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

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INFORMAČNÍCH SYSTÉMŮ**
**DEPARTMENT OF INFORMATION SYSTEMS**


# STROJOVÉ UČENÍ Z DAT SYSTÉMŮ PRO DETEKCI SÍŤOVÉHO PRŮNIKU

**MACHINE LEARNING FROM INTRUSION DETECTION SYSTEMS**

**DIPLOMOVÁ PRÁCE**
**MASTER'S THESIS**


**AUTOR PRÁCE** **Bc. MICHAL DOSTÁL**
**AUTHOR**

**VEDOUCÍ PRÁCE** **Ing. RADEK HRANICKÝ, Ph.D.**
**SUPERVISOR**



**BRNO 2023**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání diplomové práce

148697

| | |
|---|---|
| Ústav: | Ústav informačních systémů (UIFS) |
| Student: | **Dostál Michal, Bc.** |
| Program: | Informační technologie a umělá inteligence |
| Specializace: | Počítačové sítě |
| Název: | **Strojové učení z dat systémů pro detekci síťového průniku** |
| Kategorie: | Bezpečnost |
| Akademický rok: | 2022/23 |

Zadání:

1. Nastudujte problematiku strojového učení se zaměřením na detekci anomálií v síťovém provozu.
2. Nastudujte problematiku systémů detekce a prevence průniku (IDS/IPS) a porovnejte funkcionalitu existujících open-source řešení.
3. Pro vhodně zvolený IDS/IPS systém navrhněte rozšíření, jež bude aplikovat metody strojového učení na zachycené síťové toky a detekované události.
4. Po konzultaci s vedoucím implementujte navržené úpravy a aplikaci strojového učení.
5. Na vhodně zvoleném (veřejně dostupném) vzorku dat demonstrujte použitelnost Vašeho přístupu a vytvořte odpovídající testy.
6. Zhodnoťte dosažené výsledky a navrhněte možná rozšíření implementovaných technik.

Literatura:


- McHugh, J., Christie, A., & Allen, J. (2000). Defending yourself: The role of intrusion detection systems. *IEEE software*, *17*(5), 42-51.
- Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, *28* (1-2), 18-28.


Při obhajobě semestrální části projektu je požadováno:
Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

| | |
|---|---|
| Vedoucí práce: | **Hranický Radek, Ing., Ph.D.** |
| Vedoucí ústavu: | Kolář Dušan, doc. Dr. Ing. |
| Datum zadání: | 1.11.2022 |
| Termín pro odevzdání: | 31.7.2023 |
| Datum schválení: | 31.10.2022 |

# Abstrakt

Aktuální stav nástrojů pro detekci síťového průniku je nedostačující, protože tyto nástroje často fungují na základě statických pravidel a nevyužívají potenciál umělé inteligence. Cílem této práce je rozšířit open-source nástroj Snort o schopnost detekovat škodlivý síťový provoz pomocí strojového učení. Pro dosažení kvalitního klasifikátoru byly zvoleny užitečné příznaky síťového toku, které byly získány z výstupních dat aplikace Snort. Následně byly tyto toky obohaceny a označeny odpovídajícími událostmi. Experimenty vykazují velmi dobré výsledky nejenom při klasifikaci na testovacích datech, ale také v rychlosti zpracování. Z navrženého přístupu a samotných experimentů vyplývá, že tento nový přístup by mohl vykazovat dobrou úspěšnost i při práci s reálnými daty.



# Abstract

The current state of intrusion detection tools is insufficient because they often operate based on static rules and fail to leverage the potential of artificial intelligence. The aim of this work is to enhance the open-source tool Snort with the capability to detect malicious network traffic using machine learning. To achieve a robust classifier, useful features of network traffic were choosed, extracted from the output data of the Snort application. Subsequently, these traffic features were enriched and labeled with corresponding events. Experiments demonstrate excellent results not only in classification accuracy on test data but also in processing speed. The proposed approach and the conducted experiments indicate that this new method could exhibit promising performance even when dealing with real-world data.



# Klíčová slova

anomálie, IDS, IPS, Snort, Suricata, strojové učení, XGBoost, klasifikace, datová sada, hodnocení příznaků

# Keywords

anomalies, IDS, IPS, Snort, Suricata, machine learning, XGBoost, classification, dataset, feature evaluation


# Citace

DOSTÁL, Michal. *Strojové učení z dat systémů pro detekci síťového průniku.* Brno, 2023. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Radek Hranický, Ph.D.

# Strojové učení z dat systémů pro detekci síťového průniku

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Hranického Ph. D. Další informace mi poskytl Ing. Petr Chmelař. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Michal Dostál
30. července 2023

## Poděkování

Nejvíce bych chtěl poděkovat mému vedoucímu Ing. Radku Hranickému Ph. D. a konzultantu Ing. Petru Chmelařovi.

# Obsah

# Kapitola 1

# Úvod

Rychlost vývoje technologií od začátku 21. století roste s neskutečně velkou tendencí. V dnešní době je spíše normální trávit čas s technologiemi než-li bez nich. Jelikož se většina elektronických výrobců snaží lidem usnadnit život, přichází tak nové a lepší vymoženosti a najednou jsme se ocitli před neomezeně velkým výběrem elektronických výrobků, na kterých můžeme provádět nespočetné množství činností. Zde však přichází problémy a není jich malé množství. Postupně, ale jistě integrujeme na internet všechny normální běžné činnosti, které jsme doposud vykonávali fyzicky, např. ať už se jednalo o nákup v obchodě, výběr peněz z banky nebo například poslání pošty, otázka, kterou si zde můžeme položit, zní, za jakou cenu. Již delší dobu se setkáváme s krádežemi peněz na internetu a mohli bychom doufat, že tento trend bude klesat s postupně rostoucí kvalitou bezpečnosti na internetu. Bohužel tento rok jsme mohli sami zaznamenat, že krádeže na Internetu rostou, a to rychlým tempem [10].

Aktuální stav IDPS nástrojů je nedostačující, jelikož tyto nástroje fungují na základě pravidel, které mu jsou předem nastaveny. Proto tyto nástroje nedokáží detekovat nové hrozby, které ovšem mohou být detekovány pomocným modulem a strojovým učením [7].

Cílem této diplomové práce bylo navrhnout a implementovat rozšíření pro již existující `open-source` nástroj Snort, které umí detektovat stejnou část hrozeb při porovnání s detekcí o proti předem daným pravidlům, ale zároveň umí detekovat potenciálně nové hrozby, a to díky rozšíření vstupních dat a strojovému učení. Rychlost tohoto rozšíření je násobně rychlejší než rychlost samotného nástroje Snort.

Přínosem této práce je inovativní rozšíření pro nástroj Snort, které umožňuje detekovat nové hrozby na základě získaných zkušeností z předchozích dat. Tyto zkušenosti jsou získávány prostřednictvím učení na předem označených datech, kde označení je vytvořeno na základě hlášených událostí a také pomocí černých listin.

V kapitole 2 je popsáno, co je to anomálie, jaké jsou typy takových anomálii a jak tyto anomálie správně detekovat. V kapitole 3 je popsána problematika systému pro detekci narušení, jak tyto systémy fungují a jak detekují jednotlivá narušení. Kapitola 4 popisuje externí aplikace, o které byly rozšířeny síťové toky, a to hlavně `IP Geolokaci` a modul `OpenAppID`. V kapitole 5 je popsán návrh rozšíření, popis vstupních dat, která budou použita, dále je zde vektor vstupních parametrů, který bude vstupem pro trénování modelu umělé inteligence, následně stručný popis způsobů označení těchto dat a popis toho jak bude daný model trénován. Kapitola 6 nejprve popisuje instalaci a konfiguraci nástroje Snort, dále popisuje implementaci sběru dat a jeho označení, nachází se zde taky popis trénování modelu a jeho vyhodnocení, na závěr kapitola popisuje integraci modulu do systému Snort. V kapitole 7 jsou popsány experimenty, které byly provedeny nad vytvořeným pravděpodob-

nostním modelem, tyto experimenty zahrnují přesnost detekce na datech z běžného sítového provozu, rychlost tohoto rozšíření oproti rychlosti detekci na pravidlech, zhodnocení kvality rozšířených příznaků a diskuze nad falešně predikovanými instancemi. Poslední kapitola 8 shrnuje přínos práce, dosažené výsledky a diskutuje o možném rozšíření této implementace.

# Kapitola 2

# Detekce anomálií v síťovém provozu

V této kapitole bude popsáno, co je to detekce anomálií, jaké se používají techniky pro detekci anomálií, jaké jsou výstupy těchto technik. Dále bude popsáno jaké známe typy anomálií a jaké jsou výzvy při určování hranice, kdy je daná instance dat anomálií. V poslední části bude popsán vztah mezi detekcí anomálií a detekcí narušení.

## 2.1 Anomálie

Anomálie je ve vědeckém prostředí považována jako odchylka od normálu. Mnoho vědeckých a technických oborů je založeno na předpokladu, že procesy nebo chování v přírodě fungují na základě určitých pravidel a obecných principů. Z těchto dat musíme formulovat hypotézy o povaze základního chování, které lze na základě těchto dat stanovit a ověřit na základě pozorování dalších dat. Tyto hypotézy popisují normální chování procesu přičemž se implicitně předpokládá, že data použitá k jejich generování jsou pro systém v určitém smyslu typické [15].

V procesech se však mohou vyskytnout odchylky od normy, a proto systémy mohou také nabývat abnormálních stavů, což vede k pozorovatelným hodnotám dat, které jsou odlišné od normálních stavů. Úkolem detekce anomálií je odhalit takové odchylky od normy v pozorovaných stavech a v hodnotách dat, a tím odvodit odchylky v základním procesu. Základní problém je, že neexistuje jednoduchá jedinečná definice, která by nám umožnila vyhodnotit, jak jsou si dva datové body podobné, a tedy jak moc se jeden datový bod liší od ostatních a taky v jakém rozsahu se tento bod liší od celkového souboru dat. Při použití algoritmu pro detekci anomálií je podle knihy [4] potřeba řešit tři možné případy.

- **Správná detekce:** zjištěné abnormality v datech odpovídají přesně abnormalitám v procesu. True Positive (TP), True Negative (TN) data byla vyhodnocena jako správně pozitivní nebo správně negativní.
- **Falešně pozitivní:** Proces se nadále chová normálně, ale jsou zde pozorována neočekávaná data. Např. v důsledku vnitřního šumu systému.
- **Falešně negativní:** Proces se stává abnormálním, ale důsledky se zde v abnormálních datech neprojevují. Může se tak například stát z důvodu toho, že síla abnormality není dostatečně velká na to, aby přehlušila vnitřní šum v systému.

U většiny reálných systémů je nemožné zaručit 100% přesnost detekce. Proto důležitou úlohou datového analytika je si uvědomit tuto skutečnost a návrhnout mechanismus, který bude minimalizovat obecně všechna rizika, ale hlavně rizika falešně pozitivních a falešně negativních výsledků a v oborech, kde se bude daný mechanismus používat rozhodnout, zda se bude povolovat více falešně pozitivních či více falešně negativních výsledků. Pokud se analytik např. bude pohybovat v odvětví medicíny je zde lepší povolit větší množství falešných pozitiv oproti falešným negativům, a to z toho důvodu, že hranice, která by rozhodovala o tom, jestli člověk trpí vážnou nemocí byla přikloněna ke špatnému vyhodnocení, mohlo by se tak stát, že by tento pacient byl špatně diagnostikován. Ovšem nejdůležitější faktor při vytváření rozhodovací hranice v jiných odvětvích jsou asymetrické náklady spojené s těmito jevy [15].

V kontextu kybernetické bezpečnosti musí algoritmy pro detekci anomálií zohledňovat skutečnost, že procesy, které jsou předmětem zájmu, nejsou často ani deterministické, ani zcela náhodné. Potíže, s nimiž se setkáváme v aplikacích kybernetické bezpečnosti, mohou totiž často být přičítány lidským aktérům s určitou svobodnou vůlí. Jinými slovy, pozorované chování je výsledkem záměrného (nikoliv náhodného) jednání lidí, které nelze předvídat, protože jejich záměry jsou neznámé a jejich plány se mohou v průběhu času měnit neočekávaným způsobem. V nejlepším případě mohou jednotlivá pozorování v průběhu času odhalit vzorec chování, u kterého můžeme očekávat že bude pokračovat a budeme tak moci určité věci předpovídat. Lidské záměry nebo plány, které ale tento vzorec tvoří, se mohou náhle změnit, a to například v reakci na mechanismy kybernetické obrany. Nepředpokládá se, že by však k takovým změnám docházelo příliš často [4].

## 2.2 Výzvy při detekci anomálií

Na abstraktní úrovni je anomálie definována jako vzor, který neodpovídá očekávanému normálnímu chování. Přímočarý přístup k detekci anomálií tedy spočívá v definování oblasti reprezentující normální chování a prohlášení každého pozorování v datech, které nepatří do této normální oblasti, za anomálii. Ovšem několik faktorů činí tento zdánlivě jednoduchý přístup velmi náročným [3]:

- Definování normální oblasti, která zahrnuje všechna možná normální chování, je velmi obtížné. Kromě toho je hranice mezi normálním a anomálním chováním velmi často nepřesná. Proto anomální pozorování, které leží v blízkosti této hranice, může být ve skutečnosti normální a naopak.
- Pokud jsou anomálie důsledkem záškodnických akcí, jsou tyto záškodnické akce interpretovány tak, aby se anomální pozorování jevila jako normální, čímž ztěžují úkol definovat normální chování.
- V mnoha oblastech se normální chování neustále vyvíjí a současný pojem normálního chování nemusí být v budoucnu dostatečně reprezentativní.
- Přesný pojem pro anomálii se v různých vědeckých oblastech liší. Například v lékařské oblasti je malá odchylka od normálu (např. výkyvy v tělesné teplotě) anomálií, zatímco podobná odchylka v oblasti trhu s cennými papír (např. výkyvy v hodnotě akcií) může být považována za normální. Použití techniky vyvinuté v jedné doméně nelze tedy přímo aplikovat na doménu jinou.

- Dostupnost označených dat pro trénování/ověřování modelů používaných pro techniky detekce anomálií je obvykle velký problém.
- Data často obsahují šum, který bývá podobný skutečným anomáliím, a proto je obtížné je rozlišit a odstranit [4].

## 2.3 Typy anomálií

Důležitým aspektem techniky detekce anomálií je povaha dané anomálie. Dle článku [3] lze anomálie rozdělit do následujících tří kategorií:

1. **Bodové anomálie:** Pokud lze jednotlivou datovou instanci považovat za anomální vzhledem k ostatním datům, pak se tato instance označuje jako bodová anomálie. Jedná se o nejjednodušší typ anomálie, na který se zaměřuje většina výzkumů v oblasti detekce anomálií.
2. **Kontextové anomálie:** Pokud je instance dat anomální v určitém kontextu (ale ne v jiném), pak se označuje jako kontextová anomálie. Pojem kontextu je indukován strukturou v datovém souboru a musí být specifikován jako součást formulace problému. Každá instance dat je definována pomocí následujících dvou sad atributů:
   - **Kontextové atributy:** Používají se k určení kontextu a okolí pro danou instanci. Například v souborech prostorových dat jsou kontextovými atributy zeměpisná délka a šířka.
   - **Atributy chování:** Atributy chování definují nekontextové charakteristiky pro danou instanci. Například pokud máme datovou sadu, která popisuje průměrné množství srážek na celém světě, tak množství srážek na libovolném místě je atribut chování.
3. **Kolektivní anomálie:** Pokud je soubor souvisejících datových instancí anomální, vzhledem k celému souboru dat, označuje se jako kolektivní anomálie. Jednotlivé datové instance v kolektivní anomálii nemusí být samy o sobě anomáliemi, ale jejich společný výskyt v souboru je anomální.

## 2.4 Označení dat

Štítek spojený s datovou instancí označuje, zda je daná instance normální, nebo anomální. Je třeba poznamenat, že získání označených dat, která jsou přesná a zároveň reprezentativní pro všechny typy chování je velmi často neúměrně nákladné. Označování je často prováděno manuálně expertem (analytikem), a proto je potřeba značné úsilí k získání souboru označených trénovacích dat. Obvykle je získání množiny označených anomálních dat, která pokrývá všechny možné typy anomálního chování, o dost obtížnější než získat označenou množinu dat pro chování normální. Dalším problémem, který se zde vyskytuje je ten, že anomální chování je velice často dynamické povahy, to znamená, že se mohou objevit nové typy anomálií, pro které neexistují žádná označená trénovací data. V případech jako je bezpečnost letového provozu by tyto anomální případy mohly vést ke katastrofickým událostem, a proto budou velmi závažné [15].

Na základě toho do jaké míry jsou data označena, mohou techniky pro detekci anomálií pracovat v jednom ze tří režimů.

### 2.4.1 Detekce anomálií s učitelem

Techniky, které se trénují v režimu s učitelem, předpokládají na vstupu označená trénovací data, Tato označená trénovací data musí obsahovat nejenom informace a instance, které reprezentují normální chování, ale také označené anomální instance. Typickým přístupem v takových případech je pak sestavení prediktivního modelu pro normální a anomální třídu. Každá nově příchozí instance dat je vyhodnocena pomocí vzniklého modelu a následně je určena její třída. Při detekci anomálií s učitelem vznikají hned dva hlavní problémy [4].

1. Výskyt anomálních instancí v trénovací datové sadě je v porovnání s normálními případy razantně menší, a z toho následně plynou problémy, které vznikají v důsledku nevyváženého rozložení tříd. Stručně řečeno, pokud se v datové sadě nachází 99% normálních instancí a zbylé jedno procento jsou anomální instance, tak model, který je pomocí této datové sady natrénován má tendenci úplně ignorovat detekci anomálních výskytů.

2. Získání přesných a reprezentativních značek, zejména pro třídu anomálií, je obvykle náročné. Byla navržena řada technik, které vnášejí umělé anomálie do normálního soubor dat s cílem získat označený trénovací soubor dat včetně anomálií.

### 2.4.2 Detekce anomálií bez učitele

Techniky, které pracují v režimu bez učitele nevyžadují žadné trénovací data, a proto jsou nejrozšířenější. Techniky v této kategorii implicitně předpokládají, že normální případy jsou mnohem častější než anomálie v testovacích datech. Pokud tento předpoklad není pravdivý pak tyto techniky trpí vysokou mírou falešných poplachů. Mnoho polosupervisovaných technik lze upravit tak, aby fungovaly v nesupervisovaném režimu. Pomocí vzorku neoznačeného souboru dat jako trénovacích dat. Takové přizpůsobení předpokládá, že testovací data obsahují jen velmi málo anomálií a model naučený v průběhu testování se může změnit a je odolný vůči těmto několika málo anomáliím [4].

### 2.4.3 Kombinovaná detekce anomálií

Techniky, které pracují v kombinovaném režimu, předpokládají, že trénovací data mají označené případy pouze pro tzv. normální třídy. Protože nevyžadují označení pro třídu anomálií, jsou více použitelnější než techniky s učitelem. Například v případě poruch kosmických lodí by scénář anomálie znamenal nehodu, kterou není snadné modelovat. Typickým přístupem používaným v těchto technikách je vytvořit model třídy odpovídající normálnímu chování a použít tento model k identifikaci anomálií v testovacích datech. Existuje omezený soubor technik detekce anomálií, které pro trénování předpokládají dostupnost pouze instancí anomálií. Takové techniky se běžně nepoužívají, především proto, že je obtížné získat trénovací soubor dat, který by pokrýval všechny možné případy anomálního chování, které se může v datech vyskytnout [4].

## 2.5 Výstup detekce anomálií

Důležitým aspektem pro jakoukoli techniku detekce anomálií je způsob, jakým se anomálie označují. Typicky výstupy vytvořené detekcí anomálií nabývají následujících dvou typů:

- **Skóre (Score):** Skórovací techniky přiřazují anomální skóre každé instanci dat v testovacích datech na základě toho, jak moc byla daná instance vyhodnocena jako anomální. Výstupem těchto technik je tedy ohodnocený seznam anomálií. Analytik může například analyzovat několik nejvýše ohodnocených anomálií, nebo použít mezní hodnotu pro výběr anomálií.
- **Štítek (Label):** Techniky spadající do této kategorie přiřazují každé instanci dat informaci o tom, zda je normální či anomální.

Techniky detekce anomálií, které jsou založeny na skórování umožňují analytikovi použít prahovou hodnotu, která bude určovat oblast k výběru jen těch nejvýznamnějších anomálií. Toto však techniky, které přidělují binární štítky jednotlivým instancím neumožňují [3].

## 2.6 Detekování průniků pomocí detekce anomálií

Detekce narušení se týká detekce škodlivých aktivit (vloupání, průniků a další formy zneužití počítače) v systému souvisejícím s počítačem. Tyto škodlivé činnosti nebo narušení jsou zajímavé z hlediska počítačové bezpečnosti. Vniknutí se liší od normálního chování systému a proto jsou v oblasti detekce narušení použitelné techniky detekce anomálií. Klíčovou výzvou pro detekci anomálií v této oblasti je obrovský objem dat. Techniky detekce anomálií musí být výpočetně efektivní, aby bylo možné zvládnout takto velké vstupy. Data navíc obvykle přicházejí v podobě toku a vyžadují tak on-line analýzu. Dalším problémem, který vzniká kvůli velkému vstupu dat, je míra falešných poplachů. Vzhledem k tomu, že data obsahují až milióny datových objektů, může několik procent falešných poplachů způsobit, že analýza bude pro analytika zahlcující. Obvykle jsou k dispozici označená data odpovídající normálnímu chování, zatímco data, která jsou škodlivá označena nejsou. Proto je možné použít techniky detekce anomálií, kterou jsou založeny na metodě učení bez učitele a kombinované metodě s učitelem a bez [15].

Detekci narušení můžeme dále klasifikovat podle článku [9] na:

- Systémy detekce narušení založené na hostiteli (**Ossec, Sagan, Splunk**)[1]
- Systémy detekce narušení sítě (**Snort, Suricata, Zeek**)[2]

### 2.6.1 Systémy detekce narušení založené na hostiteli

Tyto systémy, které jsou označované také jako systémy detekce narušení systémových volání, se zabývají stopami volání operačního systému. Narušení mají podobu anomálních podsekvencí (kolektivních anomálií). Anomální podsekvence se promítají do škodlivých programů, neautorizovaného chování a porušení zásad. Zatímco všechny stopy obsahují události, které vypadají na první pohled normálně, je to právě společný výskyt událostí, který je klíčovým faktorem při rozlišení normálního a anomálního chování [15].

### 2.6.2 Systémy detekce narušení sítě

Tyto systémy se zabývají detekcí narušení v síťových datech. Tato narušení se obvykle vyskytují jako anomální vzorce (bodové anomálie), navzdory tomu, že některé techniky modelují data sekvenčním způsobem a detekují tak anomální následnosti. Primárním důvodem těchto anomálií jsou útoky prováděné tzv. hackery z venku, kteří chtějí získat neoprávněný přístup do sítě za účelem získání tajných či osobních informací, krádeže těchto informací, a také za účelem narušení a poškození této sítě. Typickým prostředím pro takový útok je rozsáhlá síť počítačů, která je propojená se zbytkem světa prostřednictvím internetu. Data dostupná pro systémy detekce narušení mohou mít různé úrovně granularity, např. stopy na úrovní paketů, data o síťových tocích `CISCO` atd.. Tato data závisí na časovém aspektu, ale většina technik pro detekci anomálií s tímto aspektem nepracuje. Data jsou typicky více rozměrná směsice kategoriálních a spojitých atributů. Výzva, která nastává, pro techniky detekce anomálií v této doméně je ta, že povaha anomálií se v průběhu času neustále mění, a to z toho důvodu, že útočníci a narušitelé přízpůsobují své síťové útoky, aby se vyhnuli mechanismům, které jsou aktuáně nasazeny v síti, na kterou se snaží zaútočit [15].

[1] Přehled nástrojů HIDS z dnsstuff.com
[2] Přehled nástrojů NIDS z softwaretestinghelp.com

# Kapitola 3

# Systémy IDS a IPS

V této kapitole bude popsáno, co jsou to systémy IDS[1] a IPS[2], existující IDS a IPS nástroje. Budou zde popsány nástroje `Snort`, `Suricata` a `Zeek`. Jako poslední bude provedeno porovnání výkonu mezi jednotlivými nástroji a bude provedeno jejich vyhodnocení.

## 3.1 Systém detekce narušení (IDS)

Systémy detekce narušení jsou poplašné systémy proti vloupání v oblasti počítačů a jejich bezpečnosti. Jejich cílem je bránit systém pomocí kombinace poplašného signálu (kdykoli je ohroženo zabezpečení `host-net`) a osoby nebo subjektu, nejčastěji bezpečnostního pracovníka `host-netu`, který se stará o jeho zabezpečení. [1]. Systém IDS je zodpovědný za monitorování datového provozu v síti, případného podezřelého provozu nebo také za činnosti proti zabezpečení sítě. Systém nebo správce sítě je upozorněn nebo informován, pokud jsou v síti zjištěny jakékoli hrozby nebo škodlivé činnosti. Z toho vyplývá, že hlavním účelem IDS je detekovat a hlásit pokusy o vniknutí do sítě příslušným orgánům. [24]

### 3.1.1 Typy narušení

Narušení může nabývat několika různých typů podle [1].

- Například uživatel může ukrást heslo a tím i prostředky, kterými prokazuje svoji identitu. Takovému uživateli říkáme maškaráda a odhalování takových narušitelů je jeden z důležitých problému v této oblasti.
- Další třída narušitelů, jsou lidé, kteří jsou legitimní uživatelé systému, ale zneužívají svá oprávnění

### 3.1.2 Metody detekce

Na počátku výzkumu těchto systémů byly dva hlavní principy známe jako detekce anomálií a detekce hrozeb. Nyní se metody detekce dělí do následujících kategorií:

- `Detekce na bázi signatur`
- `Detekce na bázi anomálií` viz [5].

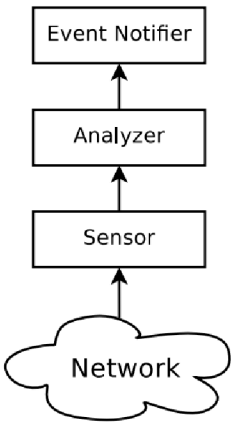
[1] Intrusion detection system (IDS), nebo-li systém detekce narušení
[2] Intrusion prevention system (IPS), nebo-li systém prevence narušení.

- `Stateful Protocol Analysis`
- `Behaviour-based`
- `Knowledge-based`
- `Network-based IDS`
- `Host-based IDS`

### 3.1.3 Přístup k detekci narušení

Práce IDS spočívá v analýze některých vstupních dat. Vstupní data mohou být různá od auditních záznamů a protokolů operačního systému nebo aplikací až po nezpracovaný síťový provoz. Podle zvolené třídy vstupních dat lze použitý systém přiřadit k jednotlivým kategoriím [12].

Navzdory rozdílům, které jsou třídám IDS vlastní, existují některé společné stavební prvky, kde lze identifikovat několik bloků a funkcí na vysoké úrovni potřebné pro zajištění bezpečnosti a splnění úkolu detekce narušení. Tyto komponenty jsou znázorněny na obr. 3.1 a jedná se o následující:




Obrázek 3.1: Struktura IDS systému převzato z [12].

- **Senzor:** ať už jsou vstupní data jakákoli, je zapotřebí komponenta, která dokáže číst a převést je do formátu, který je kompatibilní s požadovaným formátem z analyzátoru. Převod do takového formátu někdy zahrnuje extrakci některých zájmových parametrů, jejichž cílem je syntéza vlastností zařízení.
- **Analyzátor:** jakmile jsou data vymodelována do společného formátu, je třeba je analyzovat. V zásadě by složka analyzátoru takového IDS mohla být nezávislá na typu dat. Analyzátor musí znát soubor kritérií zaměřených na detekci konkrétních vlastností analyzovaných dat a v případě, že se alespoň jedna z těchto vlastností nachází v daných kritériích, musí se informovat subjekt o výskytu takové události. Pokud je každé kritérium přiřazeno k nejpravděpodobnější příčině, která mohla způsobit danou událost, je analyzátor nejen schopen upozornit v případě výskytu některých konkrétních událostí, ale je také schopen přiřadit tyto události k dané příčině, čímž umožní klasifikaci každé hlášené události.

- **Hlásič událostí:** kdykoli analyzátor ohlásí výskyt nějaké události, je nutné umožnit celému systému komunikovat s vnějším světem, aby bylo možné tyto události oznamovat. Oznamovatel událostí má na starosti interpretaci výsledků analýzy a správné formátování zpráv potřebných pro komunikaci s uživateli systému [12].

## 3.2 Systém prevence průniku (IPS)

Systém prevence průniku je zařízení pro počítačovou bezpečnost, které monitoruje síť a aktivity operačního systému na škodlivou činnost. Hlavní funkce IPS systémů jsou identifikace škodlivé činnosti, zaznamenávání informací o jejím průběhu, následném blokování této činnosti a také její nahlašování [8].

IPS systémy jsou považovány za rozšíření IDS systémů, jelikož poskytují veškeré funkce dostupné u systémů IDS. Rozšíření IDS spočívá v tom, že IPS se snaží aktivně zamezovat náhlým bezpečnostním incidentům. Systém IPS je tedy aktivní bezpečnostní prvek.

Vyčet technik pomocí kterých se IPS systémy snaží bránit útokům podle [19]:

- Systém IPS sám zastaví útok. Příklady, jak to lze provést, jsou následující:
  - Ukončit síťové připojení nebo relaci uživatele, která byla použita k útoku.
  - Zablokovat přístup k cíli (případně k dalším pravděpodobným cílům), zablokovat přístup k IP adrese útočníka nebo k jiným jeho atributům.
  - Zablokovat veškerý přístup k cílovému hostiteli, službě, aplikaci nebo jinému prostředku.
- Systém IPS změní bezpečnost prostředí: Tento systém může změnit konfiguraci jiných bezpečnostních kontrol tak, aby byl útok narušen. Běžným příkladem je změna konfigurace síťového zařízení (např., firewall, směrovač, přepínač), tak aby blokovalo přístup od útočníka. Některé IPS systémy dokonce mohou způsobit, že se na daném místě aplikují záplaty, pokud zjistí, že hostitel obsahuje zranitelnosti.
- Systém IPS změní obsah útoku: Technologie IPS mohou odstranit nebo nahradit škodlivé části útoku, aby se stal neškodným. Jednoduchým příkladem je odstranění infikovaného souboru z e-mailu a poté odeslání takto vyčištěné zprávy příjemci. Více složitější příklad může být IPS, které funguje jako proxy server a normalizuje příchozí požadavky, což znamená, že proxy server znovu zabalí obsah požadavku a odstraní informace z hlaviček. Toto může vyřadit jisté útoky v rámci procesu normalizace [19].

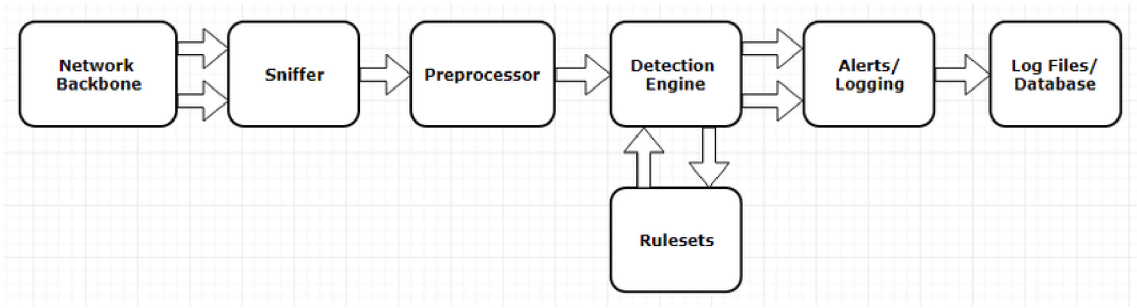
## 3.3 Přehled nejznámějších systémů IDS/IPS

V této sekci poskytneme přehled nejvíce rozšířených nástrojů pro detekci a prevenci síťových průniků (*IDPS*). Důrazněji se zaměříme na nástroje *Snort* a *Suricata* a detailně popíšeme jejich architekturu.

### 3.3.1 Snort

Snort[3]. je jedním z nejrozšířenějších řešení systém detekce a prevence narušení založených na signaturách, které podporuje režim systému pro odhalení průniku (IDS) i systém prevence průniku (IPS). Snort je cenný nástroj `network-based` IDPS, který lze snadno konfigurovat. Dokáže monitorovat provoz v síti, porovnávat přijaté pakety s podpisy, logovat útoky a je také schopen prezentovat statistiky útoků na konzoli, pokud jsou splněná pravidla [25].

Nástroj `Snort` standardně používá knihovnu `libpcap` pro zachycení paketů, na kterou pak navazuje dekodér pro dekódování paketů. Normalizace paketů provádí preprocesor, který převádí provoz do srozumitelné podoby pro detekční jádro. Detekční monitor aplikuje na provoz pravidla, aby zjistil, zda se v něm vyskytuje škodlivý paket. Snort používá pouze detekci zneužití a ve výchozím nastavení nepodporuje detekci založenou na anomáliích. V režimu `IDS` generuje pouze výstrahy na základě detekce, zatímco v režimu `IPS` blokuje škodlivé pakety. Poslední součást je výstupní blok, který jednoduše generuje textový soubor, který si uživatel může později prohlédnout [14]. Na obrázku 3.2 lze pak vidět kompletní architekturu aplikace `Snort` popsanou v této sekci.




Obrázek 3.2: Architektura programu snort převzato z [20].

Snort byl vyvinut již v roce 1998 a od té doby prošel mnoha aktualizacemi. Vícevláknová varianta Snort byla představena jako Snort 3.

Aplikaci Snort lze v základním režimu spusti v následujících 3 módech.

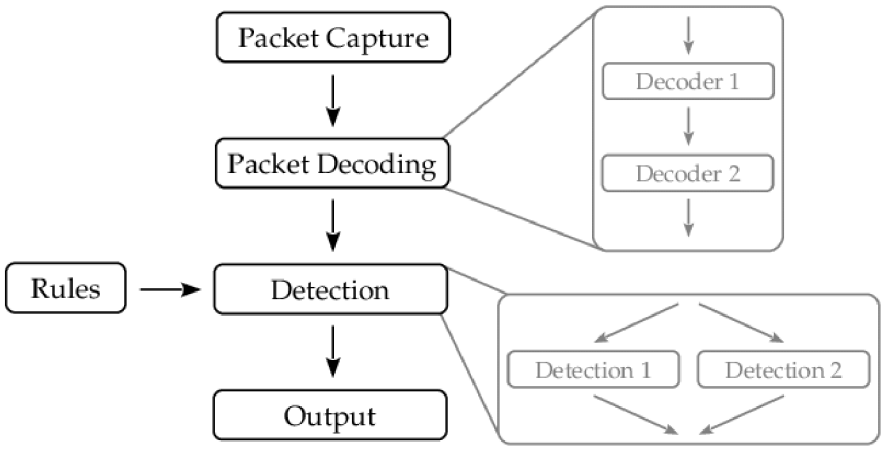
- `Sniffer mód:` Základní mód, ve kterém jsou pakety načítány jako tok ze síťového rozhraní a následně jsou vypsány do konzole.
- `Packet Logger mód:` Ukládá pakety do souboru.
- `Network Intrusion Detection System mód:` V tomto režimu `Snort` pracuje jako NIDS nástroj.

[3]Odkaz na nástroj Snort 3

### 3.3.2 Suricata

Suricata je velmi silnou alternativou network-based IDPS. Suricata podporuje více detekčních enginů jelikož má více vláknové zpracování. Pokud jde o detekci založenou na signaturách, Suricata používá stejný formát, jaký používá Snort pro deklaraci pravidel, a má také podobné detekční algoritmy. Podobně jako Snort, Suricata také podporuje režimy IDS a IPS, zatímco ve výstupní fázi mohou být výstrahy uloženy buď v jednoduchém textovém souboru, nebo mohou být uloženy ve formátu JSON [14].

Architektura nástroje `Suricata` je podobná architektuře nástroje `Snort`, ale s jedním rozdílem. To, co odpovídá části preprocesoru v systému Snort, je v architektuře Suricata rozděleno na dvě části, a to dekódování a detekci. Dekódovací moduly přidávají informace do vnitřní reprezentaci paketů v Suricatě. Detekční moduly se spoléhají na tuto vnitřní reprezentaci a poskytují klíčová slova pro použití v pravidlech [6]. Na obrázku 3.3 je znázorněna architektura nástroje Suricata.




Obrázek 3.3: Architektura programu Suricata převzato z [6].

### 3.3.3 Zeek

Zeek je další open-source nástroj NIDS, který podporuje pouze režim IDS. V rámci nástroje Zeek jsou na síťových zařízeních nasazeni agenti zvaní pracovníci, kteří odesílají své protokoly správci. Správce Zeek má v zásadě dvě komponenty, a to engine událostí, který převádí každý paket přijatý ze sítě na událost a předává ji další komponentě, kterou je interpret skriptu zásad. Hlavní funkcí interpretu skriptu zásad je aplikovat pravidla Zeek na události generované z enginu událostí, to může vést k výstrahám v případě detekce škodlivé činnosti.

Architektura Zeek je vysoce škálovatelná a lze v ní snadno dosáhnout zvýšení výkonu vyčleněním více hardwarových prostředků pro pracovníky a správce. Zeek razí svůj význam v prostředích, kde se jedná o útoky nultého dne, protože podporuje také detekci založenou na anomáliích. tato funkce chybí v systémech Snort i Suricata, které podporují pouze detekci zneužití. Zeek poskytuje hloubkovou analýzu síťového provozu a je poměrně efektivním, snadno nasaditelným a flexibilním open-source řešením, má však omezený výchozí počet

signatur/pravidel, což omezuje jeho široké využití. Naopak Snort a Suricata mají mnohem širší základnu pravidel s aktivně přispívající komunitou, a tím nacházejí větší pochopení pro nasazení [25, 17].

## 3.4 Životní cyklus IDS

Prodejci často vydávají nové systémy `IDS` a agresivně soupeří o podíl na trhu. Hodnocení těchto nových systémů není triviální úkol, z tohoto důvodu pak chybí důvěryhodné a komplexní hodnocení produktu. Najímání a udržení pracovníků pro kompetentní správu zabezpečení a zejména detekci narušení je stále náročnější. Rychlé změny v informačních technologiích ztěžují zavést účinnou organizaci dlouhodobé bezpečností strategie [14].

### Hodnocení a výběr

Pokud organizace plánuje pořídit IDS, měla by zvážit zdroje, které jsou k dispozici pro provoz a údržbu systému a vybrat si takový systém, který bude v rámci těchto omezení vyhovovat jejich potřebám. To je obtížné, protože neexistují žádné průmyslové standardy, podle kterých by bylo možné IDS porovnávat. Cyklus nových produktů pro komerční IDS je rychlý a informace a systémy rychle zastarávají [14].

### Nasazení

Problémy s nasazením, které je potřeba řešit je např. umístění senzorů tak, aby se maximalizovala ochrana nejkritičtějších prostředků a konfigurovat IDS správně tak, aby odrážel bezpečnostní politiku, dalším problémem je taky instalace vhodných signatur a dalších prvků. Uživatelé musí proto vypracovat postupy pro zpracování výstrah `IDS` a zvážit, jak korelovat výstrahy s dalšími informacemi, např. systémovými nebo aplikačními protokoly [14].

### Provoz a použití

Jakmile organizace nasadí `IDS`, musí tento systém monitorovat a reagovat na výstrahy, které systém hlásí. To znamená stanovit role a odpovědnosti za analýzu a reakci na výstrahy, sledovat výsledky jak manuálních, tak i automatických reakcí [14].

### Údržba

Činnosti údržby zahrnují instalaci nových podpisů, jakmile nějaké nastanou, a také pravidelnou aktualizaci daného IDS systému. Umístění snímačů by mělo být pravidelně revidováno, aby se zajistilo, že systém nebo změny v síti nesníží účinnost IDS [14].

## 3.5 Porovnání IDPS nástrojů

V této sekci budou porovnány nástroje `Snort`, `Suricata` a `Zeek`. Budou zde porovnány dvě verze nástroje Snort, a to verze `2.9.16` a `3.1.7`. Všechny verze nástroje `Snort`, které jsou 3 a výše podporují multi-vláknové zpracování. Dále zde budou porovnány moduly pro zpracování paketů, a to modul `Libpcap` a `AF_Packet`. Nástroje Snort i Suricata oba tyto moduly podporují a bude porovnána jejich efektivita při použití každé z nich viz [25].

### 3.5.1 Porovnání modulů pro načítání paketů

Porovnání `Libpcap` a `AF_Packet` ve Snort 2.9.16 ukazuje, že `Libpcap` začíná zahazovat pakety i při rychlosti 200 Mb/s. a tento pokles se stává podstatným při vyšších rychlostech provozu. `AF_Packet` nevykazuje žádné poklesy ani při rychlosti 1000 Mb/s. což naznačuje, že `AF_Packet` má ve srovnání s `Libpcap` vyšší kapacitu zachycení paketů. Doporučení tedy zní, aby se výchozí modul pro zachytávání paketů `Libpcap` nahradil modulem `AF_Packet` ve Snort 2.9.16, aby se zlepšila schopnost zachycení paketů.

Snort 3.1.7 porovnání výkonu mezi Libpcap a `AF_Packet` vykazuje, že Libpcap začíná zahazovat pakety už při rychlosti 500Mbps a ztráta paketů podstatně roste při vyšších rychlostech. Naopak `AF_Packet` nevykazuje žádný pokles paketů ani při rychlosti 1000 Mbps, což naznačuje převahu nad Libpcapem na základě výkonu. Maximální využití paměti při přenosové rychlosti 1000 Mb/s se ukázalo být 28,7 % pro `AF_Packet` a 18,9 % pro Libpcap, zatímco při maximálním vytížení 1000 Mb/s pro `AF_Packet` bylo maximální využití procesoru 55 % a pro Libpcap bylo 50 %.

U nástroje `Suricata` nastalo k podstatnému zahazování paketů při použití knihovny `Libpcap` již při rychlosti 100Mbps, ovšem nebylo zde pozorováno žádné zahazování paketů s použití knihovny `AF_Packet`, a to ani při rychlostech 1000Mbps [25].

### 3.5.2 Porovnání v režimu IDS

Nástroje `Snort 2.9.16`, `Snort 3.1.7`, `Suricata`, `Zeek` byly porovnány v režimu IDS, a to s celkem 42 tisíci pravidly. Nejhůře dopadl nástroj `Snort 2.9.16`, který měl nejvyšší ztrátu paketů, což je v porovnání s ostatnímy nástroji pochopitelné, jelikož Snort této verze pracuje s jedno-vláknovou architekturou, při které selhává efektivní zpracování paketů v detekčním jádře. Proto když rychlost provozu postupně roste, zvyšuje se tak i ztráta paketů.

Lépe dopadl nástroj `Snort 3.1.7` a `Suricata`, které nevykazují žádný úbytek paketů, a to díky vícevláknovému zpracování dat.

Nástroj `Zeek` také nevykazoval žádný pokles paketů, což je způsobeno tím, že pravidla využívají omezená pole, a proto vyžadují méně zpracování ve srovnání s pravidly pro zpracování dat [25].

### 3.5.3 Porovnání v režimu IPS

V této sekci budou porovnány pouze nástroje `Snort 2.9.16`, `Snort 3.1.7`, `Suricata`, nástroj `Zeek` zde uveden nebude, jelikož nepodporuje režim `IPS`. Jelikož režim IPS přidává značnou režii při zpracování příchozích paketů, může se zde stát, že při neefektivní implementaci zpracování příchozího paketu může lehce nastat zahlcení při vysokém vstupním provozu. Výsledky tohoto porovnání ukazují, že nástroj `Snort 2.9.16` již při rychlost 100Mbps zahazuje téměř 34,6% paketů. Nástroj `Snort 3.1.7` si vede srovnatelně lépe, jelikož zahazuje pouze 10 procent přijatých paketů při stejné přenosové rychlosti. Nejlépe si vede nástroje `Suricata`, který nevykazuje žádný pokles až do rychlosti 200Mbps, ovšem pokles paketů se projeví hned po překročení rychlosti 300Mbps. Hlavním důvodem poklesu paketů je skutečnost, že detekční algoritmus je při takové rychlosti omezen na efektivitě s porovnáváním všech paketů se všemi definovanými pravidly [25].

### 3.5.4 Vyhodnocení

Výběr NIDPS `open-source` řešení pro ochranu sítě v organizaci vyžaduje důkladnou analýzu výkonu jednotlivých nástrojů při zkoumání všech různých okolností. Výsledky tohoto vyhodnocení ukazují, že nástroj `Suricata` překonává zbývající nástroje ve všech ohledech. Takto dominantní výkon nástroje `Suricata` předurčuje aby se tento nástroj stal volbou č. 1 pro všechny odborníky z praxe a bylo považování za standardní `open-source` řešení, které by bylo dále zkoumáno a vylepšováno. Ačkoli `Suricata` stojí vysoko nad ostatními nástroji, stále by se zde dal zlepšit algoritmus pro porovnávání vzorů [25]. Nicméně s příchodem Snort 3 můžeme dnes říci, že Suricata již má silného konkurenta, který by si v budoucnu mohl vyhradit velmi důležité místo. Snort 3 oproti předchozím verzím výrazně zlepšil využití paměti systému Snort a rychlost zpracování paketů [2].

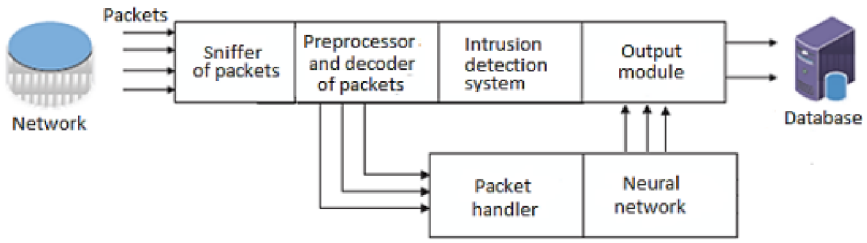
## 3.6 Existující nástroje

V článku [11] je popsáno rozšíření pro nástroj `Snort`, které používá pro detekci anomálií Neuronové sítě. Je zde rozebrána implementace a nakonec je funkčnost této implementace prověřena a otestována na reálných datech. Nejdůležitější částí této práce je popis modulu neuronové sítě.

### Popis modulu neuronové sítě

Hlavním cílem práce [11] je změnit stávající strukturu `Snort` a integrovat do ní další modul, díky němuž vznikne nová struktura. Navrhovaný adaptivní modul pracuje paralelně se sadou pravidel `Snort`. Důvod proč integrace tohoto adaptivní modulu bude běžet současně se sadou pravidel je ten, že sada pravidel `Snort` detekuje pouze známý destruktivní provoz. Tento modul tak může detekovat neznámou variantu destruktivního provozu nebo také může snížit počet falešných detekcí. Díky tomuto se pak sníží počet falešně pozitivných poplachů a zvýší se přesnost detekce. Preprocesor bude paralelně přenášet síťový provozo do modulu neuronové sítě a pravidel Snort, oba moduly tak mohou pracovat paralelně a mohou přesněji detekovat destruktivní provoz.

Na obrázku 3.4 můžeme vidět, jak bude popisovaný modul zasazen do systému `Snort`. Paralelně z preprocesoru potečou data do externího zpracování paketů a do modulu pro vyhodnocení pravidel. Následně pak výsledky vyhodnocení, jak modulu neuronové sítě, tak modulu pro vyhodnocení pravidel předají své hlášení do výstupního modulu.




Obrázek 3.4: Struktura Snort modulu převzato z [11].

Vstupní parametry musí být připraveny před jejich použitím v algoritmu strojového učení. Některé položky lze snadno zjistit, jiné je třeba zjistit experimentováním a testováním.

Použití všech prvků datové sady nemusí nutně zaručit nejlepší výsledky, naopak se tím můžou zvýšit výpočetní náklady i četnost chyb v systému. To je způsobeno tím, že některé funkce jsou pro rozlišení různých tříd útoků nadbytečné nebo nepoužitelné. Hlavní výhodou této tréningové metody je zavedení atributů navržených například expertem, které pomáhají pochopit chování různých útoků, včetně základních charakteristik detekce útoků [11].

# Kapitola 4

# Aplikace pro rozšíření dat

Kapitola se zabývá aplikacemi, které jsou použity pro rozšíření vstupních dat, díky těmto externím informacím, které dané aplikace poskytují by měla být zvýšená přesnost detekce následně trénovaného pravděpodobnostního modelu. Tato kapitola popisuje rozšíření pro aplikaci `Snort`, a to modul `OpenAppID`, který poskytuje informaci o komunikujících aplikacích a dále tato kapitola popisuje geolokaci v síťovém provozu.

## 4.1 OpenAppID

`OpenAppID` je jedním z preprocesorů pro `Snort`. Identifikuje aplikace/služby na základě předdefinovaných vzorů/pravidel uložených jako soubor detektoru `lua`. Minulé verze nástroje snort splňovaly tento požadavek tím, že umožňovaly uživatelům zadávat metadata služeb, jejich adresy URL a obsah. Tento nový preprocesor vylepšuje a přidává zjednodušené povědomí o aplikacích v jednom bodě tím, že tvůrcům pravidel Snort zpřístupňuje sadu identifikátorů aplikací `AppId`. V pravidle by stačilo místo adres URL a statického obsahu zadat pouze `AppId`.

Pro zajištění povědomí o využití sítě tento nový preprocesor vypisuje statistiky, které ukazují šířku pásma sítě využívanou jednotlivými síťovými aplikacemi. Správci mohou sledovat využití šířky pásma a mohou se rozhodnout zablokovat aplikace, které jsou neekonomické. Předprocesor umožňuje správcům vytvářet vlastní detektory aplikací pro detekci nových aplikací. Tyto detektory jsou napsány v programovacím jazyce Lua a se Snortem se propojují pomocí dobře definovaného rozhraní `API C-Lua` [16].

### Přístupy k identifikaci aplikací

Historicky firewally používaly IP adresy a čísla portů jako způsob vynucování zásad. Tato strategie vychází z předpokladu, že se uživatelé připojují k síťovému serveru z pevně stanovených míst svého uzlu a přistupují ke konkrétním prostředkům pomocí konkrétních čísel portů.

Mapování podpisů aplikací je přesná metoda identifikace aplikace, která generovala provoz v síti. Mapování podpisů funguje na 7. vrstvě a kontroluje skutečný obsah paketu.

Přístupy k identifikaci aplikací se podle [16] dělí na následující:

- `Na základě signatury:` v přístupu založeném na signatuře se pakety porovnávají s předem definovanými údaji uloženými v databázi. Tato data mohou být vyjádřena různými způsoby.

    1. `Statické prvky:` tento typ vyhledává statický hexadecimální obsah v paketu, například `01050204`, což odpovídá všem paketům, které mají v obsahu paketu `01050204`.
    2. `Prvek regulárního výrazu:` tento typ vyhledávání je určen pro hexadecimální obsah v paketu založený na regulárních výrazech, například `Ô2[159]0478`, který odpovídá všem paketům, které mají v jeho obsahu 02, následovaný buď 1 nebo 5 nebo 9, přičemž `suffix` je 0478.
    3. `Vyhledávání více prvků:` tento typ vyhledávání slouží k vyhledávání více prvků z jednoho paketu. Například `0205005403`, `0493759758`, `9387204737` vyhledá všechny tyto prvky, pokud se vyskytují v jednom paketu, kde nezáleží na místě jejich výskytu.

- `Na základě protokolu:` v přístupu založeném na protokolu jsou pakety porovnávány s protokolem a souvisejícími prvky uloženými v databázi. Příklad:

    *udp packet on dstport* 51463

    Na příkladu výše, se provede vyhledání všech paketů na protokolu `UDP` a portu `51463`.

- `Na základě statistických informací:` V přístupu statistické identifikace jsou informace týkající se relací porovnávány v reálném čase tak, aby splňovaly kritéria daných statistik.

Informace poskytnuté z OpenAppID se podle [16] dělí na následující:

- `ServiceAppID:` appId spojené s relací na straně serveru. Příklad http serveru.
- `ClientAppID:` appId přidružené k aplikaci na straně klienta relace. Například prohlížeč `Firefox`.
- `PayloAdappID:` pro služby jako http je to appId spojené s hostitelem webového serveru. Příklad `Facebook`.
- `MiscAppID:` u některých zapouzdřených protokolů je to nejvyšší úroveň zapouzdřené aplikace.

## 4.2 IP Geolokace

Metoda IP Geolokace je v rámci internetových sítí používána jako prostředek ke zjišťování pozice koncového zařízení, které je připojeno k internetu. Mezi tato zařízení může patřit například stolní počítač, přenosný počítač, mobil, tablet, server, chytré hodiny, atd. V současné době je to velmi cenný nástroj, protože pokud známe přesné umístění náhodného uživatele, který v minulosti hledal na webu nabídku ojetých automobilů, můžeme tomuto uživateli v reklamě na dále navštěvovaných stránkách poskytnout nabídky ojetých vozů z jeho pozice blízko dostupných autobazarů.

Podle článku [13] můžeme geolokaci použít v následujících případech:

- **Online reklama:** Jak bylo zmíněno v úvodu výše, geolokaci můžeme využít jako nástroj pro tzv. předhazování reklam vhodných pro daného uživatele. Pokud se uživatel z města Zlín zajímá o nabídky práce, budeme schopni díky geolokaci identifikovat jeho polohu a nabídnout mu tak vhodné pracovní pozice.

- **Vyhledání dopravního spojení:** Dnes již běžná praktika hojně využívána například v aplikaci IDOS. Uživatel otevře na mobilních datech se zaplou GPS lokací mobilní telefon např. na zastávce Semilasso a následně je aplikace schopna detekovat přesnou lokaci a automaticky tak vyplnit místo odjezdu.

- **Ochrana proti podvodům:** Případ, který bude využíván v této diplomové práci v rámci detekce škodlivých toků na základě lokace komunikujícího partnera, ale obecně se jedná o rozsáhlejší případ. Můžeme to uvést na příkladu debitních karet. Pokud jako klient například ČSOB dostanu debitní kartu, mohu si zablokovat jakékoli platby ze zahraničí, dojde tak k zabránění možnému zneužití.

- **Boj proti spamu:** Případ, který je také velmi podobný tomu, který bude interpretován v této diplomové práci. V tomto případě můžeme díky geolokaci jednoduše filtrovat e-maily na základě původní lokace odesílatele.

- **Zábava:** Díky geolokaci vzniká spoustu aplikací, které benefitují právě z této metody. Aplikací je k dnešnímu roku už spousty, ale pro příklad můžeme uvést například seznamovací aplikace, GeoCache aplikace, nebo například sociální sítě, které již při přidávání příspěvku sami detekují lokaci uživatele a doplní místo, kde byl daný příspěvek vytvořen.

### 4.2.1 Pasivní geolokační metody

Pasivní metody geolokace operují na principu získávání lokalizačních informací pouze na základě dostupných dat o daném síťovém zařízení. Poloha těchto zařízení je odvozena z údajů uložených ve veřejných nebo soukromých databázích. Podle [13] existují tři hlavní typy pasivních metod geolokace:

- Geolokace na základě IP adresy.
- Geolokace na základě DNS (Domain Name System) záznamů.
- Geolokace s využitím signálů WiFi.

Těmito metodami lze zjistit polohu zařízení bez potřeby aktivní interakce s ním, což je užitečné pro sledování a lokalizaci zařízení v rámci síťového prostředí.

#### Databáze WHOIS

V současné době je veřejná databáze Whois nejznámějším nástrojem využívaným pro získávání informací o registrovaných IP adresách a internetových doménách. Pro správu registrací těchto IP adres a domén v jednotlivých státech existují národní registrátoři, kteří spolupracují s mezinárodní organizací pro přidělování IP adres v internetu (IANA).

Databáze Whois slouží jako evidence údajů o majitelích internetových domén a IP adres. Tyto záznamy obsahují informace jako jméno správce domény, kontaktní adresu, e-mailovou adresu a případně i telefonní číslo. Tímto způsobem poskytuje důležité informace o vlastnících těchto zdrojů na internetu.

#### Databáze GEOIP

Další velmi často používanou databází je GeoIP, která obsahuje knihovnu pro programovací jazyk `python`, umožňující identifikovat zemi, ze které pochází daná IP adresa nebo název počítače. Tato knihovna využívá souborovou databázi, kde bloky IP adres slouží jako klíče.

Záznamy v databázi GeoIP obsahují informace o zemích, městech, PSČ a zeměpisných souřadnicích, které jsou přiřazeny k daným IP adresám. Některé IP adresy mají v databázi přesný záznam polohy, včetně města a konkrétních geografických souřadnic. Avšak u jiných IP adres se pouze určí země původu, a v takovém případě jsou zeměpisné souřadnice stanice nastaveny na střed této země.

### 4.2.2 Aktivní geolokační metody

Aktivní geolokační metody fungují na principu odhadu polohy stanice pomocí informací získaných z měření datového přenosu v internetové síti. Typicky se provádí měření zpoždění a analyzuje se trasa, kterou data procházejí přes síťové uzly od zdroje k cíli. Typy aktivních geolokačních metod se podle [13] dělí na:

- `Metoda GeoPing`
- `Metoda ShortestPing`
- `Metoda Constraint-Based Geolocation`
- `Metoda lokace založená na topologii sítě (TBG)`

# Kapitola 5

# Návrh rozšíření pro Snort

V této kapitole bude popsán návrh rozšíření, které bude začleněno jako nový modul pro aplikaci `Snort`, toto rozšíření bude pomocí strojového učení detekovat anomálie na výstupních datech aplikace `Snort`. Dále zde bude popsáno, jak bude toto rozšíření integrováno do nástroje `Snort`. Dále bude popsáno jak tato výstupní data (toky) budou rozšířeny o další informace např. o informace o jejich geolokaci, nebo např. jaká aplikace daný tok využívá ke své komunikaci. Jako poslední zde budou popsány techniky strojového učení, které budou pro tuto detekci použity.

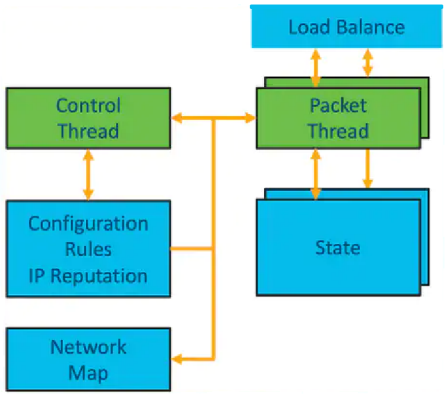
## 5.1 Úprava a rozšíření architektury nástroje Snort

Jak již bylo popsáno v sekci 3.3.1. `Snort` je `open-source` IDS nástroj, který umí pracovat i v režimu IPS. Jelikož je zdrojový kód tohoto nástroje volně dostupný, lze tak nad nástrojem implementovat libovolné úpravy. Aktuální verze zdrojového kódu viz [22]. Architektura aplikace `Snort` již byla znázorněna na obrázku 3.2 na tomto obrázku můžeme vidět jak všechny vstupní data prochází sekvenčně jednotlivými moduly. Tyto moduly pak budou dále rozšířeny a výsledkem bude následující pipeline:

1. **Sniffer:** zde je paket načten tak jak je. Tato část nástroje není potřeba dále upravovat.
2. **Preprocessor:** zde je paket upraven na čitelnou podobu pro další analýzu. Výstup tohoto preprocesoru pak půjde dále do detekčního enginu a do nově implementovaného externího modulu, který si data upraví vhodně tak, aby nad nimi šly použít jednotlivé metody strojového učení pro detekci anomálií.
3. **Modul pro přípravu dat pro ML:** Tento modul bude nově implementovaný a bude se starat o vhodné zpracování dat, jejich rozšíření a následné označení, a tato data budou dále předána modulu pro trénování modelu umělé inteligence.
4. **Detection Engine:** zde je paket porovnán s množinou pravidel a je vyhodnocen. Tato část zůstane stejná.
5. **ML externí modul:** Tento modul bude nově implementovaný a bude se starat o výslednou predikci a následně možného hlášení události, na základě vstupních dat, které mu budou předány.
6. **Alerts/Logging:** v této části dochází ke hlášení narušení a logování.

7. **Poslední fáze:** Výsledek zaeeviduje do databáze či výstupního souboru.

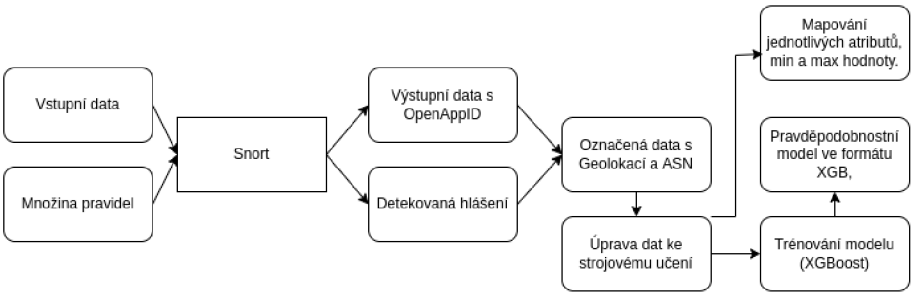
Co ovšem na obrázku 5.3 schází je to, že od verze č. 3 je aplikace `Snort` vícevláknová, to jak probíhá zprácování paketů více vlákny je znázorněno na obrázku 5.1. Na tomto obrázku můžeme vidět, že jednotlivé pakety jsou rozděleny mezi více vláken, a to tak, aby vytížení vláken bylo optimální, o tuto vlastnost se stará modul s názvem `Load Balance`. Jednotlivé pakety nejsou zpracovávány sekvenčně, ale každému paketu je přiřazeno jedno vlákno a na tomto vlákně proběhne porovnání s nastavenými pravidly a následné vyhodnocení škodlivosti. Jelikož tento externí modul pro detekci anomálií je implementován na základě metody `EventHandler` není potřeba se starat o správnou práci s paralelizaci daného rozšíření a tuto funkci vyřeší aplikace `Snort` za nás.




Obrázek 5.1: Vícevláknové zpracování v aplikaci Snort. [20]

## Architektura sběru dat

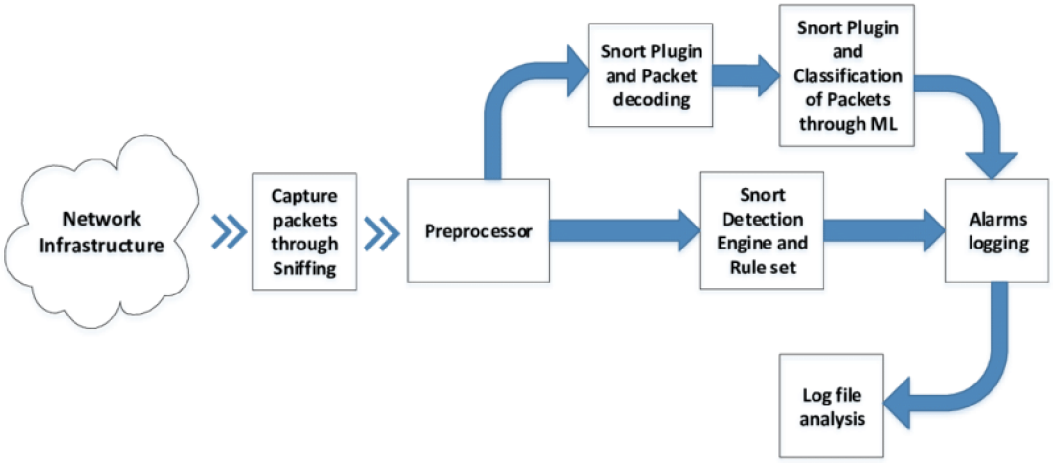
Na obrázku 5.2 je znázorněn vlastní návrh architektury sběru dat a následné trénování modelu. Jako první je potřebné zvolit vhodná vstupní data a získat použitelnou množinu pravidel. Množina pravidel a vstupní data následně budou vstupem do aplikace `Snort`. Aplikace `Snort` do souboru uloží síťové toky s informací o `OpenAppID` ve formátu `JSON` a detekované události ve formátu `CSV`. Díky těmto dvoum výstupům bude možné data označit a následně jednotlivým tokům přiřadit informace o jejich geolokaci a autonomním systému. Takto získaná data bude potřeba vhodně upravit a normalizovat ke strojovému učení. Následně bude použita knihovna `XGBoost` a výstupem bude pravděpodobnostní model.




Obrázek 5.2: Architektura sběru dat a trénování modelu.

## 5.2 Zasazení modulu

Toto rozšíření je tedy nutné vhodně zasadit jako nový modul do již existující implementace. Zasazení modulu do aplikace je znázorněno na obrázku 5.3. Ideálně by rozšíření běželo paralelně s detekcí paketu oproti množině pravidel. Rozšíření si převezme data z vnitřních struktur aplikace `Snort`, Tato data budou následně dekódována, budou vybrány vhodné atributy ke strojovému učení a k jednotlivým instancím dat se přidají další informace, které o nich dokážeme pomocí dalších nástrojů zjistit. Podrobnější popis bude v další sekci.




Obrázek 5.3: Architektura rozšíření programu Snort převzato z [20].

Pro detekci anomálií je nutné zvolit vhodné atributy z jednotlivých paketů. v tabulce 5.1 jsou uvedeny atributy pro vytváření toků, pro detekci anomálií však atributy jako je např. IP adresa nejsou vhodné. pro natrénování modelu je tyto atributy potřeba později odstranit. Pomocí těchto atributů je pak možné vytvořit dataset a natrénovat tak model, v tomto případě však budou ještě k jednotlivým tokům přidány další informace.

| # | Atribut | Český překlad |
|---|---|---|
| 1 | Date first seen | Datum prvního zjištění |
| 2 | Duration | Trvání |
| 3 | Transport protocol | Transportní protokol |
| 4 | Source IP address | Zdrojová IP adresa |
| 5 | Source port | Zdrojový port |
| 6 | Destination IP address | Cílová IP adresa |
| 7 | Destination port | Cílový port |
| 8 | Number of transmitted bytes | Počet přenesených bytů |
| 9 | Number of transmitted packets | Počet přenesených paketů |
| 10 | TCP Flags | TCP příznaky |

Tabulka 5.1: Vytvoření toků podle [18]

Jelikož zde data zpracovává `preprocessor` aplikace Snort. Je potřeba data uvedená v tabulce 5.1 převzít z jejich interní struktury.

### 5.2.1 Vektor parametrů

V tabulce 5.2 jsou popsány jednotlivé vstupní parametry, které budou vstupem pro metody strojového učení. Tyto parametry se skládají jednak z dat, které poskytuje sám `Snort`, dále jsou zde rozšířená data z `OpenAppID`, a taky data z databází `GeoLite2`. Atributy toku byly vybrány na základě článku [18], příznaky o komunikujících zemích a všechny příznaky o `OpenAppID` byly vybrány na základě konzultace, jelikož se osvědčily jako vhodné již v mé předchozí závěrečné práci.

| # | Atribut | Český překlad | Typ |
|---|---|---|---|
| 1 | server_bytes | Počet přenesených bytů na serveru | Integer (Float) |
| 2 | http_user_agent | User-Agent z HTTP požadavku | String |
| 3 | client_pkts | Počet paketů od klienta | Integer |
| 4 | apps_payload | Payload aplikace | String |
| 5 | server_pkts | Počet paketů od serveru | Integer |
| 6 | http_referrer | HTTP odkazování | String |
| 7 | user_info_username | Uživatelské jméno | String |
| 8 | dns_host | DNS host | String |
| 9 | netbios_info_netbios_domain | NetBIOS doména | String |
| 10 | proto | Transportní protokol | String |
| 11 | pkt_time | Čas paketu | Float |
| 12 | client_bytes | Počet přenesených bytů od klienta | Integer |
| 13 | client_info_version | Verze klienta | String |
| 14 | http_host | Host HTTP požadavku | String |
| 15 | apps_referred | Odkazovaná aplikace | String |
| 16 | service_info_vendor | Dodavatel služby | String |
| 17 | apps_client | Klient aplikace | String |
| 18 | service_info_subtype_version | Verze podtypu služby | String |
| 19 | user_info_id | ID uživatele | Integer |
| 20 | apps_misc | Ostatní aplikace | String |
| 21 | apps_service | Služba aplikace | String |
| 22 | service_info_port | Port služby | Integer |
| 23 | service_info_subtype_service | Podtyp služby | String |
| 24 | service_info_version | Verze služby | String |
| 25 | tls_host | TLS host | String |
| 26 | user_info_login_status | Stav přihlášení uživatele | String |
| 27 | total_flow_latency | Celková prodleva toku | Float |
| 28 | netbios_info_netbios_name | NetBIOS jméno | String |
| 29 | http_httpx_stream | Streamování HTTP | String |
| 30 | http_response_code | HTTP kód odpovědi | Integer |
| 31 | client_info_port | Port klienta | Integer |
| 32 | http_url | URL HTTP požadavku | String |
| 33 | city | Město | String |
| 34 | country | Země | String |
| 35 | asn_number | ASN číslo | Integer |
| 36 | asn_name | ASN jméno | String |

Tabulka 5.2: Atributy s českým překladem a typem

### 5.2.2 Vstupní data

Vstupní data budou použita z vědecké skupiny `Stratosphere` viz - [21]. `Stratosphere` poskytuje síťová data ve formátu `pcap`. Jejich poskytované datasety se rozdělují na 3 skupiny.

- **Malware captures:** Skupina datasetů obsahující škodlivou (anomální) komunikaci po síti. Jedná se o zachycení komunikace na sítí `Botnet` a je zde k dispozici 350 různých `pcap` souborů.
- **Normal captures:** Tato skupina obsahuje neškodlivou komunikaci po síti.
- **Mixed captures:** Skupina obsahující jak škodlivou, tak i neškodnou komunikaci po síti. Převažuje zde z větší části komunikaci neškodlivá.

### 5.2.3 OpenAppID

OpenAppId[1] je aplikační síťový bezpečnostní modul pro aplikaci Snort. OpenAppID od společnosti Cisco pomáhá zlepšit povědomí o aplikacích tím, že umožňuje uživatelům Snort detekovat, sledovat a spravovat využití aplikací v jejich síti, což umožňuje Snort používat jako open-source přizpůsobitelný aplikační firewall. Tento modul bude použit k přiřazení jednotlivých aplikací k instancím toku. Na tabulce 5.2 lze již vidět parametry, o které toky budou rozšířeny. Původně OpenAppId poskytovala pouze informace, které jsou uvedeny níže.

```
statTime="1393807860",appName="doubleclick",txBytes="5543",rxBytes="2598"
```

`statTime` zde udává, v jaký čas začala daná aplikace komunikovat, `appName` udává název aplikace, zde se může jednat o škodlivou aplikaci `doubleclick`, jelikož se jedná o reklamní server. Poslední dva parametry udávají velikost přijatých a odeslaných bytů.

Jelikož je ale modul `OpenAppId` v pokročilejší verzi je možné z něj získat daleko více informací. Zde uvedu podrobnější popis.

- `HTTP` - Podrobné informace o `HTTP` požadavku.
  1. `User agent` - agent `HTTP` požadavku.
  2. `URL` - `URL HTTP` požadavku.
- `apps` - Podrobné informace o použité aplikaci.
  1. `payload` - Obsah samotné aplikace.
  2. `client` - Název použitého klienta.
- `user info` - Informace o uživateli.
  1. `login status` - Informace o přihlášení.
  2. `username` - Přihlašovací jméno uživatele.
- `netbios info` - Informace o NetBIOS.
  1. `netbios name` - Název NetBIOS.
  2. `nebios domain` - Doména NetBIOS.

[1] Odkaz na nástroj OpenAppID

### 5.2.4 Informace o geolokaci

Ke každé instanci toku je možné přidat informaci o tom, z kama daný tok pochází, k tomu je možné použít hned několika nástrojů. Nejjednodušší je použití databáze Whois, která je veřejná a volně dostupná, poskytuje kompletní evidenci o majitelích domén a IP adres. Dalším možným nástrojem je databáze GeoIP, tato databáze je sice placená, ale její méně přesná kopie GeoLite je zdarma, bohužel poskytuje přesné záznamy jen pro některé IP adresy. Verze GeoLite2 však poskytuje informace o anonymních IP adresách a ASN systémech, proto je vhodná pro účely této práce. Jelikož se jedná o offline databáze, nemůže tak nastat problém s limitací počtu dotazů či nefunkčnost samotného dotazu. Dotaz bude proveden do lokální databáze a odpověď je okamžitá bez nutnosti připojení k internetu. Databáze `GeoLite2` se skládá ze dvou hlavních databází, a to:

1. `GeoLite2 ASN` - Databáze, která poskytuje k jednotlivým IP adresám informaci o autonomním systému, v případě této práce budou použity atributy číslo autonomního systému a název autonomního systému.

2. `GeoLite2 Country` - Databáze poskytující informace o zemích a městech z kama daná IP Adresa pochází, v této práci budou použity dva parametry, a to země a město odkud daná `IP Adresa` pochází.

### 5.2.5 Označení dat

Získané síťové toky bylo potřeba označit, a to bylo provedeno následujícími dvěma způsoby.

#### Označení dat pomocí čených listin

Další velmi důležitou přidanou hodnotou k jednotlivým tokům může být porovnání IP adresy instance toku oproti databázi blacklistovaných IP adres. Jedna z nejdůležitějších databází čených listin je například projekt `NERD` od společnosti `CESNET`, jedná se o databázi reputace síťových entit. Vytvoření databáze černých listin proběhne spuštěním skriptu `blacklist.py`, který bude dále zmíněn v kapitole 6. Tento skript naplní databází typu `postgres` daty o doménách a `IP Adresách`, které budou získány z důvěryhodných zdrojů.

#### Označení dat pomocí událostí

Data budou dále označena na anomální či normální, podle hlášení poskytnuté samotnou aplikací `Snort`. Budou stažena pravidla, která jsou dostupná pro registrované uživatele, tato množina obsahuje okolo 40000 pravidel, které budou využity na detekci škodlivých toků ve vstupních datech. Tato hlášení budou poté prozkoumána a pro každý vstupní soubor budou sesbírána data o tocích a jednotlivých hlášení, kdy následně budou nasbírané toky označeny za škodlivé pokud se shodují s daným hlášením.

### 5.2.6 Učení s učitelem nad označenými daty

Poté, co budou data správně označena, bude zvolen vhodný model pro trénování. Pro trénování a testování přesnosti modelu bude použit jazyk `python` s knihovnou pro strojové učení. Bude použita funkce `GridSearchCV` pro nalezení nejlepších parametrů pro trénování modelu, a tyto parametry budou použity jako finální pro predikční model. Následně tento nejlepší model bude použit pro predikci na testovacích datech a provedou se experimenty a vyhodnocení na matici záměn.

# Kapitola 6

# Implementace

## 6.1 Instalace a konfigurace aplikace Snort

Pro integraci tohoto rozšíření byla zvolena nejnovější verze nástroje `Snort`, a to přímo z jejích oficiálního repozitáře na Githubu[1]. Po stažení tohoto repozitáře, bylo potřeba aplikaci správně nainstalovat, to zahrnovalo doinstalování všech potřebných knihoven. Hlavní výhodou stáhnutí zdrojového kódu a následné instalace je možnost jakýchkoliv úprav.

### Konfigurační soubor snort.lua

Nejdůležitější součátí při používání aplikace `Snort` je soubor, který se nachází ve složce `etc/snort`, a to `snort.lua`, jedná se o `Lua` skript, ve kterém se povolují všechny možné pluginy, moduly, dále je zde potřeba uvést cesty k používaným pravidlům a zvolit vhodné hlášení detekovaných událostí.

### 6.1.1 Integrace modulu OpenAppId

`OpenAppId` je rozšíření, které je v této práci používáno a dodává k tokům jednotlivé informace jako je např.

- `service` - informace o službě (HTTP)
- `client` - Informace o použitém klientu (Chrome)
- `host` - informace o DNS (geo.yahoo.com)
- `version` - Informace o verzi použitého klienta (44.0.2403.107 - Chrome)

Těchto informacích je daleko víc. Aby toto rozšíření fungovalo je potřeba stáhnout soubor `snort-openappid.tar.gz`[2], který obsahuje detektory jednotlivých služeb, aktuálně dokáže rozpoznat více než 7000 aplikací. Dále je taky potřeba, jak již bylo zmíněno v sekci výše, nastavit v konfiguračním souboru `snort.lua` správnou cestu k těmto detektorům a toto rozšíření povolit.

[1] https://github.com/snort3/snort3
[2] https://www.snort.org/downloads/openappid/33380

```
appid =
{
    -- appid requires this to use appids in rules
    app_detector_dir = './'
}
```

Obrázek 6.1: Ukázka kódu pro použití appid.

### 6.1.2 Použití Snort Extra

Pro tvoření modulů komunita `Snort` vytvořila repozitář s názvem `snort3_extra`[3]. Díky tomuto repozitáři je tvoření nových pluginů do aplikace `Snort` jednodušší. Implementace nového modulu tak lze provést pomocí návodu, který je uveden v souboru `readme.txt`. V tomto repozitáři se nachází rozšíření `appid_listener`, které bylo rozšířeno o informace o jednotlivých tocích. Následně v tomto rozšíření jsou data upravená do vhodné podoby a jsou vyhodnocena pravděpodobnostním modelem [23].

### 6.1.3 Použitá pravidla

Pro sběr dat a jeho následující označení byla použita množina pravidel, která není volně dostupná. Pro veřejnost je možnost stáhnutí `snort3-community-rules.tar.gz`, to však pro tuto práci není dostačující, a proto byla zvolena rozšířená pravidla `snortrules-snapshot-31470.tar.gz`, která jsou dostupná pro registrované uživatele. Tato množina pravidel je velmi rozsáhlá a obsahuje následující položky:

- `includes.rules`
- `rulestates-balanced-ips.states`
- `rulestates-connectivity-ips.states`
- `rulestates-max-detect-ips.states`
- `rulestates-no-rules-active.states`
- `rulestates-security-ips.states`
- `snort3-app-detect.rules`
- `snort3-browser-firefox.rules`
- `snort3-pua-p2p.rules`
- `snort3-pua-toolbars.rules`
- `snort3-server-apache.rules`
- `snort3-server-mail.rules`
- `snort3-server-mssql.rules`
- A mnoho dalších pravidel...

[3] https://github.com/snort3/snort3_extra

## 6.2 Sběr dat

Pro sběr dat byl rozšířen externí modul `appid_listener`, který na počátku poskytoval pouze informace o OpenAppID (Informace o použité aplikaci v toku). Tento modul byl rozšířen o informace daného toku, které jsou uvedeny v hlavičkovém souboru `flow.h`. Dále tyto toky byly rozšířeny o informace jako je číslo autonomního systému, organizace autonomního systému, země a město, ze kterého tok komunikuje (IP Adresa).

### 6.2.1 Zvolená data

Pro vytoření počáteční datové sady byly vybrány soubory typu `pcap` od organizace `Stratosphere`, jedná se o věděckou skupinu z univerzity ČVÚT, která se zabývá kyberbezpečností. Jejich datové sady se dělí na

- `MALWARE CAPTURES` - Data obsahující výhradně malware.
- `NORMAL CAPTURES` - Data obsahující normální provoz.
- `MIXED CAPTURES` - Data obsahující anomální i normální provoz na síti.

Aby bylo zajištěno, že pro model umělé inteligence, který se bude trénovat, bude dostatečný výskyt anomálních instancí, byla zvolena data z kategorie `MIXED CAPTURES`[4]. Důležité při tvorbě těchto dat je zvolit taková data, která budou obsahovat alespoň podobný počet normálních a anomálních instancí.

### 6.2.2 Implementace sběru dat

Sběr dat byl implementován pomocí metody `EventHandler`, která jako vstupní parametry dostane `DataEvent o AppID` - informace o `AppID` eventu a `Flow` - informace daného toku. Ze struktury toku jsou následně extrahovany informace o:

- `client_pkts` - počet odeslaných paketů klientem.
- `server_pkts` - počet odeslaných paketů serverem.
- `client_bytes` - velikost odeslaných dat klientem.
- `server_bytes` - velikost odeslaných dat serverem.
- `total_flow_latency` - délka trvání toku.
- `proto` - typ protokolu.
- `client_ip` - IP Adresa klienta.
- `server_ip` - IP Adresa serveru.
- `client_port` - port klienta.
- `server_port` - port serveru.

Poté, co jsou zpracovány všechny atributy z toku, jsou dále zpracovány informace o AppID:

[4] https://www.stratosphereips.org/datasets-mixed

```
{"apps": {"service": "HTTP",
          "client": "Chrome",
          "payload": "unknown",
          "misc": null,
          "referred": null },
"proto": "TCP",
"client_info": {"version": "44.0.2403.107" },
"service_info": {"version": null,
                 "vendor": "ATS" },
"user_info": {"id": 0,
              "username": null,
              "login_status": "n/a" },
"tls_host": null,
"dns_host": null,
"netbios_info": {"netbios_name": null, "netbios_domain": null },
"http": {"httpx_stream": null,
         "host": "geo.yahoo.com",
         "url": "http://geo.yahoo.com/p?t=0.7395445310976356&_V=V&s=81121452&V",
         "user_agent": "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
            like Gecko) Chrome/44.0.2403.107 Safari/537.36",
         "response_code": "200",
         "referrer": "http://news.yahoo.com/" }}
```

Výpis 6.1: Ukázka dolovaných dat z OpenAppID

Na obrázku 6.1. lze vidět všechny informace, které jsou poskytnuty z modulu OpenAppID. Tyto informace jsou velmi důležité, jak pro trénování modelu, tak pro samotné pochopení dat. Na základě těchto přesných informací se pak natrénovaný model může snadno rozhodovat, zda instanci toku vyhodnotí jako anomální. Tyto toky jsou zpracovávány paralelně, a proto výstup z tohoto rozšíření může být pokaždé jiný, avšak data by měla být stejná, pouze mohou být v prohozeném pořadí, ale to na výslednou aplikaci nemá vliv.

### 6.2.3 Formát výstupu

Data o tocích a informace o OpenAppID, jsou poté ukládány do souboru, který je nastaven v konfiguračním souboru `snort.lua`, a to v části `appid_listener` pod proměnnou `file`, pro takový výstup je také potřeba nastavit proměnnou `json_logging` na hodnotu `true`. Data jsou ukládána ve formátu `json`, a to z toho důvodu, že jich může spousta chybět, ať už kvůli tomu, že třeba `DNS` dotaz nenese skoro žádné informace, které jsou obsaženy v `OpenAppID`, a taky proto, že je to následně jednodušší ke zpracování do datového rámce.

### 6.2.4 Zpracování výstupu

Výstupní `JSON` formát je dále zpracován skriptem v jazyce `Python`, Vyskytl se zde problém s kódováním, a proto bylo potřeba nalézt správné kódování, a to pomocí knihovny `codec`. Tato knihovna nalezla kódování tohoto souboru jako `Windows-1252`, a proto toto kódování je dále používáno ke zpracování všech dat. Jelikož u některých toků, jako je např `DNS` chybí spoustu informací, a v souboru `JSON` se tak hodnota tohoto atributu tváří jako prázdný řetězec je potřeba tento řetězec upravit. Tyto hodnoty byly nahrazeny řetězcem `unknown`, který bude dále použit pro všechny chybějící hodnoty pro atributy, které jsou nenumerické. Pro atributy, které se skládají z pole znaků, nebo-li řetězce je pak jednodušší pokud všechny

hodnoty atributů všech instancí nabývají nějaké hodnoty ve formátu řetězce, protože je pak jednoduché použít na taková data `LabelEncoder`, který bude dále popsán v textu a je v této práci hojně využíván.

### 6.2.5 Detekované události

Pro označení dat, které bude dále popsáno v následující sekci bylo potřeba rozumně nasbírat data o jednotlivých událostech a hlášení, ať už škodlivá hlášení či normální. Tato hlášení jsou vyprodukována díky pravidlům, které byly uvedeny v sekci 6.1.3. Pro získání takových hlášení je jednak potřeba mít množinu pravidel, a to nejlépe obrovskou množinu, jelikož v této práci byla vyzkoušena množina pravidel o velikosti 4000, a to kvalitativně vůbec nestačilo ke správnému označení škodlivých toků, tato komunitní množina pravidel hlásila hlavně 2 typy událostí a to `DNS Spoofing` - podvržení `DNS` dotazu a `UPNP Discovery`. Z tohoto důvodu byla vybrána množina pravidel, která je dostupná pro registrované uživatele a byla lépe popsána v sekci 6.1.3. Tato množina již na datech, na které se tato práce zaměřuje, detekovala kvalitativní, kvantitativní a rozmanité množství jednotlivých hlášení. Pro příklad:

- "SERVER-WEBAPP generic SQL select statement possible sql injection"
- "SERVER-OTHER Wavelink Emulation License Server HTTP header overflow attempt"
- "SERVER-WEBAPP CoreHTTP Long buffer overflow attempt"

## 6.3 Rozšíření dat

Dále bylo v této práci navrženo rozšířit jednotlivé toky o další informace. Ačkoli samotné IP Adresy v sobě nenesou žádnou vlastnost, která by strojovému učení mohla nějak v detekci pomoci, jsou v této práci aspoň použity k rozšíření vektorů příznaků pomocí geolokačních databází.

### Geolokace

Data uvedena v předchozí kapitole byla rozšířena o další atributy. Tyto atributy byly získány z databázi `GeoLite2`. Tato databáze poskytuje informace o geolokaci koncového uživatele na základě jeho IP adresy. Jelikož se v dolovaných datech před jejich odstraněním (IP adresy nejsou vhodné pro trénování modelu), lze pomocí jednoduchého dotazu do databáze získat další důležité atributy k trénováno modelu. Nejprve je potřeba tyto databáze stáhnout, a to na oficiálních stránkách společnosti `MaxMind`[5], kde jsou dostupné pro přihlášeného uživatele. Poté, co jsou tyto databáze staženy, jedná se o

- `GeoLite2 ASN` - Databáze obsahující informace o autonomních systémech.
- `GeoLite2 City` - Databáze poskytující informace o zemích.

Je potřeba nainstalovat knihovnu pro následující práci s nimi. Příklad pro jazyk python, kde lze knihovna jednoduše nainstalovat příkazem

```
pip3 install geoip2
```

[5] https://dev.maxmind.com/geoip/geolite2-free-geolocation-data

Následně je potřeba tuto databázi v kódu načíst a vytvořit instanci `Reader` pro čtení dat z této databáze. Následně je možné pomocí jednoduchého kódu např.

```
reader.city('203.0.113.0')
```

Vytvořit strukturu, která v sobě nese následující informace:

- `country.name` - Název země, odkud daná IP adresa pochází.
- `country.iso_code` - ISO kód země.
- `postal.code` - Poštovní směrovací číslo pro IP Adresu.
- `city.name` Název města pro danou IP adresu.

Pro databázi ASN (Autonomních systému) je inicializace struktury podobné, ale poskytuje další důležité informace.

- `autonomous_system_number` - číslo autonomního systému.
- `autonomous_system_organization` - Název organizace autonomního systému.

Tyto informace jsou při průchodu jednotlivých toků přidány do vektorů příznaků jako další vhodné atributy pro strojové učení. Důležité je zmínit, že pokud jsou poskytnutá data z této databáze numerická, stačí je pouze v další práci normalizovat, ovšem v případě pokud se jedná o názvy, je potřeba tyto názvy zakódovat na čísla, pomocí vhodného enkodéru.

## 6.4 Označení nasbíraných dat

Jelikož se tato práce zabývá hlavně detekcí anomálií na základě klasifikace, je potřeba získané instance dat rozšířit o tzv. skóre, které strojovému učení poskytuje informaci o tom, zda je tato instance při trénování anomální či nikoli. O rozšíření tohoto atributu, který bude nazván pro další používání `blacklisted` se budou starat dvě metody, které budou podrobně popsány v následujících dvou podsekcích.

### 6.4.1 Označení pomocí databáze černých listin

Pro správné označení dat se autor v práci rozhodl získat databázi černých listin obsahující `IP Adresy`. Tato databáze je vytvořená pomocí skriptu `blacklist.py` a tento skript mi byl poskytnut s licencí `GNU-GPL2` od konzultanta Ing. Petra Chmelaře. Tato databáze využívá ke své práci databázi typu `Postgres`. Skript `blacklist.py` Získává data z ověřených zdrojů, které si načítá z přiloženého souboru typu `csv`.

| Sloupec | Typ | Nullable |
|---|---|---|
| ip | character varying(255) | not null |
| sourceid | integer | not null |
| domain | character varying(1024)[] | |
| first_occ | timestamp with time zone | |
| update_occ | timestamp with time zone | |
| no_occ | timestamp with time zone | |

Tabulka 6.1: Tabulka pro IP adresy

V tabulce 6.1 uvedené výše lze vidět schéma dané tabulky, která je vytvořena pro každou získanou `IP Adresu`. Tato tabulka obsahuje důležité sloupce o informacích, jako je `ip` samotná `IP Adresa` uložena v datovém typu řetězec charakterů, dále obsahuje informaci `sourceid` - zdrojové id, a také obsahuje sloupec `domain` - doména patřící k dané `IP Adrese`. Díky této databázi jsou externí `IP Adresy` porovnány s hodnotami ve vytvořené databázi a pokud je nalezena shoda, tak je tok následně označen jako škodlivý.

### 6.4.2 Označení podle hlášení

Dále bylo pro správné označení dat využito hlášení, které generuje aplikace `Snort`, pro jednotlivé instance dat. Hlášení v tomto programu se dělí na následující typy:

- `alert_fast`
- `alert_full` - Hlášení obsahující všechny možné data o události
- `alert_csv` - Hlášení ve formátu `csv`
- `alert_syslog` - Zasílání hlášení na `syslog` server

Pro bližší zkoumání chování vhodně zvolených souborů typu `pcap` bylo zvoleno použití typu `alert_full`, jelikož poskytuje kompletní informaci o dané události a je tak jednoduché se zorientovat nad tím, co se v danou chvíli na síti odehrálo. Při zkoumání vybraných dat jsem narazil na následující hlášení:

```
[**] [1:254:16] "PROTOCOL-DNS SPOOF query response with TTL of 1 min.
and no authority" [**] [Classification: Potentially Bad Traffic]
[Priority: 2] [AppID: DNS] 07/27-00:06:23.039570
8.8.8.8:53 -> 10.0.0.45:57887
UDP TTL:49 TOS:0x0 ID:42853 IpLen:20 DgmLen:108 Len: 80
```

Výpis 6.2: Ukázka falešně pozitivního hlášení z alert full

Hlášení uvedeno výše oznamuje, že při detekci dat oproti množině pravidel došlo k možnému útoku `DNS` spoof, jelikož se ale v přiložené události vyskytuje `IP Adresa` 8.8.8.8, která patří nejpoužívanějšímu `DNS` serveru googlu, je velmi málo pravděpodobné, že se jedná o anomální instanci. Z tohoto důvodu bylo potřeba více prozkoumat dané hlášení a vybrat jenom ty, které jsou také spíše spekulativně škodlivé, ale s větší mírou pravděpodobnosti.

```
[**] [1:20620:7] "SERVER-WEBAPP CoreHTTP Long buffer overflow attempt" [**]
[Classification: Attempted User Privilege Gain] [Priority: 1] [AppID: Chrome]
07/28-08:17:48.945134 10.0.0.45:52606 -> 217.12.13.41:80
TCP TTL:128 TOS:0x0 ID:6737 IpLen:20 DgmLen:1021 DF
***AP*** Seq: 0xDE2C668E Ack: 0x877FF33E Win: 0x102 TcpLen: 20
```

Výpis 6.3: Ukázka pozitivního hlášení z alert full

Na výše uvedeném hlášení už se jedná s větší pravděpodobností o potenciální škodlivý úmysl ze strany odesílatele, a to napadnou klienta pomocí útoku typu `buffer owerflow` a získat tak vyšší práva, ke kterým by daný odesílatel neměl mít přístup.

Pro přiřazení událostí k jednotlivým tokům (instancím dat) byla použita metoda hlášení `alert_csv`, a to díky její kompaktnosti, zpracování a rychlosti. Jelikož lze formát typu `csv` lehce zpracovávat.

```
04/12-16:08:48.974694, 730, ICMP, 3, 160, C2S, 112.203.201.9:1, 192.168.50.92:1
04/12-16:08:49.235959, 741, ICMP, 3, 160, C2S, 124.120.192.198:1, 192.168.50.92:1
04/12-16:08:55.436153, 828, ICMP, 3, 160, C2S, 134.65.233.40:1, 192.168.50.92:1
04/12-16:09:00.425649, 921, ICMP, 3, 160, C2S, 110.137.156.39:1, 192.168.50.92:1
04/12-16:09:06.257212, 928, UDP, 3, 132, C2S, 46.219.211.227:80, 192.168.50.92:80
```

Výpis 6.4: Ukázka hlášení z alert csv

Na obrázku výše lze vidět příklady hlášení ve fomátu `csv`. Tyto hlášení jsou dále zpracovány do seznamu jednoduchých struktur obsahující:

- `Čas příchodu paketu` (1. sloupec na obrázku výše).
- `Typ protokolu` (3. sloupec na obrázku výše).
- `Zdrojová IP Adresa` (7. sloupec).
- `Cílová IP Adresa` (8. Sloupec).

Díky těmto prvkům v dané struktuře lze jednoduše zjistit, který tok takové hlášení vyvolal, porovnáním společných atributů.

## 6.5 Zpracování dat ke strojovému učení

Jelikož data v původní podobě, která jsou nasbírána z modulu pro sběr dat nejdou přímo použít jako vstupní data pro strojové učení, je třeba tyto data následovně upravit. Chybějící nenumerická data je potřeba zakódovat na slovní hodnotu, dále bylo potřeba tyto nenemurická data upravit na slova a následně všechna tato data převést na typ `float`, což bude podrobněji popsáno v následujících podsekcích.

### 6.5.1 Zpracování chybějících hodnot

V nasbíráných datech je u určitých atributů velký výskyt chybějících dat. Například `ICMP` dotaz v sobě nenese informaci o tom, jakou aplikaci podle modulu `OpenAppID` použil, nebo dotaz typu `DNS` v sobě tyto informace také nenese. Z tohoto důvodu bylo potřeba následující chybějící data nahradit za stejnou slovní hodnotu, která v tomto případě představuje slovo `"unknown"`. Na tabulce 6.2 lze vidět pro kolik instancí z dané datové sady, bylo potřeba doplnit chybějící hodnoty. Toho bylo docíleno pomocí následující úpravy.

```
for column in non_numeric_columns:
    df[column].fillna('unknown', inplace=True)
```

Výpis 6.5: "Ukázka doplnění chybějících hodnot"

| Hodnota atributu 'apps_service' | Počet výskytů |
|---|---|
| IGMP | 17182 |
| MDNS | 16178 |
| HTTPS | 15274 |
| DHCP | 13355 |
| ICMP | 10058 |
| DNS | 10027 |
| NetBIOS-ns | 117 |
| WSDD | 107 |
| TeamSound | 99 |
| NTP | 95 |
| HTTP | 92 |
| NNTP | 50 |
| STUN | 30 |
| QUIC | 15 |
| Ganglia | 10 |
| RTMP | 9 |
| Skype Auth | 7 |

Tabulka 6.2: Počet výskytů hodnoty 'unknown'

| Nenumerický atribut | Počet unikátních hodnot |
|---|---|
| http_user_agent | 151 |
| apps_payload | 1624 |
| http_referrer | 1519 |
| dns_host | 5727 |
| proto | 5 |
| client_info_version | 26 |
| http_host | 2323 |
| apps_referred | 87 |
| service_info_vendor | 145 |
| apps_client | 25 |
| service_info_subtype_version | 21 |
| apps_service | 28 |
| service_info_subtype_service | 9 |
| service_info_version | 335 |
| tls_host | 968 |
| http_url | 56671 |
| city | 594 |
| country | 98 |
| asn_name | 954 |

Tabulka 6.3: Počet unikátních hodnot nenumerických atributů

### 6.5.2 Převod nenumerických hodnot

Na tabulce 6.3 lze pozorovat dva sloupce, první sloupec s názvem **Nenumerický atribut** a druhý sloupec s názvem **Počet unikátních hodnot**. Tato tabulka znázorňuje počet neunikátných hodnot k jednotlivým sloupcům, které bylo potřeba zakódovat do číselného formátu `float64`, aby bylo možné následně tato data normalizovat. Po zakódování nenumerických atributů bylo potřeba ke každému uložit tzv. `mapping`, který je použit při následné predikci, kdy všechny nenumerické atributy v instanci se nahradí za stejnou hodnotu, jako tomu bylo v počátečním trénování modelu a hodnota, který nebude známá ponese hodnotu `"unknown"`.

```
from sklearn.preprocessing import LabelEncoder
import joblib

label_encoder = LabelEncoder()
mappings = {}

for column in non_numeric_columns:
    df[column] = label_encoder.fit_transform(df[column])
    mappings[column] = {value: label for label,
                        value in enumerate(label_encoder.classes_)}
    filename = f"{column}_mapping.pkl"
    joblib.dump(mappings[column], "./mapping/" + filename)
```

Výpis 6.6: Převod nenumerických hodnot a uložení mapování

Na výpisu 6.6 lze vidět vytvoření instance `LabelEncoder`, a slovníku `mappings`. Tato instance `LabelEncoder` postupně zakóduje všechny hodnoty v nenumerických atributech na stejné číslo a následně je pomocí funkce `dump` z knihovny `joblib` uloží pro další zpracování.

### 6.5.3 Normalizace dat

Jelikož data obsahují atributy s různým formátem číselného typu, bylo potřeba tyto hodnoty sjednotit do jednoho formátu a to `float64`, poté co jsou všechny atributy homogenního typu byla provedena normalizace. Normalizaci jsem se rozhodl zvolit typu `MinMax`. Jelikož se bude trénovaný model dále používat pro predikci, je potřeba si maximální a minimální hodnoty pro každý jeden atribut uložit, to je opět provedeno pomocí knihovny `joblib` a funkce `dump`.

**Algorithm 1** Min-Max Normalizace

**Vstup :** pole *data* s hodnotami pro normalizaci
**Výstup:** pole *data* s normalizovanými hodnotami

1 $min\_val \leftarrow \min(data)$ $max\_val \leftarrow \max(data)$
2 **for** $i \leftarrow 0$ **to** $length(data) - 1$ **do**
3 $\quad normalized\_value \leftarrow \frac{data[i]-min\_val}{max\_val-min\_val}$
4 $\quad data[i] \leftarrow normalized\_value$
5 **return** *data*

Algoritmus 6.5.3 popisuje, jak tato normalizace probíhá, v této implementaci to znamená, že pro všechny atributy v datové sadě převede stávající hodnotu hodnotou, která je vypočítáná pomocí hodnoty samotné a maximalní a minimální hodnoty pro daný atribut.

```
import pandas as pd
import joblib

def min_max_scale_dataframe(df):
    min_vals = df.min()
    max_vals = df.max()

    normalization_values = {}
    for column in df.columns:
        normalization_values[column] =
            {'min_val': min_vals[column], 'max_val': max_vals[column]}

    scaled_df = (df - min_vals) / (max_vals - min_vals)
    return scaled_df, normalization_values

joblib.dump(normalization_values, 'normalization_values.pkl')
```

Výpis 6.7: Normalizace a ukládání hodnot pro normalizaci

Data jsou v tuto chvíli plně připravena ke strojovému učení, a taky k samotné predikci, jelikož při predikci potřebujeme znát původní mapování slov na jednotlivá čísla a hodnoty potřebné pro samotnou normalizaci.

## 6.6 Trénování modelu

Po přípravě dat bylo potřeba natrénovat správny model. Modelu bylo poskytnuto na trénování celkově 38 vstupních atributů a půl milionu instancí jednotlivých toků. Předtím než se samotný model dá trénovat je potřeba udělat nějaké práce na připravené datově sadě. To zahrnuje:

- Oddělit vstupní parametry (učící) od výstupních (skóre).
- Rozdělit datovou sadu na trénovací a testovací.
- Zvolit správny poměr trénovacích a testovacích instancí.

Jako trénovácí model byl zvolen `Decision Tree Classifier`. po vytvoření instance tohoto modelu bylo potřeba následně zvolit vhodné parametry tohoto klasifikátoru, jelikož se tyto parametry mohou různě kombinovat a těchto kombinací může být v řádech stovek. Byla použita funkce `GridSearchCV` z knihovny `sklearn.model_selection`.

```
param_grid = {
    'criterion': ['gini', 'entropy'], # Kriterium pro mereni kvality deleni
    'splitter': ['best', 'random'], # Strategie pro vyber deliciho bodu
    'max_depth': [None, 5, 10, 15], # Maximalni hloubka stromu
    'min_samples_split': [2, 5, 10], # Minimalni pocet vz. pro deleni uzlu
    'min_samples_leaf': [1, 2, 4] # Minimalni pocet vzorku v listech stromu
}
```

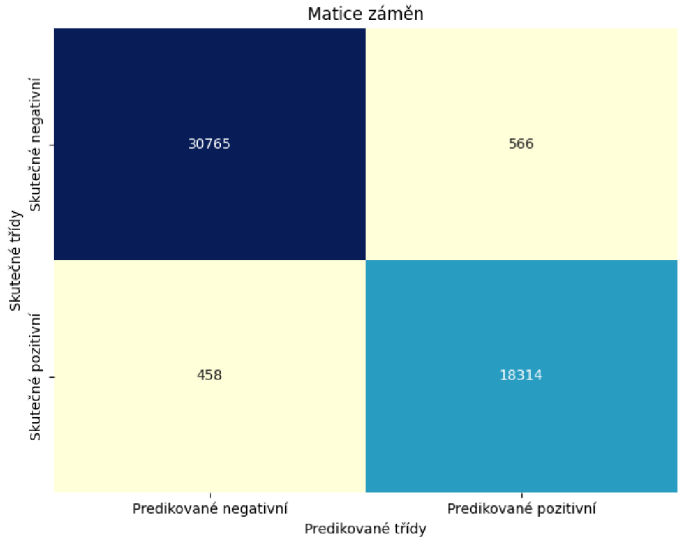
Výpis 6.8: Parametry DecisionTreeClassifier

Na výpisu 6.8 lze vidět slovník obsahující různé hodnoty parametrů, které chceme otestovat při ladění modelu `DecisionTreeClassifier` pomocí metody `GridSearchCV` z knihovny `scikit-learn`.

Pro získání nejlepšího modelu bylo potřeba provést následující kroky:

- `GridSearchCV(model, param_grid)` - Volání funkce s instancí modelu a slovníkem parametrů.
- `grid_search.fit(X_train, y_train)` - Zavolání funkce fit na modelu se všemy parametry.
- `best_model = grid_search.best_estimator_` - Uložení nejlepšího modelu.

### 6.6.1 Matice záměn




Obrázek 6.2: Matice záměn počátečního modelu

Na obrázku 6.2 lze vidět matici záměn počátečné natrénovaného modelu, který byl natrénován na 70 procentech dat a na zbylém vzorku dat testován. Ohodnocení tohoto

modelu na testovacích datech bylo **97,9%**. Predikovala 566 falešných pozitiv a 458 falešných negativ.

### 6.6.2 Finální model

Jelikož tento počáteční model se mi zdál ne příliš přesný a při experimentovaní s ním nevykazoval dobré výsledky, bylo mi navrženo použít jinou knihovnu pro strojové učení a natrénovat jiný a nový model.

Po konzultaci s vedoucím bylo doporučeno vyzkoušet knihovnu `xgboost` a model `xgb_gpu.XGBClassifier(tree_method='gpu_hist')`. `gpu` v názvu tohoto modelu pouze znamená, že trénování modelu probíhalo na grafické kartě, a to `RTX 3070`. Trénování modelu probíhalo v následujícím pořadí:

- Zavolání funkce `StratifiedKFold` z knihovny `sklearn`, tato funkce vytváří instanci křízové validace pro získání spolehlivějších odhadů.
- Zavolání funkce `GridSearchCV`, stejně jako u počátečního modelu, byla použita tato funkce, která pomocí funkce `fit`, hledá nejlepší parametry modelu pomocí předem definované tabulky parametrů.
- Následně byly naleznuty nejlepší hyperparametry pro tento model a daná data a tyto parametry jsou použity i pro finální model.
- Uložení finálního modelu pro další predikci a parametry.

```
param_grid = {
    'max_depth': [11, 12, 13, 14, 15, 16],
    'learning_rate': [0.1, 0.001, 0.0001],
    'n_estimators': [235, 240, 245]
}
```

Výpis 6.9: Tabulka parametrů pro XGBClassifier

Použitá tabulka hyperparametrů je uvedena ve výpisu 6.9. Je důležité zmínit že tuto tabulku je potřeba zvolit rozvážně, protože počet modelů k vyzkoušení násobně roste.

Výsledná kombinace nejlepších hyperparametrů byla následující:

```
{'learning_rate': 0.1, 'max_depth': 16, 'n_estimators': 240}
```

Výpis 6.10: Nejlepší parametry pro XGBClassifier

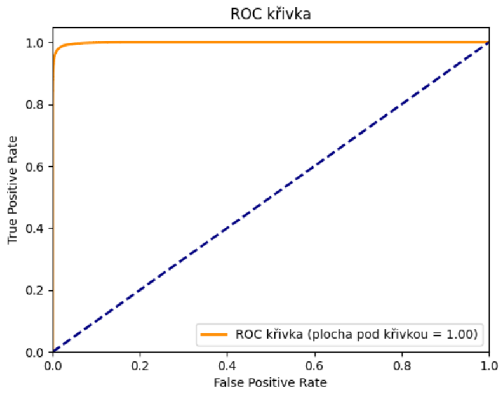
Model byl s těmito nejlepšími zvolenými parametry vyhodnocen na testovacích datech a vykazoval výsledky znázorněné na tabulce 6.4.

| | |
|---|---|
| Skóre F1 | 0.9690952511042246 |
| Skóre modelu | 0.986052866052866 |

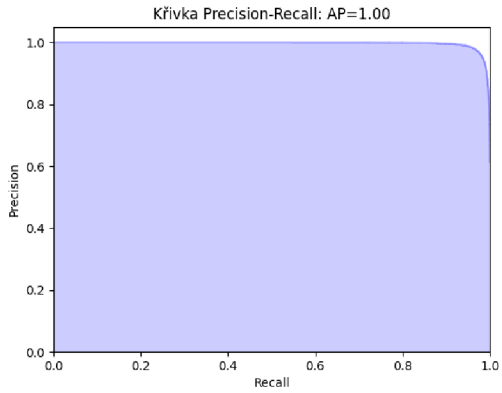
Tabulka 6.4: Skóre dosažené nejlepší kombinací hyperparametrů

| | |
|---|---|
| Skóre F1 | 0.9710952511042246 |
| Skóre modelu | 0.9858924858924859 |

Tabulka 6.5: Dosažené skóre na validačních datech

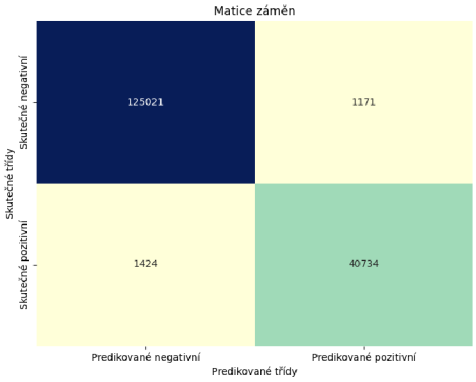



Obrázek 6.3: ROC křivka

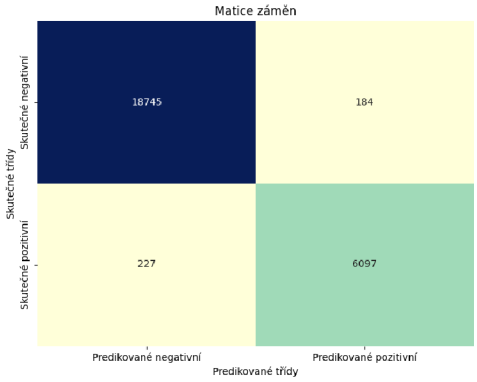



Obrázek 6.4: PR křivka

Na obrázku 6.3 lze vidět ROC křivka, kde modrá přerušovaná čára představuje predikční model, který by klasifikoval pouze na základě náhody, oranžová křivka představuje hodnoty implementovaného predikčního modelu a znázorňuje, že výstup implementovaného predikčního modelu je ideální. Na grafu 6.4 je znázorněna křivka `Precision Recall`, křivka je v případě natrénovaného modelu také ideální.




Obrázek 6.5: CM - testovací data




Obrázek 6.6: CM - validační data

Na obrázcích 6.6 a 6.5 lze vidět vyhodnocení modelu na testovacích a validačních datech, v případě dat testovacích model správně předikoval více než 161 tisíc instancí dat a pouze 1171 vyhodnotil jako falešná pozitiva a 1424 instancí vyhodnotil jako falešná negativa. Přesnost modelu na testovacích datech vykazuje hodnoty 98,5% a skóre `F1` činí 96,5%. Na validačních datech tento pravděpodobnostní model správně vyhodnotil 25 tisíc instancí dat a pouze 400 z těchto instancí klasifikoval do špatných tříd. Ohodnocení i `F1` skóre je podobné tomu, které bylo získáno na datech testovacích. Matice záměn na obrázích 6.5, 6.6 a 6.2 jsou uvedeny v této kapitole z toho důvodu, že se jedná stále o vyhodnocení na podobných datech jako jsou data trénovací. V kapitole 7 bude tento model řádně a lépe otestován na datech z reálného provozu.

## 6.7 Integrace predikčního modelu

Model je integrován do aplikace `Snort`, pomocí rozšíření `Snort3 Extra`, kde se modul nachází ve stejném zdrojovém souboru, kde je i sběr dat. Při vytváření toků a jejich předání strojovému učení jsou nejprve potřeba upravit do správné podoby. Před předáním samotné instance toku je nejprve třeba doplnit chybějící hodnoty na `"unknown"` a poté použít mapování, které bylo vytvořeno při vytváření počátečního modelu.

**Algorithm 2** Převod slovníkových hodnot na číselné

**Vstup :** Seznam atributů: ['http_user_agent', 'apps_payload', ...]
**Výstup:** Namapované atributy

```
for každý sloupec v ['http_user_agent', 'apps_payload', ...] do
    mapping ← load "init_model/mapping/"+ sloupec + "_mapping.pkl
     sloupec ← sloupec.map(mapping)
end
```

Výše je uveden algoritmus 2, který převádí jednotlivé nenumerické hodnoty na numerické, které byly použity při vytváření modelu, aby se zachovala správnost predikce pro nové neviděné data. Dále bylo potřeba tato data normalizovat pomocí původních hodnot.

### Použití predikčního modulu

Před samotným použitím nástroje Snort pro predikci anomálií je potřeba nastavit konfigurační soubor `snort.lua`, kde je potřeba přidat následující konfiguraci 6.11:

```
appid_listener = {
    anomaly_detection = true,
    json_logging = true,
    file = 'test.json'
}
```

Výpis 6.11: Nastavení pro detekci anomálií

Poté je možné pustit `Snort` na jakémkoli `pcap` souboru, nebo je možné predikovat anomálie přímo na síťovém rozhraní. Je ale potřeba uvést cestu k těmto externím pluginům.

```
/bin/snort -c "/etc/snort/snort.lua" -r "$file" --plugin-path "/lib/snort"
```

Výpis 6.12: Spuštění detekce nad souborem.

```
/bin/snort -c "/etc/snort/snort.lua" -i eno1 --plugin-path "/lib/snort"
```

Výpis 6.13: Spuštění detekce nad síťovým rozhraním.

Na výpisu 6.12 lze vidět, jak se rozšíření spouští pro soubor, může se jednat o soubor typu `pcap` a `pcapng`. Výpis 6.13 popisuje, jak spustit rozšíření nad síťovým rozhraním. Snort začne zachytávat síťové toky na daném rozhraní a bude na něm aplikovat detekci anomálií.

Predikce je umístěna ve zdrojovém souboru `appid_listener_event_handler.cc`, byla vytvořena funkce `model_predict`, která pro předem specifikovaný počet toků, kdy velikost toků byla vybrána na základě efektivnosti, zavolá skript `prediction.py` a upraví data, tak jak bylo zmíněno na začátku sekce a vypíše predikované události do souboru `alerts.json`.

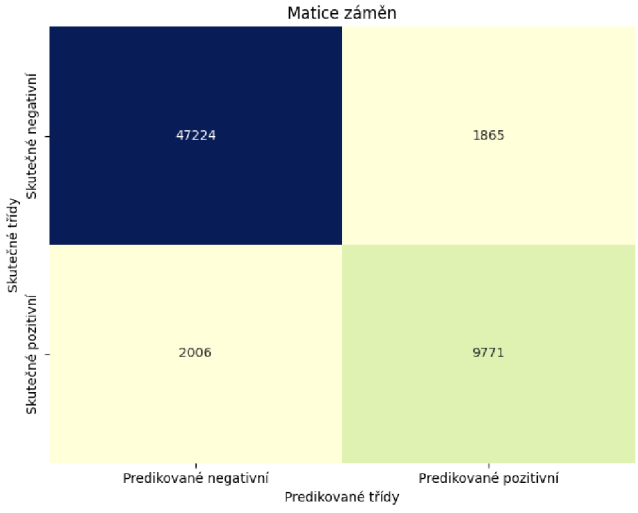
# Kapitola 7

# Experimenty

Tato kapitola popisuje vyhodnocení natrénovaného modelu na reálných datech běžného síťového provozu, která jsou volně dostupná a prezentuje jeho využitelnost. Tato kapitola dále popisuje přínos jednotlivých vstupních příznaků pro vytvořený predikční model a zabývá se možným vylepšením.

## 7.1 Vyhodnocení na normálních datech

Natrénovaný model byl vyhodnocen na datech, které poskytnula věděcká skupina stratosphere, a to na třídě normální dat, která by neměla obsahovat škodlivé instance toků, avšak `Snort` přesto hlásí velký počet událostí. První experiment proběhl na souboru `2017-05-01_normal.pcap`[1], který obsahuje běžný síťový přenos, který proběhnul na systému `Kali Linux`.

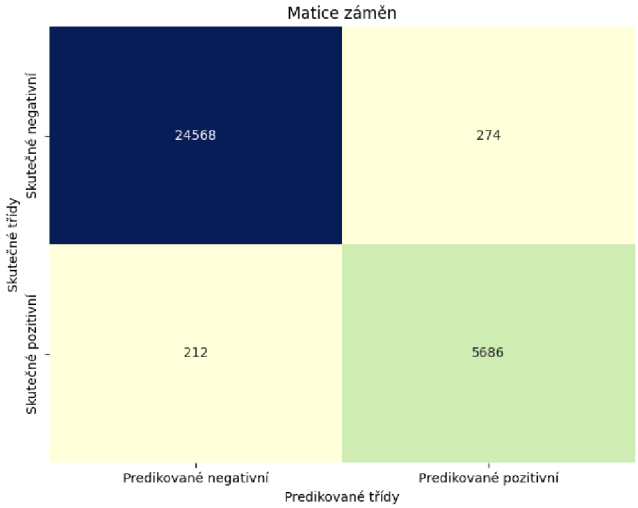



Obrázek 7.1: Matice záměn pro uvedený soubor.

Na obrázku 7.1 lze vidět matici záměn pro první testovaný síťový přenos. Jelikož jsou data méně podobná těm, na kterých byl původní model natrénován výsledek není úplně ideální. Skóre predikce v tomto případě je 93,6% a skóre F1 pouze 83,5%.

[1] https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-29/

Výsledek predikce se ale razantně změní, pokud se model natrénuje pouze na polovině vstupních dat.

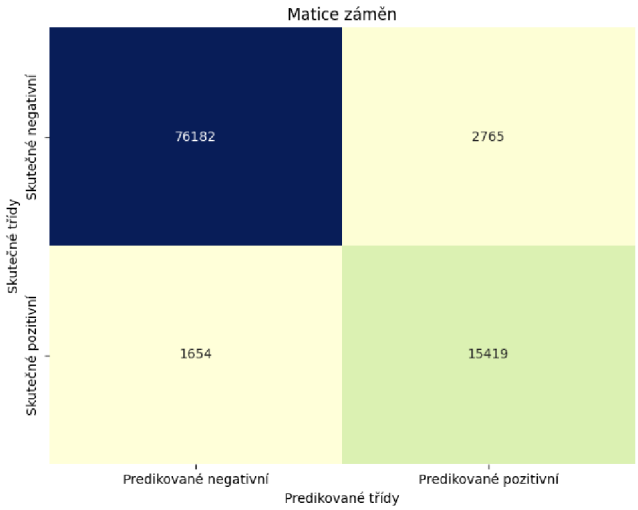



Obrázek 7.2: Matice záměn pro uvedený soubor po doučení modelu.

Na obrázku 7.2 lze pozorovat razantní zlepšení predikce modelu, a to pouze při doučení na polovině vstupních dat. Výsledné skóre modelu je 98,42% a skóre F1 dosahuje 95,9%.

### Druhé vyhodnocení na normálních datech.

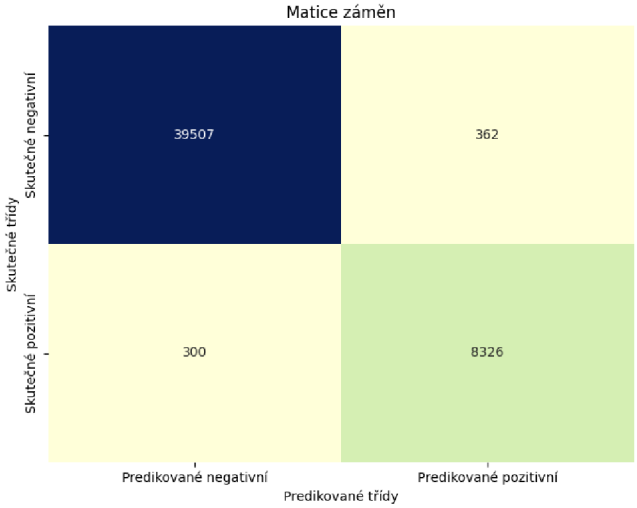
Druhý experiment proběhl na souboru `2017-05-01_normal.pcap`[2], který obsahuje běžný síťový přenos, který proběhnul na systému `Kali Linux`. Oproti prvnímu experimentu se tento soubor liší v samotné velikosti a celkové doby zachyceného síťového přenosu.




Obrázek 7.3: Matice záměn pro druhý experiment.

[2] https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-30/

Na obrázku 7.3 lze vidět vyhodnocení predikce počátečně natrénovaného modelu na síťovém provozu z výše uvedeného souboru. Skóre predikce v tomto případě je 95,4% a skóre F1 pouze 87,4%. Ohodnocení i skóre F1 je v tomto případě o něco lepší než u předešlého experimentu, a to časový úsek tohoto síťového provozu je o 2 hodiny delší.




Obrázek 7.4: Matice záměn pro druhý experiment po doučení modelu.

Na obrázku 7.4 lze pozorovat razantní zlepšení predikce modelu, a to pouze při doučení na polovině vstupních dat. Výsledné ohodnocení modelu je 98,64% a skóre F1 dosahuje 96,2%.

### Vyhodnocení na vlastních datech.

Dále proběhlo vyhodnocení na vlastním nasbíraném vzorku, který se nachzáí v souboru `DHT1.pcapng`. V této zachycené síťové komunikaci se převážně jedná o inicializaci klienta ke stahování `torrentů`, a to `BitTorrent`, kde s velkou pravděpodobností se nejedná o žádný škodlivý provoz, avšak nástroj `Snort` na použitých pravidlech hlásí 38 falešných pozitiv a čas vyhodnocení tohoto vzorku oproti pravidlům ukazuje 7,15 sekund.
Pokud spustím vyhodnocení na mnou vytvořeném predikčním modelu, model vykazuje přesně 0 detekovaných hlášení, což by se mělo jednat o spravné vyhodnocení a doba běhu programu je pouze 1 sekunda.

## 7.2 Vyhodnocení rychlosti predikce oproti pravidlům.

Na tabulce 7.1 Je znázorněno srovnání rychlosti predikce modelu a vyhodnocení oproti pravidlům, z tabulky můžeme vyčíst, že model dokáže dané soubory zpracovat a vypsat plné hlášení ve formě `JSON` 3x rychleji. Jedná se tak o důležitý přínos implementovaného přístupu, jelikož aplikace `Snort` má pří velkém průtoku dat tendenci zahazovat pakety.

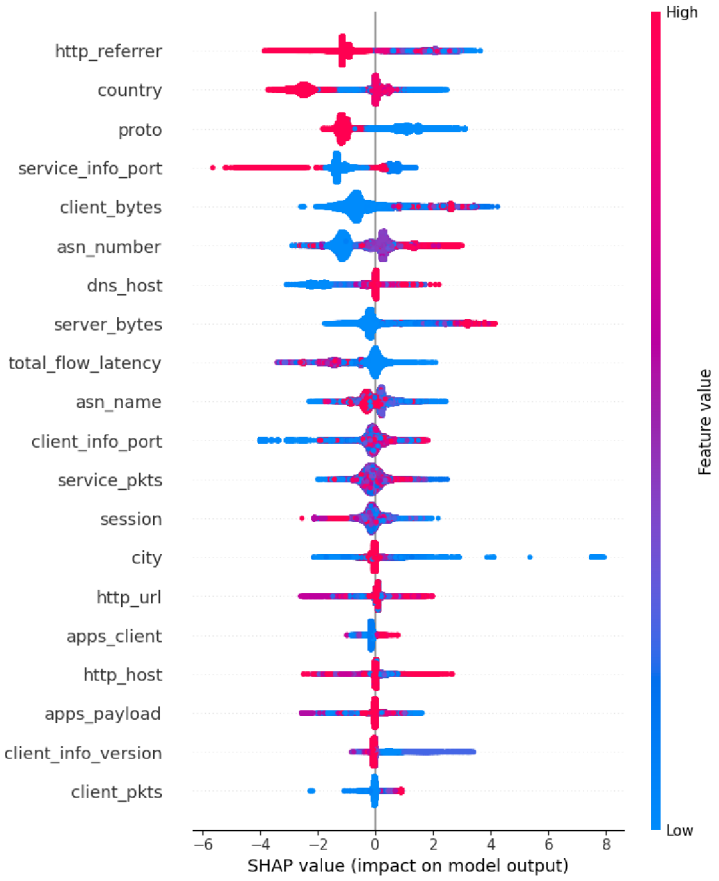
| Název souboru | Velikost souboru | Čas predikce modelu | Čas detekce na pravidlech |
|---|---|---|---|
| DHT1 | 8,1 MB | 1,1 s | 7,6 s |
| experiment1 | 457 MB | 9.9 s | 27,314 s |
| experiment2 | 873 MB | 14,927 s | 42,681 s |

Tabulka 7.1: Srovnání rychlosti predikce modelu, oproti vyhodnocení na pravidlech.

V tabulce 7.1 soubor `DHT1` představuje vlastní zachycený provoz popsaný v sekci výše, Soubory `experiment1` a `experiment2` jsou popsány v sekci 7.1 ve stejném pořadí.

## 7.3 Graf SHAP

Jelikož k tokům byly přidány rozšiřující atributy, bylo potřebné vyzkoumat, zda tyto atributy mají pro predikci modelu nějakou přidanou hodnotu. Na základě toho byl vytvořen `SHAP` graf, který můžete vidět na obrázku 7.5, který zdůrazňuje vliv jednotlivých atributů na následný výstup modelu.

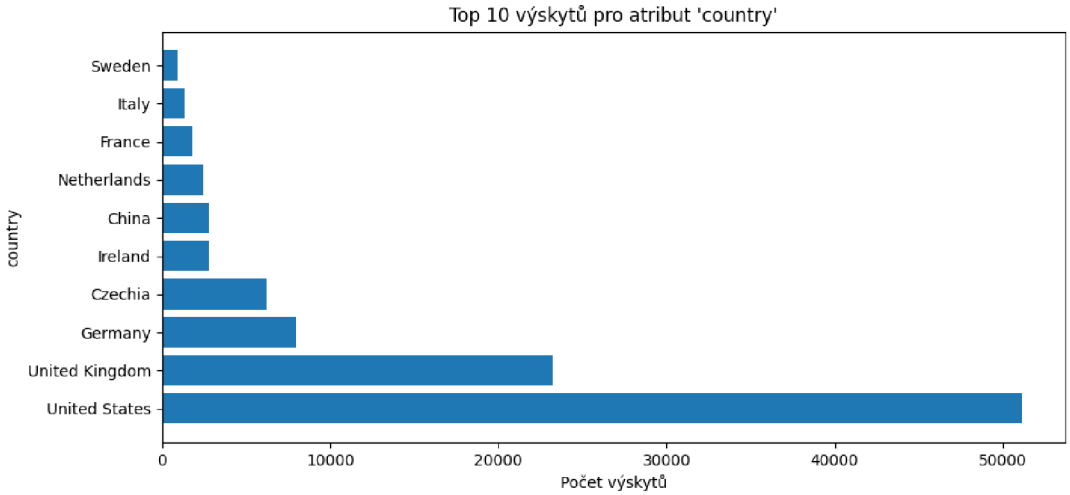



Obrázek 7.5: SHAP graf.

Důležité je poznamenat, že na základě grafu 7.5 opravdu mají přidané atributy značný podíl na výsledné predikci. Na prvním místě se nachází atribut `http_referrer`, který byl získán na základě informací z externího modelu `openAppID` a podle vyhodnocení se tak jedná o velmi důležitý příznak. Na druhém místě se nachází atribut `country`, který určuje odkud daný tok komunikuje a byl také dodatečně přidán. Oba tyto atributy zasádně přebíjí třetí příznak, který je získán jako běžná informace z toku. Dále je nutné poznamenat, že tento graf 7.5 neobsahuje všechny atributy ze vstupního vektoru příznaků, ale jen ty nejdůležitější. Na základě tohoto grafu můžeme konstatovat, že přidané atributy z obou aplikací, ať už se jedná o `openAppID` nebo `GeoLite2` mají pro predikcni model značný význam.

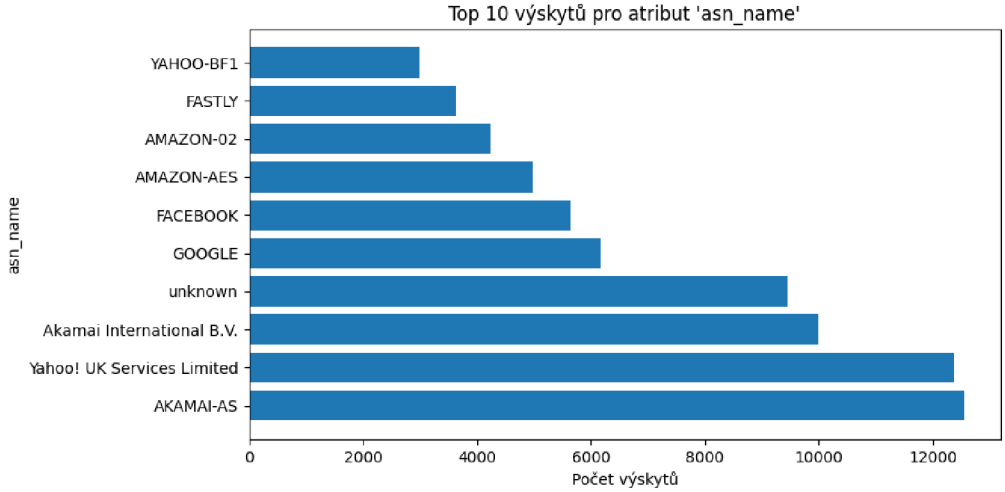
## 7.4 Zhodnocení výstupu predikčního modelu

Na obrázku 7.6 lze vidět, že největší počet útoků pochází ze Spojených států, více než 50000 škodlivých instancí, na druhém místě se nachází Spojené království, které nabývá polovinu hodnot oproti Spojeným státům. Toto se může zdát jako nežádoucí faktor pro trénování modelu, protože je možné, že bude mít tendenci označovat toky ze Spojených Států jako škodlivé, což se projeví na grafu 7.9.



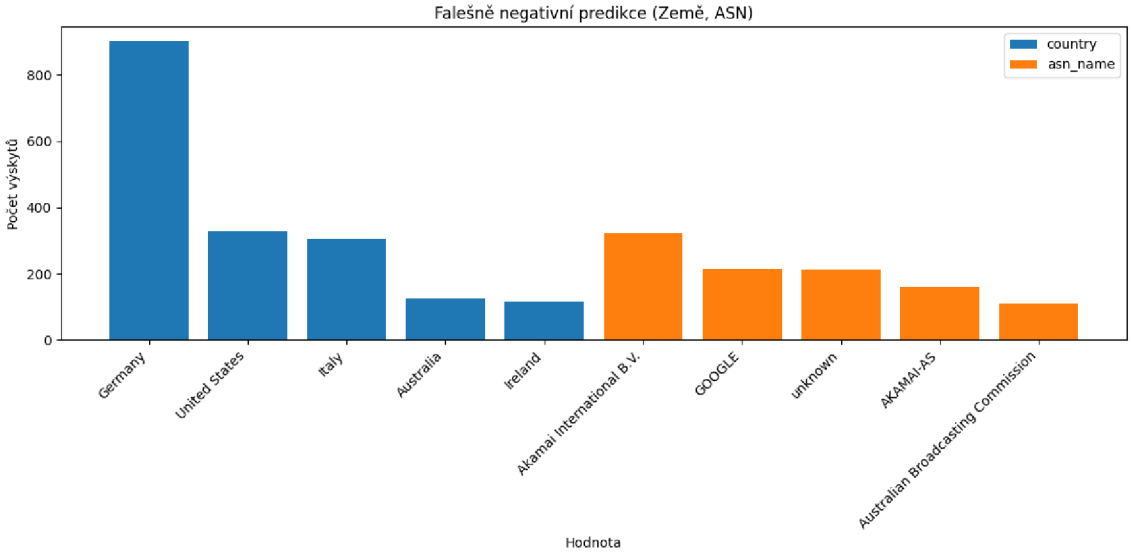

Obrázek 7.6: Top 10 útoků podle země.

Na grafu 7.7 lze vidět názvy jednotlivých autonomních systému a počet útoků daných autonomních systému, bylo vybráno pouze top 10. Nejvíce útoků pochází z autonomního systému `AKAMAI-AS`. Počet výskytů se zdá být poměrně vyvážený a výběr příznaku ASN je pro strojové učení vhodný.



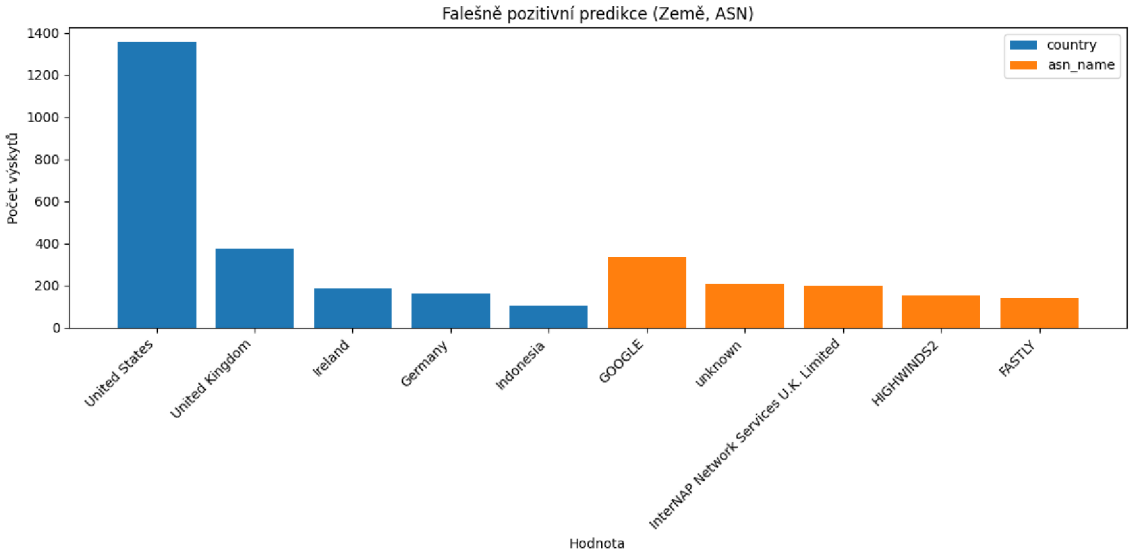

Obrázek 7.7: Top 10 útoků podle názvu autonomního systému.

Na grafu 7.8 lze vidět země a ASN systémy pro falešně negativně vyhodnocené instance. Model na tomto příznaku neměl tendenci se přeučit, ale naopak některé instance označil špatně i přesto, že se vyskytují poměrně často v trénovací datové sadě.




Obrázek 7.8: Falešně negativní predikce podle země a ASN systému.

Na grafu 7.9 lze vidět země a ASN systémy pro falešně pozitivně vyhodnocené instance. Na zemích jde vidět jistá korelace s grafem 7.6, kdy se model nejspíše přeučil a dal velkou váhu u atributu země pro hodnotu Spojených států a následně má model tendenci falešně predikovat instance toků pocházející z této země. Tento problém by se dal vyřešit vyvážením datové sady na základě hodnot zemí, ze kterých škodlivé toky směřují. Toto ovšem nemusí nutně znamenat, že toky byly špatně ohodnoceny, jelikož byly porovnány na základě detekovaných hlášení, která mohou být falešně negativní.




Obrázek 7.9: Falešně pozitivní predikce podle země a ASN systému.

## 7.5 Celkové shrnutí výsledků

Na tabulce 7.2 lze vidět celkové shrnutí výsledků testovaného provozu. Model vykazuje velmi kvalitní výsledky na testovací i validační datové sadě, ale aby byl model pořádně otestován, bylo potřeba použít i odlišné vstupní data. Jedná se o provoz `2017-05-01_normal.pcap č.1` a `2017-05-01_normal.pcap č.2`, tento provoz byl více rozepsán v sekci 7.1. Možný důvod proč model vykazuje horší výsledky na odlišném provozu je popsán v sekci výše, může to být z důvodu přetrénování modelu na škodlivých tocích směrovaných ze Spojených Států nebo se taky může jednat o nedostatek trénovacích dat. Soubor `DHT1.pcap` vyhodnotil však zcela správně, jedná se totiž o mnou zachycený síťový tok, kde se pouze inicializuje klient `Bittorrent` a stáhne se jeden soubor, výsledek je v tomto případě v porovnání s pravidly výrazně lepší. Dále bylo zjištěno, že model je velice přizpůsobivý novým datům a jeho přesnost razantně roste už při dotrénování na polovině vstupních dat.

| Testovaná data | Skóre F1 | Skóre modelu |
|---|---|---|
| Testovací data | 97,1% | 98,6% |
| Validační data | 97% | 98,6% |
| 2017-05-01_normal.pcap č.1 | 83,5% | 93,6% |
| 2017-05-01_normal.pcap č.2 | 87,4% | 95,4% |
| DHT1.pcap | 100% | 100% |

Tabulka 7.2: Výsledky predikce

Dále byla v této sekci zhodnocena rychlost predikce v porovnání s pravidly a výsledky jsou znázorněny v tabulce 7.1, v tomto případě se dá konstatovat, že ačkoli je natrénovaný model o něco méně přesnější je skoro 3x rychlejší a jelikož aplikace `Snort` trpí zahlcením a následným zahazováním paketů, je toto velmi důležitý benefit navrženého a implementovaného přístupu.

Jako poslední byla zhodnocena kvalita použitých příznaků a přínos jednotlivých atributů je znázorněn na grafu 7.5. Jak již bylo konstatováno, atributy, o které byly instance síťového toku rozšířeny, jsou pro trénování a následnou predikci modelu důležité a mají větší dopad na výstup, než informace získané pouze ze samotných toků.

Jelikož je model velice přizpůsobivý novým datům a jeho ohodnocení razantně vzrostlo už při dotrénování na polovině vstupních dat, tak se dá předpokládat, že tento přístup je použitelný na reálných datech, a to nejenom z hlediska funkčnosti, ale i rychlosti zpracování.

# Kapitola 8

# Závěr

V rámci diplomové práce byla nastudována problematika strojového učení se zaměřením na detekci anomálií v síťovém provozu. Tato problematika byla popsána v kapitole 2. Dále byla nastudována problematika systémů detekce a prevence průniku (IDS/IPS) a byla porovnána funkcionalita existujících `open-source` řešení, tato problematika byla popsána a jednotlivá řešení byla porovnána v kapitole 3. Pro nástroj `Snort` bylo navrženo rozšíření, které zlepšuje výstupní data získaná touto aplikací, a to přiřazením geolokace a komunikujících aplikací. Toto rozšíření dále označuje tato data na základě databáze černých listin a detekovaných hlášení. Toto rozšíření pak pomocí metod strojového učení vytvoří pravděpodobnostní model, který byl integrován do aplikace `Snort`. Toto rozšíření je popsáno v kapitole 5. Byla vytvořena funkční implementace rozšíření, která je popsána v kapitole 6. Tento model byl řádně otestován a jednotlivé experimenty jsou popsány v kapitole 7.

Při vyhodnocení správnosti predikce vytvořeného modelu bylo u testovacích i validačních dat dosaženo velmi kvalitních výsledků, a to ohodnocení `98,6%`. Jelikož ale validační i testovací data byla ze stejné datové sady, byl tento model dále otestován na nových datech zachycující běžný síťový provoz. Vytvořený model na těchto datech vykazuje ohodnocení v nejlepším případě `95,4%`, což by se dalo zhodnotit jako uspokojivý výsledek. Důvod proč je výsledek horší než na počátečních datech, by nejspíš odpovídal tomu, že model má tendenci predikovat falešná pozitiva na toky směřující ze Spojených států, a to z toho důvodu, že jejich výskyt byl při trénování příliš velký oproti ostatním zemím.

V porovnání s přístupem v článku 3.6 se tato implementace liší tím, že nepoužívá pouze data o tocích, která jsou lehce získatelná přimo v aplikaci, ale toky jsou obohaceny o příznaky, které byly získány externími aplikacemi.

V rámci implementace modelu byl pokus rozšířit datovou sadu i o provoz zachycený v organizaci `Netresec` na lokální síti, toto se ovšem osvědčilo jako nefunkční, jednak z důvodu že z lokální sítě nejde získat žádné informace o geolokaci, ale taky v zachycených síťových tocích bylo velmi malé procento informací o použitých aplikacích.

Toto rozšíření je plánováno na další zdokonalení v rámci spolupráce s externí firmou. Konkrétně má být rozšířeno o databázi reputací a lepší sadu pravidel s menším počtem falešných pozitiv. Jelikož tento přístup není nijak vázaný na konkretní data a je plně kompatibilní s nástrojem `Snort`, je možné jej dotrénovat na libovolných datech z reálného síťového provozu a zařadit jej tak k detekci v jakémkoli prostředí.

# Literatura


[1] Axelsson, S. Intrusion Detection Systems: A Survey and Taxonomy. In:. 2002. Dostupné z: https://api.semanticscholar.org/CorpusID:1609389.

[2] Boukebous, A. A. E., Fettache, M. I., Bendiab, G. a Shiaeles, S. A Comparative Analysis of Snort 3 and Suricata. In: *2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET)*. 2023, s. 1–6. DOI: 10.1109/GlobConET56651.2023.10150141.

[3] Chandola, V., Banerjee, A. a Kumar, V. Anomaly detection: A survey. *ACM computing surveys*. New York, NY: ACM. 2009, sv. 41, č. 3, s. 1–58. ISSN 0360-0300.

[4] Dhruba K. Bhattacharyya, J. K. Network anomaly detection; a machine learning perspective. *Reference Research Book News*. Portland: Ringgold, Inc. 2013, sv. 28, č. 6. ISSN 0887-3763.

[5] Garcia Teodoro, P., Diaz Verdejo, J., Macia Fernandez, G. a Vazquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers security*. Amsterdam: Elsevier Ltd. 2009, sv. 28, č. 1, s. 18–28. ISSN 0167-4048.

[6] Ghafir, I., Prenosil, V., Svoboda, J. a Hammoudeh, M. A Survey on Network Security Monitoring Systems. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2016, s. 77–82. DOI: 10.1109/W-FiCloud.2016.30.

[7] Karatas, G., Demir, O. a Koray Sahingoz, O. Deep Learning in Intrusion Detection Systems. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. 2018, s. 113–116. DOI: 10.1109/IBIGDELFT.2018.8625278.

[8] Karen Scarfone, P. M. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. CreateSpace Independent Publishing Platform, 2013. ISBN 978-1494758813.

[9] Keary, T. *IDS vs IPS* [online]. [cit. 2023-21-05]. Dostupné z: https://www.comparitech.com/net-admin/ids-vs-ips/.

[10] Khraisat, A., Gondal, I., Vamplew, P. a Kamruzzaman, J. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*. 2019, sv. 2, č. 1, s. 20. DOI: 10.1186/s42400-019-0038-7. ISSN 2523-3246. Dostupné z: https://doi.org/10.1186/s42400-019-0038-7.

[11] Klotsa, Y., Titova, V., Petliak, N., Cheshun, V. a Salem, A.-B. M. Research of the Neural Network Module for Detecting Anomalies in Network Traffic. *IntelITSIS 2022*. 2022, sv. 3156, s. 1–12. ISSN 0167-4049.

[12] Luigi V. Mancini, R. P. *Intrusion detection systems*. New York: Springer, 2008. Advances in information security. ISBN 978-0-387-77265-3.

[13] Lukáš Verner, D. K. Geolokace síťových zařízení v internetových sítích. *Elektrorevue*. 2011, sv. 13, č. 3. ISSN 1213-1539.

[14] McHugh, J., Christie, A. a Allen, J. Defending Yourself: The Role of Intrusion Detection Systems. *IEEE software*. Los Alamitos: IEEE. 2000, sv. 17, č. 5, s. 42–51. ISSN 0740-7459.

[15] Mehrotra, K. G., Mohan, C. K. a Huang, H. *Anomaly Detection Principles and Algorithms*. Cham: Springer International Publishing AG, 2018. Terrorism, Security, and Computation. ISBN 9783319675244.

[16] Patel, N. V., Patel, N. M. a Kleopa, C. OpenAppID - application identification framework next generation of firewalls. In: *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. 2016, s. 1–5. DOI: 10.1109/GET.2016.7916721.

[17] Paxson, V. *Zeek* [online]. [cit. 2023-21-03]. Dostupné z: https://zeek.org/.

[18] Ring, M., Wunderlich, S., Scheuring, D., Landes, D. a Hotho, A. A survey of network-based intrusion detection data sets. *Computers and Security*. 2019, sv. 86, s. 147–167. ISSN 0167-4048.

[19] Scarfone, K. A. a Mell, P. M. *SP 800-94. Guide to Intrusion Detection and Prevention Systems (IDPS)*. Gaithersburg, MD, USA, 2007.

[20] Shah, S., Issac, B. a Jacob, S. Intelligent Intrusion Detection System Through Combined and Optimized Machine Learning. *International Journal of Computational Intelligence and Applications*. Červen 2018, sv. 17, s. 1850007. DOI: 10.1142/S1469026818500074.

[21] Stratosphere. *Stratosphere Laboratory Datasets* [online]. 2015 [cit. 2022-21-12]. Dostupné z: https://www.stratosphereips.org/datasets-overview.

[22] Systems, C. *Snort 3* [online]. [cit. 2022-21-12]. Dostupné z: https://github.com/snort3/snort3.

[23] Systems, C. *Snort 3 Extra* [online]. [cit. 2023-21-03]. Dostupné z: https://github.com/snort3/snort3_extra.

[24] Thapa, S. a Mailewa, A. The role of intrusion detection/prevention systems in modern computer networks: A review. In: *Conference: Midwest Instruction and Computing Symposium (MICS)*. 2020, sv. 53, s. 1–14.

[25] Waleed, A., Jamali, A. F. a Masood, A. Which open-source IDS? Snort, Suricata or Zeek. *Computer networks (Amsterdam, Netherlands : 1999)*. Amsterdam: Elsevier B.V. 2022, sv. 213, s. 109116. ISSN 1389-1286.

# Příloha A

# Obsah paměťového média

Na přiloženém paměťovém médiu jsou uložené všechny zdrojové soubory potřebné pro běh aplikace, popis instalace, zdrojové soubory pro LaTeX, ale také vygenerována tato práce ve formátu PDF. Nacházejí se na něm i všechny ostatní soubory potřebné pro překlad a správné fungování programu a skriptů. Obsah jednotlivých složek je popsán zde:

**tex:** Zdrojové LaTeX soubory pro vytvoření tohoto pdf.

**tex/obrazky-figures:** Obrázky použité v této práci.

**pdf:** Soubor s touto diplomovou prací.

**src:** Zdrojové soubory, skripty pro spuštění, popis instalace.

**src/postgres:** Záloha databáze.

**src/*pcaps:** Vstupní data ve formátu *pcap*.

**src/snort3:** Zdrojové kódy pro aplikaci *Snort*.

**src/snort3_extra:** Zdrojové kódy pro rozšíření *Snort3_Extra*.

**src/odp:** Deskriptory pro *OpenAppID*.