměřuje na definování jednotlivých scénářů pro testování. Byla navržena a vytvořena experimentální síť, v rámci které dochází k implementaci vybraných protokolů Modbus/TCP a DNP3 a simulaci bezpečnostních incidentů.

Ve třetí kapitole jsou provedeny jednotlivé navržené scénáře. Nejprve je provedena detekce signatury v protokolu Modbus/TCP pomocí IDS systému Snort detekující nadměrný paket a nedefinované IP adresy. K detekci bylo nutné vytvořit pravidla, která filtrují příchozí provoz a testují definovaná kritéria. Vytvořená pravidla byla schopna úspěšně detekovat tyto bezpečnostní incidenty. Dále se tato kapitola zaměřuje na využití IDS nástroje Zeek, určeného k detekci anomálií v síťovém provozu u obou vybraných průmyslových protokolů. Nejprve bylo testování zaměřeno na protokol Modbus, u kterého jsou nejprve provedeny simulace bezpečnostních incidentů, jako je skenování UID, útok DoS a skenování podporovaných function kódů. Na základě analýzy jsou navrženy metody k detekci, které jsou následně implementovány a nasazeny v systému Zeek.

Následně je testování zaměřeno na protokol DNP3. Zde jsou nejprve simulovány bezpečnostní incidenty, jako útoku pomocí nástroje nmap, provedení operace mimo definovaný interval a detekce útoku man in the middle. Na základě analýzy jsou následně vytvořeny metody pro detekci, které jsou poté implementovány v IDS systému Zeek. Jako alternativní přístup k detekci anomálií v síťovém provozu bylo využito strojového učení s učitelem i bez. U protokolu Modbus/TCP k detekci útoku man in the middle a u protokolu DNP3 k detekci operace mimo interval a detekci útoku man in the middle. Vytvořené metody byly nasazeny na cílové stanici, kde docházelo k úspěšné detekci bezpečnostních incidentů. Většina vytvořených metod vyžaduje předcházející fázi „učení“, která je kritická na nepřítomnost útočníka v této fázi. Detekce jednotlivých incidentů byly reprezentovány formou vytvořeného logu. Jako rozšíření této práce je možné tyto metody nasadit na síťovou sondu, kde by docházelo k rozpoznávání bezpečnostních incidentů.

Literatura

[1] COLLANTES, Miguel Herrero a Antonio López PADILLA. Protocols and Network Security in ICS infrastructures. *The Spanish National Institute for Cyber-security* [online]. 2015, 2015, 39 [cit. 2020-05-28]. Dostupné z: https://www.incibe-cert.es/ sites/default/files/contenidos/guias/doc/incibe_protocol_net_security_ics.pdf

[2] VLČEK, Lukáš. Modbus: Úvod do protokolu. *Lukas-vlcek.cz* [online]. 2017 [cit. 2019-10-28]. Dostupné z: https://www.lukas-vlcek.cz/iot/modbus-uvod-do-protokolu/

[3] GASTREICH, Wally. What is Modbus? *Realpars.com* [online]. 2018 [cit. 2019-10-28]. Dostupné z: https://realpars.com/modbus/

[4] GASTREICH, Wally. How does Modbus Communication Protocol work? *Realpars.com* [online]. 2018 [cit. 2019-10-28]. Dostupné z: https://realpars.com/modbus-protocol/

[5] Detailed description of the Modbus TCP protocol with command examples. *Ipc2u.com* [online]. [cit. 2019-10-29]. Dostupné z: https://ipc2u.com/articles/knowledge-base/detailed-description-of-the-modbus-tcp-protocol-with-command-examples/

[6] Modbus 101 - Introduction to Modbus. *Csimn.com* [online]. [cit. 2019-10-29]. Dostupné z: https://www.csimn.com/CSI_pages/Modbus101.html

[7] SCADA MODBUS Protocol Vulnerabilities. *Cyberbit.com* [online]. 2017 [cit. 2019-10-29]. Dostupné z: https://www.cyberbit.com/blog/ot-security/scada-modbus-protocol-vulnerabilities/

[8] ICS Advisory (ICSA-17-101-01). *Us-cert.gov* [online]. 2017 [cit. 2019-10-29]. Dostupné z: https://www.us-cert.gov/ics/advisories/ICSA-17-101-01

[9] CVE-2017-6034 Detail. *Nvd.nist.gov* [online]. 2017 [cit. 2019-10-29]. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2017-6034

[10] Security alert 53425. *Cisco.com* [online]. [cit. 2019-09-10]. Dostupné z: https://tools.cisco.com/security/center/viewAlert.x?alertId=53425

[11] Cagalaban, Giovanni & So, Yohwan & Kim, Seoksoo. (0002). *SCADA Network Insecurity: Securing Critical Infrastructures through SCADA Security Exploitation.*

[12] I. EVANGELIA, Evangeliou. *VULNERABILITIES OF THE-MODBUS PROTOCOL* [online]. [cit. 2019-11-11]. Dostupné z: http://dione.lib.unipi.gr/xmlui/bitstream/handle/unipi/11394 /Evangeliou_1508.pdf?sequence=1&isAllowed=y

[13] GABRIEL, Sanchez. *Man-In-The-Middle AttackAgainst Modbus TCPIllustrated with Wireshark* [online]. [cit. 2019-11-11]. Dostupné z: https://www.sans.org/reading-room/whitepapers/ICS/man-in-the-middle-attack-modbus-tcp-illustrated-wireshark-38095

[14] DARWISH, Ihab, Obinna IGBE, Orhan CELEBI, Tarek SAADAWI a Joseph SORYAL. Smart Grid DNP3 Vulnerability Analysis and Experimentation. *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing* [online]. IEEE, 2015, 2015, , 141-147 [cit. 2020-05-28]. DOI: 10.1109/CSCloud.2015.86. ISBN 978-1-4673-9300-3.

[15] ROSBOROUGH, Clifford, Colin GORDON a Brian WALDRON. *All About Eve: Comparing DNP3Secure Authentication With Standard Security Technologies for SCADA Communications* [online]. 2019 [cit. 2020-05-06]. Dostupné z: https://ccaps.umn.edu/documents/CPE-Conferences/MIPSYCON-Papers/2019/AllAboutEve.pdf

[16] ICS Advisory (ICSA-13-291-01B). *Us-cert.gov* [online]. [cit. 2019-11-11]. Dostupné z: https://www.us-cert.gov/ics/advisories/ICSA-13-291-01B#footnotea_8rer6kt

[17] Security alert 36654. *Cisco.com* [online]. [cit. 2019-09-10]. Dostupné z: https://tools.cisco.com/security/center/viewAlert.x?alertId=36654]

[18] Security alert 30390. *Cisco.com* [online]. [cit. 2019-09-10]. Dostupné z: https://tools.cisco.com/security/center/viewAlert.x?alertId=30390]

[19] *EtherNet/IP Quick Start for Vendors Handbook*. 2008, (PUB00213R0).

[20] YUAN ZHOU, DAN CHAI, MINGSHAN LIU, FENGXUE LIN, WENDONG SHANG a LIANG WANG. Research on the security mechanism for interconnection between PROFIBUS and Internet. *Proceeding of the 11th World Congress on Intelligent Control and Automation* [online]. IEEE, 2014, 2014, , 5972-5976 [cit. 2020-05-30]. DOI: 10.1109/WCICA.2014.7053743. ISBN 978-1-4799-5825-2.

[21] *Computer Safety, Reliability, and Security: Exploring Network Security in PROFIsafe*. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04468-7.

[22] XU, Haiya, Yan GAO, Kemin LIU, Bo ZHU a Chao ZHANG. Research on cross-communication based on real-time Ethernet POWERLINK. *The 26th Chinese Control and Decision Conference (2014 CCDC)* [online]. IEEE, 2014, 2014, , 2577-2581 [cit. 2020-05-28]. DOI: 10.1109/CCDC.2014.6852608. ISBN 978-1-4799-3708-0.

[23] *Security of industrial control systems and cyber-physical systems* [online]. New York, NY: Springer Berlin Heidelberg, 2017 [cit. 2019-10-29]. ISBN 978-3-319-61436-6.

[24] SHIMANUKI, Y. OLE for process control (OPC) for new industrial automation systems. *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)* [online]. IEEE, 1999, , 1048-1050 [cit. 2020-05-28]. DOI: 10.1109/ICSMC.1999.816721. ISBN 0-7803-5731-0.

[25] KNAPP, Eric D. a Joel Thomas LANGILL. Industrial Network Protocols. *Industrial Network Security* [online]. Elsevier, 2015, 2015, , 121-169 [cit. 2020-05-28]. DOI: 10.1016/B978-0-12-420114-9.00006-X. ISBN 9780124201149.

[26] *EtherCAT and EtherCAT P Slave Implementation Guide: ETG.2200* [online]. 2018 [cit. 2019-10-29]. Dostupné z: https://www.ethercat.org/download/documents/ETG2200_V3i0i4_G_R_SlaveImplementationGuide.pdf

[27] CALDWELL, Tracey. Plugging IT/OT vulnerabilities — part 1. *Network Security* [online]. 2018, **2018**(8), 9-14 [cit. 2020-05-28]. DOI: 10.1016/S1353-4858(18)30078-3. ISSN 13534858.

[28] CALDWELL, Tracey. Plugging IT/OT vulnerabilities — part 2. *Network Security* [online]. 2018, **2018**(9), 10-15 [cit. 2020-05-28]. DOI: 10.1016/S1353-4858(18)30089-8. ISSN 13534858.

[29] FOVINO, Igor Nai, Marcelo MASERA, Luca GUIDI a Giorgio CARPI. An experimental platform for assessing SCADA vulnerabilities and countermeasures in power plants. *3rd International Conference on Human System Interaction* [online]. IEEE, 2010, 2010, , 679-686 [cit. 2020-05-28]. DOI: 10.1109/HSI.2010.5514494. ISBN 978-1-4244-7560-5.

[30] AMOAH, Raphael, Seyit CAMTEPE a Ernest FOO. Securing DNP3 Broadcast Communications in SCADA Systems. *IEEE Transactions on Industrial Informatics* [online]. 2016, **12**(4), 1474-1485 [cit. 2020-05-27]. DOI: 10.3233/JCS-181139. ISSN 1551-3203.

[31] CREMERS, Cas, Martin DEHNEL-WILD a Kevin MILNER. Secure authentication in the grid: A formal analysis of DNP3 SAv5. *Journal of Computer Security* [online]. 2019, **27**(2), 203-232 [cit. 2020-05-27]. DOI: 10.3233/JCS-181139. ISSN 18758924.

[32] BOU-HARB, Elias, Nasir GHANI, Abdelkarim ERRADI a Khaled SHABAN. Passive inference of attacks on CPS communication protocols. *Journal of Information Security and Applications* [online]. 2018, **43**, 110-122 [cit. 2020-05-28]. DOI: 10.1016/j.jisa.2018.10.002. ISSN 22142126.

[33] MODBUS/TCP Security: Protocol Specification. Modbus.org [online]. 2018 [cit. 2020-04-29]. Dostupné z: http://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf

[34] Overview of DNP3 Security Version 6. Dnp.org [online]. 2020 [cit. 2020-04-29]. Dostupné z: https://www.dnp.org/LinkClick.aspx?fileticket=hyvYMYugaQI%3d&tabid=66&portalid=0&mid=447&forcedownload=true

[35] Pymodbus 2.3.0. *Pypi.org* [online]. [cit. 2019-12-08]. Dostupné z: https://pypi.org/project/pymodbus/

[36] Pydnp3 0.1.0. *Pypi.org* [online]. [cit. 2019-12-08]. Dostupné z: https://pypi.org/project/pydnp3/

[37] Synchronous Client Example. *Pymodbus.readthedocs.io* [online]. [cit. 2019-12-08]. Dostupné z: https://pymodbus.readthedocs.io/en/v1.3.2/examples/synchronous-client.html

[38] Updating Server Example. *Pymodbus.readthedocs.io* [online]. [cit. 2019-12-08]. Dostupné z: https://pymodbus.readthedocs.io/en/v1.3.2/examples/updating-server.html

[39] Python bindings for opendnp3 libary. *Github.com* [online]. [cit. 2020-03-21]. Dostupné z: https://github.com/ChargePoint/pydnp3

[40] Snort. *Snort.org* [online]. [cit. 2019-12-08]. Dostupné z: https://www.snort.org/

[41] MODBUS Penetration Testing Framework. *Github.com* [online]. [cit. 2020-03-12]. Dostupné z: https://github.com/theralfbrown/smod-1

[42] Nmap-NSEs. *Github.com* [online]. [cit. 2020-03-21]. Dostupné z: https://github.com/sjhilt/Nmap-NSEs/blob/master/dnp3-info.nse

[43] SUABOOT, Jakapan, Adil FAHAD, Zahir TARI, John GRUNDY, Abdun Naser MAHMOOD, Abdulmohsen ALMALAWI, Albert Y. ZOMAYA a Khalil DRIRA. A Taxonomy of Supervised Learning for IDSs in SCADA Environments. *ACM Computing Surveys* [online]. 2020, **53**(2), 1-37 [cit. 2020-05-28]. DOI: 10.1145/3379499. ISSN 0360-0300.

[44] DING, Derui, Qing-Long HAN, Yang XIANG, Xiaohua GE a Xian-Ming ZHANG. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing* [online]. 2018, **275**, 1674-1683 [cit. 2020-05-28]. DOI: 10.1016/j.neucom.2017.10.009. ISSN 09252312.

[45] DESAI, Aaditya a Sunil RAI. *Analysis of Machine Learning Algorithms using Weka* [online]. 2013, , 7 [cit. 2020-05-28].

[46] LI, Yali, Ling WANG, Zhihong LIU, Chanjuan LI, Jiake XU, Qiong GU a Jun XU. Predicting selective liver X receptor β agonists using multiple machine learning methods. *Molecular BioSystems* [online]. 2015, **11**(5), 1241-1250 [cit. 2020-05-28]. DOI: 10.1039/C4MB00718B. ISSN 1742-206X.

[47] PAL, Saurabh. Early Prediction of Heart Diseases Using Data Mining Techniques. *Caribbean Journal of Science and Technology* [online]. 2013, (1), 10 [cit. 2020-05-28]. Dostupné z: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2991237

[48] ZHANG, Yudong, Siyuan LU, Xingxing ZHOU, Ming YANG, Lenan WU, Bin LIU, Preetha PHILLIPS a Shuihua WANG. Comparison of machine learning methods for stationary wavelet entropy-based multiple sclerosis detection: decision tree, k-nearest neighbors, and support vector machine. *SIMULATION* [online]. 2016, **92**(9), 861-871 [cit. 2020-05-28]. DOI: 10.1177/0037549716666962. ISSN 0037-5497.

[49] Unsupervised Outlier Detection: OneClassSVM. *Scikit-learn.org* [online]. 2019, 2019 [cit. 2020-05-28]. Dostupné z: https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

[50] Anomaly Detection. *Github.com* [online]. [cit. 2020-04-12]. Dostupné z: https://github.com/aqibsaeed/Anomaly-Detection

Seznam symbolů, veličin a zkratek

ICS	Industrial Control System
IT	Information Technology
OT	Operation Technology
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
DPCS	Discrete Process Control Systems
SCADA	Supervisory Control and Data Acquisition
HMI	Human Machine Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
LDAP	Lightweight Directory Access Protocol
NTP	Network Time Protocol
CAN	Controller Area Network
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
PDU	Protocol Data Unit
HDLC	High-level Data Link Control
ADU	Application Data Unit
DoS	Denial of service
CRC	Cyclic Redundancy Check
IDS/IPS	Intrusion Detection System/Intrusion Prevention System
AH	Application Header
TH	Transport Header
LH	data Link Header
APDU	Application Protocol Data Unit
ASDU	Application Service Data Unit
APCI	Application Protocol Control Info
TPDU	Transport Protocol Data Unit
LPDU	Link Protocol Data Unit
FDL	Fieldbus Data Link
SCNM	Slot Communication Network Management
MN	Management Node
CN	Controlled Node
OPC	OLE for Process Control
RPC	Remote Procedure Call
OPC-UA	OPC Unified Architecture

SOAP	Simple Object Access Protocol
VPN	Virtual Private Network
EtherCAT	Ethernet for Control Automation Technology
WSR	Write Single Register
DoS	Denial of Service
MITM	Man In The Middle
ARP	Address Resolution Protocol
ACK	Acknowledgement
RTT Delay	Round-Trip Time Delay
ML	Machine Learning