FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Klasifikace provozu a zařízení v počítačových sítích na základě toků |
| **Student:** | Matej Hulák |
| **Vedoucí:** | Ing. Tomáš Čejka, Ph.D. |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Bezpečnost a informační technologie |
| **Katedra:** | Katedra počítačových systémů |
| **Platnost zadání:** | Do konce zimního semestru 2020/21 |

## Pokyny pro vypracování

Seznamte se s aktuálními technologiemi monitorování síťového provozu na základě síťových toků. Zaměřte se na systém NEMEA [1] pro analýzu síťového provozu a detekci anomálií.
Prostudujte přínosy bakalářské práce [2] a analyzujte možnosti existujícího klasifikačního NEMEA modulu.
Navrhněte nový rozšířený modul systému NEMEA, který bude klasifikovat síťový provoz lokálních zařízení a rozeznávat klienty a servery v lokální síti na základě síťových toků (základních i rozšířených o aplikační informace).
Otestujte funkcionalitu a přesnost klasifikace modulu s použitím testovacích sad (vytvořených ve spolupráci s vedoucím práce).
Vyhodnoťte výkon vytvořeného modulu, především jeho propustnost.

## Seznam odborné literatury

[1] T. Cejka, V. Bartoš, M. Svepes, Z. Rosa, and H. Kubatova: "NEMEA: A Framework for Network Traffic Analysis," in 12th International Conference on Network and Service Management (CNSM 2016), Montreal, Canada, 2016
[2] Zdeněk Kasner: Flow-Based Classification of Devices in Computer Networks. Bachelor thesis, Faculty of information technology, CTU in Prague, 2016

prof. Ing. Pavel Tvrdík, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 20. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Klasifikace provozu a zařízení v počítačových sítích na základě toků

***Matej Hulák***

Katedra počítačových systémů
Vedoucí práce: Ing. Tomáš Čejka, Ph.D.

4. června 2020

# Poděkování

V první řadě bych chtěl poděkovat svému vedoucímu práce Ing. Tomáši Čejkovi, Ph.D., za odborné vedení, rady a vstřícnost. Dále děkuji své rodině, která mě celou dobu podporovala a umožnila mi studium na vysoké škole.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principu při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 4. června 2020 . . . . . .

### Odkaz na tuto práci

Hulák, Matej. *Klasifikace provozu a zařízení v počítačových sítích na základě toků*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

# Abstrakt

Práce se zabývá klasifikací síťového provozu a tvorbou klasifikačního modulu pro systém NEMEA. První část práce popisuje existující nástroje a metody klasifikace. Teoretická část se zaměřuje na návrh klasifikačního algoritmu a klasifikačního modulu pro systém NEMEA. Praktická část popisuje tvorbu anotovaných datových sad a implementaci modulu. Závěr práce se je věnován testování klasifikační úspěšnosti a časové náročnosti vytvořeného modulu.

Výstupem práce je vytvoření devíti anotovaných datových sad a funkční klasifikační modul pro systém NEMEA, který je schopný klasifikovat síťový provoz v reálném čase.

**Klíčová slova** klasifikace síťového provozu, síťové toky, analýza síťového provozu, NEMEA, IPFIX

# Abstract

This thesis focuses on the network traffic classification and on the creation of a classification module for the NEMEA system. Firstly, the thesis examines the existing tools and classification methods. The theoretical section aims at creating of a classification algorithm along with a classification module for the NEMEA system. The practical section of the thesis describes the creation of the annotated datasets and the module implementation. The final section of the thesis is dedicated to the evaluation of the classification accuracy as well as to the time comlexity of the created module.

This thesis resulted in the creation of nine annotated datasets and a fully functional classification module for the NEMEA system, capable of real-time network traffic classification.

**Keywords** network traffic classification, network flows, network traffic analysis, NEMEA, IPFIX

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

S nárůstem síťového provozu, roste také potřeba efektivní analýzy provozu, za účelem administrace sítě, detekce bezpečnostních událostí a obrany před kybernetickými útoky. Za posledních deset let síťový provoz vzrostl více než desetkrát a předpokládá se, že objem síťového provozu bude narůstat i nadále [1]. Proto je třeba vylepšovat kromě vlastní síťové infrastruktury i ekosystém nástrojů pro bezpečnost a přehled správců o tom, co se v síti děje (tzv. *situational awareness* jako se diskutuje např. v [2]). S velikostí sítě úměrně narůstá i obtížnost administrace, filtrace síťového provozu za účelem zajištění bezpečnostních politik a bezpečnostní analýzy obecně. Kvůli ochraně uživatelů a provozuschopnosti sítě je nutné monitorovat provoz na síti a detekovat možné hrozby. V současné době již existují mnohé společnosti, které se na tuto problematiku zaměřují a vyvíjejí nové nástroje, jež tento úkol usnadňují.

Tato bakalářská práce se zaměřuje na klasifikaci síťového provozu na úrovni síťových toků. Klasifikace síťového provozu by měla značně pomoci při administraci a při detekci bezpečnostních hrozeb, neboť je to první krok k identifikaci podezřelého provozu. Například identifikací komunikace na aplikační vrstvě můžeme snadno rozlišit škodlivou komunikaci na nestandardních portech od běžné. Na tuto problematiku se zaměřuje řada soukromých firem, avšak většina jejich řešení je proprietární a jejich implementace a efektivita není veřejně známá.

Na základě těchto skutečností jsem se rozhodl prozkoumat možnosti klasifikace síťového provozu a následně navrhnout a vytvořit prototyp klasifikačního modulu. Tento modul, který vznikl jako rozšíření open-source systému NEMEA, je schopen klasifikovat provoz na úrovni síťových toků. Znamená to, že k tokům známého provozu bude schopen přiřadit štítek s významem síťové komunikace. Díky tomuto rozpoznání komunikace na základě statistických vlastností a charakteristického chování aplikací bude možné rozšířit množinu informací pro bezpečnostní analýzu a to přispěje k zajištění bezpečnosti sítě a zlepšení možnosti její administrace.

## Cíle práce

Cílem práce je vytvoření klasifikačního modulu, který bude schopný klasifikovat síťový provoz v reálném čase. Nejprve je třeba prozkoumat existující nástroje a metody klasifikace a poté navrhnout klasifikační algoritmus, který bude využit pro tvorbu klasifikačního modulu. Posledním cílem je otestovat navržený modul a ověřit jeho úspěšnost v reálném provozu.

Kapitola 1

# Analýza a rešerše

## 1.1 Síťová komunikace a monitorovací nástroje

Jakákoliv komunikace mezi dvěma a více zařízeními musí mít předem definovaná pravidla, která závisí na typu a účelu spojení. Ve světě informačních technologií existují mnohé komunikační metody, které slouží k propojení zařízení, ať už pomocí kabelu, nebo bezdrátově. Každá metoda komunikace však musí být podporována všemi zařízeními, která chtějí přijímat nebo posílat zprávy. Proto musí být každá takováto metoda řádně definovaná, aby každý, kdo je připojen k síti, věděl, jak v jejím rámci komunikovat a jaká pravidla je nutné dodržet. Definované metody komunikace nazýváme protokoly.

V současné době je největší a nejpoužívanější komunikační sítí internet [3]. Tato síť začala vznikat v roce 1969 pod názvem ARPANET [4] a propojovala čtyři zařízení. Postupně se rozšířila a momentálně je používána na celém světě. Stejně jako na každé síti je nutné zejména na takto rozsáhlé síti provoz usměrňovat a zabezpečit, aby se odeslaná data dostala ke svému adresátovi. Pro tento účel je nejvíce používána kombinace komunikačních protokolů TCP/IP. Tyto protokoly zabezpečují přenos zpráv a určují syntax a význam hlaviček každé přenášené zprávy.

Vzhledem k velikosti a složitosti daného problému byla definována architektura tzv. síťových vrstev (většinou se používá TCP/IP model nebo ISO/OSI model), která jej rozděluje na více dílčích. Síťové vrstvy jsou organizovány hierarchicky, přičemž každá vrstva poskytuje služby vyšší vrstvě a využívá služeb nižších vrstev. Každá vrstva se stará o specifickou část síťové komunikace.

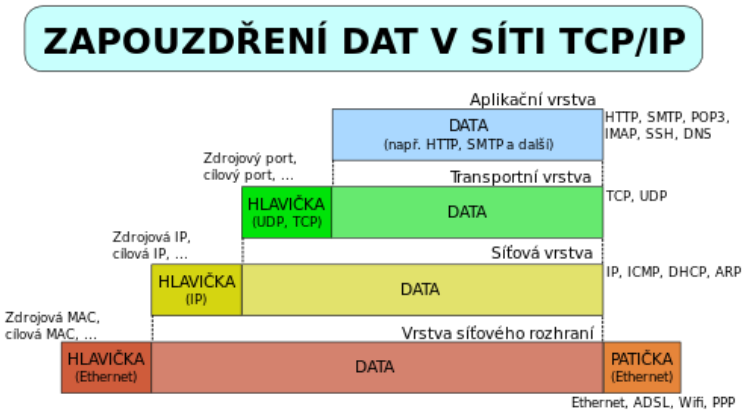
U TCP/IP modelu je problém rozdělen na čtyři vrstvy [5].

- aplikační vrstva
- transportní vrstva
- síťová vrstva
- vrstva síťového rozhraní

Vrstva síťového rozhraní je nejnižší a umožňuje přístup k fyzickému přenosovému médiu jako například ethernet. Síťová vrstva řeší problém adresace a směrování dat. Nejpoužívanějším protokolem na této vrstvě je právě protokol IP, který zabezpečuje adresaci zařízení a také směrování dat v rámci sítě. Transportní vrstva zabezpečuje plynulost a celistvost komunikace, hlavními představiteli jsou protokoly TCP a UDP. Poslední a nejvyšší vrstvou je vrstva aplikační, která poskytuje aplikacím přístup ke komunikační síti a zabezpečuje veškerou potřebnou komunikaci pro aplikace a uživatele. Na této vrstvě pracuje mnoho protokolů, přičemž každý je specializován pro konkrétní potřeby. Mezi nejpoužívanější protokoly patří HTTP, DHCP a DNS. Pro lepší rozlišení byla protokolům na aplikační vrstvě přidělena konkrétní čísla portů, které by měly být použity při komunikaci konkrétním protokolem. Díky tomu můžeme i u neznámého adresáta předpokládat, na kterém portu pracuje konkrétní služba/protokol. O přidělování a spravování čísel portů se stará společnost IANA [6].

Všechna data, která jsou na síti přenášena, jsou rozdělena v závislosti na použitém protokolu do menších zpráv, takzvaných paketů. O toto rozdělení se stará transportní vrstva. Pakety jsou nejmenší a nejzákladnější formou přenášených dat. Každý paket obsahuje přenášená data a hlavičky použitých síťových protokolů. Hlavičky obsahují data potřebná pro směrování paketů k adresátovi a slouží také ke opětovnému složení dat rozdělených do více paketů. Protokol každé vrstvy přidá vždy jednu hlavičku a výsledný paket je tak zapouzdřen hlavičkami protokolů. Díky těmto hlavičkám je tedy možné určit, odkud a kam paket směřuje a jaký typ dat obsahuje. V závislosti na použitém protokolu mohou, ale nemusí být přenášená data šifrovaná. Šifrování hlaviček je ovšem problematické, protože pro doručení paketů potřebují směrovače znát adresáta a další nezbytné údaje o paketu. Strukturu a umístění hlaviček názorně ukazuje obrázek 1.1.

Z pohledu operátora sítě je pozorování paketů způsob, jak sledovat, jaký provoz se přes síť přenáší. Proto mnoho monitorovacích systémů využívá pro analýzu data z paketů. Pomocí nahlédnutí do hlaviček paketů je možné určit odesílatele a příjemce, a dokonce i jaký typ provozu obsahují. Díky tomu mohou monitorovací nástroje snadno určit typ síťového provozu. Maximální velikost paketů (Maximal Transmission Unit – MTU) však bývá nastavena na pouhých 1,5 kB [8]. Z toho plyne, že i na přenesení jednoduché zprávy nebo




Obrázek 1.1: Zapouzdření paketů v síti TCP/IP [7]

obrázku je nutné přenést posloupnost více paketů. Jediné zařízení tak může za hodinu přijmout a odeslat desítky tisíc paketů. Analýza paketů jediného zařízení není problematická. Problém ale nastává, pokud chceme monitorovat ne jedno, ale několik zařízení, nebo dokonce celou síť. V takovém případě může být náročné zachytávat a analyzovat všechna přenášená data. Kvůli tomuto problému byl vyvinut a definován pojem síťový tok.

Síťový tok reprezentuje komunikaci mezi dvěma zařízeními za stejným účelem, tedy posloupnost paketů, jak popisuje např. [9]. Síťový tok je identifikován n-ticí informací, kterou mají všechny související pakety stejnou, tzn. odesílatelem, příjemcem, transportním protokolem a jeho porty. Síťový tok v základní podobě obsahuje:

- IP adresa odesílatele
- IP adresa příjemce
- číslo portu odesílatele
- číslo portu příjemce
- počet přenesených bajtů
- časová známka
- použitý protokol transportní vrstvy

Síťový tok však může obsahovat i méně nebo více informací podle zvoleného nastavení a použitého protokolu. Největší výhodou síťových toků je jejich velikost. Síťové toky totiž neobsahují uživatelský obsah komunikace a všechny hlavičky paketů jsou agregovány. Díky tomu je jejich velikost v porovnání s velikostí přenášených paketů minimální. Pomocí síťových toků je tedy možné

monitorovat provoz i na větších sítích, protože výpočetní i paměťová náročnost je několikanásobně menší než u tradiční analýzy paketů.

Pojem síťový tok byl prvně definován už v roce 1996 společností Cisco. Jednalo se o NetFlow verze 1 [10]. NetFlow v1 však nebyl nikdy používán ve větší míře. NetFlow v2, v3 a v4 ani nebyl nikdy oficiálně vydaný. První verze, která byla hromadně využívána, byla verze 5. Tato verze byla velice oblíbená a je podporována mnohými zařízeními až dodnes.

V roce 2008 byl také definován organizací IETF nový standard IPFIX [11] vycházející z NetFlow v9. Tento standard byl zpětně kompatibilní se všemi standardy NetFlow a umožňoval větší flexibilitu nastavení. Taktéž se jedná o neproprietární protokol, což umožňovalo jeho rozšíření v ještě větší míře.

## 1.2 Monitorovací nástroje

Tato podkapitola popisuje základní nástroje používané pro analýzu datového provozu.

### 1.2.1 Wireshark

Jedním z nejpoužívanějších nástrojů pro monitorování a analýzu síťového provozu je aplikace Wireshark [12] nebo její terminálová podoba Tshark, distribuovaná pod licencí GPL v2. Jedná se o multiplatformní aplikaci, která pracuje na úrovni paketů a díky tomu poskytuje uživateli veškerá dostupná data o síťovém provozu. Aplikace samotná obsahuje zejména nástroje pro analýzu provozu a grafické rozhraní, viz ukázka výpisu čtyř DNS paketů na obrázku 1.2. O záchyt samotného provozu se stará knihovna pcap, která je v různých verzích také dostupná pro většinu platforem. Knihovna pcap umožňuje zachycení veškerých dat ve formě paketů na sledovaném zařízení. Tato data mohou být okamžitě odeslána do aplikace Wireshark nebo uložena do souboru pro pozdější analýzu.

Pomocí Wiresharku je tedy možné analyzovat veškerá data, která prošla monitorovacím zařízením. Tato data můžeme analyzovat díky dostupným nástrojům, které jsou obsaženy v samotné aplikaci. Nástroje dokážou rozeznat, jakým protokolem a jakým způsobem byl paket zpracován. Umí rozeznat a zobrazit konkrétní hlavičky síťových protokolů, ze kterých můžeme získat všechny potřebné informace o paketu. Wireshark také obsahuje řadu pokročilých filtrovacích nástrojů, s nimiž je možné síťový provoz filtrovat a zobrazit pouze informace, jež jsou pro uživatele důležité.

| Time | Source | Destination | Protoco | Length | Info |
|---|---|---|---|---|---|
| 8.555806 | fit.cvut.cz | 192.168.1.11 | TCP | 1506 | https(443) → 50474 |
| 8.555863 | 192.168.1.11 | fit.cvut.cz | TCP | 54 | 50474 → https(443) |
| 8.598779 | fit.cvut.cz | 192.168.1.11 | TCP | 1506 | https(443) → 50474 |
| 8.599604 | fit.cvut.cz | 192.168.1.11 | TCP | 1506 | https(443) → 50474 |

Obrázek 1.2: Wireshark GUI [13]

### 1.2.2 Libprotoident

Libprotoident [14] je open-source knihovna, nově licencovaná pod licencí LGPL v3, která se zaměřuje na klasifikaci a identifikaci síťového provozu. Byla vyvinuta na univerzitě Waikato výzkumnou skupinou WAND. Tato knihovna byla vyvinuta pouze pro unixové platformy a neobsahuje grafické rozhraní. Její implementace a rozhraní jsou velice jednoduché a intuitivní.

Na obrázku 1.3 můžeme vidět ukázkový výstup klasifikačního nástroje lpi_protoident z knihovny libprotoident.

```
protident label, source IP, dest IP, src port, dest port, transport protocol, start timestamp,
end timestamp, src bytes, dst bytes, first four bytes of payload(hex)

DNS_TCP 192.58.128.30 193.87.0.34 53 55695 6 1584016481.075 1584016481.081 1171 43 0491545c
Unknown_TCP 192.203.230.10 158.197.8.8 53 37082 6 1584016502.141 1584016546.400 1039 51 040d19a8
DNS_TCP 192.58.128.30 193.87.0.34 53 41585 6 1584016481.075 1584016481.081 1171 43 04916e1a
DNS_TCP 192.58.128.30 193.87.0.34 53 41845 6 1584016481.075 1584016481.081 1099 30 0449f7c9
```

Obrázek 1.3: Ukázka výstupu libprotoident

### 1.2.3 Cisco NBAR

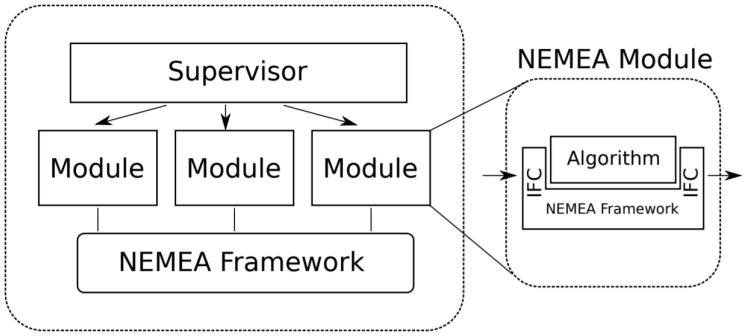
Zařízení společnosti Cisco Systems využívají mnohé metody a nástroje pro klasifikaci síťového provozu. Jedním z populárních nástrojů je NBAR [15] a jeho novější verze NBAR 2. Obě verze podporují většinu protokolů aplikační vrstvy a pro klasifikace využívají zejména hloubkovou analýzu paketů. Do nové verze NBAR 2 byla přidána podpora síťových toků a také možnost rozšířeného nastavení analýzy paketů.

Výsledky klasifikace těchto nástrojů se používají také pro optimalizaci propustnosti sítě ve spolupráci s nástrojem QoS. Nástroj NBAR byl zahrnut do Cisco IOS od verze 12.0(5)XE2 a je využíván dodnes.

### 1.2.4 NEMEA systém

V rámci tématu práce bylo třeba se seznámit se systémem NEMEA [16]. Tento systém se zaměřuje na automatickou analýzu a detekci hrozeb. Byl vyvinut výzkumnou skupinou Liberouter sdružení CESNET ve spolupráci s českými univerzitami. Systém byl navržen tak, aby byl schopen monitorovat i sítě s

velkým síťovým provozem v reálném čase. Celý systém je složen z mnoha nezávislých modulů, které jsou propojené pomocí speciálních komunikačních rozhraní implementovaných v NEMEA framework knihovnách. Komunikační rozhraní (IFC) umožňují rychlou a stabilní komunikaci. Systém místo hloubkové analýzy paketů využívá pro analýzu síťové toky. Použití síťových toků značně snižuje paměťovou náročnost, protože systém nemusí ukládat a zpracovávat celou komunikaci včetně přenášených dat. Systém je veřejně dostupný pod licencí GPL v2.




Obrázek 1.4: Struktura NEMEA sytému [16]

Na obrázku 1.4 můžeme vidět základní strukturu systému NEMEA, která se skládá ze tří základních komponentů:

- NEMEA Framework
- NEMEA Supervisor
- moduly a detektory

Framework obsahuje všechny potřebné knihovny pro chod systému a modulů. Mezi nejdůležitější patří TRAP, UniRec a Common. Supervisor slouží na řízení a správu běžících modulů a poskytuje jednoduché rozhraní, přes které je možné spravovat všechny běžící moduly. Poslední částí jsou moduly a detektory. Detektory slouží k detekci škodlivého provozu a hrozeb, jako jsou například DoS, DNS tunel a skenování portů. Moduly slouží zejména na zpracování, uložení, agregaci a analýzu dat.

Všechny tyto součásti jsou spolu propojené přes IFC, které jsou implementované v knihovně libtrap. Knihovna libtrap je jednou z nejdůležitějších částí systému. Komunikace mezi NEMEA moduly probíhá jednosměrně, rozhraní mohou fungovat jako vstupní nebo výstupní, přičemž každý modul může mít v podstatě libovolný počet vstupných a výstupních rozhraní. Ke každému výstupnímu rozhraní modulu může být připojen jeden nebo více NEMEA modulů. Přes toto rozhraní je možné posílat zprávy o maximální velikosti 65

kB, avšak množství zpráv není omezeno. Rozeznáváme čtyři typy základních TRAP rozhraní:

- UNIX rozhraní
- TCP rozhraní
- soubor
- černá díra (zahodí všechny zprávy, používá se pro měření propustnosti)

Všechna tato rozhraní jsou v rámci systému ekvivalentní a není nutné je při vývoji NEMEA modulu rozlišovat, což umožňuje vysokou abstrakci vstupních a výstupních rozhraní. Moduly tedy využívají TRAP rozhraní stejným způsobem, bez ohledu na to, jaký typ se při spuštění použije. O správné směrování a transformaci dat se stará samotná knihovna automaticky.

Přes TRAP rozhraní je možné přenášet tři datové formáty, a to nestrukturovaná data, JSON a UniRec. UniRec [17] je binární formát, specifický pro systém NEMEA, který může přenášet libovolná data. Vyznačuje se flexibilitou a efektivností zejména pro záznamy síťových toků. Formát je velice podobný C strukturám a je vhodný pro práci s nestrukturovanými daty a umožňuje taktéž práci s proměnnými variabilní délky. Formát umožňuje nadefinovaní struktury za běhu programu. Jeho hlavní výhodou je rychlost přístupu k datům uloženým ve struktuře, protože pozice a struktura dat je vždy přesně určena vzorem. Vzor neobsahuje žádná data, je proto velice malý a může být lehce přenesen spolu se samotným UniRec záznamem.

Aby bylo možné monitorovat i větší sítě, systém NEMEA dokáže sbírat síťové toky z vícerých míst pomocí sond. Sondy jsou zařízení, na nichž probíhá nebo může probíhat záchyt síťového provozu. Sondy musí pouze podporovat export síťových toků ve formátu, který je podporován kolektorem. Kolektor toky zpracuje a následně je odešle do systému k analýze. V rámci existujících systémů NEMEA je nejvíce využíván IPFIX formát síťových toků. Systém pro zpracování exportovaných toků využívá vlastní IPFIXcol [18] kolektor nebo jeho novější verzi IPFIXcol2 [19]. Tento kolektor je schopen zpracovat příchozí toky ve formátech IPFIX, NetFlow v5, NetFlow v9 a poskytuje také velké množství výstupních formátů jako například UniRec, JSON a FDS. Pomocí kolektoru IPFIXcol2 je možné také anonymizovat provoz, například použitím Crypto-PAn algoritmu a nahrazením IP adres. Avšak systém není závislý na použití konkrétních sond a kolektorů, a proto je možné využít i jiné nástroje.

Pro použití na menších sítích systém obsahuje také flow_meter, který je schopen zpracovávat PCAP soubory nebo přímo zachytávat síťový provoz na síťovém rozhraní.

## 1.3 Metody klasifikace síťového provozu

Tato podkapitola popisuje nejpoužívanější metody klasifikace síťového provozu a ukazuje příklady jejich využití v existujících nástrojích.

### 1.3.1 Analýza čísla portu

Nejjednodušší metodou klasifikace provozu je analýza čísla portu. Každé zařízení v síti by mělo pro komunikaci určitým aplikačním protokolem využívat číslo portu přidělené společností IANA. Díky číslu portů by tedy mělo být možné jednoduše identifikovat použitý protokol.

Přidělená čísla portů jsou využívaná zejména pro komunikaci s neznámým zařízením. Díky přiděleným číslům portů totiž můžeme předpokládat, na kterém portu běží požadovaná služba. Avšak nastavení čísla portů není závazné a za jeho nastavení zodpovídá samotné zařízení. Proto není zaručeno, že zařízení pro komunikaci využijí definovaná čísla portů. Pomocí této metody tedy není možné hodnověrně klasifikovat síťový provoz.

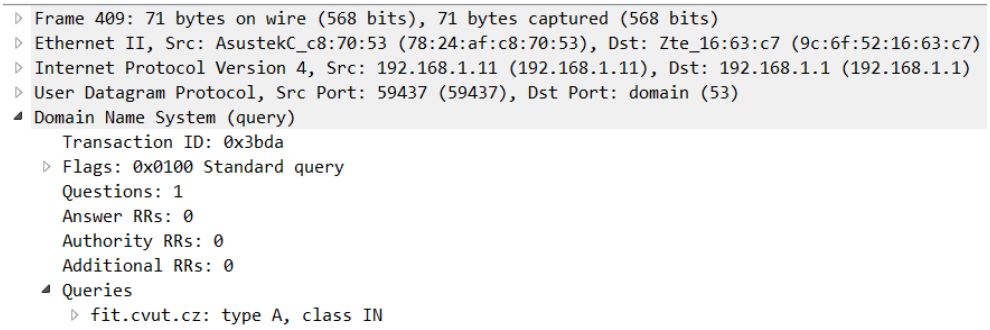
### 1.3.2 Hloubková analýza paketů

Pomocí hloubkové analýzy je možné klasifikovat provoz nahlédnutím do hlaviček paketů. Také je možné nahlédnout na několik prvních bajtů přenášených dat, kde se často nachází v závislosti na použitém protokolu další hlavička obsahující informace o přenášených datech. Tato metoda klasifikace je velice přesná a účinná a je využívaná síťovými analytiky po celém světě.

Jedním z nástrojů, který využívá tuto metodu, je Wireshark a Cisco NBAR. Na obrázku 1.5 můžeme vidět příklad využití hloubkové analýzy paketů pomocí aplikace Wireshark.

Pro přesnou klasifikaci je však nutné zachytávat všechna data přenášená sledováním zařízení, včetně dat přenášených v paketech. V závislosti na typu a vytíženosti zařízení může být klasifikace v reálném čase velice paměťově a výpočetně náročná. Taktéž záchytem a analýzou paketů můžeme porušit práva na soukromí uživatelů. Analýza provozu na úrovni paketů totiž zachytává mimo hlaviček také přenášená uživatelská data. Tato data však mohou být citlivá, a proto nám uživatel nemusí dovolit je zachytávat a analyzovat.

Aplikace využívající hloubkovou analýzu paketů jsou tedy velice užitečné, zejména pro detailní analýzu a klasifikaci síťového provozu. Avšak klasifikace všech dat v reálném čase by vyžadovala extrémní výpočetní a paměťovou kapacitu. Taktéž každý uživatel si musí být vědom toho, že jeho data mohou být analyzována, a musí s tím souhlasit.




Obrázek 1.5: Hloubková analýza DNS paketu pomocí aplikace Wireshark

### 1.3.3 Omezená hloubková analýza paketů

Knihovna libprotoident umožňuje klasifikaci obousměrných síťových toků a pro klasifikaci využívá omezenou hloubkovou analýzu paketů. Tato analýza je omezená pouze na hlavičky paketů a první čtyři bajty přenášených dat. Knihovna v současnosti podporuje víc než čtyři sta protokolů aplikační vrstvy.

Podle práce zveřejněné skupinou WAND [20], ve které byly porovnávány open-source klasifikační metody, dokázal libprotoident klasifikovat v závislosti na použité datové sadě 86 % – 99 % toků.

Aplikace NBAR 2 také využívá tuto metodu a navíc umožňuje rozšířené nastavení, díky kterému je možné nastavit hloubku prohledávání paketů. Přesná implementace však není známa, neboť se jedná o proprietární aplikaci.

Tyto knihovny tedy poskytují validní a efektivnější způsob klasifikace síťového provozu. Avšak stále využívají jistou metodu hloubkové analýzy paketů, která má téměř stejné nevýhody, jež byly popsány v předcházející podkapitole.

### 1.3.4 Analýza síťových toků

Problematikou klasifikace provozu a síťových zařízení pomocí síťových toků se zaobíral Zdeněk Kasner ve své bakalářské práci [21], publikované v roce 2016. Práce se zaměřovala na automatické rozeznávání síťových zařízení pomocí analýzy síťových toků. V práci navrhl metodu, která je schopna pomocí statistických informací o tocích klasifikovat síťové zařízení. Metoda pro vyhodnocování naměřených statistik využívá strojové učení, konkrétně Support Vector Machine (SVM).

Vstupem do algoritmu strojového učení bylo šest statistických hodnot získaných ze síťových toků. Tyto hodnoty by měly nejlépe určovat, o jaký typ zařízení a provozu se jedná:

- Init Ratio — podíl komunikace iniciované zařízením vůči celkovému počtu komunikací
- Data Ratio — podíl odeslaných bajtů vůči celkově přeneseným bajtům
- Packets/Flow — odchylka od předpokládaného poměru odesílaných paketů/toků
- Time/Flow — odchylka od předpokládaného poměru sekunda/počet toků
- TCP Ratio — podíl TCP komunikace vůči celkové komunikaci
- Port Usage — čísla použitých portů

Pro trénování a klasifikaci byla použita metoda podpůrných vektorů. Tato metoda se využívá zejména pro klasifikaci a regresní analýzu, a proto nejvíce vyhovovala požadavkům. Výstupem modulu byl seznam identifikováních IP adres, přičemž každá adresa byla označena klasifikačními štítky, které určovaly, zda se jedná o klienta nebo server a jaké aplikační protokoly byly adresou použity. Příklad výstupu můžeme vidět na obrázku 1.6.

```
ip_address      labels
66.189.174.146  CLIENT
76.241.17.159   SERVER
92.155.236.218  SERVER,MAIL
107.41.42.63    SERVER,MAIL
128.135.183.34  SERVER,HTTP
140.231.116.196 SERVER,HTTP,FTP
145.22.147.241  SERVER,HTTP
```

Obrázek 1.6: Ukázka výstupu modulu Zdeňka Kasnera [21]

Podle testů vykonaných v rámci této bakalářské práce byla úspěšnost modulu v průměru 97,50 %. Testy byly prováděny pomocí upravené verze desetinásobné křížové validace na datové sadě zachycené na síti CESNET2.

## 1.4 Shrnutí používaných metod a nástrojů

V rámci rešerše jsme tedy prozkoumali nástroje, které se využívají na analýzu síťového provozu. Každý ze zmíněných nástrojů pracuje se síťovým provozem rozdílným způsobem, neboť každý nástroj využívá rozdílnou metodu klasifikace. Každá z těchto metod má své výhody a nevýhody, a proto je nutné využít takovou metodu, která nejlépe vyhovuje řešenému problému. Rychlost a přesnost klasifikace se totiž přímo odvíjí od použité metody.

Jedním z hlavních cílů bakalářské práce je vytvoření klasifikačního modulu, který bude schopný klasifikovat provoz v reálném čase. Použití hloubkové analýzy paketů proto není vhodné, ačkoliv je tato metoda využívána většinou nástrojů, protože paměťová a výpočetní náročnost je příliš vysoká. Proto budeme pracovat pouze s metodou analýzy síťových toků, která je pro tento účel vhodnější. Tato metoda však poskytuje o něco méně informací než metoda hloubkové analýzy, nicméně Zdeněk Kasner ve své práci dokázal úspěšně klasifikovat zařízení pouze za pomoci síťových toků a vypočtených statistických informací. Pomocí této metody by tedy mělo být možné klasifikovat i samotnou kategorii síťového provozu.

Metoda analýzy síťových toků je vhodná také proto, že nenarušuje soukromí uživatelů a nezachytává soukromá data uživatelů. V systému NEMEA je rovněž možné síťové toky plně anonymizovat, což je velice přínosná funkcionalita.

Kapitola 2

# Návrh

Tato kapitola popisuje návrh klasifikačního modulu. Důraz byl kladen zejména na hodnocení časové náročnosti a na úspěšnost přidělování klasifikačních štítků.

## 2.1 Požadavky

Hlavním požadavkem na modul je korektní a efektivní přidělování klasifikačních štítků. Také by měl být schopen pracovat samostatně bez potřeby použití dalších nástrojů. Modul může být nasazen i na větší sítě a proto musí návrh počítat s velikým zatížením zejména při ověřovaní toků a přidělování klasifikačních štítků. Je nutné brát ohled na to, že cílem je vyvinout modul, který bude klasifikovat síťové toky i v reálném čase, ne pouze nad zachycenými daty.

Modul musí být také schopen spolupracovat se systémem NEMEA, který pro komunikaci využívá knihovnu TRAP. Musí proto umět komunikovat pomocí této knihovny. Také je nutné brát v úvahu přenositelnost modulu mezi různými unixovými distribucemi, kde modul musí pracovat plně samostatně, případně pod správou NEMEA supervisora.

Důležitá je také minimalizace zásahů člověka. Modul by měl po nasazení pracovat samostatně a bez potřeby další údržby nebo obsluhy.

## 2.2 Vstup

Modul bude přijímat síťové toky prostřednictvím rozhraní TRAP. Knihovna libtrap poskytuje všechny potřebné funkce pro práci se síťovými toky, přenášenými ve formátu UniRec. Záznam síťového toku může obsahovat různá políčka, avšak nejčastěji obsahuje alespoň deset základních políček, která jsou zobrazena na obrázku 2.1. Tato políčka reprezentují základní informace o síťové komunikaci, které poskytují všechny současné monitorovací systémy založené na NetFlow nebo IPFIX reprezentaci dat. V základním nastavení by tato políčka měla být vždy dostupná i v modulech NEMEA systému. V rámci

| Data Type | Field Name | Description |
|---|---|---|
| ipaddr | SRC_IP | Source address of a flow |
| ipaddr | DST_IP | Destination address of a flow |
| uint16 | SRC_PORT | Source transport-layer port |
| uint16 | DST_PORT | Destination transport-layer port |
| uint8 | PROTOCOL | L4 protocol (TCP, UDP, ICMP, etc.) |
| uint32 | PACKETS | Number of packets in a flow or in an interval |
| uint64 | BYTES | Number of bytes in a flow or in an interval |
| time | TIME_FIRST | Timestamp of the first packet of a flow |
| time | TIME_LAST | Timestamp of the last packet of a flow |
| uint8 | TCP_FLAGS | TCP flags of a flow (logical OR over TCP flags field of all packets) |

Obrázek 2.1: UniRec základní políčka síťových toků

návrhu algoritmu řešení budeme brát v úvahu pouze tato políčka, abychom zajistili co největší univerzálnost.

Přes TRAP tedy budou proudit do modulu síťové toky ve formátu UniRec, které je třeba přijmout, spravovat a výsledek zpracování odeslat na výstup. Pro příjem a odesílání UniRec záznamu budou použity již existující metody z knihovny, a proto se modul musí starat zejména o zpracování a vyhodnocení síťových toků.

## 2.3 Používané pojmy

### 2.3.1 Kategorie

V daném rámci bakalářské práce není možné vytvořit vzory pro všechny existující protokoly a kategorie síťového provozu. Proto bylo nutné vybrat podmnožinu síťového provozu, na které otestujeme úspěšnost navrženého algoritmu. Snahou bylo vybrat co největší podmnožinu síťového provozu obsahující typické a převládající protokoly, protože díky tomu bude modul užitečný i v tomto omezeném stavu na jakékoliv síti.

Nejprve byly vytvořeny základní neboli všeobecné kategorie síťového provozu, které by měly odrážet nejčastější provoz. Následně bylo ke každé kategorii přidáno několik protokolů, které jsou pro danou kategorií typické. V rámci této práce se budeme snažit klasifikovat „všeobecné kategorie“ i „protokolové kategorie“.

- Web
  - http
  - https
- Mail
  - imap
  - pop3
  - smtp
- Vzdálený přístup
  - ssh
  - telnet
  - ftp
  - rdp
- Síťová obsluha
  - ntp
  - dns
  - dhcp
  - rip

Celkově byly navrženy čtyři všeobecné kategorie, přičemž každá z těchto kategorií obsahuje minimálně dva protokoly. Tyto kategorie by se měly vyskytovat na každé síti, jejich klasifikace by měla zvýšit bezpečnost a ulehčit administraci sítě.

### 2.3.2 *Vzor*

Všechny statistické informace o síťových kategoriích budou uloženy ve vzorech. Každý z těchto vzorů bude reprezentovat jednou síťovou kategorii, ať už všeobecnou, nebo protokolovou. Vzory budou využívané při klasifikaci síťových toků a budou vznikat procesem trénování.

Vzor síťového provozu musí obsahovat veškeré informace o kategorii síťového provozu, jeho název, jeho charakteristiky a všechny potřebné informace pro klasifikaci dané kategorie, protože tyto informace se v modulu nebudou nacházet nikde jinde. Díky uložení všech dat o síťové kategorii do jediné struktury bude možné informace jednoduše upravovat a revidovat.

Vzor bude obsahovat svůj název pro odlišení od ostatních vzorů (protokolové vzory *upper_case*, všeobecné vzory *lower_case*), klasifikační štítek, který bude přidělován tokům, informace o natrénovaných datech, číslo používaného protokolu, číslo nebo čísla portů a samozřejmě intervaly spolehlivosti jednotlivých metrik. Dále bude obsahovat hodnoty evaluačních bodů a pro každou

metriku používanou ke klasifikaci bude obsahovat také násobící koeficient evaluačních bodů. Význam těchto hodnot je detailně vysvětlen v oddílu 2.4.1, implementace pak v podkapitole 3.2.

### 2.3.3 Klasifikační štítek

Každému toku, který bude klasifikován pomocí nějakého *vzoru*, bude přidělen klasifikační štítek daného *vzoru*. Štítek může obsahovat libovolný text, který bude označovat použitý aplikační protokol nebo všeobecnou kategorii. Ke každému toku, který bude procházet modulem, bude přidáno poličko, do něhož bude vložen příslušný klasifikační štítek nebo konstanta „NO_TAG“, která označuje neznámý provoz.

### 2.3.4 Metrika toku

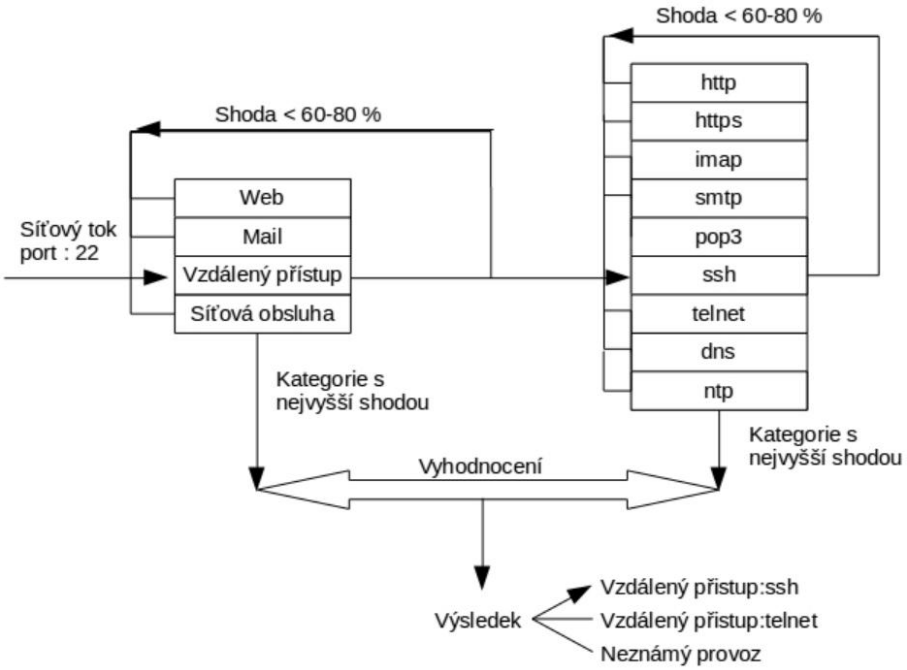
Metrika toku označuje hodnoty, které jsme schopni získat ze záznamu síťového toku, například počet přenesených bajtů nebo paketů, použitá čísla portů atd. Výjimkou jsou IP adresy, protože neříkají nic o samotném toku, ale pouze určují odesílatele a příjemce.

## 2.4 Zpracování toků

Modul bude schopen jak klasifikace toků, tak trénování vzorů. Modul bude vyhodnocovat a přidělovat štítky na základě klasifikačních vzorů. Vzory budou uložené v konfiguračním souboru a díky tomu bude možné vzory jednoduše ladit, upravovat a přenášet. Vzory budou trénované pomocí funkce modulu a bude tedy možné kdykoliv natrénovat nový vzor. Modul tak bude schopný klasifikovat libovolnou kategorii síťového provozu pouze na základě poskytnutých trénovacích dat.

Při klasifikaci bude modul přijímat síťové toky přes TRAP rozhraní. Příchozí toky budou následně porovnávány s natrénovanými vzory a na základě výsledků bude toku přidělen příslušný klasifikační štítek, který bude označovat, do které kategorie síťového provozu tok patří. Po vyhodnocení bude tok s přiděleným štítkem odeslán přes výstupní TRAP rozhraní modulu.

Tok bude porovnáván se vzory postupně. Nejprve bude tok porovnán dle portu s příslušejícím všeobecným vzorem, v případě dosažení nízké shody bude porovnán i s ostatními všeobecnými vzory a dle dosaženého hodnocení bude vybrán vzor s nejvyšší shodou. V případě, že dosáhne dostatečné shody, tok nebude dále porovnáván s jinými všeobecnými vzory. Následně bude tok porovnán stejným způsobem s příslušejícím protokolovým vzorem a v případě nízké shody bude tok porovnán také s ostatními vzory. Tento postup je znázorněn na obrázku 2.2.




Obrázek 2.2: Postup klasifikace toků

Na závěr se ověří, jestli všeobecný vzor odpovídá protokolovému. Pokud ano, kategorie testovaného toku je zaručeně správná a můžeme toku přidělit klasifikační štítek. V opačném případě bude porovnávána procentuální shoda obou vzorů. V závislosti na rozdílu procentuální úspěšnosti může být toku přidělen štítek podle vzoru s vyšší procentuální shodou. V případě podobných nebo nejasných výsledků nebude toku přidělen žádný štítek.

Cílem trénování je vytvořit vzor síťové kategorie, který bude používán pro klasifikaci. Trénování může probíhat buď přes TRAP rozhraní, nebo načtením dat z CSV souboru. Po načtení do paměti modul zpracuje získané údaje a vytvoří nebo zaktualizuje příslušný vzor. Validní vzor není možné vytvořit na základě jednoho síťového toku, ale je pro něj nutné zpracovat dostatečné množství dat. Modul tato data ze vstupu automaticky agreguje a následně je po celcích vyhodnocuje. Výsledné natrénované vzory budou uloženy do souboru tak, aby mohly být opětovně použity při dalších spuštěních modulu (kvůli perzistenci dat). Dále bude možné natrénované vzory tímto způsobem sdílet i mezi různými zařízeními, např. síťovými kolektory. Trénování vzorů není častá operace, provádí se zpravidla na začátku nasazení systému anebo při nasbírání dalších trénovacích datových sad. Proto zde není nutné klást velký důraz na rychlost procesu trénování, které navíc může probíhat i na nezávislém stroji s vyhrazenými výpočetními prostředky.

### 2.4.1 Algoritmus klasifikace

Při návrhu algoritmu klasifikace byly používány pouze základní UniRec políčka. Jako nejdůležitější informace se jeví tato políčka:

- src_port — číslo portu odesílatele
- dst_port — číslo portu příjemce
- time_first — čas začátku komunikace
- time_last — čas ukončení komunikace
- bytes — počet přenesených bajtů
- packets — počet přenesených paketů
- protokol — použitý protokol transportní vrstvy

Hodnoty těchto políček závisí na použitém síťovém protokolu a mají tedy informační hodnotu. Políčka jako src ip, src mac, tcp flags a ttl neříkají nic o kategorii síťového provozu, vypovídají pouze o způsobu přenosu, a proto nejsou při tvorbě algoritmu klasifikace podstatná. Hlavními metrikami algoritmu klasifikace jsou čísla portů, trvání toku, počet přenesených bajtů a paketů, použitý protokol a odvozená hodnota BPP, která reprezentuje průměrný počet bajtů v jednom paketu. Tato políčka by měla nejlépe vypovídat o typu přenášeného provozu.

Nad těmito poměrně omezenými daty je nutno vymyslet efektivní způsob klasifikace. Jako první se nám nabízí použití metod strojového učení. Jednou z nejvhodnějších metod pro tento účel je metoda rozhodovacího stromu. Tato metoda je poměrně známá a používaná v mnohých systémech. Avšak metody strojového učení mohou být často nepřehledné a ladění a pochopení jejich postupů vyhodnocování nemusí být vždy možné. Taktéž časová náročnost těchto metod může být znatelně vyšší než u statistických metod. Proto bylo rozhodnuto navrhnout statistickou metodu klasifikace, která bude mít deterministické chování a předvídatelné výsledky. Po vytvoření a otestování statistické verze může být záměrem budoucí práce vytvořit také verzi využívající strojové učení. Bude-li tato verze vytvořena, bude také možné porovnat její úspěšnost vůči existujícímu statistickému řešení.

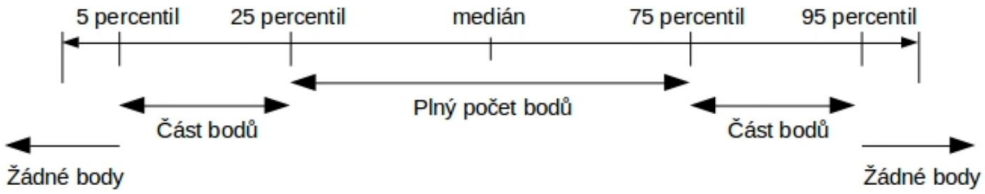
Pro kategorické metriky, jejichž hodnota nemá žádnou kvantitu a pouze označuje použitý protokol nebo port, budeme hledat nejvyšší výskyt nebo budou určeny na základě známých pravidel síťového provozu. Avšak pro ordinální metriky je zapotřebí najít jinou metodu.

Na základě experimentálního vyhodnocení zachycených vzorků komunikace různých protokolů byly identifikovány metriky, které je možné odvodit ze základních síťových toků. Tyto metriky je následně možné využít pomocí spolehlivostních intervalů, které reprezentují okolí mediánu. Předpokládáme totiž, že každý protokol je využíván jiným účelem (provoz se chová odlišně) a bude mít proto i rozdílné charakteristiky toku. Podle těchto charakteristik (statistických vlastností) je možné různé chování s určitou pravděpodobností rozeznat na základě vypočítaných metrik u každého síťového toku. Pomocí

okolí mediánu těchto metrik jsme schopni určit, zda hodnota testované metriky odpovídá zjištěným hodnotám z trénovacích dat. Návrh klasifikačního modulu využívá tuto metodu založenou na vzdálenosti hodnoty metriky od odvozeného mediánu se zohledněním blízkého okolí — spolehlivostního intervalu.

Každý klasifikační vzor bude obsahovat několik klasifikačních metrik, jako je například počet přenesených bajtů a paketů a trvání přenosu. Pro každou metriku budou vytvořeny dva spolehlivostní intervaly pro větší přesnost hodnocení. Vnější spolehlivostní interval bude reprezentovat širší okolí mediánu, vnitřní interval zase bližší okolí mediánu. Při klasifikaci se bude posuzovat, zda hodnoty metrik testovaného toku patří do jednoho z těchto okolí, a na základě toho budeme hodnotit, jestli tok odpovídá danému vzoru nebo ne. Jednotlivé metriky budou vyhodnocovány samostatně a s každým vyhodnocením metriky bude testovaný tok získávat evaluační body, které budou reprezentovat výsledek testu dané metriky. Po otestování všech metrik bude podle počtu získaných a ztracených bodů odvozena procentuální pravděpodobnost příslušnosti testovaného toku ke klasifikačnímu vzoru.

Evaluační body budou přidělovány podle toho, jak daná metrika testovaného toku odpovídá spolehlivostním intervalům vzoru. Množství získaných bodů bude záviset na tom, do kterého spolehlivostního intervalu bude metrika testovaného toku patřit. Jestliže metrika nepatří ani do jednoho spolehlivostního intervalu, nezíská žádné body, jestliže bude patřit do intervalu reprezentujícího širší okolí, získá nějakou část bodů, a jestliže bude patřit do intervalu bližšího okolí, získá plný počet bodů (viz obrázek 2.3). Pro každou metriku bude stanoven maximální počet bodů, které lze získat. Toto maximum může být pro každý vzor a každou metriku rozdílné a díky tomu bude možné natrénované vzory ručně ladit pro dosažení ještě lepších výsledků.




Obrázek 2.3: Návrh spolehlivostních intervalů

### 2.4.2 Algoritmus trénovaní

Validní vzory klasifikačních kategorií jsou neodmyslitelnou součástí algoritmu, protože veškeré vyhodnocování je založeno na referenčních datech uložených ve vzorech. Vytvoření všech vzorů ručně by bylo velmi zdlouhavé a nepraktické, proto je nutné vytvořit metodu, která tento proces zautomatizuje a umožní jednoduché natrénování nového vzoru. Pro každý vzor je nutné vytěžit data

z minimálně několika tisíc toků. Funkce trénování bude zahrnuta v hlavním klasifikačním modulu, aby bylo možné kdykoliv a kdekoliv modul trénovat a vytvářet nebo upravovat nové vzory. Vytváření vzorů bude možné pomocí dat z CSV souboru anebo přímo přes TRAP rozhraní modulu.

Důležitou součástí trénovacího algoritmu je vhodná volba datových sad. Data, nad kterými budeme vytvářet vzor, musí reprezentovat výhradně kategorii síťového provozu, kterou chceme klasifikovat. Při volbě nevhodných dat pro trénování nebude modul pracovat správně, protože algoritmus klasifikace je založen na spolehlivostních intervalech základních metrik toků. Zamíchání jiné kategorie sníží přesnost intervalů a klasifikace bude méně přesná. Je tedy nezbytné, aby byla trénovací data anotovaná a nebylo pochyb o jejich validitě a příslušnosti k dané kategorii.

Pro účel trénování je tedy nejprve nutné připravit kvalitní anotované datové sady. Každá datová sada by měla obsahovat alespoň několik tisíc síťových toků a měla by být co nejvíce různorodá. Různorodost datové sady je potřebná zejména kvůli správnému vytvoření vzorů, nerůznorodá datová sada může způsobit, že vzor bude odpovídat jenom malé podmnožině klasifikační kategorie, a proto bude správně fungovat jenom pro malou podmnožinu síťového provozu kategorie, nikoliv pro celu síťovou kategorii.

Cílem trénování je najít pro každou metriku, s výjimkou kategorických metrik, bližší a širší okolí mediánu. Výpočet mediánu a jeho okolí však není možné rozumně provádět postupně na procházejících datech. Proto musíme získaná data metrik nejprve uložit a po nasbírání dostatečného množství toků vypočítat okolí mediánu. Vzhledem k velikosti dostupné paměti a datové sady nemusí být vždy možné uložit hodnoty celé datové sady, a proto bude vhodné data zpracovávat po určitých celcích. Velikost celku bude záviset na velikosti dostupné paměti.

Aby bylo možné vzory zlepšovat a přidat také nově získané datové sady, bude třeba každý vzor trénovat více krát. Trénování bude taktéž podporovat jednoduchou anotaci příchozích dat dle čísla portu. Tento způsob anotace je nedostačující pro samotnou klasifikaci, ale může být užitečný pro určitou podmnožinu síťového provozu a také pro porovnávací testy.

Data z příchozích toků je nutné rozumně ukládat. Pro účel trénování a zmenšení paměťové náročnosti se budeme snažit omezit množství ukládaných dat. To můžeme docílit ořezáním síťového toku, protože příchozí tok bude obsahovat informace, které pro nás nejsou důležité. Data jako ip adresa zdroje a cíle, tcp flags, ttl budeme zahazovat a tím snížíme paměťovou náročnost. Ukládat budeme pouze data, která jsou pro algoritmus klasifikace relevantní.

Výsledné vzory s vypočtenými intervaly spolehlivosti a informacemi o síťové kategorii budou uloženy do souboru. Formát uložení vzorů je velice podstatný, chceme totiž, aby bylo možné vzory upravovat a revidovat. Proto nemůžeme použít binární soubor, zobrazení vzorů by pak vyžadovalo další nástroj pro otevření a zobrazení v čitelné podobě. Budeme tedy vzory ukládat do textového souboru, neboť velikost vzorů bude malá a nemusíme proto řešit

problémy optimalizace velikosti souboru. Mezi nejpoužívanější formáty textových souborů patří json, xml, CSV. Ani jeden z těchto formátů však není příliš přehledný, a proto bylo rozhodnuto použít vlastní formát. Navíc pravděpodobnost, že vzory budou použity mimo navrhovaný modul, je nízká. Kdyby však tato situace nastala, je možné kdykoliv data převést do požadovaného formátu.

Kapitola 3

# Realizace

Tato kapitola popisuje práci na tvorbě datových sad, implementaci modulu a všechny nezbytnosti týkající se vývoje a testování klasifikačního modulu. Implementace musí brát v úvahu požadavky na rychlost a udržitelnost. NEMEA systém podporuje programovací jazyky C, C++ a Python pro implementaci NEMEA modulů. Vybrán byl jazyk C++, který je sice v porovnání s jazykem Python složitější a náročnější na implementaci, ale výsledný modul by měl být rychlejší. Při implementaci je třeba brát ohled na to, že algoritmus klasifikace a trénování může být změněn a nahrazen jiným algoritmem, například algoritmem strojového učení. Modul bude taktéž podroben mnohým testům, a je proto nutné, aby obsahoval testovací metody. Implementace by měla být co nejpřehlednější kvůli následné údržbě.

## 3.1 Datové sady

Tvorba anotovaných datových sad byla jednou s klíčových úloh. Anotované datové sady nejsou totiž nutné jen kvůli testům klasifikačního modulu, ale také kvůli odvození metrik a vlastností toků. V žádných dostupných zdrojích nebyly k dispozici vyhovující anotované datové sady, a proto bylo nutné vytvořit vlastní. Pro tvorbu nových anotovaných datových sad bylo potřeba provést záchyt síťového provozu a následnou anotaci zachycených dat.

### 3.1.1 Záchyt

Pro účely této práce bylo provedeno několik záchytů na různých sítích a různých zařízeních. Pilotní datové sady pro analýzu provozu a hledání vhodných metrik probíhaly na počítači autora práce a na domácí síti. Pomocí těchto experimentů a zachycených dat bylo hledáno nejlepší vytváření datových sad a jejich anotací. Záchyty na domácí a pracovní síti vykazovaly značné nedostatky. Jedním z hlavních nedostatků byla masivní převaha určitých protokolů, zatímco některé jiné protokoly v provozu úplně chyběly, a to vedlo k nevyvá-

ženým datovým sadám. Téměř devadesát procent zachyceného provozu bylo tvořeno stejným protokolem. Bylo by tedy velice neefektivní zachytávat všechen provoz bez nějaké filtrace, protože sběr dostatečného množství některých méně používaných protokolů by byl velice paměťově náročný. Proto bylo rozhodnuto zachytávat provoz pouze na jediném portu, který by měl odpovídat protokolu, který chceme zachytit. Tento způsob fungoval o mnoho lépe, protože množství zbytečného provozu značně pokleslo. V rámci pokusů bylo také zjištěno, že pro správnou anotaci zachyceného provozu bude třeba zachytávat celé pakety ve formátu pcap (viz další oddíl).

Touto metodou byl prováděn i finální záchyt datových sad na síti CESNET. Datové sady byly zachytávané zvlášť a pro filtrování provozu byla využívána čísla portů. Každá datová sada byla vytvořena z několika nezávislých záchytů, které byly prováděné v rozdílných časech, aby byla zachycena data co nejvíce různorodá. Provoz byl zachytáván ve formátu pcap a následně byl také transformován do formátu IPFIX. Všechna zachycená data byla anonymizována.

### 3.1.2 Anotace

Zachycená data bylo potřeba anotovat a ubezpečit se, jestli opravdu odpovídají očekávanému protokolu. Pro tento účel byly použity již existující nástroje, které jsou popsány v podkapitole 1.2. Data budeme anotovat pomocí nástroje Wireshark, který by měl být nejpřesnější a nejvhodnějším pro tento účel, protože nám nezáleží na časové ani paměťové náročnosti, ale pouze na kvalitě anotace.

Zachycená data však byla ve formátu pcap a obsahovala tedy samotné pakety. Tyto pakety bylo zapotřebí anotovat pomocí Wiresharku a následně spárovat se získanými síťovými toky. Pomocí Wiresharku se tedy určilo, které pakety využívaly požadovaný protokol. Následně byl pomocí filtrů vytvořen seznam IP adres, které spolu komunikovaly požadovaným protokolem spolu s čísly použitých portů. Tento seznam bylo možné spárovat se síťovými toky porovnáním IP adres a portů odesílatele a příjemce. Tato metoda však není dokonalá a existuje malá pravděpodobnost, že dvě IP adresy spolu mohly komunikovat na stejných portech pomocí rozdílného protokolu. Proto bylo třeba tyto případy pečlivě kontrolovat a při zjištění, že stejné IP adresy komunikovaly přes stejné porty rozdílným protokolem, vyloučit jejich síťové toky z datové sady. Pro toto párování seznamu IP adres se síťovými toky byl vytvořen jednoduchý program, který zabezpečoval správné párování.

V rámci vytváření párovacího programu bylo rozhodnuto přidat k síťovým tokům také klasifikační štítky knihovny libprotoident. Tyto štítky mohou být využity pro porovnání úspěšnosti klasifikace. Knihovna libprotoident přiděluje štítky obousměrným síťovým tokům, a proto bylo párovaní toků v porovnání s Wiresharkem mnohem jednodušší. Párování probíhalo pomocí IP adres, čísel portů a použitého protokolu. Při párování byl jeden obousměrný tok spáro-

ván s dvěma jednosměrnými pomocí výměny IP adresy a portu odesílatele s příjemcem.

Výsledkem bylo devět anotovaných datových sad vytvořených z následujících protokolů, které tvoří v reálném síťovém provozu vysoký podíl:

- http
- https
- imap
- pop3
- smtp
- ssh
- telnet
- dns
- ntp

Nejmenší datová sada obsahovala 18 tisíc toků a největší 200 tisíc toků. Datové sady obsahují síťové toky a jsou uloženy ve formátu CSV. Ke každému toku byla přidána dvě políčka, jedno označovalo použitý protokol dle Wiresharku a druhé klasifikační štítek knihovny libprotoident. Na obrázku 3.1 je vidět ukázka několika toků z vytvořené datové sady.

```
BYTES, TIME_FIRST, TIME_LAST, PACKETS, DST_PORT, SRC_PORT, PROTOCOL, TTL,
PROTOIDENT_LABEL, TSHARK_LABEL

2472,2020-02-23T15:23:10.037,2020-02-23T15:23:12.316,6321,22,18999,6,122,SSH,ssh
4535,2020-02-23T14:02:30.054,2020-02-23T14:02:37.518,17,34510,22,6,58,NONE,ssh
```

Obrázek 3.1: Ukázka anotovaných toků z vytvořené datové sady

## 3.2 Implementace klasifikačního modulu

Při implementaci klasifikačního modulu bylo vycházeno z příkladu dostupného v github repositáři systému NEMEA [22]. Tento příklad byl navržen pro vytváření nových modulů pro NEMEA systém. Implementace modulu je rozdělena na dvě hlavní části, a to na klasifikační část a trénovací část.

Klasifikační *vzory* budou v paměti reprezentovány třídou pattern, jejíž struktura je zobrazena na obrázku 3.3. Každá instance této třídy bude reprezentovat právě jeden klasifikační *vzor*. Třída bude obsahovat funkce potřebné pro klasifikaci i trénování. Instance třídy budou uložené v asociativním kontejneru podle čísla portu, aby k nim bylo možné přistupovat s konstantní časovou náročností. Samotné spolehlivostní intervaly budou implementované čtyřmi proměnnými a budou reprezentovat 5., 25., 75. a 95. percentil. Kategorické metriky budou reprezentované pomocí pole nebo proměnné. Každý

klasifikační vzor bude obsahovat dvě hodnoty evaluačních bodů pro částečnou a úplnou shodu. Každá metrika bude také obsahovat násobící koeficient evaluačních bodů, který bude sloužit pro zvýhodňování a znevýhodňování jednotlivých metrik podle toho, která klasifikační metrika je důležitější a naopak.

Násobící koeficienty evaluačních bodů není možné nastavit přesně automaticky, a proto může být vhodné je ručně nastavit po vytvoření *vzoru*. V základním nastavení může každá metrika získat stejný počet bodů. Správné nastavení však může značně zvýšit úspěšnost klasifikace zvýhodněním metrik typických pro danou kategorii.

### 3.2.1 Implementace klasifikace

Pro klasifikaci bude sloužit funkce verify_pattern, třídy pattern. Funkci bude pomocí parametru předán síťový tok ve formátu UniRec. Funkce bude postupně extrahovat data z UniRec struktury. Při extrakci je nutné vždy ověřit validitu načtených hodnot, není totiž zaručeno, že data musí být korektní. Data budou extrahována postupně od nejdůležitějších. Některé metriky mohou totiž ověřování urychlit odmítnutím toku, jako je například metrika číslo protokolu.

Po extrakci bude každá metrika porovnána s natrénovanými metrikami vzoru pomocí if/else výrazů. Výsledek každého porovnání bude reprezentován evaluačními body, jejichž přidělování je popsané na konci oddílu 2.4.1. Po porovnání všech metrik dostupných ze vzoru proběhne celkové vyhodnocení. Vyhodnocení probíhá pomocí jednoduché pravděpodobnostní funkce, kde počet získaných bodů dělíme maximálním dosažitelným počtem bodů daného vzoru. Výsledek je ve formě procent vrácen volajícímu.

Při implementaci tohoto řešen bylo nutné brát ohled na to, že ne každý vzor musí mít definované všechny metriky. Proto se maximální dosažitelný počet bodů počítá přímo ve funkci.

### 3.2.2 Implementace trénování

Implementace trénování je o něco komplikovanější. Je totiž nutné počítat s poměrně komplexními požadavky. Vzory mohou být trénovány z CSV souboru nebo přes TRAP rozhraní, proto je nutné připravit potřebné funkce pro obě varianty. Vzory se mohou trénovat z anotovaných datových sad nebo je možné vzory trénovat na neanotovaných datech přímo ze síťového provozu. Ve druhém případě se data budou anotovat pouze podle čísla portu. Trénování také nemusí být jednorázovým procesem a musíme počítat s možností dodatečného trénování.

Trénování proto bylo rozděleno do více funkcí. Hlavní funkce se bude starat o celkovou organizaci trénování. Hlavní funkce bude určovat metodu trénování, formát vstupních dat a bude zabezpečovat uložení vzorů. Další dvě funkce se budou starat o načítání dat z CSV souboru a přes TRAP rozhraní. Poslední

funkce bude realizovat samotný proces trénování instance třídy neboli klasifikačního vzoru.

Hlavní funkci trénování bude pomocí parametrů z příkazové řádky předán přepínač určující metodu trénování a případná cesta k souboru s datovou sadou. V případě, že bude trénovat vzory z neanotované datové sady, funkce bude postupně podle čísla portu odesílat toky do samotných vzorů, kde budou zpracované. Při vyhodnocování čísla portu se bude funkce řídit číslem portu s nižší hodnotou, protože nižší čísla portů jsou většinou přiřazované definovaným protokolům. V případě trénování z anotované datové sady musí být jako parametr příkazové řádky předán taktéž název vzoru, který bude trénován. V případě, že daný vzor neexistuje, bude vytvořen. Naopak pokud vzor existuje, bude dle jména vyhledán a budou mu předána všechna data pro trénování.

Funkce načítání budou přijímat data přes TRAP rozhraní nebo ze souboru. Obě funkce budou mít shodný výstup, rozdílný bude pouze zdroj dat. Obě funkce musí kontrolovat načítaná data a ověřovat jejich validitu. Funkce, která bude načítat data ze souboru, bude o něco složitější, protože textový soubor neposkytuje žádné záruky správného formátu a lehce může dojít k přeházení pořadí nebo výpadku dat. Kvůli zjednodušení práce programu musí CSV soubor přesně odpovídat požadovanému formátu a musí obsahovat zadaná políčka v přesném pořadí. Tato podmínka se může zdát příliš přísná, avšak NEMEA systém obsahuje nástroje, které umí zajistit přesně stanovený formát CSV souboru. Funkce budou všechna data ukládat do kontejnerů *vector*, které na konci předají další funkci, jež vypočte samotné vzorové hodnoty metrik. Každý kontejner bude náležet právě jedné metrice.

Samotná funkce trénování bude zpracovávat metriky uložené v kontejnerech a jejím cílem bude v závislosti na metrice najít širší a bližší okolí mediánu nebo nejčastější výskyt. Proto si funkce na začátku všechna dostupná data seřadí pomocí funkce *quicksort* a následně vybere potřebné percentily jednoduchým výběrem prvků z pole na vypočteném místě. V případě, že vzor nebyl ještě trénován, budou získaná data přímo uložena jako vzorové metriky spolu s počtem toků, které se zpracovaly. Pokud však vzor už nějaká data obsahoval, je nutné získané metriky sloučit s původními. Nejjednodušším a nejefektivnějším způsobem sloučení je zprůměrování hodnot. Avšak pro validní výpočet průměru musíme znát počet toků, ze kterých byly jednotlivé hodnoty vypočteny. Není ale žádoucí průměrovat hodnoty rovným dílem. Je totiž možné, že původní nebo aktuálně získaná data jsou vypočtena z mnohem větší nebo menší množiny toků. Proto si každý vzor zachovává počet toků, ze kterých byly vypočteny jeho vzorové metriky.

Taktéž je při trénování nutné brát ohledy na paměťovou náročnost. Funkce načítání mají proto nastavené omezení maximálního počtu načtených toků. V případě přesáhnutí maximálního limitu jsou kontejnery s metrikami toků odeslány na zpracování. Po zpracování načtených toků se kontejnery vyprázdní a pokračuje se v načítání dat.

## 3.3 Implementace uložení a načítaní *vzorů*

Klasifikační vzory budou uložené kvůli persistenci dat v souboru. V rámci návrhu bylo vyhodnoceno, že bude vhodnější použít pro uložení vlastní formát. Při návrhu formátu byla zohledněna hlavně čitelnost a možnost ruční úpravy vzorů. Všechny vzory budou uloženy v jediném souboru, je proto potřeba je také znatelně oddělit. Není zaručeno, že všechny vzory budou obsahovat všechny metriky, proto je nutné počítat s možností uložení neúplného nebo prázdného vzoru. Pro lepší čitelnost bude každý obsažený element vzoru uložen na samostatném řádku. První řádek bude vždy název vzoru a za ním budou následovat všechny ostatní elementy. Na pořadí jednotlivých elementů nebude záležet, ale v rámci čitelnosti by bylo vhodné dodržovat předepsané pořadí. Každý řádek bude začínat klíčovým slovem, které bude označovat, o jakou metriku se jedná. Pro každé klíčové slovo je definována přesná struktura řádku, podle níž bude možné data načítat. Řádek, který nezačíná klíčovým vzorem, bude považován za komentář a nebude brán v úvahu. Příklad jednoho vzoru reprezentovaného navrženým formátem je zobrazen na obrázku 3.2. Vysvětlení jednotlivých elementů a struktury formátu je uvedeno na obrázku A.1 v příloze A.

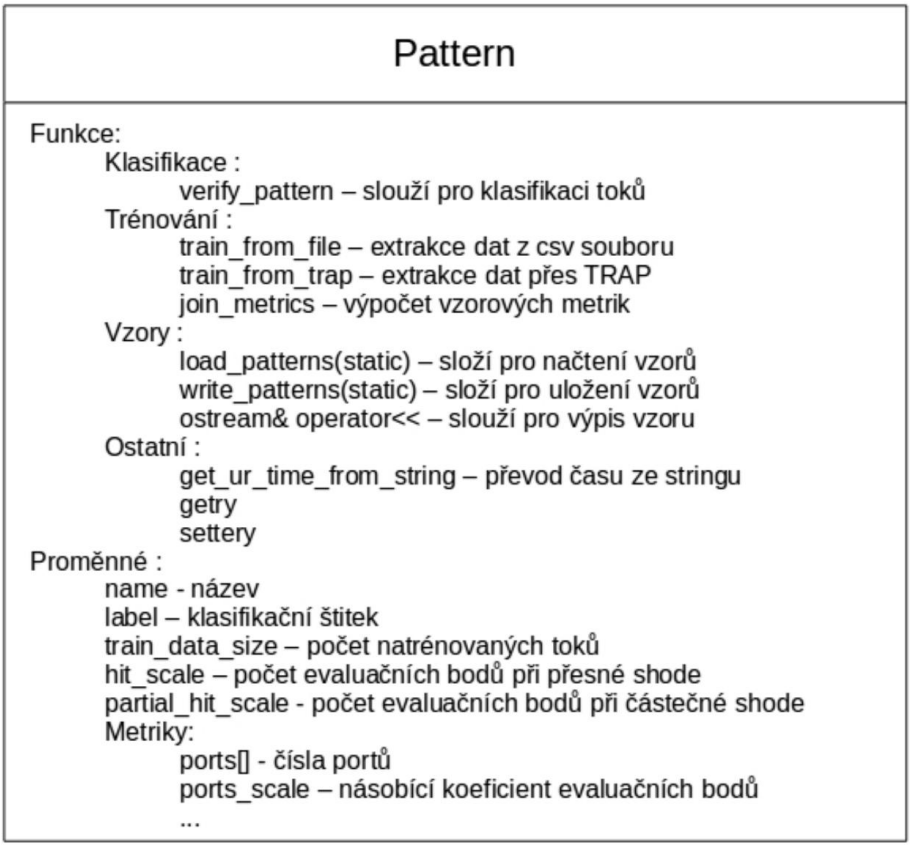
```
NAME: SSH
TAG: ssh
DATA: 33282
HIT_SCALES: 20 12
PORTS: 5 22
PROTOCOLS: 1 6
DURATION: 1 350 2398 7731 16881
PACKETS: 3 5 14 19 26
BYTES: 1 650 2497 3457 4450
BPP: 3 65 160 194 230
```

Obrázek 3.2: Ukázka formátu uložení *vzorů*

Načítání vzorů ze souboru bude realizovat statická funkce *load_patterns*. Této funkci bude parametrem předáno umístění souboru a kontejner určený pro načítání vzorů. Funkce si nejprve ověří platnost zadané cesty a následně otevře soubor. Funkce bude procházet postupně všechny řádky, dokud nenarazí na klíčové slovo „NAME“. Následně vytvoří v kontejneru nový vzor požadovaného jména. Dále bude funkce procházet řádky a hledat klíčová slova reprezentující elementy vzoru, které bude následně ukládat do vzoru v kontejneru. Prohledávání se zastaví až nalezením dalšího klíčového slova „NAME“. V tomto případě bude stávající vzor uložen a bude vytvořen nový vzor požadovaného jména. Uvedený postup bude pokračovat až do konce souboru.

O ukládání vzorů se bude starat *operator «*. Tento operátor slouží k výpisu jediného vzoru. Vzor bude vypsán v předdefinovaném pořadí a každý řádek bude označen příslušným klíčovým slovem. Třída bude obsahovat také funkci *write_patterns*, která vypíše všechny vzory do určeného souboru pomocí *ope-*

*rátoru «*. Navíc bude tato funkce vymazávat obsah souboru, aby se zamezilo smíchání dat.




Obrázek 3.3: Celkový náhled na strukturu třídy pattern, zobrazuje implementované metody a použité proměnné

Kapitola 4

# Testy a vyhodnocení

Tato kapitola popisuje postup a výsledky testování. Jednou z nejdůležitějších částí bylo měření výkonnosti a spolehlivosti algoritmu, neboť to bylo nezbytné pro verifikaci správné funkčnosti modulu.

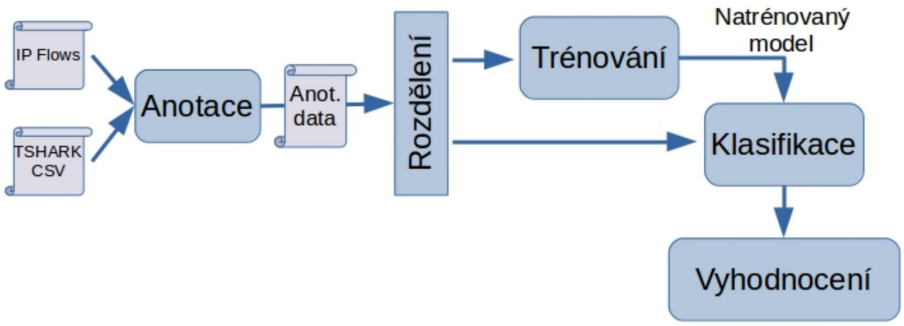
## 4.1 Metoda testování

První fáze měření byla zaměřena na testování spolehlivosti klasifikace. Spolehlivost, s jakou algoritmus klasifikuje síťové toky, je pro nás prvořadá. Při testování byl kladen důraz zejména na odhalení falešně pozitivních toků. Falešně pozitivní toky jsou toky, které byly modulem vyhodnocené a klasifikované nesprávně. Tyto falešně pozitivní toky jsou totiž velice zavádějící pro jakýkoliv modul nebo program, který bude využívat výstup tohoto modulu. Je proto nutné zajistit co nejmenší výskyt takovýchto toků, a to i za cenu nižšího počtu klasifikovatelných toků.

Druhou částí testů bylo vyhodnocení časové náročnosti klasifikace. Časová náročnost je důležitá zejména pro klasifikaci provozu v reálném čase, kde je zapotřebí, aby byl modul schopen zpracovávat příchozí toky dostatečně rychle.

Testování musí probíhat na anotovaných datových sadách, aby bylo možné verifikovat výsledky klasifikace. Na testování proto byly využity vytvořené anotované datové sady. Avšak kvůli věrohodnosti testu nebylo možné trénovat a zároveň testovat klasifikaci na stejných datech. Tyto výsledky by nebyly ani zdaleka podobné výsledkům v reálném provozu zejména kvůli přetrénování. Přetrénování nastává, když je algoritmus laděn a testován na stejné omezené množině dat. Algoritmus pak dosahuje lepších výsledků, ale pouze na dané množině. Při testech na jiné množině dat se jeho úspěšnost dramaticky zhorší, protože pravidla jsou nastavena příliš striktně. Bylo proto nutné datové sady rozdělit na menší části, které se budou používat vždy jen pro trénování, anebo testování. Tím vzniká problém nevyužití celé datové sady a také problém možné nevyváženosti datových sad. Datová sada by se měla otestovat celá, protože se může stát, že toky na začátku mají rozdílné charakteristiky než ty

na konci. Tím by mohlo dojít k nežádoucímu zlepšení nebo zhoršení výsledků. Postup testování je znázorněn na obrázku 4.1.




Obrázek 4.1: Postup testování

Pro testování proto byla využita metoda k-násobné křížové validace [23]. Křížová validace využívá pro testování celou datovou sadu a zaručuje, že testování bude probíhat na jiné datové podmnožině než trénování. Zároveň řeší problém možné nevyváženosti datových sad, protože každá část datové sady je testována a výsledná úspěšnost je sestavena průměrem výsledků ze všech testů. Datová sada je rozdělena na $k$ disjunktních podmnožin stejné velikosti. Každá podmnožina je použita na testování právě jednou, přičemž na trénování jsou použity všechny zbylé podmnožiny. Tento postup je aplikován na každou podmnožinu. Výsledkem je pak průměr výsledků všech $k$ testů.

## 4.2 Výsledky testů

Testování probíhalo na vytvořených datových sadách. Každá datová sada byla testovaná samostatně, avšak modul vždy obsahoval *vzory* všech síťových kategorií, aby byly simulovány reálně podmínky, protože modul bude pracovat v reálném provozu s větším počtem *vzorů*. Testovaná kategorie byla trénovaná a testovaná dle desetinásobné křížové validace, naopak netestované kategorie byly natrénovány na celé své datové sadě. Každý tok byl vždy ověřován pomocí všech *vzorů* a simuloval se tak nejhorší případ, kdy číslo portu neodpovídá použitému protokolu.

Výsledky zobrazené v tabulce 4.1 ukázaly, že algoritmus klasifikuje s průměrnou úspěšností 92,82 %, u 4,34 % dat nepřiřadí žádnou kategorii a pouze 2,84 % výsledků je klasifikováno chybně. Tyto výsledky ukazují, že algoritmus funguje správně. Úspěšnost modulu je dostatečná pro reálné nasazení.

Tabulka 4.1: Souhrn výsledků testů protokolových *vzorů*, tabulka obsahuje výsledky všech testů, první sloupec označuje testovanou datovou sadu, sloupec **Celkem** označuje celkový počet toků v datové sadě, sloupec **Úspěch** označuje počet úspěšně klasifikovaných toků, sloupec **Bez** označuje počet nezařazených toků, sloupec **Falešné** označuje počet falešně pozitivních toků, poslední sloupec **Úspěšnost** označuje procentuální úspěšnost klasifikace

| **Datová sada** | **Celkem** | **Úspěch** | **Bez** | **Falešné** | **Úspěšnost** |
|---|---|---|---|---|---|
| http | 16 131 | 16 059 | 72 | 0 | 99,55 % |
| https | 14 697 | 14 173 | 523 | 1 | 96,43 % |
| imap | 21 725 | 20 478 | 849 | 398 | 94,26 % |
| pop3 | 20 121 | 17 429 | 1854 | 838 | 86,62 % |
| smtp | 33 020 | 29 105 | 2952 | 936 | 88,22 % |
| ssh | 33 281 | 29 842 | 444 | 2995 | 89,67 % |
| telnet | 39 471 | 36 043 | 1519 | 1909 | 91,29 % |
| dns | 61 772 | 59 257 | 745 | 1770 | 95,93 % |
| ntp | 2 750 865 | 2 569 889 | 180 976 | 0 | 93,42 % |
| | | | | **průměr** | **92,82 %** |

## 4.3 Testování všeobecných kategorií

Toto testování mělo ukázat, zda je možné sjednotit víceré klasifikační *vzory* do jednoho všeobecného. Sjednocením vícerých kategorií do jedné by bylo bezpochyby praktičtější, protože by stačilo udržovat a trénovat pouze jeden klasifikační *vzor*. Tyto *vzory* budou použity pro postupnou klasifikaci, která je popsána v podkapitole 2.4, nebo mohou být použity pro rychlejší klasifikaci, u níž nás nezajímá konkrétní kategorie protokolu a postačí nám všeobecná kategorie.

Pro tvorbu všeobecných *vzorů* byly použity anotované datové sady, sjednocené podle jejich kategorie:

- Web: http, https
- Mail: imap, pop3, smtp
- Vzdálený přístup: ssh, telnet
- Síťová obsluha: dns, ntp

Testování probíhalo obdobně jako testování klasických kategorií. Pro testování byla využita desetinásobná křížová validace. Modul obsahoval vždy všechny čtyři všeobecné *vzory*, přičemž netestované vzory byly trénované na všech svých datech.

Výsledky zobrazené v tabulce 4.2 ukázaly, že modul byl pomocí všeobecných kategorií schopný správně klasifikovat 85,64 % dat. Tento výsledek ukazuje, že sjednocení *vzorů* je možné, avšak za cenu snížení přesnosti klasifikace.

Tabulka 4.2: Souhrn výsledků testů všeobecných *vzorů*, tabulka obsahuje výsledky všech testů, první sloupec označuje testovanou datovou sadu, sloupec **Celkem** označuje celkový počet toků v datové sadě, sloupec **Úspěch** označuje počet úspěšně klasifikovaných toků, sloupec **Bez** označuje počet nezařazených toků, sloupec **Falešné** označuje počet falešně pozitivních toků, poslední sloupec **Úspěšnost** označuje procentuální úspěšnost klasifikace

| **Datová sada** | **Celkem** | **Úspěch** | **Bez** | **Falešně** | **Úspěšnost** |
|---|---|---|---|---|---|
| Web | 30 828 | 25 769 | 4929 | 130 | 83,59 % |
| Mail | 74 841 | 69 869 | 4426 | 546 | 93,36 % |
| Vzdálený přístup | 72 753 | 57 689 | 13 742 | 1322 | 79,29 % |
| Síťová obsluha | 14 7594 | 12 7558 | 18 168 | 1868 | 86,42 % |
| | | | | **průměr** | **85,67 %** |

## 4.4 Porovnání s knihovnou libprotoident

Knihovna libprotoident dosahovala dle práce [20] vynikající výsledky klasifikace, a proto došlo k rozhodnutí porovnat její úspěšnost s úspěšností navrženého modulu. Při tvorbě datových sad, popsaných v oddílu 3.1.2, byl ke každému toku přidán také klasifikační štítek nástroje *lpi_protoident*. Test tak bylo možné jednoduše uskutečnit.Test probíhal na vytvořených anotovaných sadách, které byly použity i pro testování navrženého modulu.

Tabulka 4.3: Porovnání úspěšnosti modulu s libprotoident, tabulka obsahuje výslednou procentuální úspěšnost klasifikace navrženého modulu a libprotoidentu

| **Datová sada** | **Úspěšnost protoident** | **Úspěšnost mého modulu** |
|---|---|---|
| http | 91,01 % | 99,55 % |
| https | 33,90 % | 96,43 % |
| imap | 93,55 % | 94,26 % |
| pop3 | 96,55 % | 86,62 % |
| smtp | 56,93 % | 88,22 % |
| ssh | 79,33 % | 89,67 % |
| telnet | 48,92 % | 91,29 % |
| dns | 47,10 % | 95,93 % |
| ntp | 99,95 % | 93,42 % |
| **průměr** | **73,68 %** | **92,82 %** |

Výsledná úspěšnost klasifikace libprotoidentu je uvedena v tabulce 4.3 spolu s výsledky úspěšnosti navrženého modulu z podkapitoly 4.2. V tabulce můžeme vidět, že libprotoident dosáhl velice nerovnoměrných výsledků. Jeho průměrná úspěšnost 73,68 % je poměrně nízká, zejména když víme, že pro klasifikaci využívá také omezenou hloubkovou analýzu paketů.

Jednou z příčin nízké úspěšnosti je nízký počet výstupních toků z klasifikačního nástroje *lpi_protoident*. Počet toků na výstupu nástroje byl často i 3krát menší než počet výstupních toků NEMEA nástroje *flow_meter*. Tento rozdíl je příliš veliký, i když vezmeme v úvahu, že výstupní toky *flow_meteru* byly pouze jednosměrné a výstupní toky z *lpi_protoident* byly obousměrné. Kvůli ověření tohoto nepoměru byly za pomoci nástrojů systému NEMEA spárovány i obousměrné toky. Výsledek byl stejný, počet výstupních toků z *lpi_protoident* byl někdy i 1,5krát menší než počet toků ze systému NEMEA. Tento rozdíl však nastává pouze u některých protokolů a není tedy jasné, zda se jedná o chybu klasifikace nebo zpracování síťových toků.

## 4.5 Měření časové náročnosti

Měření časové náročnosti probíhalo na notebooku autora práce, vybaveném procesorem Intel Core i7-4710HQ se základní frekvencí 2,5 GHz, 16 GB RAM paměti a Intel SSD diskem. Test probíhal na 64bitovém Linuxu Mint verze Serena.

Pro testování byla využita největší vytvořená datová sada protokolu ntp, která obsahuje 2 750 865 síťových toků. Modul byl zkompilován s optimalizačním přepínačem *-O3* a všechny textové výpisy modulu byly zakomentovány, ponechána byla pouze komunikace přes TRAP rozhraní (pro měření byl použit typ blackhole). Datová sada byla předána na vstupní TRAP rozhraní pomocí souboru obsahujícího UniRec zprávy.

Testovány byly dva případy, a to ideální případ, kdy je porovnán pouze jeden všeobecný a jeden protokolový *vzor*, a nejhorší případ, kdy jsou porovnány všechny načítané *vzory*. Během testu modul obsahoval všech 13 *vzorů* (9 protokolových a 4 všeobecné). Pro každý případ proběhlo celkem 5 testů, výsledný čas je vypočten jako aritmetický průměr ze všech testů.

Tabulka 4.4: Časová náročnost modulu, tabulka obsahuje výsledky testů časové náročnosti modulu, první sloupec označuje testovaný případ, druhý sloupec označuje počet zpracovaných toků, třetí sloupec označuje celkový čas od spuštění až po ukončení programu, poslední sloupec označuje počet zpracovaných toků za sekundu

| **Test. případ** | **Počet toků** | **Celkový čas** | **Počet toků za sekundu** |
|---|---|---|---|
| Ideální | 2 750 865 | 1,105 s | 2 489 471 |
| Nejhorší | 2 750 865 | 1,606 s | 1 712 867 |
| | | **průměr** | **2 101 169** |

Výsledky testů jsou zobrazeny v tabulce 4.4. Modul byl schopen v průměru zpracovat 2,1 milionu toků za sekundu. Ve výsledcích je jasně vidět, že v nejhorším případě závisí časová náročnost na počtu načtených *vzorů*. Tato závislost však není velká. Naopak v ideálním případě by časová náročnost

neměla záviset na počtu načtených *vzorů*, protože se vždy budou porovnávat pouze dva *vzory*.

Kapitola 5

# Využití modulu

Využití modulu je široké, zejména díky jeho úspěšnosti a rychlosti klasifikace. První využití modulu by mělo proběhnout v rámci projektu ADiCT, jehož cílem je vytvoření znalostní databáze. Modul by měl pracovat jako vstupní modul systému a klasifikací síťových toků by měl poskytovat užitečné informace o síťovém provozu, které budou následně zpracovány dalšími moduly.

Na základě dosažených výsledků byl modul nasazen do testovacího provozu na síť „netmonlab“. Tato síť je v podstatě infrastrukturou využívanou výzkumným týmem laboratoře monitorování síťového provozu, jež působí na FIT ČVUT v Praze. Na této síti je nasazen monitorovací systém NEMEA, který monitoruje veškerý provoz na síti. Síť netmonlab se také používá pro testování nového projektu ADiCT a v současné době probíhají práce na začlenění navrženého modulu do tohoto systému.

Modul byl několikrát použit pro klasifikaci určitého provozu a aktuálně probíhají testy klasifikace v reálném čase, kdy modul zpracovává veškerý provoz na síti. Veškeré testy proběhly bez problémů, ve vyvinutém klasifikačním modulu nedošlo k žádným chybám.

# Závěr

Vytvoření modulu schopného klasifikovat síťový provoz v reálném čase by mělo značně přispět k bezpečnosti a administraci sítě. Klasifikace síťového provozu značně zvýší šanci detekce škodlivého provozu zejména proto, že minimalizujeme podíl neznámého provozu na síti. Díky tomu se značně zmenší prostor, který může útočník využít k útoku na síť, a také umožníme přesnější zacílení detekčních nástrojů. Klasifikací provozu získáme také větší kontrolu a přehled o sledované síti.

V rámci této práce bylo vytvořeno devět anotovaných datových sad, které byly použity pro trénování a testování navrženého modulu. Tyto datové sady mohou být nyní použity k dalšímu výzkumu. Anotované datové sady jsou totiž velice užitečné pro výzkum a testování nových i existujících nástrojů nejen v oblasti klasifikace.

Hlavním přínosem práce je návrh a implementace klasifikačního algoritmu. Algoritmus využívá pouze základní informace o síťových tocích (porty, délka toku, počet paketů, počet bajtů apod.), které jsou dostupné ve všech současně používaných monitorovacích systémech.

Funkčnost klasifikačního algoritmu byla ověřena implementací prototypu modulu v jazyce C++. Implementace obsahuje i natrénované *vzory* a modul je tak schopen klasifikovat čtyři všeobecné síťové kategorie a devět protokolů aplikační vrstvy. Díky flexibilnímu návrhu je také možné sadu *vzorů* libovolně rozšiřovat, a to bez nutnosti úprav zdrojového kódu.

Pomocí testů bylo ověřeno, že modul dokáže klasifikovat provoz s průměrnou úspěšností 92,82 % a zvládne klasifikovat až 2,1 miliónu toků za sekundu. Tyto hodnoty jsou velice příznivé a dokazují, že algoritmus i modul fungují podle předpokladů.

Modul byl vzhledem k úspěšným výsledkům nasazen do pilotního provozu na síti „netmonlab“ v laboratoři monitorování síťového provozu na FIT ČVUT v Praze. Aktuálně se pracuje na začlenění modulu do projektu ADiCT, který cílí na vytvoření sady informačních zdrojů a znalostní databáze.

Tato bakalářská práce prezentovala vytvořený funkční klasifikační modul,

který bude v budoucnu dále využitelný jako referenční řešení pro příští výzkum a vývoj.

### Budoucí práce

V rámci budoucí práce bude stávající funkční verze klasifikačního modulu porovnána s dalšími klasifikačními metodami, které budou navíc využívat i rozšířené informace v síťových tocích. V první řadě bude vyhodnocen vliv informací, jako jsou třeba hlavičky aplikačních protokolů anebo naopak nízkoúrovňové charakteristiky na paketové vrstvě (rozestupy paketů a jejich velikosti v toku). Uvedené rozšířené informace nebyly v této práci využity záměrně, protože by se tím výrazně omezila použitelnost klasifikačního modulu jen na sítě s pokročilejším exportérem síťových toků. Cílem budoucí práce bude porovnání různých množin informací a posouzení jejich vlivu na přesnost klasifikace.

# Literatura


[1] Labovitz, C.: Internet Traffic 2009-2019. *Presentation at NANOG*, ročník 76, 2019: s. 4–6.

[2] Pahi, T.; Leitner, M.; Skopik, F.: Analysis and Assessment of Situational Awareness Models for National Cyber Security Centers. In *ICISSP*, 2017, s. 334–345.

[3] Leiner, B. M.; Cerf, V. G.; Clark, D. D.; aj.: A brief history of the Internet. *ACM SIGCOMM Computer Communication Review*, ročník 39, č. 5, 2009: s. 22–31.

[4] Hauben, M.: History of ARPANET. *Site de l'Instituto Superior de Engenharia do Porto*, ročník 17, 2007.

[5] Braden, R. T.: Requirements for Internet Hosts - Communication Layers. RFC 1122, Říjen 1989, doi:10.17487/RFC1122. Dostupné z: `https://rfc-editor.org/rfc/rfc1122.txt`

[6] Cotton, M.; Eggert, L.; Touch, D. J. D.; aj.: Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, Srpen 2011, doi:10.17487/RFC6335. Dostupné z: `https://rfc-editor.org/rfc/rfc6335.txt`

[7] Mudrák, D.: Schéma zapouzdření aplikačních dat na jednotlivých vrstvách rodiny protokolů TCP/IP [online]. Wikimedia Commons [vid. 2020-03-24]. Dostupné z: `https://commons.wikimedia.org/wiki/File:Tcpip_zapouzdreni.svg`

[8] Sinha, R.; Papadopoulos, C.; Heidemann, J.: Internet packet size distributions: Some observations. *USC/Information Sciences Institute, Tech. Rep. ISI-TR-2007-643*, 2007.

[9] Hofstede, R.; Čeleda, P.; Trammell, B.; aj.: Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE Communications Surveys Tutorials*, ročník 16, č. 4, 2014: s. 2037–2064.

[10] Cisco Systems, Inc.: NetFlow gives network managers a detailed view of application flows on the network [online]. Cisco Systems, Inc. [vid. 2020-05-24]. Dostupné z: https://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_case_study0900aecd80311fc2.pdf

[11] Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101, Leden 2008, doi:10.17487/RFC5101. Dostupné z: https://rfc-editor.org/rfc/rfc5101.txt

[12] wireshark.org [online]. Wireshark. [vid. 2020-05-15]. Dostupné z: https://www.wireshark.org/

[13] Wireshark User's Guide [online]. Wireshark. [vid. 2020-05-15]. Dostupné z: https://www.wireshark.org/docs/wsug_html/wsug_graphics/ws-main.png

[14] libprotoident [online]. WAND Network Research Group. [vid. 2020-05-19]. Dostupné z: https://research.wand.net.nz/software/libprotoident.php

[15] Cisco Systems, Inc.: Classifying Network Traffic Using NBAR. 2006. Dostupné z: ftp://ns.nobatel.com/pub/manuales/cisco/pdfs/qsnbar1.pdf

[16] Cejka, T.; Bartoš, V.; Svepes, M.; aj.: NEMEA: A Framework for Network Traffic Analysis. In *12th International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, Říjen 2016, doi:10.1109/CNSM.2016.7818417. Dostupné z: http://dx.doi.org/10.1109/CNSM.2016.7818417

[17] CESNET: UniRec [online]. GitHub. [vid. 2020-05-05]. Dostupné z: https://github.com/CESNET/Nemea-Framework/tree/master/unirec

[18] Velan, P.; Krejčí, R.: Flow information storage assessment using IPFIXcol. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, 2012, s. 155–158.

[19] Hutak, L.: A New Generation of an IPFIX Collector. In *7 th Prague Embedded Systems Workshop*, 2019, str. 33.

[20] Alcock, S.; Nelson, R.: Libprotoident: traffic classification using lightweight packet inspection. *WAND Network Research Group, Tech. Rep.*, 2012.

[21] Kasner, Z.: *Klasifikace zařízení na základě tok v počítačovỳch sítích*. bakalářská práce, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2016.

[22] CESNET: Example module [online]. GitHub. [vid. 2019-10-05]. Dostupné z: `https://github.com/CESNET/Nemea-Framework/tree/master/examples/c/module`

[23] Nature Inspired Technologies Group: Testování modelů a jejich výsledků [online]. ČVUT. [vid. 2020-05-01]. Dostupné z: `https://cw.fel.cvut.cz/old/_media/courses/a6m33dvz/03-testovanimodelu.pdf`

Příloha **A**

# Instalační příručka

Tato příloha popisuje, jak zkompilovat a zprovoznit klasifikační modul na systému s linuxovou distribucí.

## A.1 NEMEA framework

Pro spuštění modulu je nezbytný NEMEA framework, který je možné stáhnout z github repozitáře na adrese *https://github.com/CESNET/Nemea-Framework* nebo pomocí příkazu :

- git clone --recursive https://github.com/CESNET/Nemea-Framework

Pro kompilaci je potřeba mít nainstalované tyto balíky: gcc, gcc-c++, make, autoconf, automake, libtool, pkg-config, libxml2-dev a libpcap-dev.
Systém lze poté zkompilovat pomocí následujících příkazů:

1. ./bootstrap.sh
2. ./configure
3. make
4. sudo make install

## A.2 Klasifikační modul

Zdrojové kódy klasifikačního modulu jsou na přiloženém CD. Modul lze zkompilovat pomocí následujících příkazů:

1. autoreconf -i
2. ./configure
3. make
4. make install

## A.3 Nastavení modulu

Klasifikační modul vyžaduje pro práci soubor pro uložení klasifikačních *vzorů*. Cestu k tomuto souboru je nutné předat modulu pomocí parametru „-p“. Formát a struktura formátu tohoto souboru je zobrazena na obrázku A.1 a příklad *vzoru* je zobrazen na obrázku 3.2.

```
#NAME: "název vzoru"
#TAG: "klasifikační štítek"
#DATA: "celkový počet toků z kterých byl vzor natrénován"
#HIT_SCALES: "plný počet evaluačních bodů" "částečný počet evaluačních bodů"
#PORTS: "násobíci koeficient" "číslo portu" . . .
#PROTOCOLS: " násobíci koeficient" "číslo protokolu" . . .
#DURATION: "násobíci koeficient" "5 percentil" "25 percentil" "75 percentil" "95 percentil"
#PACKETS: "násobíci koeficient" "5 percentil" "25 percentil" "75 percentil" "95 percentil"
#BYTES: "násobíci koeficient" "5 percentil" "25 percentil" "75 percentil" "95 percentil"
#BPP: "násobíci koeficient" "5 percentil" "25 percentil" "75 percentil" "95 percentil"
```

Obrázek A.1: Vysvětlení formátu uložení *vzorů*

Všechny potřebné informace pro spuštění modulu je možné získat použitím přepínače „-h“.

Příloha B

# Seznam použitých zkratek

**ADiCT** Asset Discovery, Classification and Tagging

**BPP** Bytes per packet

**CESNET** Czech Educational and Scientific NETwork

**Cisco IOS** Cisco Internetwork Operating System

**CSV** Comma-separated values

**ČVUT** České vysoké učení technické v Praze

**DNS** Domain Name System

**DoS** Denial of service

**FDS** FDS file format

**FIT** Fakulta informačních technologií

**GPL** General Public License

**GUI** Graphical user interface

**HTTP** Hypertext Transfer Protocol

**HTTPS** Secure Hypertext Transfer Protocol

**IANA** Internet Assigned Numbers Authority

**IETF** Internet Engineering Task Force

**IMAP** Internet Message Access Protocol

**IP** Internet protocol

**IPFIX** IP Flow Information Export

**JSON** JavaScript Object Notation

**LGPL** Lesser General Public License

**NBAR** Network-Based Application Recognition

**NEMEA** Network Measurements Analysis

**NTP** Network Time Protocol

**POP3** Post Office Protocol 3

**QoS** Quality of service

**RAM** Random Access Memory

**RFC** Request For Comments

**SMTP** Simple Mail Transfer Protocol

**SSD** Solid-state drive

**SSH** Secure Shell

**TCP** Transmission Control Protocol

**TELNET** Network Virtual Terminal Protocol

**TMC** Traffic monitoring and control

**TRAP** Traffic Analysis Platform

**TRAPCAP** TRAP Capture

**UDP** User Datagram Protocol

**UniRec** Unified Record

**UNIX** Uniplexed Information and Computer Systems

Příloha C

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF