VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ROZŠÍŘENÍ NETFLOW ZÁZNAMŮ PRO ZLEPŠENÍ MOŽNOSTÍ KLASIFIKACE ŠIFROVANÉHO PROVOZU

EXTENDING NETFLOW RECORDS FOR INCREASING ENCRYPTED TRAFFIC CLASSIFICATION CAPABILITIES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. PETER ŠUHAJ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MARTIN HOLKOVIČ**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Šuhaj Peter, Bc.**
Program: Informační technologie Obor: Počítačové sítě a komunikace
Název: **Rozšíření NetFlow záznamů pro zlepšení možností klasifikace šifrovaného provozu**
Extending NetFlow Records for Increasing Encrypted Traffic Classification Capabilities
Kategorie: Počítačové sítě

Zadání:

1. Nastudujte vhodné algoritmy pro klasifikaci šifrovaných síťových dat.
2. Nastudujte parametry síťového provozu vhodné pro klasifikaci šifrovaných dat.
3. Nastudujte techniku monitorování počítačových sítí protokolem NetFlow.
4. Navrhněte a implementujte rozšíření NetFlow záznamů o data vhodná pro klasifikaci šifrovaných síťových toků.
5. Navrhněte a implementujte algoritmus pro klasifikaci šifrovaného síťového provozu z rozšířených NetFlow dat.
6. Vytvořte testovací datasety a proveďte validaci.
7. Vyhodnoťte vytvořené řešení.

Literatura:

- Zhang, Jun, et al. "Robust Network Traffic Identification with Unknown Applications." Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications security. ACM, 2013.
- Lotfollahi, Mohammad, et al. "Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning." Soft Computing (2017): 1-14.
- Nguyen, Thuy TT, and Grenville Armitage. "A Survey of Techniques for Internet Traffic Classification using Machine Learning." IEEE communications surveys & tutorials 10.4 (2008): 56-76.
- Crotti, Manuel, et al. "Traffic Classification through Simple Statistical Fingerprinting." ACM SIGCOMM Computer Communication Review 37.1 (2007): 5-16.
- Williams, Nigel, Sebastian Zander, and Grenville Armitage. "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification." ACM SIGCOMM Computer Communication Review 36.5 (2006): 5-16.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 ze zadání, částečne body 4 a 5.

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

Vedoucí práce: **Holkovič Martin, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 20. května 2020
Datum schválení: 29. října 2019

Abstrakt

Diplomová práca sa zaoberá výberom atribútov sieťových tokov vhodných pre klasifikáciu šifrovanej prevádzky, rozšírením záznamov NetFlow týmito atribútmi a vytvorením nástroja pre klasifikáciu šifrovaných tokov, ktoré používajú protokol TLS. Ako atribúty pre klasifikáciu boli vybrané: veľkosti paketov, medzipaketové medzery, počet paketov v toku a veľkosť toku. Po zvolení atribútov nasleduje návrh rozšírenia záznamov NetFlow o tieto atribúty a návrh algoritmu pre klasifikáciu šifrovanej sieťovej prevádzky. Rozšírenie záznamov bolo implementované v jazyku C v rámci exportéru od Flowmon Networks a.s.. Klasifikátor pre kolektor bol implementovaný v jazyku Python.Klasifikačný algoritmus používa model, k čomu bolo potrebné získať trénovacie dáta. Algoritmus bol pridaný aj na exportér, miesto klasifikácie je možné zvoliť. Po implementácií nasledovalo vytvorenie testovacích dát a vyhodnotenie úspešnosti algoritmu a taktiež testovanie rýchlosti klasifikácie. V najlepšom prípade bola dosiahnutá úspešnosť 47%.

Abstract

Master's thesis deals with selection of attributes proper for classification of encrypted traffic, with the extension of NetFlow entries with these attributes and with creating a tool for classify encrypted TLS traffic. The following attributes were selected: size of packets, interpacket arrival times, number of packets in flow and size of the flow. Selection of attributes was followed by design of extending NetFlow records with these attributes for classifying encrypted traffic. Extension of records was implemented in language C for exporter of the company Flowmon Networks a.s.. Classifier for collector was implemented in language Python. Classifier is based on a model, for which training data were needed. The exporter contains the classifying algorithm too, the place of the classification can be set. The implementation was followed by creation of testing data and evaluation of the accuracy. The speed of the classifier was tested too. In the best case scenario 47% accuracy was achieved.

Kľúčové slová

Šifrovaná prevádzka, NetFlow, klasifikácia, sieťové protokoly.

Keywords

Encrypted traffic, NetFlow, classification, network protocols.

Citácia

ŠUHAJ, Peter. *Rozšíření NetFlow záznamů pro zlepšení možností klasifikace šifrovaného provozu*. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Holkovič

Rozšíření NetFlow záznamů pro zlepšení možností klasifikace šifrovaného provozu

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Martina Holkoviča. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Šuhaj
3. júna 2020

Poďakovanie

Chcel by som poďakovať Ing. Martinovi Holkovičovi za odborné vedenie mojej práce, za cenné rady na konzultáciách. Poďakovanie patrí aj mojej rodine, priateľom, priateľke a každému, kto ma podporoval počas štúdia.

Obsah

1 Úvod **3**

2 Algoritmy pre klasifikáciu šifrovaných sieťových dát **4**
2.1 Strojové učenie 7
2.1.1 Učenie s učiteľom 8
2.1.2 Učenie bez učiteľa 10

3 Monitorovanie počítačových sietí protokolom NetFlow **12**
3.1 Architektúra 12
3.2 Exportér 13
3.2.1 Záznamy NetFlow 14
3.3 Komunikačný protokol 15
3.4 Kolektor 15
3.4.1 Aplikácie 16
3.5 Verzie NetFlow 16
3.5.1 NetFlow v9 17
3.5.2 IPFIX 17

4 Parametre sieťovej komunikácie vhodné pre klasifikáciu šifrovaných dát **18**
4.1 Veľkosť paketov 19
4.2 Medzipaketové medzery 20
4.3 Veľkosť toku 22
4.4 Vybrané atribúty sieťových tokov 22

5 Klasifikačný algoritmus **23**
5.1 Návrh 23
5.2 Implementácia 27

6 Rozšírenie záznamov NetFlow **29**
6.1 Návrh 29
6.1.1 Záznamy NetFlow 29
6.1.2 Plugin pre exportér 30
6.2 Implementácia 34
6.2.1 Pridanie atribútov 35
6.2.2 Spracovanie paketov 36

7 Testovanie a vyhodnotenie **39**
7.1 Testovacie dáta 39

7.2 Vyhodnotenie 39
7.2.1 Rýchlosť 44

8 Záver **45**

Literatúra **46**

A Obsah pamäťového média **50**

B Profily jednotlivých protokolov **51**

C Zdroje súborov PCAP **58**

Kapitola 1

Úvod

Počiatkom vzniku internetu aplikačné protokoly nepoužívali žiadne šifrovanie, čím sa komunikácia dala odpočúvať a ľahko získať súkromné dáta používateľov. Výhodou nešifrovanej komunikácie ale je, že správcovia majú lepší prehľad o dianí na sieti a môžu klasifikovať prevádzku aj na základe aplikačných dát. Postupne sa ale pre zvýšenie súkromia zaviedlo šifrovanie komunikácie, čoho najrozšírenejším predstaviteľom je rodina protokolov SSL/TLS. Tieto protokoly zabezpečujú dôvernosť a integritu dát. Rozšírenie šifrovania komunikácie robí klasifikáciu čoraz obťažnejšou. Ak je nejaká prevádzka plne šifrovaná a používa neštandardné porty, tak sa o nej ťažko získajú nejaké podstatné informácie, ako napríklad o aký aplikačný protokol sa jedná. Postupne preto boli skúmané nové prístupy, ktoré do určitej miery vedia identifikovať o aký aplikačný protokol alebo o akú aplikáciu ide.

Ešte predtým ako je možné klasifikovať prevádzku, potrebujeme ju zachytiť alebo ukladať informácie z paketov, ktoré prechádzajú cez monitorovacie zariadenie. Existujú rôzne metódy a protokoly na monitorovanie sieťovej prevádzky. Jedným z rozšírených protokolov je NetFlow, ktorý poskytuje metaúdaje o jednotlivých tokoch sieťovej prevádzky. Monitorovanie sa vykonáva na aktívnych sieťových prvkoch, ktoré musia protokol podporovať alebo na sonde. Získané údaje sú potom cenným zdrojom informácií pre správu. Príkladom ich využitia je optimalizovanie vyťaženia siete, škálovanie siete, zabezpečenie kvality služieb alebo monitorovanie bezpečnosti siete.

Cieľom tejto práce je navrhnúť a implementovať rozšírenie záznamov NetFlow o parametre sieťových tokov vhodné pre klasifikáciu šifrovaných dát a navrhnutie nástroja, ktorý pomocou údajov zo záznamov klasifikuje toky do tried na základe aplikačného protokolu. Štruktúra práce je nasledujúca. V kapitole 2 sú popísané existujúce metódy používané pre klasifikáciu šifrovanej komunikácie. V kapitole 3 je popísaný protokol NetFlow, ďalej kapitola 4 sa zaoberá atribútmi vhodnými pre klasifikáciu. Po naštudovaní potrebných informácií nasleduje v kapitole 5 návrh a implementácia klasifikačného algoritmu, ktorý podľa vybraných atribútov klasifikuje toky. Ďalej kapitola 6 sa zaoberá návrhom a implementáciou modulu pre exportér firmy Flowmon Networks a. s., ktorý poskytuje rozšírenie záznamov a exportovanie potrebných atribútov. Na záver v kapitole 7 je algoritmus vyhodnotený.

Kapitola 2

Algoritmy pre klasifikáciu šifrovaných sieťových dát

Existuje niekoľko možností na klasifikáciu tokov sieťovej komunikácie, z ktorých každá metóda aplikuje iné princípy. Cieľom klasifikácie je zistiť aký typ aplikačného protokolu (napríklad HTTP, SMTP alebo DNS) bol použitý alebo identifikácia konkrétnej aplikácie (napríklad Skype, Spotify, atď.). Tradične sa používa prístup klasifikácie na základe portov. Tieto prístupy ale v dnešnej dobe nie vždy môžu byť použiteľné. Sieťové aplikácie nemusia používať štandardné porty predpísané v zozname *Internet Assigned Numbers Authority* (IANA[1]), čím sa znemožní identifikácia na základe portov. Taktiež nie každá aplikácia musí mať registrovaný port v IANA [33].

Ďalšou často používanou metódou je klasifikácia na základe obsahu paketov. Nazýva sa aj ako *Deep Packet Inspection* (DPI). Väčšina týchto metód je založená na vyhľadávaní signatúr v aplikačných dátach. Signatúry sú reťazce, ktoré pomáhajú k identifikácií daného protokolu či aplikácie. Na vyhľadávanie reťazcov sa používajú regulárne výrazy, ako napríklad nasledujúci výraz pre HTTP[2]:

```
http/(0\.9|1\.0|1\.1) [1-5][0-9][0-9] [\x09-\x0d -~]*(connection:
|content-type:|content-length:|date:)|post [\x09-\x0d -~]* http/[01]\.[019]
```

Najprv je potrebné vytvoriť databázu signatúr, ktoré budú potom vyhľadávané v paketoch [40]. Metóda sa nespolieha na čísla portov, čím umožňuje klasifikáciu aj pri zmenených portoch. Je ale náročnejšia na výpočet a signatúry je potrebné aktualizovať pri zmene protokolu. Taktiež rozšírením šifrovania aplikačných dát sa znížila jeho použiteľnosť [20].

Nové prístupy identifikujú toky sieťovej komunikácie na základe štatistických vlastností, nespoliehajú sa tak na porty ani nepotrebujú prístup k aplikačným dátam. Tým sú tieto metódy vhodné aj pre klasifikáciu šifrovanej komunikácie [36]. Tieto prístupy na základe vybraných atribútov sieťových tokov klasifikujú toky do jednotlivých kategórií. Táto klasifikácia vychádza z predpokladu, že rôzne typy sieťovej prevádzky pre jednotlivé protokoly alebo aplikácie majú jedinečné štatistické vlastnosti. Tento prístup je vhodný aj pre využitie v tejto práci, keďže potrebné údaje vieme získať použitím NetFlow.

Väčšina štatistických prístupov je založená na strojovom učení. Pre tieto algoritmy je najprv potrebné nazbierať vstupné dáta (zachytiť dostatočný počet tokov), na základe

[1] https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[2] http://l7-filter.sourceforge.net/layer7-protocols/protocols/http.pat

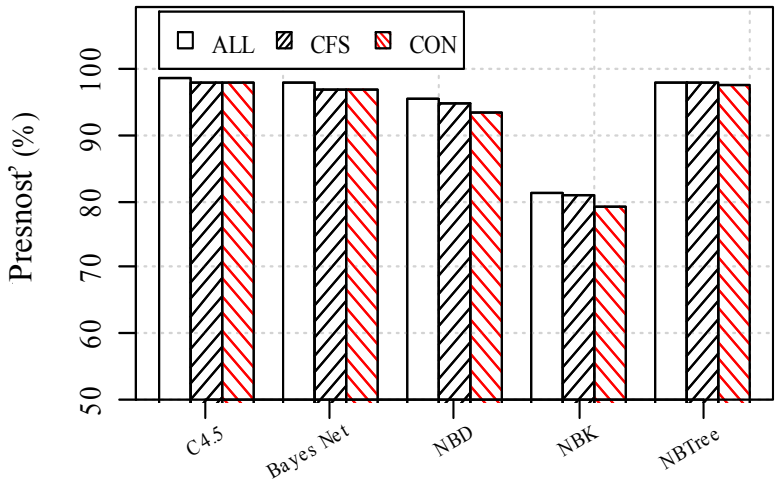
ktorých sa vytvorí model a nasledovne sa budú neznáme toky klasifikovať podľa vytvorených modelov [31]. Pri klasifikácií sieťovej komunikácie mapujeme inštancie sieťových tokov do rôznych tried prevádzky, ktoré môžu byť vytvorené na základe:

- **typu prevádzky:** prenos súborov, multimédia, prehľadávanie webu, atď.;
- **aplikačného protokolu:** HTTP, SMTP, FTP, atď.;
- **aplikácie:** Skype, Spotify, Messenger, atď.;
- **zdroju prevádzky:** užívateľ, stroj.

Toky sú reprezentované pomocou jednotlivých vlastností (napr. veľkosti jednotlivých paketov) a hodnotami týchto vlastností. Jednotlivé toky sú charakterizované množinou týchto vlastností a na základe nich klasifikované [44].

V existujúcich výskumoch sa často aplikujú metódy strojového učenia s učiteľom. Moore [47] použil naivný Bayesov klasifikátor pre klasifikáciu prevádzky na základe typu aplikácie (web, e-mail, atď.). Použitím základnej verzie dosiahol presnosť 65%, rôznymi vylepšeniami až 95%. Tento algoritmus na vytvorenie modelu aplikoval aj Nguyen [34], ktorý klasifikoval konkrétnu aplikáciu na základe obmedzeného počtu paketov, keďže všetky pakety toku nemusia byť vždy dostupné. Williams [44] redukoval vlastnosti použité na trénovanie a klasifikáciu pomocou rôznych algoritmov a porovnával presnosť vybraných algoritmov strojového učenia. Jeden z nich bol naivný Bayesov klasifikátor. Pri tejto metóde je ale potrebné aby vlastnosti mali konečný počet hodnôt, čo pri spojitých veličinách nie je zaručené. Na to sa použije buď diskretizácia (NBD) [45], alebo odhad hustoty jadra (NBK) [29]. V tomto porovnaní použil oba prístupy, pri NBK dosahoval presnosť okolo 80% a pri NBD okolo 94% pri klasifikácií vybraných aplikačných protokolov.

V rovnakej štúdií Williams skúmal aj rozhodovacie stromy typu C4.5 a dosiahol presnosť až 98%. Tento prístup bol aj najrýchlejší pri klasifikácií. Okrem týchto algoritmov porovnával ešte Bayesovskú sieť a naivný Bayesov strom. Pri testovaní použili nasledujúce množiny vlastností tokov: všetky, vlastnosti vybrané na základe korelácie (CFS) a vlastnosti vybrané na základe konzistencie (CON). Obrázok 2.1 zobrazuje presnosť a rýchlosť jednotlivých algoritmov. Nguyen [35] v ďalšom výskume podobnému k [34] využil aj rozhodovacie stromy, kde dosiahol presnosti okolo 98%.



Obr. 2.1: Presnosť jednotlivých algoritmov porovnávaných Williams [44].

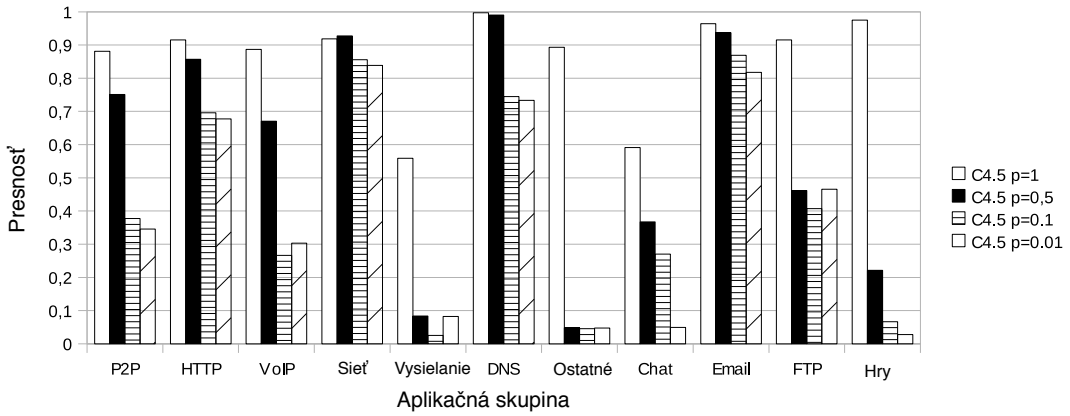
Niektoré štúdie používajú vlastne vytvorené modely. Crotti [23] vytvoril model pomocou funkcie hustoty pravdepodobnosti. Pri vytváraní modelu a pri klasifikácií použil hodnoty prvých N paketov a medzipaketových medzier toku. Klasifikoval vybrané aplikačné protokoly. Wang [43] použil rovnaký model, zdrojom dát boli iba veľkosti prvých N paketov a dosiahol presnosť 87%.

Pri klasifikácií šifrovanej prevádzky sa používa aj učenie bez učiteľa, kde rozšíreným algoritmom je zhlukovací algoritmus K–means, ktorý aplikoval Erman [25, 26]. Identifikoval aplikačné protokoly, zvyšovaním počtu zhlukov dosiahol presnosť okolo 80%. V jednom z týchto výskumov [25] použil ešte zhlukovacie algoritmy DBSCAN a AutoClass, celkovo najpresnejší bol AutoClass s presnosťou okolo 92%. Bernaille [18] použil K–means na klasifikáciu aplikačných protokolov pomocou prvých N paketov, kde viac ako 80% tokov vybraných protokolov identifikoval správne.

Jaber [28] tiež využil algoritmus K–means a tiež klasifikoval na základe veľkostí prvých N paketov. Pre jednotlivé pakety v poradí aplikoval zhlukovanie, kde vytvoril 20 tried na jednotlivých úrovniach. Pre tieto triedy určil pravdepodobnosti aplikačného protokolu pri pakete neznámeho toku, ktorý padne do tejto triedy. Pri príchode nového toku pre jednotlivé pakety sa určili zhluky a pravdepodobnosti, z toho sa vypočítala výsledná pravdepodobnosť. Dosiahol úspešnosť okolo 97%, pri podobnej štúdií [27], ale použitím medzipaketových medzier, dosiahol presnosť okolo 99%. Ich metódy poskytujú aj možnosť použitia portov pri klasifikácií, kedy administrátor môže určiť mieru dôvery v použitých portoch. Ďalšie štúdie a v nich použité prístupy môžeme nájsť v zhrnutí od Nguyen [33] alebo Deebalakshmi [24].

Vyššie uvedené štúdie klasifikovali prevádzku na základe informácií na úrovní paketov, niektoré použili aj informácie na úrovni toku. Zachytávali celú prevádzku pre trénovanie, potom vyextrahovali jednotlivé toky a potrebné údaje. Existujú aj štúdie ktoré používajú protokol NetFlow a pri klasifikácií používajú vlastnosti na úrovni toku. Carela [21] použil tieto atribúty a odvodené vlastnosti (napr. priemerná veľkosť paketov). Aplikoval učenie s učiteľom, model vytvoril pomocou rozhodovacieho stromu C4.5. Cieľom bolo klasifikovať vzorkovanú prevádzku do tried podľa vybraných aplikácií. Najprv použil úplné toky (tj. bez vzorkovania) pre trénovanie, kde pri klasifikácií aplikoval vzorkovanie a postupne znižoval frekvenciu. Znižovaním postupne klesala presnosť klasifikácie. Pri použití vzorkovaných tokov pri trénovaní (frekvencia vzorkovania sa zhodovala pri trénovaní a klasifikácií) sa

výrazne zvýšila presnosť, pri väčšine aplikácií dosiahla hodnotu okolo 90%. Obrázok 2.2 zobrazuje presnosť pri použití nevzorkovaných tokov pre trénovanie (parameter p udáva frekvenciu vzorkovania).



Obr. 2.2: Presnosť algoritmu pri použití nevzorkovaných dát na trénovanie [21].

Bakshi [17] využil taktiež NetFlow, kde záznamy o oboch smeroch toku agregoval do jedného záznamu a rozšíril ešte ďalšími informáciami, ako napríklad prenosová rýchlosť v bitoch (bity/s) alebo v paketoch (pakety/s). Toky pochádzali od 15 najpopulárnejších webových aplikácií (napr. Youtube, Messenger). Použil dvojfázový algoritmus, kde v prvej fáze pomocou algoritmu K–means zhlukoval toky do tried, kde dostal 12 unikátnych zhlukov. Ďalej tieto triedy použil pri trénovaní rozhodovacieho stromu C5.0. Dosiahol úspešnosť 92.37%, použitím adaptívneho boostingu až 96.67%.

Existuje veľa rôznych prístupov štatistickej klasifikácie. Líšia sa v použitých vlastnostiach prevádzky, používajú rôzne algoritmy. Neexistuje jeden dokonalý prístup pre klasifikáciu šifrovanej prevádzky. Presnosť týchto algoritmov závisí aj od množstva a kvalite trénovacích dát, keďže sú založené na nejakom modelu. Datasety výskumov ale často nie sú publikované, alebo nie sú označené. Ostáva možnosť spoliehať sa na porty pri označovaní dostupných dát, alebo vytvoriť nové vlastné datasety. Často sa ale vytvorí menší dataset, ako by bolo potrebné pre presnú klasifikáciu [19].

Vo zvyšnej časti kapitoly budú postupne popísané najčastejšie používané algoritmy, konkrétne rôzne typy strojového učenia. V prvom rade sa kapitola zaoberá s metódou učenia s učiteľom, v rámci nej s naivným Bayesovským klasifikátorom, rozhodovacími stromami a sú spomenuté aj vlastné modely. Ďalšia časť popisuje učenie bez učiteľa a podrobnejšie sa zaoberá so zhlukovacím algoritmom K–means.

2.1 Strojové učenie

Strojové učenie je technika, ktorá umožňuje počítačom učiť sa a rozhodovať sa bez explicitného programovania [7]. Algoritmy strojového učenia používajú iba svoje vstupné dáta (*dataset*, resp. trénovacie dáta), podľa ktorých sa naučia mapovať neznáme vstupy na požadované výstupy. Zväčšovaním vstupu zvyšujú svoju presnosť. Vo vstupne nachádzajú vzory, ktoré pre človeka nemusia byť jasne viditeľné. Dnes je strojové učenie rozšírené, používa sa napríklad na spracovanie obrazu, rozpoznávanie reči, vo finančnom sektore, v priemysle,

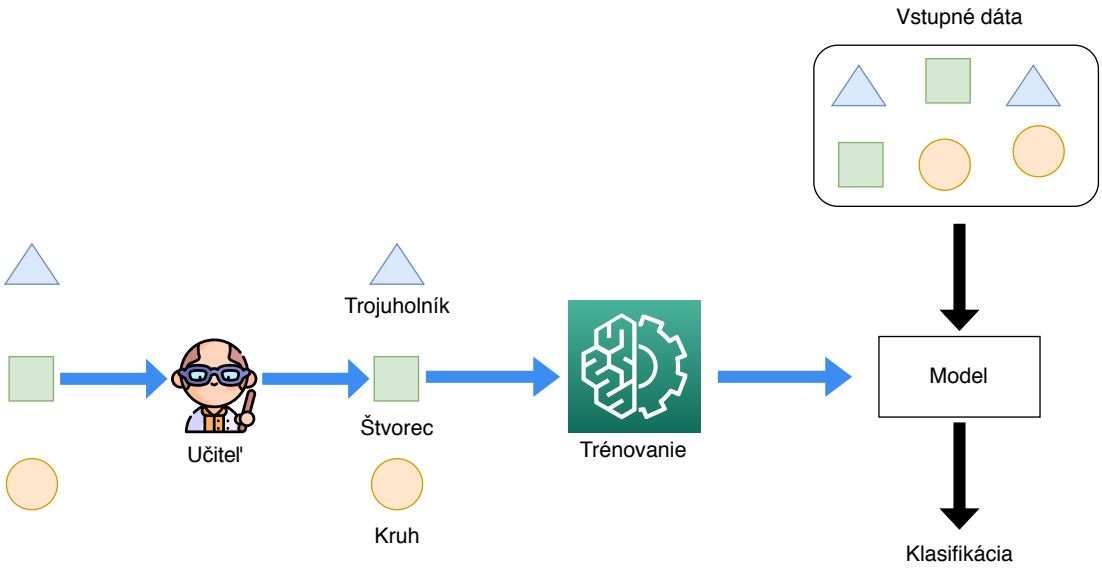
atď. Taktiež sa začalo používať na klasifikáciu sieťovej komunikácie. Strojové učenie zahŕňa viacero metód, ako napríklad učenie s učiteľom a učenie bez učiteľa [8].

2.1.1 Učenie s učiteľom

Pri učení s učiteľom trénovacie dáta obsahujú dvojice vstupných dát (*features*), hodnoty ktorých sú použité pri vytváraní modelu a neskôr klasifikácií, a k nim požadované výstupy (*labels*). Učiteľom je vlastne človek, ktorý označuje vstupné trénovacie dáta s požadovanými výstupmi. Použitý algoritmus vytvorí model, buď v jednom kroku, alebo iteratívne. V prípade iteratívneho modelu na základe označenia požadovaného výstupu model určí, či správne klasifikoval nejaký trénovací vstup. V prípade nesprávnej klasifikácie sa model sám koriguje, tým sa postupne vylepšuje. Trénovanie sa vykonáva dovtedy, kým model nedosiahne učiteľom požadovanú presnosť.

Problémom pri učení s učiteľom môže byť preučenie (*overfitting*), kedy model má slabú schopnosť generalizácie (je príliš prispôsobený trénovacím dátam). Jedná sa o jav, keď model sa naučí nevhodné detaily trénovacieho vstupu (tzv. šum) a nie ten správny vzor, ktorý potrebujeme pre klasifikáciu neznámych vstupov. Pri validácií potom model dosahuje nízke presnosti, keďže hľadá tie nevhodné vzory. Preučeniu môžeme predísť napríklad zväčšením trénovacích dát, krížovou validáciou, znížením počtu vlastností [6].

Výstupom modelu môže byť trieda pri klasifikácií, alebo predikovaná číselná hodnota pri regresií [9]. Existujú rôzne metódy, ako rozhodovacie stromy, naivný Bayesov klasifikátor, neurónové siete, atď. Obrázok 2.3 zobrazuje kroky pri učení s učiteľom. Pri klasifikácií sieťovej komunikácie trénovacie dáta sú vlastnosti jednotlivých tokov a k nim odpovedajúce protokoly.



Obr. 2.3: Algoritmus učenia s učiteľom [9].

Naivný Bayesov klasifikátor

Táto metóda je založená na Bayesovskej teórií. Nech $x = (x_1, ..., x_n)$ sú vzorky, ktoré v tomto prípade reprezentujú jednotlivé toky. Každý tok x_i je popísaný pomocou m diskriminátorov $\{d_1^{(i)}, ..., d_m^{(i)}\}$, ktoré môžu nadobúdať numerické alebo diskrétne hodnoty. Pri

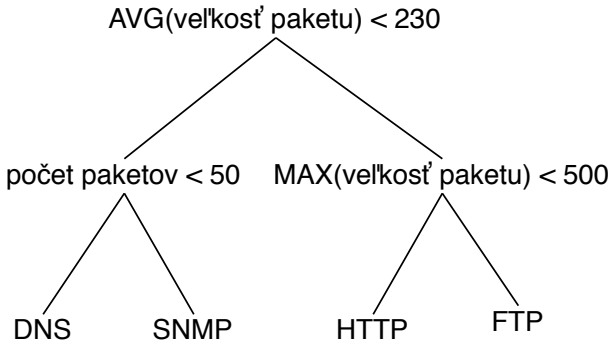
sieťovej prevádzke $d_j^{(i)}$ je j-tý diskriminátor i-tého toku, napríklad priemerná veľkosť paketov v toku. Bayesov klasifikačný problém sa zaoberá s vytvorením štatistického modelu, kde každý nový tok y je klasifikovaný do jednej z tried c_j podľa vzorca

$$p(c_j|y) = \frac{p(c_j) * f(y|c_i)}{\sum_{c_j} p(c_j) * f(y|c_j)}$$

, kde $p(c_j)$ je pravdepodobnosť príslušnosti do triedy c_j nezávisle na vstupu y, $f(y|c_j)$ je podmienená pravdepodobnosť y za predpokladu c_j, menovateľ je normalizačná konštanta. Vstup je klasifikovaný do tej triedy c_j, pri ktorej hodnota $p(c_j|y)$ je maximálna spomedzi všetkých pravdepodobností. Predpokladá nezávislosť diskriminátorov $\{d_1, ..., d_m\}$, čo ale pri sieťových tokoch nie je realistické, napriek tomu sa dá tento algoritmus použiť [47].

Rozhodovacie stromy

Jedná sa o typ klasifikačného modelu. Je definovaný ako strom, kde uzly stromu predstavujú testy na hodnoty jednotlivých atribútov, vetvy z konkrétneho uzlu reprezentujú možné výsledky testu a listy stromu reprezentujú jednotlivé triedy. Klasifikácia začína od koreňa stromu, v každom uzle je otestovaná hodnota patričného atribútu a na základe výsledku sa pokračuje ďalej. Klasifikácia končí keď sa dôjde na listový uzol. Príklady algoritmov používajúce rozhodovacie stromy sú ID3, C4.5, C5.0 [39]. Obrázok 2.4 zobrazuje jednoduchý príklad rozhodovacieho stromu.

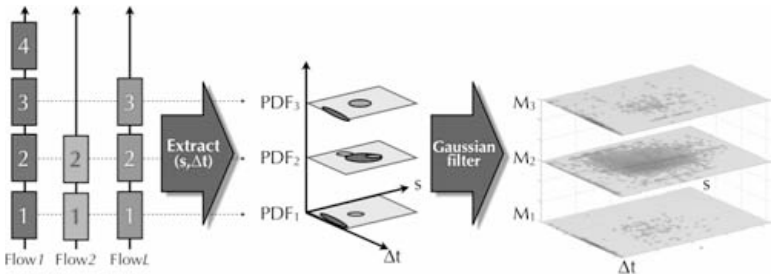


Obr. 2.4: Rozhodovací strom založený na vlastnostiach paketov vybraného toku.

Vlastné modely

Oproti vyššie spomenutých modelov existujú štúdie, v ktorých autori vytvárali vlastný model. Crotti et. al. [23] vo svojom výskume použili veľkosti jednotlivých paketov, ich poradie a medzipaketové medzery. Pre jednotlivé pakety v poradí vytvoril matice použitím dvojíc (veľkosť paketu, medzipaketová medzera) z funkcie hustoty pravdepodobnosti použitím Gaussovského filtra. Tieto matice reprezentujú masku daného protokolu. Obrázok 2.5 zobrazuje proces výpočtu týchto matíc. Ďalej boli vyrátané prahové hodnoty pre jednotlivé protokoly pre rôzne počty použitých dvojíc. Prahová hodnota sa vypočíta ako súčet priemeru a štandardnej odchýlky skóre anomálie jednotlivých vstupných tokov daného protokolu oproti masky tohto protokolu. Tento postup bol vykonaný pre každý protokol, tým sa vytvorili takzvané „odtlačky“ jednotlivých protokolov, ktoré obsahujú dvojicu maska a prah (zoznam „odtlačkov“ reprezentuje klasifikačný model). Pri príchode nového toku sa

vyráta skóre anomálie protokolu oproti každého protokolu pomocou masky do prvých n paketov, ktoré sa ešte normalizuje (vydelí) hodnotou prahu. Tok sa zaradí do tej triedy, pri ktorej je táto hodnota najnižšia [23].



Obr. 2.5: Proces vytvorenia matíc, ktoré použili Crotti et. al ako základ modelu [23].

2.1.2 Učenie bez učiteľa

Učenie bez učiteľa (*unsupervised learning*) zoskupuje vstupné dáta na základe podobných vlastností, nazýva sa aj ako zhlukovanie (*clustering*). Tento prístup na rozdiel od učenia s učiteľom nepoužíva označené vstupné dáta. Cieľom je o neznámych dátach získať nejaké nové znalosti, ktoré umožnia klasifikáciu. Neoznačené vstupné dáta sa zoskupujú do zhlukov podľa vybraného algoritmu a potom jednotlivé zhluky sú označené. Výhodou tohto prístupu pri klasifikácií sieťovej prevádzky je, že sa môžu identifikovať nové, neznáme aplikácie či protokoly [25]. Príkladom metódy tohto typu je zhlukovacia metóda K-means [9]. Proces učenia bez učiteľa zobrazuje obrázok 2.6.

Obr. 2.6: Algoritmus učenia bez učiteľa [9].

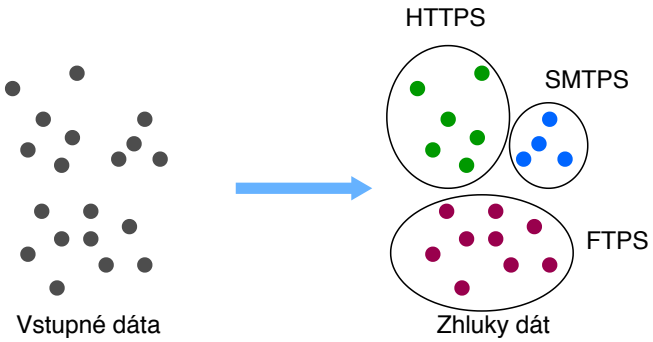
K–means

Jedná sa o jednoduchú ale rozšírenú zhlukovaciu metódu. Algoritmus začína so vstupnými dátami a zadaným počtom požadovaných zhlukov. Na začiatku náhodne určí stredy (centroidy) jednotlivých zhlukov. Ostatné prvky sú priradené do zhlukov na základe Euklidovskej vzdialenosti od stredného bodu, ktorá je definovaná ako:

$$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

, kde n je počet dimenzií priestoru. Prvok sa priradí do toho zhluku, k centroidu ktorého je najbližšie. Ďalej iteratívne sa znovu vypočítajú centroidy zhlukov a prvky sa preusporiadajú na základe vzdialeností. Tento proces sa vykonáva, až kým sa zhluky neustália, tj. kým kvadratická chyba nedosiahne svoje minimum. Pri klasifikácií sieťového toku sa ako

vektor môžu použiť napríklad niektoré štatistické vlastnosti, počet dimenzií môže byť počet protokolov ktoré budú klasifikované [46]. Obrázok 2.7 zobrazuje dáta pred a po vykonaní zhlukovania.



Obr. 2.7: Dáta pred–po zhlukovaní.

Kapitola 3

Monitorovanie počítačových sietí protokolom NetFlow

NetFlow je otvorený protokol organizácie IETF, pôvodne vyvinutý firmou Cisco, pre monitorovanie sieťovej komunikácie na základe sieťových tokov. Cieľom protokolu je zlepšenie škálovateľnosti. Je to nástrojom pre sieťových administrátorov, ktorým umožňuje napríklad monitorovanie siete, monitorovanie a profilovanie aplikácií alebo používateľov, plánovanie siete, bezpečnostnú analýzu, účtovanie [22].

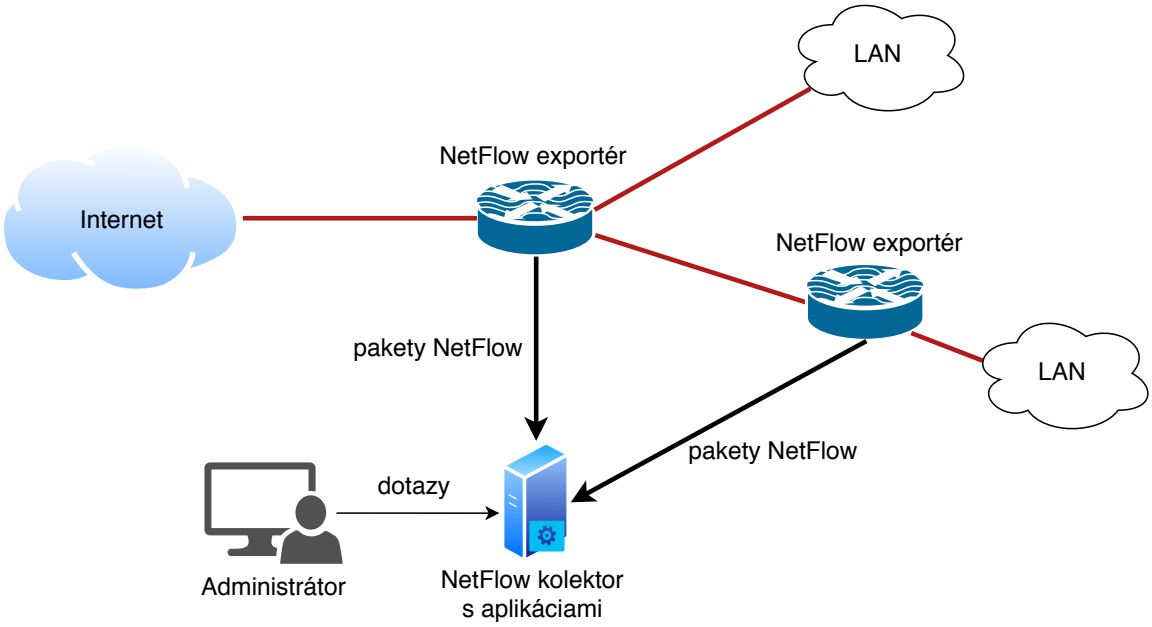
Pred samotným popisom protokolu NetFlow je potrebné definovať sieťový tok, ktorý môžeme definovať viacerými spôsobmi. Podľa RFC3954 [13] je tok definovaný ako jednosmerná sekvencia paketov majúce nejakú spoločnú vlastnosť, ktorá prechádza cez sieťové zariadenie. Protokol NetFlow definuje sieťové toky pomocou kombinácie kľúčových atribútov paketu. Paket patrí do konkrétneho toku, ak má všetky kľúčové atribúty zhodné s atribútmi daného toku. Prijatie paketu s rozličnými atribútmi je považovaný za nový tok. Kľúčové atribúty sú nasledujúce [22]:

- zdrojová IP adresa,
- cieľová IP adresa,
- zdrojový port,
- cieľový port,
- typ IP protokolu,
- typ služby,
- rozhranie (vstupné/výstupné).

3.1 Architektúra

Architektúra NetFlow využíva distribúciu jednotlivých úloh potrebných pre monitorovanie sieťovej komunikácie. Prevádzka je zachytávaná na sieťovom zariadení, spracované údaje sú potom odoslané na iné zariadenie, kde sa vykonáva analýza. Ako prvé je potreba spracovať pakety a uložiť potrebné informácie. Toto vykonáva zariadenie s názvom *exportér*, ktorý spracuje pakety a získava informácie pre vytvorenie/aktualizáciu záznamov (napr. cieľová IP adresa, veľkosť paketu, atď.). Následne sú záznamy o tokoch z jedného alebo z viacerých

exportérov odoslané na zariadenie s názvom *kolektor*. Kolektorov v architektúre môže byť viac. Na prenos dát sa používa *protokol* NetFlow, formát paketov závisí od verzie. Exportované dáta sa na kolektore analyzujú pomocou *aplikácií*, ktoré poskytujú pre správcov užitočné informácie získané z tokov. Architektúra NetFlow je zobrazená na obrázku 3.1.



Obr. 3.1: Príklad architektúry NetFlow.

3.2 Exportér

Exportér môže byť sieťový prvok na ktorom beží NetFlow, alebo dedikované zariadenie, tzv. sonda. Výhodou sondy je, že sa zachová výkon sieťových prvkov, keďže sa nemusia zaoberať s vytváraním, aktualizáciou a exportovaním záznamov. Ďalšími výhodami sú presnejšie dáta a jednoduchšia aktualizácia. Sondu môžeme do siete zapojiť dvomi spôsobmi. Prvá možnosť je pripojenie technológiou SPAN (*Switched Port Analyzer*), kde sa využíva zrkadlenie portov. Prevádzka z jedného alebo viacerých portov sieťového zariadenia je preposielaná na port sondy. Ďalšia možnosť je použitie zariadenia TAP (*Test Access Point*), čo je pasívne zariadenie, ktoré sa zapája medzi dve sieťové zariadenia. Dva porty sú zapojené do siete, na tretí port sa preposielajú oba smery komunikácie, kde je zapojený exportér. Výhodou oproti SPAN je, že môže byť umiestnená kdekoľvek v sieti a je ťažko detekovateľná (nemá MAC adresu) [5].

Na sieťových prvkoch, cez ktoré prechádzajú gigabity dát by ukladanie údajov o každom pakete mohlo byť veľmi náročné na výpočet. Riešením tohto problému je nastavenie vzorkovania (*sampling*) prevádzky, kedy sa nespracuje každý paket, iba vybrané pakety. Vzorkovanie je technika výberu podmnožiny celkových dát v sieti, kde cieľom je čo najpresnejšia reprezentácia charakteristík pôvodnej prevádzky [22]. Celkové dáta sa nazývajú ako rodičovská populácia, vzorkované dáta ako populácia potomkov. Výber paketu pre spracovanie závisí na typu vzorkovania. Prvá možnosť je deterministické vzorkovanie, ktoré má viacero zastúpení:

- **vzorkovanie na základe počtu** – tzv. vzorkovanie 1:N, kedy sa deterministicky vyberá jeden paket z N po sebe nasledujúcich paketov, teda každý N-tý paket (napr. každý 100. paket);
- **vzorkovanie v čase** – spracujú sa pakety vždy po nejakom periodickom časovom intervale za určitú dobu (napr. každých 100 ms sa spracuje 5ms komunikácie);
- **vzorkovanie na základe veľkostí** – spracujú sa pakety, ktoré spĺňajú podmienku pre ich veľkosť (napr. pakety väčšie ako 1000 bajtov).

Nevýhodou tohto typu vzorkovania je, že vyberá pakety v určitom poradí, čo v niektorých prípadoch môže byť problém. Ak sa nejaká komunikácia vyskytuje periodicky (napr. VoIP) tak sa môže stať, že sa väčšinou spracujú tieto pakety, kým ostatné nie. Môže to potom skresliť výsledok využitia siete daným protokolom, resp. aplikáciou. Napríklad ak VoIP tvorí reálne iba 10% prevádzky ale väčšina paketov sa spracuje z tejto komunikácie kvôli periodickému vzorkovaniu, tak výsledkom bude vyšší pomer využitia siete daným protokolom v porovnaní s ostatnou prevádzkou.

Druhý typ vzorkovania je náhodné vzorkovanie, kedy sa z daného intervalu vyberajú spracované pakety náhodne. Tento prístup generuje vzorkované dáta, ktoré lepšie reprezentujú vlastnosti pôvodných dát oproti deterministickému vzorkovaniu. Odporúča sa použitie tohto typu vzorkovania [22]. Obrázok 3.2 zobrazuje porovnanie deterministického a náhodného vzorkovania.

Obr. 3.2: Ukážka deterministického a náhodného vzorkovania [22].

3.2.1 Záznamy NetFlow

Pri spracovaní paketu, ktorý patrí do zatiaľ neznámeho toku sa vytvorí nový záznam (*flow record*). Následne pri prijatí paketu už známeho toku sa záznam iba aktualizuje o potrebné informácie. Exportér si záznamy ukladá do svojej vyrovnávacej pamäte, do tzv. *NetFlow cache*. Jednotlivé záznamy obsahujú rôzne položky, ktoré podľa typu môžeme rozdeliť nasledovne:

- **kľúčové atribúty** – Atribúty, ktoré jednoznačne definujú tok;
- **čítače** – Slúžia ako počítadlá nejakej hodnoty, napríklad počet paketov alebo veľkosť toku v bajtoch;
- **časovače** – Každý záznam má svoje časovače. Prvým je aktívny, ktorý počíta dobu od prijatí prvého paketu toku. Druhý je pasívny, ktorý počíta dobu od prijatia posledného paketu v danom toku. Časovače sa používajú potom na určenie expirácie záznamu. Nenachádzajú sa explicitne v zázname;
- **dodatočné informácie** – Dodatočné parametre komunikácie, ako napríklad IP adresa ďalšieho smerovača (*next hop*), príznaky TCP, atď. V novších verziách môžu byť pridané aj nové atribúty, čím si organizácie môžu rozšíriť formát záznamov o nové informácie.

Príklad záznamov je zobrazený na obrázku 3.3, kde kľúčové atribúty sú označené hrubým písmom.

SrcIf	**SrcIPadd**	DstIf	**DstIPadd**	**Protocol**	**TOS**	Flgs	Pkts	**Src Port**	Src AS	**Dst Port**	Dst AS	NextHop	Bytes/ Pkt	Active	Idle
Fa1/0	173.100.21.2	Fa0/0	10.0.227.12	11	80	10	11000	162	5	162	15	10.0.23.2	1528	1745	4
Fa1/0	173.100.3.2	Fa0/0	10.0.227.12	6	40	0	2491	15	196	15	15	10.0.23.2	740	41.5	1
Fa1/0	173.100.20.2	Fa0/0	10.0.227.12	11	80	10	10000	161	180	161	15	10.0.23.2	1428	1145.5	3

Obr. 3.3: Ukážka záznamov NetFlow.

Jednotlivé záznamy z NetFlow *cache* sa exportujú na kolektor. Odoslanie a odstránenie záznamu z cache sa vykoná pri splnení niektorej z podmienok [22]:

- **Aktívny časovač** – Tento časovač začína počítať čas od prijatí prvého paketu daného toku. Záznamy o tokoch, ktorých doba presahuje prahovú hodnotu sú exportované.
- **Inaktívny časovač** – Počíta čas od prijatí posledného paketu konkrétneho toku, pri prijatí paketu je vynulovaný a počíta znovu. Ak po nastavenú časovú hodnotu sa neprijme žiadny paket toku, záznam sa exportuje.
- **Veľkosť cache** – Pri zaplnení NetFlow cache na zariadení je potrebné uvoľniť miesto pre nové záznamy. V tomto prípade sa pomocou rôznych heuristík vyberú záznamy, ktoré sa exportujú a budú odstránené z cache.
- **Koniec TCP spojenia** – V prípade, že sa prijme paket obsahujúci príznak na ukončenie TCP spojenia (FIN alebo RST) záznam sa exportuje.
- **Vynútenie aplikáciou na sonde** – V niektorých prípadoch chceme exportovať záznam ešte pred ukončením toku. Napríklad v prípade komunikácie VoIP po nadviazaní spojenia už vieme kto s kým komunikuje a túto informáciu chceme exportovať. Existuje možnosť explicitného exportovania záznamu, kde na základe podmienky danou aplikáciou, ktorá beží na sonde, sa exportuje záznam ešte pred ukončením spojenia.

3.3 Komunikačný protokol

Na odosielanie sa používa komunikačný protokol NetFlow. Pakety prenášajúce dáta NetFlow môžu obsahovať viacero expirovaných záznamov, ich množstvo a obsah závisí na verzii protokolu. Taktiež sa líši aj formát paketu pre jednotlivé verzie. Paket sa skladá z hlavičky NetFlow a z tela, hlavička obsahuje informácie ako verzia protokolu, počet záznamov, atď., telo obsahuje záznamy jednotlivých tokov. Dáta sa odosielajú transportným protokolom UDP, pričom nemá štandardizovaný port. Najčastejšie sa používajú porty 2055, 9995 a 9996. Neskôr vo verzií 9 bola pridaná podpora pre transportný protokol SCTP.

3.4 Kolektor

Kolektor prichádzajúce pakety NetFlow spracuje a uloží si ich na disk alebo do databáze. Môže spracovať pakety z viacerých exportérov. Ďalej sa vykonáva analýza týchto dát pomocou rôznych aplikácií.

V niektorých prípadoch sú potrebné údaje o oboch smeroch komunikácie v jednom zázname (tzv. *biflow*). V prípade použití verzie, ktorá neumožňuje vytváranie tohto typu

záznamu na exportéru sa táto činnosť vykonáva na kolektore. Kolektor príjme dva záznamy konkrétneho spojenia a vytvorí z nich jeden. Ďalšou možnou funkciou kolektoru je tzv. deduplikácia, ktorá odstraňuje duplicitné záznamy. V prípade, ak v sieťovej štruktúre je nasadených viacero exportérov, záznam o jednom toku sa môže odoslať na kolektor viackrát. Jedná sa ale o tú istú komunikáciu a použitie viacerých záznamov o tom istom toku by viedlo ku skreslení štatistík [3].

3.4.1 Aplikácie

Exportované údaje je možné na kolektore analyzovať a získať z nich nejaké užitočné informácie. Na to sa používajú rôzne aplikácie, ktoré spracúvajú dáta. Výsledkom môže byť informácia o tom že ktorý protokol, resp. aplikácia sa najčastejšie používa, ktorú IP adresu navštevoval daný užívateľ najčastejšie, atď. Umožňujú aj identifikáciu staníc z najvyšším dátovým prenosom. Poskytovatelia internetu môžu použiť aplikácie na účtovanie, kde po vyčerpaní dátového limitu sa užívateľovi spomalí rýchlosť alebo sa odoberie prístup. Ďalej niektoré firmy poskytujú aj pokročilejšie aplikácie. Riešenie od firmy Flowmon Networks a. s. umožňuje napríklad detegovať anomálie siete, ako útoky typu DDoS.

Dnes používané nástroje poskytujú webové rozhranie pomocou ktorého administrátori sa dotazujú na informácie, sú dostupné rôzne štatistiky, správy, atď. Obrázok 3.4 zobrazuje webové rozhranie Flowmon Monitoring Centrum.

Obr. 3.4: Flowmon Monitoring Centrum zobrazujúci rôzne štatistiky [1].

3.5 Verzie NetFlow

Počas vývoja protokolu boli postupne vydané nové verzie, ktoré prinášali rôzne rozšírenia. Pôvodná verzia je 1, ktorá je už ale zastaraná a prakticky sa nepoužíva. Najčastejšie používaná verzia je 5, ktorá má pevný formát záznamov a je obmedzená na IPv4. Verzia 9 prináša podporu IPv6, nové atribúty a flexibilitu pomocou šablón. Posledná verzia je 10 nazývaná ako IPFIX, ktorá je založená na verzii 9 ale je štandardizovaná organizáciou IETF a prináša ďalšie atribúty a rozšírenia [22].

3.5.1 NetFlow v9

Verzia 9 je popísaná v RFC3954 [13], rozširuje staršie verzie o šablóny a o informácie vrstvy sieťového rozhrania. Administrátor si môže nadefinovať šablónu (*template record*), v ktorej určí aké dáta (t.j. ktoré atribúty) sa budú exportovať. V predošlých verziách protokolu ak bola potreba zmena atribútov tak bolo potrebné vydať novú verziu protokolu. Pridaním šablón už tento problém nenastáva. Ďalšou výhodou je, že firmy si môžu prispôsobiť množinu exportovaných dát na základe svojich potrieb. Je možné zvoliť konkrétne atribúty, ktoré je potrebné exportovať. Nemusia sa tak posielať všetky atribúty, čím sa zníži záťaž siete.

Pred odoslaním samotných záznamov z exportéru je najprv potreba poslať šablónu, aby kolektor rozumel daným informáciám. Jednotlivé šablóny (*Template Record*) obsahujú svoje jedinečné identifikačné číslo a počet atribútov, ďalej pre každý atribút jeho typ a dĺžku. Posielajú sa v *Template FlowSet*, ktorý má vždy identifikátor 0. Záznamy o tokoch sa posielajú v *Data FlowSet*, ktorý obsahuje identifikátor šablóny a záznamy o tokoch (*Flow Record*), ktoré obsahujú hodnoty pre jednotlivé atribúty konkrétnej šablóny [13]. Príklad šablóny a záznamu pre danú šablónu zobrazuje obrázok 3.5.

FlowSet ID = 0
Length = 28 bytes
Template ID = 256
Field Count = 5
IPV4_SRCADDR(0x0008)
Length = 4
IPV4_DSTADDR(0x000C)
Length = 4
IPV4_NEXT_HOP(0x000E)
Length = 4
PKTS_32(0x0002)
Length = 4
BYTES_32(0x0001)
Length = 4

šablóna

FlowSet ID = 256	Length = 64 bytes
192.168.1.12	
10.5.12.254	
192.168.1.1	
5009	
5344385	
⋮	

záznam

Obr. 3.5: Príklad šablóny a záznamu obsahujúceho konkrétne hodnoty [2].

3.5.2 IPFIX

Štandard IPFIX (*IP Flow Information Export*) bol vytvorený združením IETF ako otvorený protokol pre exportovanie informácií o sieťových tokoch. Prvá verzia bola zverejnená v roku 2008 publikovaním RFC5101 [14], ktorá bola aktualizovaná v roku 2013 s RFC7011 [16]. Označuje sa aj ako NetFlow v10, atribút hlavičky paketu udávajúci verziu protokolu obsahuje hodnotu 10 [14]. Vychádza z NetFlow v9 oproti ktorému prináša rôzne rozšírenia. Obsahuje viacero predefinovaných atribútov, ktoré môžu byť exportované, všetky podporované dátové typy a atribúty sú popísané v RFC5102 [37] a RFC7012 [15].

Pre zvýšenie zabezpečenia je možné použiť protokol TLS alebo DTLS na šifrovanie pri posielaní záznamov. IPFIX je nezávislý na transportnom protokole, ale každá implementácia musí podporovať aspoň SCTP protokol s rozšírením PR–SCTP, ďalej môže implementovať protokoly UDP aj TCP [14].

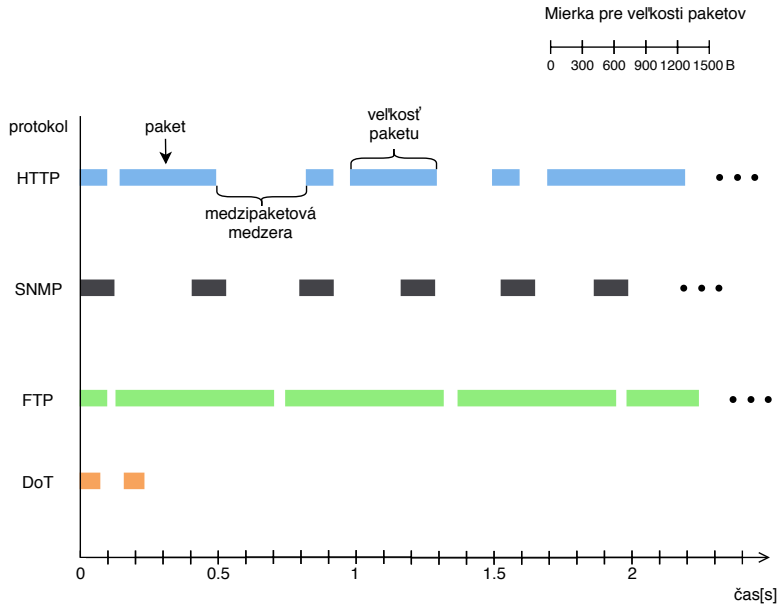
V predošlých verziách NetFlow oba smery tej istej komunikácie bolo možné reprezentovať iba pomocou dvoch záznamov, jeden pre prichádzajúcu, druhý pre odchádzajúcu komunikáciu. IPFIX poskytuje exportovanie údajov o oboch smeroch komunikácie pomocou jedného záznamu (tzv. *biflow*). V zázname je každý nekľúčový atribút obsadený dvakrát (pre oba smery). Kľúčové atribúty obsahuje záznam iba raz, ich hodnoty sú rovnaké pre oba smery iba je vymenená rola odosielateľ–prijímateľ [41].

Kapitola 4

Parametre sieťovej komunikácie vhodné pre klasifikáciu šifrovaných dát

Pre klasifikáciu šifrovanej prevádzky je vhodná štatistická klasifikácia, ktorá bude použitá v tejto práci. Výhodou tejto klasifikácie je že stačia informácie o vlastnosti tokov a paketov, samotný obsah aplikačných dát nie je potrebný. Na základe týchto vlastností sa potom tok zaradí do klasifikačnej kategórie. V rámci práce sa bude pracovať so šifrovanými tokmi, ktoré používajú protokol TLS a budú sa identifikovať použité aplikačné protokoly.

Obrázok 4.1 zobrazuje niekoľko tokov rozličných protokolov. Obdĺžniky reprezentujú jednotlivé pakety, šírka obdĺžnika (paketu) udáva jeho veľkosť podľa mierky zobrazenej vpravo hore, medzery predstavujú medzipaketové medzery v sekundách podľa osy x. Nižšie uvedené sekcie budú popisovať atribúty a tiež obsahovať tabuľky s hodnotami daných atribútov, ktoré vychádzajú z tohto obrázku.



Obr. 4.1: Príklad tokov rôznych protokolov, veľkostí ich paketov a medzipaketových medzier.

Práca sa bude zaoberať iba tokmi TCP, ktoré obsahujú TLS dáta. Prvých niekoľko paketov toku pri klasifikácií nie je vhodné použiť. Jedná sa o pakety pre naviazanie spojenia TCP (*3 way–handshake*) a o pakety pre naviazanie spojenia TLS (*TLS handshake*). Tieto sú rovnaké pre rôzne aplikačné protokoly a preto neposkytujú žiadnu pridanú hodnotu pri klasifikácií. V ďalších odstavcoch sa toky uvažujú bez týchto paketov, takže paket číslo 1 reprezentuje prvý paket po TCP a TLS *handshake* a medzera číslo 1 predstavuje medzipaketovú medzeru po tomto pakete, teda medzi prvým a druhým paketom.

Na klasifikáciu prevádzky sa budú používať dva typy údajov: na úrovni paketov (detailné) a na úrovni toku (agregované). Detailné sú údaje o konkrétnych paketoch danej komunikácie, ako veľkosti paketov a medzipaketové medzery. Tieto hodnoty nám poskytujú údaje o toku na úrovni paketov. Exportovanie údajov o každom pakete v toku je nereálne, generovalo by to veľké záznamy a bolo by to výkonovo náročné na spracovanie. Pre klasifikáciu sa použije iba prvých niekoľko paketov a medzipaketových medzier. Klasifikácia na základe paketov na začiatku komunikácie je založená na pozorovaní, že v počiatočnej fáze každého spojenia na transportnej vrstve štatistiky uvedených vlastností závisia na stavovom automate protokolu aplikačnej vrstvy [23]. Jednotlivé aplikačné protokoly posielajú rôzne, predefinované sekvencie správ, ktoré sú pre daný protokol jedinečné [42]. Napríklad je rozdiel v paketoch pri nadviazaní spojenia SMTP oproti POP3 alebo oproti HTTP dotazu.

Druhý typ údajov sú agregované štatistické údaje, ako napríklad dĺžka toku, priemer alebo štandardná odchýlka veľkostí paketov. Tieto informácie nám poskytujú súhrnné informácie o celom toku, napríklad vysoká priemerná veľkosť môže indikovať prenos súboru. Jednotlivé atribúty a ich typ zobrazuje tabuľka 4.4.

4.1 Veľkosť paketov

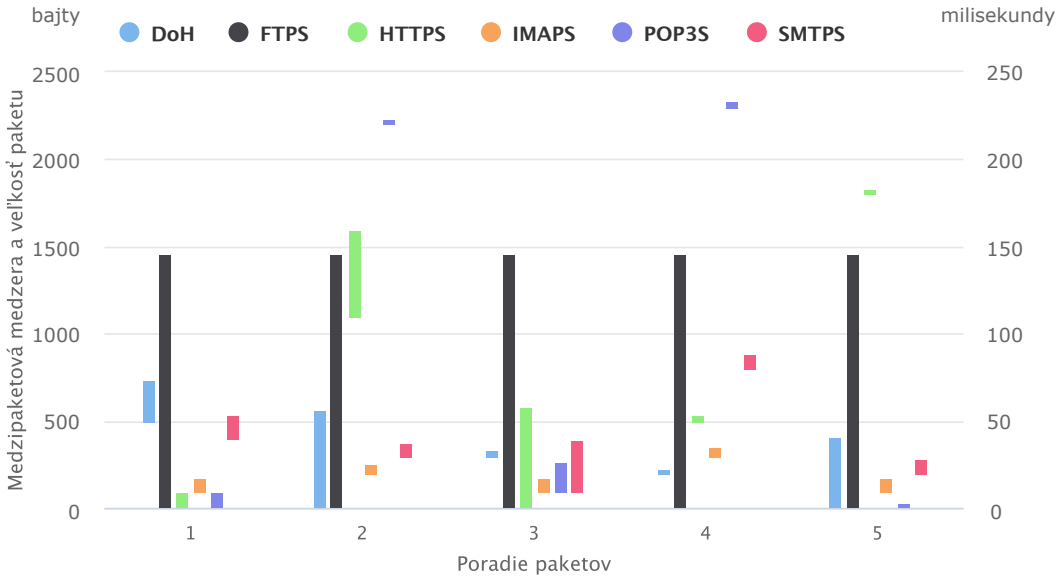
Veľkosť paketov závisí na sémantike protokolu, na množstve odoslaných aplikačných dát. Pod veľkosťou paketov v rámci práce sa bude rozumieť veľkosť aplikačných dát paketu v bajtoch. Môže nadobúdať hodnotu až 1460 bajtov pri protokolu Ethernet [10]. Počas prenosu sa veľkosť niektorých paketov môže zmeniť v prípade ak veľkosť IP datagramu presahuje hodnotu *Maximum Transmission Unit* (MTU) a paket sa fragmentuje na menšie časti. Hodnota MTU pri IPv4 môže nadobúdať hodnoty od 576 B az po 1500 B. Veľkosť paketov teda závisí aj od sieťovej infraštruktúry. Na závislosti od umiestnenia monitorovacieho zariadenia môže veľkosť nadobúdať rozličné hodnoty (pred/po fragmentácií).

Vlastnosť / Protokol	paket 1	paket 2	paket 3	paket 4	AVG[1]	STDEV[2]	MIN[3]	MAX[4]
HTTP	251	1143	240	945	564.9	252.3	112	1500
SNMP	320	312	300	309	308.7	10.4	290	345
FTP	235	1500	1500	1500	1441.8	50.6	210	1500
DNS over TLS	201	193	N/A[5]	N/A[5]	197	7.5	192	201

[1] Priemerná veľkosť paketov.
[2] Štandardná odchýlka veľkostí paketov.
[3] Minimálna veľkosť paketu.
[4] Maximálna veľkosť paketu.
[5] Žiadna hodnota, tok obsahuje iba dva pakety a jednu medzeru (dotaz a odpoveď).

Tabuľka 4.1: Príklad veľkostí paketov a vlastností odvodených od nich v bajtoch.

Väčšina paketov sa dá kategorizovať ako veľmi malý paket alebo veľmi veľký paket [12]. Pakety prenášajúce kontrolné informácie sú zvyčajne menšie ako 200 bajtov, na druhej strane pakety prenášajúce dáta majú vo väčšine prípadov veľkosť nad 200 bajtov [11]. Tabuľka 4.1 zobrazuje veľkosti prvých štyroch paketov tokov zobrazených na obrázku 4.1, ďalej obsahuje aj štatistické údaje o veľkostí paketov z celého toku. Obrázok 4.2 zobrazuje veľkosti prvých päť paketov rôznych aplikačných protokolov, kde výška stĺpca vyjadruje veľkosť v bajtoch na základe ľavej osy y.



Obr. 4.2: Veľkosti prvých päť paketov tokov rôznych protokolov a medzipaketové medzery.

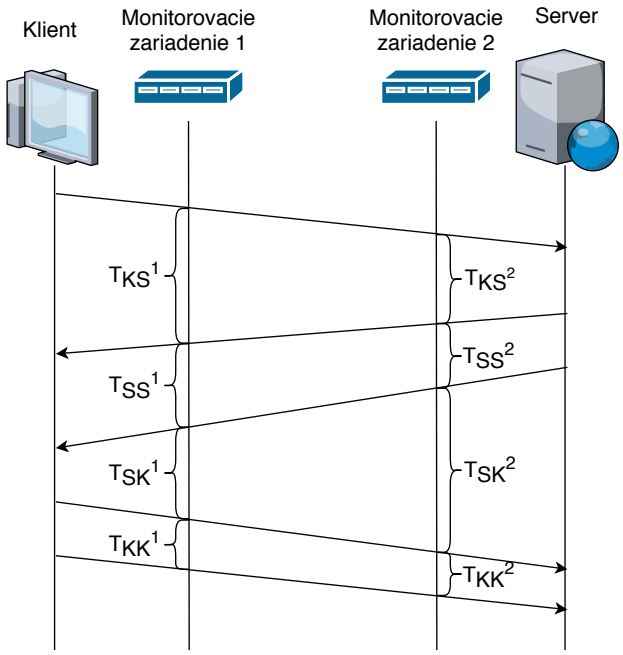
4.2 Medzipaketové medzery

Medzipaketové medzery sú časové intervaly medzi jednotlivými za sebou nasledujúcimi paketmi toku. Tento rozdiel medzi dvomi paketmi závisí od viacerých faktorov. Hodnota primárne závisí od použitej aplikácie, že ako často, v akom poradí a s akým oneskorením sú posielané pakety. Taktiež závisí od doby potrebnej pre aplikáciu na vygenerovanie dát a odoslanie ich na transportnú vrstvu [27]. Má na to vplyv aj infraštruktúra a prenosová rýchlosť siete, ďalej taktiež zahltenie siete. Medzipaketové medzery v obojsmernom toku zobrazuje obrázok 4.3. Medzery môžeme zaradiť do nasledujúcich tried [27]:

- T_{KK} – medzera medzi dvomi paketmi vygenerovanými klientom,
- T_{KS} – medzera medzi paketom odoslaným klientom a prijatým od servera,
- T_{SK} – medzera medzi paketom odoslaným serverom a prijatým od klienta,
- T_{SS} – medzera medzi dvomi paketmi vygenerovanými serverom.

Hodnoty medzipaketových medzier sú ovplyvnené aj umiestnením monitorovacieho zariadenia. Obrázok 4.3 zobrazuje prípad použitia dvoch zariadení, kedy jedno je bližšie ku

klientu (zariadenie 1) a druhé k serveru (zariadenie 2). Hodnoty na zariadení bližšiemu ku klientu sú znázornené indexom 1, hodnoty na zariadení bližšiemu k serveru s indexom 2. V tomto prípade sa výraznejšie líšia hodnoty T_{KS} a T_{SK}. Hodnota T_{KS} bude väčšia na monitorovacom bode 1, hodnota T_{KS} zas bude väčšia na zariadení 2. Hodnoty T_{KK} a T_{SS} budú podobné, môžu sa trochu líšiť v závislosti na zahltení siete.



Obr. 4.3: Komunikácia medzi klientom a serverom znázorňujúca medzipaketové medzery.

Príklad medzipaketových medzier vybraných protokolov na základe obrázku 4.1 zobrazuje tabuľka 4.2. Obsahuje aj štatistické údaje medzipaketových medzier celého toku. Medzera pred prvým paketom je nulová, tá nebude použitá. Prvá medzera v tomto kontexte predstavuje rozdiel medzi prvým a druhým paketom, atď. Ďalej obrázok 4.2 okrem veľkostí paketov zobrazuje aj prvých päť medzipaketových medzier pre rôzne aplikačné protokoly. Časovú medzeru vyjadruje vzdialenosť počiatku stĺpca od osy x v milisekundách na základe pravej osy y.

Vlastnosť / Protokol	medzera 1	medzera 2	medzera 3	medzera 4	AVG[1]	STDEV[2]	MIN[3]	MAX[4]
HTTP	51	340	67	212	119	91	12	2117
SNMP	243	231	239	220	245	35	205	674
FTP	27	42	52	41	43	11	27	310
DNS over TLS	72	N/A[5]	N/A[5]	N/A[5]	72	0	72	72

[1] Priemerná veľkosť medzipaketových medzier.
[2] Štandardná odchýlka medzipaketových medzier.
[3] Minimálna medzipaketová medzera.
[4] Maximálna medzipaketová medzera.
[5] Žiadna hodnota, tok obsahuje iba dva pakety a jednu medzeru (dotaz a odpoveď).

Tabuľka 4.2: Príklad veľkostí medzipaketových medzier a vlastností odvodených od nich v milisekundách.

4.3 Veľkosť toku

Veľkosť toku môžeme charakterizovať pomocou počtu paketov v toku alebo celkovej veľkosti toku v bajtoch. Počet paketov zahŕňa všetky pakety od začiatku prenosu aplikačných dát až po ukončenie spojenia, veľkosť toku v bajtoch vyjadruje súčet veľkostí týchto paketov. Krátke toky s malým počtom paketov sa nazývajú aj ako *mouse flow*, dlhé toky s vyšším počtom paketov ako *elephant flow* [32]. Toto rozdelenie sa taktiež používa aj na základe veľkosti toku v bajtoch [30]. Tabuľka 4.3 obsahuje rôzne reprezentácie veľkostí tokov a ich hodnoty.

Vlastnosť / Protokol	**COUNT(pakety)**[1]	**SUM(veľkosti)**[2]	**Dĺžka toku [s]**
HTTP	72	40673	14.74
SNMP	39	12040	5.03
FTP	64	92276	30.85
DNS over TLS	2	394	0.31

[1] Počet paketov.
[2] Súčet veľkostí paketov.

Tabuľka 4.3: Príklad veľkostí tokov ako veľkosť v bajtoch, počet paketov a trvanie v sekundách.

4.4 Vybrané atribúty sieťových tokov

V tejto kapitole boli popísané rôzne atribúty sieťových tokov. V rámci práce sa budú používať vlastnosti na úrovni paketov a taktiež na úrovni tokov. Konkrétne budú použité veľkosti aplikačných dát prvých niekoľkých paketov a medzipaketové medzery medzi nimi. Ďalej bude použitá veľkosť aplikačných dát toku a počet paketov v toku. Súhrn vlastností tokov popísaných v kapitole zobrazuje tabuľka 4.4.

Atribút	Typ
veľkosť paketu	úroveň paketov
medzipaketová medzera	úroveň paketov
minimálna veľkosť paketu/medzip. medz.	úroveň toku
maximálna veľkosť paketu/medzip. medz.	úroveň toku
štandardná odchýlka veľkostí paketov/medzip. medz.	úroveň toku
priemerná veľkosť paketov/medzip. medz.	úroveň toku
veľkosť toku	úroveň toku
počet paketov	úroveň toku
dĺžka toku	úroveň toku

Tabuľka 4.4: Jednotlivé vlastnosti a ich typy.

Kapitola 5

Klasifikačný algoritmus

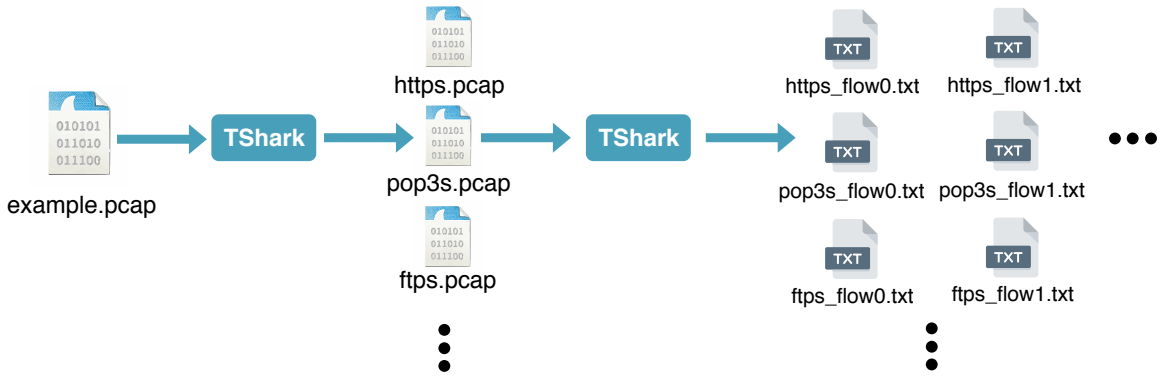
Jedným z cieľov tejto práce je vytvorenie algoritmu, ktorý bude použitý na klasifikáciu šifrovanej sieťovej prevádzky. Táto kapitola sa zaoberá návrhom klasifikačného algoritmu a nasledovne popisuje jeho implementáciu. Algoritmus bude použitý na kolektoru aj na exportéru. Na základe popísaných vlastností v kapitole 4 bude klasifikovať neznáme sieťové toky TLS do tried, ktoré predstavujú jednotlivé aplikačné protokoly. V rámci práce pre ukážku boli zvolené protokoly HTTPS, POP3S, IMAPS, SMTPS, FTPS, DNS over HTTPS, DNS over TLS. Zoznam podporovaných protokolov je možné rozšíriť o ďalšie.

5.1 Návrh

Algoritmus pre klasifikáciu šifrovaných tokov TLS je založený na modelu, ktorý je neskôr v tejto sekcií popísaný. K vytvoreniu modelu sú potrebné trénovacie dáta, čo sú vlastne vlastnosti tokov jednotlivých protokolov. K tomu je potrebné získať dostatočný počet tokov. Tieto toky boli pre túto prácu získané z online dostupných súborov PCAP, z ktorých sa vyfiltrujú potrebné protokoly na základe portov. Konkrétne odkazy obsahuje príloha C. Jedná sa o komunikácie z rôznych zdrojov v rámci rôznych sietí, čo má vplyv na jednotlivé vlastnosti tokov. Táto možnosť bola zvolená kvôli absencie možnosti zachytiť dáta pomocou sondy v rámci nejakej siete. Sieťový administrátor si môže zachytiť dáta v sieti, kde sa nástroj bude aplikovať a použije ich pre vytvoreniu modelu. To potom vedie k presnejšej klasifikácie v danej sieti.

Pre niektoré protokoly, o ktorých je málo dostupných súborov online, sú komunikácie vytvorené a zachytené ručne pomocou programu `Wireshark`. Ďalej je potrebné tieto komunikácie rozdeliť na jednotlivé toky. Na filtrovanie portov a tokov sa použije nástroj `TShark`[1]. Pri rozdelení na toky sa taktiež exportujú informácie o tokoch popísané v kapitole 4. Obrázok 5.1 ilustruje proces získavania týchto informácií.

[1] https://www.wireshark.org/docs/man-pages/tshark.html



Obr. 5.1: Extrakcia údajov TLS o tokoch zo súboru PCAP.

Vyexportované informácie sa použijú na výpočet potrebných štatistických informácií pre jednotlivé protokoly, ktoré budú použité v samotnom klasifikátore a budú tvoriť model. Pre vytvorenie modelu je potrebné vytvoriť tzv. profily jednotlivých protokolov, kde sa použijú vlastnosti tokov daného protokolu. Konkrétne veľkosť aplikačných dát paketov TLS a medzipaktové medzery po N-tý paket, kde N je maximálny počet paketov na základe ktorých bude klasifikátor pracovať. Presnosť medzipaketových medzier bude na úrovni milisekúnd (hodnoty od exportéru v nanosekundách sa zaokrúhlia). Taktiež sa použije počet paketov TLS a veľkosť TLS dát v toku. Výpočet pre jednotlivé vlastnosti prebieha nasledovne:

1. Hodnoty daného atribútu sa zoradia zostupne a použijú sa tri skupiny hodnôt. Prvá skupina reprezentuje pôvodné dáta, v druhej skupine sa zahodí prvých a posledných 10% hodnôt, čím zostaňe 80% pôvodných dát. V tretej skupine sa zahodí prvých a posledných 20% hodnôt, čím zostaňe 60% pôvodných dát. Odstránenie krajných hodnôt bolo zvolené kvôli vyfiltrovaniu prípadného šumu a výkyvov v komunikácií.

2. Pre jednotlivé skupiny hodnôt sa vypočíta minimum, maximum a štandardná odchýlka. Odchýlka sa od minima odpočíta a k maximu sa pripočíta. Získajú sa tri rôzne intervaly. Prípadné záporné hodnoty sa zaokrúhlia na nulu, čím sa nezníži presnosť keďže vlastnosti toky budú mať hodnoty väčšie ako nula. Intervaly vytvorené pomocou minimálnych a maximálnych hodnôt nezahŕňajú všetky možné hodnoty pre dané protokoly. Vlastnosti neznámeho toku, ktorý obsahuje niektorý z podporovaných protokolov, sa môžu podobať k profilu tohto protokolu ale aj tak padnúť mimo intervalov. Kvôli týmto prípadom boli intervaly rozšírené o štandardnú odchýlku.

```
   #paket 1           #paket 2       ...
[[[100%, 80%, 60%], [100%, 80%, 60%], ...  #minimálna veľkosť paketu
 [[100%, 80%, 60%], [100%, 80%, 60%], ...  #maximálna veľkosť paketu
   #medzera1          #medzera2      ...
 [[100%, 80%, 60%], [100%, 80%, 60%], ...  #minimálna medzipaketová medzera
 [[100%, 80%, 60%], [100%, 80%, 60%], ...  #maximálna medzipaketová medzera
 [100%, 80%, 60%],                         #minimálny počet paketov
 [100%, 80%, 60%],                         #maximálny počet paketov
 [100%, 80%, 60%],                         #minimálna veľkosť toku
 [100%, 80%, 60%]]                         #maximálna veľkosť toku
```

Výpis 5.1: Všeobecný formát pre uloženie profilu protokolu.

Vo výsledku sa získa množina intervalov, ktoré tvoria profil protokolu. Formát profilu konkrétneho protokolu zobrazuje výpis 5.1. Podobným spôsobom sa vytvoria profily ďalších protokolov, ktoré spoločne tvoria model. Tabuľka 5.1 zobrazuje príklad údajov pre veľkosti paketov protokolu POP3S v bajtoch, tabuľka 5.2 údaje pre medzipaketové medzery (hodnoty sú uvedené v milisekundách).

Údaj / Poradie	Min 100%	Max 100%	σ 100%	Min 80%	Max 80%	σ 80%	Min 60%	Max 60%	σ 60%
1.	39	1418	343.54	39	101	15.67	94	96	0.66
2.	34	1418	355.93	35	41	2.23	35	37	0.66
3.	36	1418	334.13	45	164	43.98	149	164	4.96
4.	36	1418	354.46	36	69	8.2	41	41	0.0
5.	33	1418	356.13	33	53	5.48	33	37	1.39

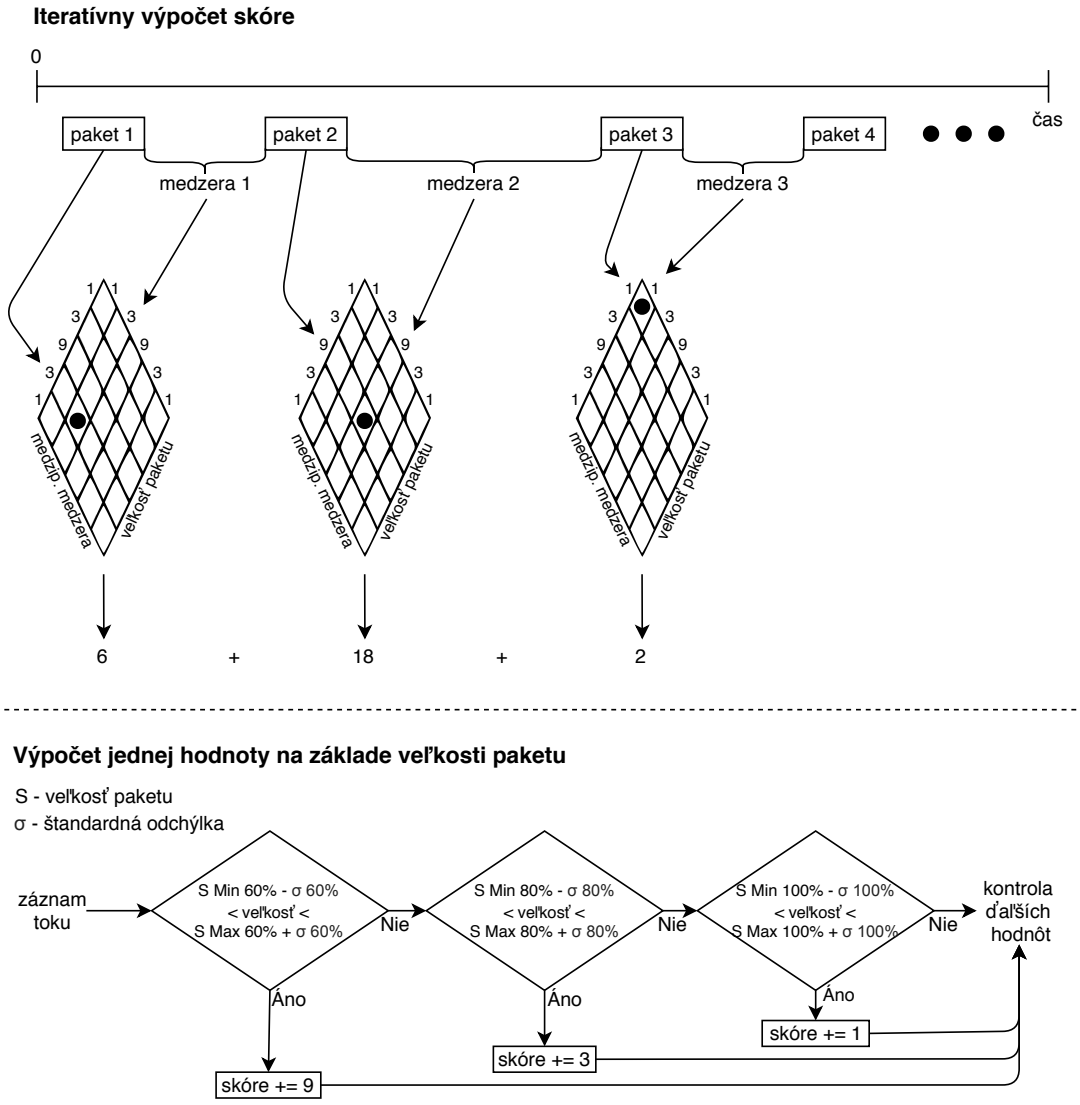
Tabuľka 5.1: Štatistické hodnoty pre veľkosti paketov protokolu POP3S.

Vypočítané údaje sa použijú pri klasifikácií. Klasifikácia bude možná na dvoch miestach a to na kolektore pomocou exportovaných hodnôt, alebo rovno na exportéru. Model bude kvôli tomu umiestnený aj na exportéru, algoritmus klasifikácie bude rovnaký v oboch prípadoch. Výhodou umiestnenia na exportéru môže byť rýchlejšia klasifikácia v prípade výkonnej sondy. Pri klasifikácií na kolektore sa zas záťaž sondy znižuje čo môže byť výhodné pri menej výkonnej sonde. Výhodou klasifikácie na kolektore je tiež jednoduchšia aktualizácia modelu, v prípade exportéru je aktualizácia obťažnejšia.

Údaj / Poradie	Min 100%	Max 100%	σ 100%	Min 80%	Max 80%	σ 80%	Min 60%	Max 60%	σ 60%
1.	0	5334	1373.42	0	5	1.48	0	2	0.78
2.	0	42	9.85	16	42	6.26	22	22	0.0
3.	1	12	2.76	1	3	0.64	1	2	0.5
4.	0	647	160.31	22	63	13.13	22	39	5.54

Tabuľka 5.2: Štatistické hodnoty pre medzipaketové medzery protokolu POP3S.

Počet použitých paketov závisí na nastavení exportéru, kde sa určí koľko údajov sa má exportovať. Postupne sa budú prechádzať hodnoty, prvý paket, prvá medzipaketová medzera, potom druhý paket, atď. Tieto údaje sa potom porovnávajú s profilmi jednotlivých protokolov. Ak veľkosť paketu padne do intervalu (Min 60% $-\sigma$ 60%, Max 60% $+\sigma$ 60%) priradí sa skóre 9, ak padne do intervalu (Min 80% $-\sigma$ 80%, Max 80% $+\sigma$ 80%) sa priradí skóre 3, ak do intervalu (Min 100% $-\sigma$ 100%, Max 100% $+\sigma$ 100%) tak skóre 1, v prípade že hodnota nepadne ani do jedného z intervalov tak sa priradí skóre 0.



Obr. 5.2: Postupný výpočet skóre pre jednotlivé úrovne a výpočet pre konkrétny atribút.

Rovnaké vyhodnotenie sa vykoná pre medzipaketové medzery a jednotlivé skóre sa sčítajú. Toto vyhodnotenie prebieha pre každý paket až kým sa nespracuje každá exportovaná hodnota a skóre na jednotlivých úrovniach sa sčítajú. Postupné vyhodnocovanie zobrazuje horná časť obrázku 5.2. Dolná časť zobrazuje výpočet konkrétneho skóre na základe veľkosti paketu podľa vyššie popísaného postupu.

Po vyhodnotení hodnôt na úrovni paketov sa ešte získa skóre na základe veľkosti toku a počtu paketov v toku, kde podľa vyššie uvedených intervalov sa priradia skóre 10, 50, 100, alebo 0. Tieto atribúty predstavujú väčšiu váhu, keďže sa jedná o vlastnosti na úrovni celého toku. Všetky skóre sa sčítajú a tým sa získa celkové skóre toku pre daný protokol. Skóre sa vypočíta pre každý protokol a výsledkom bude protokol, pri ktorom tok získa najvyššie skóre. Skóre musí dosiahnuť aspoň 50% celkového možného skóre. Pri nižšom skóre je málo pravdepodobné, že tok obsahuje daný aplikačný protokol. Vzhľadom k tomu, že nie je možné vytvoriť model pre každý protokol, tak vstupom môže byť aj neznámy protokol. Preto v prípade ak ani pri jednom protokolu sa táto hodnota nedosiahne klasifikuje sa tok ako neznámy.

5.2 Implementácia

Klasifikátor bol implementovaný v jazyku Python vo verzii 3.6 pre kolektor a v jazyku C pre exportér. Kód klasifikátoru obsahuje už vypočítané trénovacie dáta, tj. intervaly hodnôt popísané v predošlej sekcií pre jednotlivé vlastnosti toku. Tieto údaje pre jednotlivé protokoly sú reprezentované zoznamom. Profily jednotlivých protokolov boli získané metódou popísanou v návrhu použitím skriptov. Formát profilu v jazyku Python pre konkrétny protokol zobrazuje ukážka 5.2. Profily jednotlivých protokolov obsahuje príloha B.

```
    #paket1         paket2 ...
  #100% 80% 60%  100% 80% 60%
[[[0, 0, 64.39], [0, 0, 51.3], ...                          #min. velkost paketu
 [[1566.6, 1294.8, 959.6], [2397.5, 1566.6, 1487.7], ... #max. velkost paketu
  #medzera1  medzera 2  ...
 [[0, 0, 0], [0, 0, 0], ...                                 #min. medzip. medzera
 [[633.6, 107.8, 81.4], [34448.7, 13699.4, 94.6], ...       #max. medzip. medzera
 [[0, 0, 0]],                                               #min. pocet paketov
 [[7331.6, 209.7, 66.5]],                                   #max. pocet paketov
 [[0, 0, 2612.5]],                                          #min. velkost toku
 [[9322375.0, 179869.7, 94862.5]]]                          #max. velkost toku
```

Výpis 5.2: Príklad časti profilu protokolu HTTPS.

Všetky modely v rámci tejto práce obsahujú intervaly pre prvých 10 paketov a 9 medzipaketových medzier. V prípade potreby je možné profily protokolov rozšíriť o ďalšie intervaly. Pre rozšírenie modelu o profil ďalšieho protokolu je potrebný tento profil pridať vo formáte uvedenom vo výpise 5.1. Na vygenerovanie profilu požadovanom formáte pre kolektor je dostupný skript `generate_data_collector`, na generovanie profilu pre exportér slúži skript `generate_data_exporter`. Presný popis použitia obsahuje súbor `README`. Tento profil je už iba potrebné vložiť do kódu.

Klasifikácia sa vykoná pomocou metódy popísanej v návrhu. V prípade ak sa nepoužijú údaje na úrovni toku tak tieto vlastnosti sa vynechajú pri klasifikácií. Počet použitých paketov a medzipaketových medzier závisí na nastavení exportéru. V prípade ak model niektorého protokolu obsahuje dáta pre menej paketov ako počet vstupných paketov tak sa použije iba dostupný počet paketov pri klasifikácií. Pri jednotlivých hodnotách sa kontroluje do ktorého intervalu hodnota padla a podľa toho sa priradí príslušné skóre. Výpočet skóre podľa veľkosti aplikačných dát paketu TLS v jazyku Python zobrazuje ukážka 5.3. Rovnaký výpočet sa vykoná v prípade klasifikácie na exportéru, len v jazyku C. Výpočet prebieha analogicky pre ostatné vlastnosti. Výstupom je ten protokol, pri ktorom sa získa najvyššie skóre a dosiahne sa minimálny počet bodov (50% z maximálnej hodnoty). Taktiež sa vypočíta koľko percent predstavuje výsledné skóre oproti maximálnemu skóre. Maximálne skóre sa mení na závislosti počtu použitých atribútov a zo samotného skóre by bolo ťažšie identifikovať dané skóre je dobré alebo zlé. Kvôli tomu boli zvolené percentá, ktoré univerzálne reprezentujú výsledok nezávisle na počte použitých atribútov. Ak viacero protokolov dosiahne rovnaké skóre, tak sa všetky vypíšu oddelené medzerou.

```
def get_score(self, sizes, ipts, packetnum, flowsize, level):
    self.score = 0
    for i in range(level):
        if (sizes[i] > self.min_size[i][2]) and (sizes[i] < self.max_size[i][2]):
            self.score += 9
        elif (sizes[i] > self.min_size[i][1]) and (sizes[i] < self.max_size[i][1]):
            self.score += 3
        elif (sizes[i] > self.min_size[i][0]) and (sizes[i] < self.max_size[i][0]):
            self.score += 1
    ...
```

Výpis 5.3: Výpočet skóre pre veľkosť paketu.

Kolektor

Po spustení klasifikátoru na kolektore sa z profilov protokolov pre každý protokol vytvorí objekt, ktorý obsahuje hodnoty jednotlivých údajov a metódu pre výpočet skóre. Táto metóda ako vstup dostane hodnoty atribútov neznámeho toku a vypočíta skóre protokolu. Po vytvorení objektov klasifikátor ako prvé načíta vstupný súbor CSV vytvorený exportérom, ktorý obsahuje záznamy tokov. Názov vstupného súboru sa zadáva parametrom `-i`. Záznamy sa postupne prechádzajú a spracujú. Spracovanie závisí od toho, ako je nastavený exportér, tj. či sa exportujú aj údaje na úrovni toku alebo či sa klasifikuje na exportéri. V rámci NetFlow atribúty, ktoré neboli nastavené na exportéri obsahujú hodnotu NIL. Pomocou tejto hodnoty je možné zistiť ako je nastavený exportér. V prípade ak každý exportovaný atribút obsahuje hodnotu NIL tak sa nejedná o TLS tok a nevykoná sa žiadna akcia. V prípade ak sú naplnené atribúty pre protokoly a percentá a ostatné atribúty obsahujú NIL, tak klasifikácia sa vykonala na exportéru a výsledok sa iba interpretuje.

Ak atribúty pre protokoly a percentá obsahujú NIL, tak klasifikácia sa má vykonať na kolektoru. V tomto prípade sa skontrolujú dve veci. Ako prvé, či boli exportované údaje na úrovni toku. Ako druhé ešte ta skontroluje, že údaje o koľkých paketoch boli exportované. Ďalej sa vypočítajú skóre a získa sa protokol pomocou vyššie popísanej metódy. Použitím parametru `-n` sa dá nastaviť koľko výsledkov sa má zobraziť (od 1 do 3). Pri viacerých výsledkoch sa jednotlivé protokoly a percentá vypíšu zostupne. Ak parameter nie je zadaný, tak sa výstupom je iba najlepší výsledok.

Formát výstupu je konfigurovateľný pomocou parametru `-t`, ktorý môže nadobúdať hodnoty PLAIN, CSV alebo JSON. V prípade PLAIN sa na štandardný výstup vypíše zdrojová a cieľová IP adresa toku, protokol a získané percento. Pri CSV a JSON sa tieto údaje zapíšu do výstupného súboru v danom formáte. Názov výstupného súboru sa zadáva pomocou parametru `-o`.

Exportér

Na exportéru sa požadované atribúty tokov ukladajú interne pri spracovaní paketov. Klasifikácia sa vykoná pred exportovaním záznamu rovnakým spôsobom ako na kolektore. Rozdielom je formát uloženia modelu a výstup. Keďže sa jedná o jazyk C profily protokolov sú uložené ako viacrozmerné pole a nie ako zoznam. O jednotlivých vlastnostiach sa ukladá rozdielne množstvo trojíc intervalov (pre 100, 80 a 60 percent pôvodných dát). Napríklad pre veľkosti prvých desiatich paketov je potrebných desať takýchto trojíc, pre veľkosť TLS dát v toku stačí jedna. V jazyku C musí mať viacrozmerné pole každú časť rovnako veľkú, kvôli tomu nevyužité indexy boli naplnené nulovými hodnotami. Tieto hodnoty sa nebudú používať, ale kvôli obmedzeniam jazyka je nutné ich použiť. Po klasifikácií sa do exportovaného záznamu sa uložia tri protokoly sa najvyšším skóre a ich percentá.

Kapitola 6

Rozšírenie záznamov NetFlow

K tomu aby sa šifrované toky dali klasifikovať pomocou navrhnutého algoritmu je potrebné rozšíriť záznamy NetFlow o nové atribúty šifrovanej prevádzky. Tieto atribúty bude možné potom využiť aj v ďalších aplikáciách, ktoré pracujú s tokmi TLS. Existuje viacero riešení NetFlow exportérov a kolektorov od rôznych výrobcov. Táto kapitola sa zaoberá návrhom rozšírenia záznamov a návrhom modulu pre exportér od brnenskej firmy Flowmon Networks a. s.. Ďalej nasleduje popis samotnej implementácie modulu, ktorý zaisťuje exportovanie potrebných údajov a podporuje aj klasifikáciu.

6.1 Návrh

Návrh pozostáva z dvoch častí. Ako prvé je potrebné navrhnúť rozšírenie záznamov NetFlow. Ďalej je potrebné vytvoriť návrh pre modul, ktorý bude na exportéri získavať hodnoty potrebných atribútov TLS tokov a exportovať ich. S tokmi na exportéru sa pracuje po paketoch, ktoré sú reprezentované ako pole bajtov. Bude teda potrebné určiť ktoré bajty treba kontrolovať a aké hodnoty ukladať aby sme získali požadované údaje.

6.1.1 Záznamy NetFlow

V prvom rade je pre klasifikáciu potrebné exportovať údaje o TLS toku uvedené v kapitole 4, konkrétne veľkosti aplikačných dát prvých N paketov TLS s obsahom *Application Data*, prvých N-1 medzipaketových medzier, počet paketov TLS v toku a celkovú veľkosť TLS správ v bajtoch. K tomu treba rozšíriť záznamy NetFlow o tieto atribúty, čo sa dosiahne vytvorením novej šablóny. Potom sa atribúty môžu ukladať do NetFlow cache a následne exportovať.

NetFlow podporuje ukladanie počtu paketov a veľkosť toku v bajtoch. Pre klasifikáciu budú ale použité iba aplikačné dáta TLS, konkrétne iba správy typu *Application Data*. K tomu je potrebné pridať nové atribúty pre exportovanie veľkostí týchto dát a počtu týchto paketov. Exportovanie týchto údajov môže byť vypnuté.

Veľkosti paketov a medzipaketové medzery bude potrebné pridať do záznamov pre klasifikáciu. Počet týchto atribútov bude obmedzený, vychádzame z predpokladu že na identifikáciu protokolu stačí niekoľko prvých paketov toku (po naviazaní spojenia TLS). Taktiež exportovanie týchto údajov o každom pakete je nereálne, generovalo by to veľké záznamy a vyžadovalo by to výkon na exportéri. Počet exportovaných hodnôt bude konfigurovateľný na exportéru. Maximálny počet exportovaných hodnôt závisí na počte pridaných atribútov pre veľkosti paketov a medzipaketové medzery do záznamov NetFlow. To, že koľko týchto

údajov je vhodné použiť pre klasifikáciu sa vyhodnotí neskôr pri testovaní. Nové vlastnosti budú reprezentované ako samostatné atribúty s unikátnym názvom.

Okrem exportovaní údajov o TLS tokoch bude možné vykonať klasifikáciu rovno na exportéru a exportovať tri protokoly s najvyšším dosiahnutým skóre. Pre túto možnosť boli pridané atribúty pre názvy protokolov a ich získané skóre v percentách oproti maximálnemu skóre.

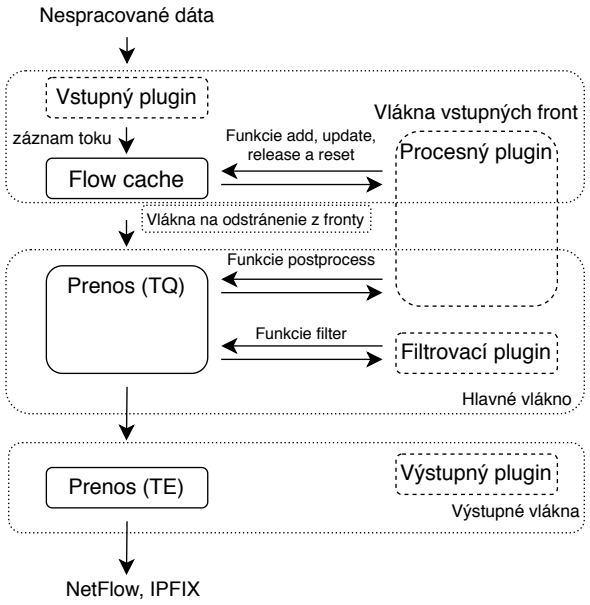
V rámci `flowmonexp5` sa pre každý modul používa maximálne jedna šablóna, kvôli tomu sa vo všetkých prípadoch použije tá istá šablóna. To znamená, že bez ohľadu na nastavenie a dáta sa vždy exportujú všetky atribúty. V závislosti na nastavení exportéru sa ale niektoré nemusia naplniť. Príklad rozšírených záznamov zobrazuje obrázok 6.1.

Veľkosť dát TLS	34741	NIL	NIL
Počet paketov TLS	65	NIL	NIL
Veľkosť paketu 1	1451	NIL	265
...	...	...	...
Veľkosť paketu N	162	NIL	15
Medzipaketová medzera 1	50531	NIL	71623
...	...	...	...
Medzipaketová medzera N-1	61942	NIL	236105
Protokol 1	NIL	HTTPS	NIL
Percento 1	NIL	91	NIL
Protokol 2	NIL	SMTPS	NIL
Percento 2	NIL	76	NIL
Protokol 3	NIL	FTPS	NIL
Percento 3	NIL	62	NIL

Tabuľka 6.1: NetFlow záznamy rozšírené o potrebné atribúty.

6.1.2 Plugin pre exportér

Na analýzu tokov a exportovanie záznamov bude použitý exportér Flowmon. Jedná sa o softvér `flowmonexp5` s modulárnou architektúrou, ktorý vykonáva analýzu a vytvára záznamy. Na monitorovanie prevádzky sa môže použiť hardvérové zariadenie (Flowmon sonda), alebo môže sa použiť aj softvérové zariadenie (virtuálny stroj). Cieľom exportéru je transformácia sieťovej prevádzky na záznamy o tokoch. Na základe informácií z hlavičiek vrstvy sieťového rozhrania až transportnej vrstvy sa vytvoria záznamy, ktoré môžu byť rozšírené o dodatočné informácie [4]. Jednotlivé moduly exportéru sa nazývajú ako *plugin*. Architektúru `flowmonexp5` zobrazuje obrázok 6.1.



Obr. 6.1: Moduly exportéru `flowmonexp5` a ich komunikácia.

Jednotlivé pluginy sú rozdelené do nasledujúcich kategórií:

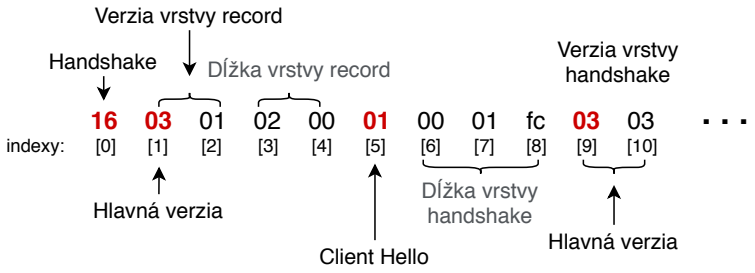
- **vstupný** – poskytuje dáta pre exportér vo forme paketov, vždy musí byť použitý práve jeden plugin tohto typu;
- **procesný** – spracuje jednotlivé pakety, vytvára a aktualizuje záznamy o tokoch, umožňuje rozšírenie exportovaných údajov;
- **filtrovací** – rozhoduje o tom, ktoré záznamy sa exportujú a ktoré nie;
- **výstupný** – exportuje záznamy vo vybranom formáte (NetFlow, IPFIX, csv, atď.), musí byť použitý aspoň jeden.

Táto práca sa zaoberá implementáciou procesného pluginu pre spracovanie TLS prevádzky a exportovanie údajov popísaných v kapitole 4. Každý plugin musí obsahovať volanie makra `FLOWMON_MODULE_INIT`, pomocou ktorého sa plugin registruje a ešte musí obsahovať naplnenú štruktúru typu `flowmon_plugin_t`. Pluginy taktiež implementujú funkcie (tzv. *hooks*), ktoré sa volajú pri špecifických udalostiach. Nasledujúci zoznam popisuje jednotlivé funkcie:

- Povinné
  - **init** – alokuje zdroje, spracuje parametre, registruje atribúty a voliteľné funkcie (postprocess, split, callback);
  - **init queue** – inicializuje dáta špecifické pre frontu a registruje voliteľné funkcie pre frontu (add, update, filter, release);
  - **shutdown** – volá sa pri ukončení pluginu, uvoľní zdroje pluginu;
  - **shutdown queue** – tiež sa volá pri ukončení pluginu, uvoľní zdroje špecifické pre frontu;

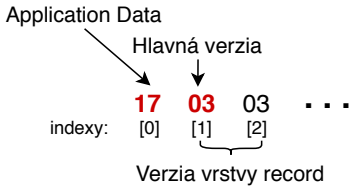
- Voliteľné
  - **filter** – volá sa pri každom spracovanom pakete, môže zahadzovať nepotrebné pakety;
  - **update** – volá sa pri prijatí paketu existujúceho toku, tu prebieha samotné spracovanie a parsovanie; môže byť pre tok zrušený pomocou `flowmon_hook_update_remove()`;
  - **add** – volá sa pri prvom príchode paketu neznámeho toku, špeciálny prípad funkcie update;
  - **release** – volá sa pri uvoľnení záznamu z cache, prebieha hlboká kópia dát tokov;
  - **postprocess** – volá sa pred exportovaním, pripraví dáta na exportovanie;
  - **split** – rozdelí *biflow* na dva toky;
  - **callback** – používa sa na uvoľnenie dynamicky alokovanej pamäte;

Pre klasifikáciu sa budú ukladať informácie o paketoch, ktoré obsahujú dáta TLS. K tomu je potrebné identifikovať či obsahuje spracovaný tok protokol TLS. V prvom rade sa skontroluje či sa jedná o TCP tok. Bude sa kontrolovať, či prvý paket toku obsahuje príznak SYN, ktorý indikuje začiatok naviazania spojenia TCP. Ďalej je potrebné identifikovať či sa jedná o tok TLS. Spojenie TLS začína správou typu *Client Hello*, v ktorej klient oznamuje serveru potrebné parametre k zahájeniu spojenia. Bude sa teda kontrolovať prvý paket s obsahom dát na aplikačnej vrstve, kde sa budú porovnávať vybrané bajty s hodnotami, ktoré identifikujú správu *Client Hello*. Jedná sa o bajty ktoré indikujú, že sa jedná o správu typu *Handshake*, kontkrétne *Client Hello*. Ďalej sa ešte skontroluje hlavná verzia TLS. Jednotlivé bajty na začiatku správy *Client Hello* zobrazuje obrázok 6.2, kontrolovať sa budú zvýraznené hodnoty. V prípade zhody sa daný tok bude naďalej spracovať pluginom, inak sa spracovanie ruší.



Obr. 6.2: Správa typu *Client Hello*.

Nasledovne sa budú spracovať iba pakety obsahujúce správu typu *Application Data*, ktoré obsahujú šifrované aplikačné dáta zapuzdreného protokolu, pakety bez aplikačných dát sa nespracujú ďalej. O prvých N paketoch sa uloží veľkosť aplikačných dát a medzipaketové medzery medzi nimi v nanosekundách. Ďalej pri každom TLS pakete sa inkrementuje počítadlo paketov TLS a pričíta sa veľkosť dát k celkovej veľkosti TLS dát. Pre identifikáciu *Application Data* sa použijú bajty zvýraznené na obrázku 6.3.



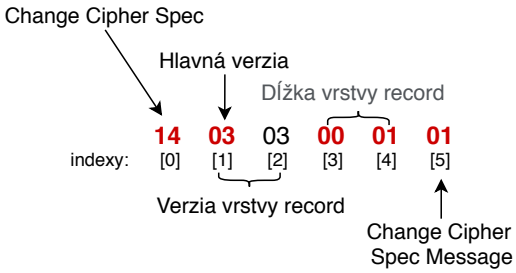
Obr. 6.3: Správa typu *Application Data*.

Pri TLS 1.3 správy pre nadviazanie spojenia (okrem *Client Hello* a *Server Hello*) sú interpretované ako *Application Data*. Pri nižších verziách po dohodnutí parametrov sa posiela správa typu *Change Cipher Spec*, ktorá indikuje že ďalšie správy budú šifrované dohodnutým algoritmom. Podľa RFC8446 [38] sa vo verzií 1.3 tento typ správy nemá vyskytovať, ale kvôli zachovaniu kompatibility sa môže použiť. Môže sa vyskytnúť situácia, kedy sa medzi dvomi správami *Change Cipher Spec* vyskytne paket s obsahom *Application Data*, ktorú zobrazuje obrázok 6.4.

```
TLSv1.3   571 Client Hello
TCP        54 443 → 48172 [ACK] Seq=1 Ack=518 Win=30720 Len=0
TLSv1.3  1514 Server Hello, Change Cipher Spec
TCP        54 48172 → 443 [ACK] Seq=518 Ack=1461 Win=32128 Len=0
TLSv1.3  1344 Application Data
TCP        54 48172 → 443 [ACK] Seq=518 Ack=2751 Win=35072 Len=0
TLSv1.3   134 Change Cipher Spec  Application Data
TLSv1.3   285 Application Data, Application Data, Application Data, Application Data
```

Obr. 6.4: Ukážka výskytu *Application Data* v rámci nadviazania spojenia TLS v programe Wireshark.

Paket s obsahom *Application Data* ale v tomto prípade nepotrebujeme, keďže sa zjavne jedná o paket pre naviazanie spojenia. Je teda potrebné kontrolovať, či sa nevyskytne po správach typu *Application Data* správa *Change Cipher Spec* a v prípade výskytu ignorovať doterajšie pakety s *Application Data*. Správa *Change Cipher Spec* je zobrazená na obrázku 6.5, kde sú zvýraznené použité bajty pre detekciu.



Obr. 6.5: Správa typu *Change Cipher Spec*.

Pri TLS 1.3 sa môže vyskytnúť situácia, keď sa aplikačné dáta posielajú už v rámci nadviazania spojenia TLS. Jedná sa o tzv. 0-RTT spojenia, kde sa klient už skorej pripájal na server. V tomto prípade nemusí prebehnúť výmena kľúčov, klient aj server majú lokálne uložený predzdieľaný kľúč spojenia (*pre-shared key*). To sa ale nedá identifikovať, či sa jedná o dáta aplikácií alebo ide o pakety pre nadviazanie spojenia, takže sa ignorujú. Budú sa spracovávať iba pakety obsahujúce správy *Application Data* po nadviazaní TLS spojenia. Ak

trénovacie dáta obsahujú málo tokov so spojením 0-RTT, tak sa neznámy tok so spojením 0-RTT nemusí správne klasifikovať, keďže aplikačné dáta sa na začiatku ignorujú a klasifikácia začína až s nasledujúcimi dátami. Napríklad prvý paket s obsahom *Application Data* môže už obsahovať tretiu správu zapuzdreného aplikačného protokolu, ale pri klasifikácií sa bude považovať za prvú správu.

Môže sa ešte stať, že prvé správy *Application Data* sú súčasťou nadviazania spojenia, ale to nedokážeme rozlíšiť. Týmto spôsobom sa spracujú aj trénovacie toky aj toky pri klasifikácií. V prípade dostatočného počtu tokov so spojením 0-RTT v trénovacích dátach by to nemalo mať vplyv na výsledok.

Exportér okrem exportovaniu všetkých údajov bude podporovať aj možnosť exportovanie iba údajov na úrovne paketov. V tomto prípade nie je potrebné počkať na koniec toku, stačí spracovať prvých N paketov a záznam sa môže exportovať. Výhodou je, že tok sa môže klasifikovať už na začiatku komunikácie. Nevýhodou zas môže byť prípadná nižšia presnosť kvôli absencie detailov na úrovne toku, to sa vyhodnotí až v testovaní.

Ďalšou možnosťou bude samotná klasifikácia na exportéru. Kvôli tomu bude treba pridať model aj na exportér. V tomto prípade sa naplnia atribúty týkajúce sa protokolov a ich percentá, ostatné sa nenaplnia.

6.2 Implementácia

Po návrhu rozšírenia záznamov nasledovala implementácia modulu pre exportér, ktorý rozširuje záznamy NetFlow o potrebné atribúty. Rozšírenie záznamov NetFlow o potrebné informácie bolo implementované pridaním modulu do exportéru `flowmonexp5`. Implementácia exportéru používa jazyk C, tento jazyk bol použitý aj pre implementáciu nového modulu s názvom `plugin_proces_eta.c`. Názov určuje že sa jedná o procesný plugin pre eta (*encrypted traffic analysis*), tj. pre analýzu šifrovanej prevádzky. Modul bude podporovať aj klasifikáciu na exportéru a taktiež bude možnosť zvoliť či sa má pracovať aj s údajmi na úrovni toku alebo bez nich.

Pri vývoji bol použitý virtuálny obraz *Flowmon Collector Virtual Appliance* dostupný po registrácií na stránke Flowmon Portal[1]. Tento obraz obsahuje, okrem iného, exportér `flowmonexp5`. Ďalej bolo potrebné vytvoriť konfiguračný súbor pre nastavenie a spustenie exportéru s novým modulom. Tu sa nastavujú rôzne parametre, ako formát vstupu (prevádzka v reálnom čase, PCAP, atď.) či výstupu (IPFIX, CSV, atď.). Pre preklad modulu bol použitý `Makefile` dostupný v dokumentácií. V rámci práce sa použije súbor CSV, ktorý potom slúži ako vstup pre klasifikátor.

V prvom rade boli implementované povinné časti pluginu. Ako prvé je potrebné zavolať makro `FLOWMON_MODULE_INIT`, čím sa plugin registruje (bude spustený pri spustení exportéru). Ďalej treba inicializovať štruktúru typu `flowmon_plugin_t`, ktorá obsahuje informácie pluginu, ako typ pluginu, názov pluginu, názov inicializačnej funkcie, atď.

```
typedef struct {
  size_t data_offset;
  int plugin_id;
  int queue_id;
  int optional_param;   // voliteľný parameter pluginu
  //int mandatory_param;  // povinný parameter pluginu
  // veľkosti prvých N TLS aplikačných dát po nadviazaní spojenia TLS
  flow_record_getter_t *g_sizes[MAX_PACKETS_STORED];
```

[1] https://portal.flowmon.com/s/login/

```
  // prvých N−1 medzipaketových medzier po nadviazaní spojenia TLS
  flow_record_getter_t *g_ipts[MAX_PACKETS_STORED−1];
  // súčet paketov TLS a ich veľkostí
  flow_record_getter_t *g_tls_packets;
  flow_record_getter_t *g_tls_octets;
  // model, aplikačný protokol a dosiahnuté percento oproti maximálnemu skóre
  // potrebné v prípade ak sa klasifikuje na exportéri
  float model[SUPPORTED_PROTOCOLS][8][MAX_PACKETS_STORED][3];
  flow_record_getter_t *protocols[3];
  flow_record_getter_t *percentages[3];
} eta_private_t;
```

Výpis 6.1: Privátna štruktúra pluginu `plugin_process_eta`.

V module boli ešte definované dve ďalšie štruktúry. Prvou je `eta_private_t` zobrazená na ukážke 6.1, ktorá je privátnou štruktúra pluginu a obsahuje atribúty, ktoré budú exportované. Druhou je `plugin_record_t` zobrazená na ukážke 6.2, ktorá definuje dáta pridané do záznamov tokov a poskytuje ukladanie interných informácií ktoré nebudú exportované.

```
typedef struct {
  // pre uloženie časového razítka posledného TLS paketu
  // použité pri výpočte medzipaketových medzier
  uint64_t last_timestamp;
  // príznak určujúci že sa jedná o TLS tok (nastavený po Client Hello)
  uint8_t is_tls;
  // pre počítanie počtu TLS paketov a veľkostí dát
  // pred exportovaním sú zapísané do g_tls_octets a g_tls_packets
  uint32_t packets;
  uint64_t octets;
  // veľkosť dát TLS požadovaných paketov a medzipaketové medzery
  // potrebné v prípade ak sa klasifikuje na exportéru
  uint16_t sizes[PACKETS_STORED];
  uint64_t ipts[PACKETS_STORED−1];
  // príznak určujúci, že medzipaketová medzera sa má vypočítať a uložiť
  uint8_t store_ipt;
  // príznak indikujúci, že správa TLS je obsadená vo viacerých paketoch
uint8_t reassembled_pdu;
// uloženie veľkosti dát TLS pre tento prípad
uint32_t reassembled_pdu_size;
} plugin_record_t;
```

Výpis 6.2: Štruktúra obsahujúce dáta pridané do záznamov.

Ďalej boli implementované inicializačné funkcie `process_eta_init` a `process_eta_init_queue`. V prvej funkcií sa alokuje privátna štruktúra záznamu, spracujú sa parametre pluginu, registrujú sa nové atribúty a registrujú sa voliteľné globálne funkcie. V druhej funkcií sa registrujú voliteľné funkcie špecifické pre frontu a vytvorí sa kópia privátnej štruktúry. Bolo ešte potrebné implementovať funkcie `process_eta_shutdown` a `process_eta_shutdown_queue`, ktoré sa volajú pri ukončení exportéru a uvoľnia alokované zdroje.

6.2.1 Pridanie atribútov

Nové atribúty záznamov je potrebné definovať v inicializačnej funkcií `process_eta_init`. Počet exportovaných veľkostí paketov a medzipaketových medzier je konfigurovateľný v tejto časti kódu, závisí na počte pridaných atribútov. Počet uložených hodnôt sa nastaví makrom `PACKETS_STORED`, makro `MAX_PACKETS_STORED` obsahuje maximálny počet exportovaných paketov. Do exportovaných záznamov boli pridané nasledujúce atribúty:

- **PACKET_SIZE_1 ... PACKET_SIZE_N** – veľkosti dát TLS prvých N paketov;
- **INTER_PACKET_TIME_1 ... INTER_PACKET_TIME_N-1** – prvých N-1 medzipaketových medzier;
- **TLS_OCTETS** – veľkosť dát TLS s obsahom `Application Data` v toku;
- **TLS_PACKETS** – počet paketov TLS s obsahom `Application Data` v toku;
- **PROTOCOL_1 ... PROTOCOL_3** – protokoly s tromi najvyššími skóremi v prípade klasifikácie na exportéru;
- **PERCENTAGE_1 ... PERCENTAGE_3** – dosiahnuté percentá protokolov.

Tieto atribúty sa pridávajú pomocou funkcie `flowmon_attribute_add` a ukladajú sa do štruktúry `eta_private_t`. Tu sa nastavuje názov nového atribútu, veľkosť a dátový typ. Pre veľkosti jednotlivých paketov sa použije 16 bitov, pre jednotlivé medzipaketové medzery 64 bitov. V rámci práce model klasifikátoru podporuje maximálne 10 paketov, kvôli tomu bolo pridaných 10 atribútov pre exportovanie veľkostí a 9 pre medzery. V prípade potreby je možné atribúty rozšíriť. Pomocou skriptu `generate_attributes` po zadaní požadovaného počtu paketov sa vygeneruje kód pre vytvorenie atribútov, tento kód stačí vložiť do inicializačnej funkcie na vyznačené miesto. Pre exportovanie počtu paketov v toku sa použije 32 bitov, pre veľkosť dát v toku 64 bitov. Vo všetkých prípadoch sa použije bezznamienkový (*unsigned*) typ. Pre jednotlivé protokoly sa použijú reťazce, teda pole znakov, pre percentá bezznamienkový dátový typ veľkosti 8 bitov. Registrácia nového atribútu pre veľkosť toku zobrazuje nasledujúci príklad:

```
flowmon_attribute_add(getter_list, "TLS_OCTETS", sizeof(uint64_t),
RECORD_VALUE_TYPE_UNSIGNED, RECORD_GETTER_GROUP_ID_NEW);
```

Výpis 6.3: Registrácia nového atribútu pomocou funkcie `flowmon_attribute_add`.

Po vytvorení k atribútom je možné pristúpiť pomocou privátnej štruktúry `eta_private_t` a nastaviť ich hodnotu pomocou funkcie `flowmon_getter_set`. Pri volaní tejto funkcie sa zvolí atribút, jeho hodnota a veľkosť a či sa má prepísať aktuálna hodnota. Tieto atribúty sa potom pridajú k existujúcim atribútom exportéru.

V prípade ak bola zapnutá klasifikácia nastavením makra `CLASSIFIER` na nenulovú hodnotu, tak sa v inicializačnej funkcií pluginu inicializuje model pre klasifikáciu. Profily jednotlivých protokolov sú uložené do poľa `model` v štruktúre `eta_private_t`. Pre indexovanie bol vytvorený výčet (`enum`) protokolov, ktorý umožňuje indexovať profil protokolu na základe zadania jeho skratky ako index.

6.2.2 Spracovanie paketov

Spracovanie paketov TLS a ukladanie potrebných informácií sa vykonáva vo funkcií `process_eta_update`. Táto funkcia sa volá pri prijatí paketu ktoréhokoľvek toku. Ako prvé je potrebné určiť či sa jedná o šifrovaný tok s obsahom TLS paketov. To je dosiahnuté kontrolou obsahu prvého paketu, ktorý obsahuje aplikačné dáta (teda prvým paketom po nadviazaní TCP spojenia). Kontroluje sa, či tento paket obsahuje správu typu *Client Hello*. Skontrolujú sa bajty indukujúce že sa jedná o správu *Handshake*, konrétne o *Client Hello* a bajty obsahujúce hlavnú verziu TLS. V prípade ak sa jedná o TLS spojenie nastaví sa

príznak `is_tls` v štruktúre `flowmon_plugin_t`, aby kontrola už znovu neprebiehala. Inak sa volanie funkcie zruší pre aktuálny tok. Toky UDP sa vyfiltrujú už pri príchode prvého paketu nového toku vo funkcií `process_eta_add`, kde sa kontroluje či prvý paket je TCP alebo UDP.

Ďalej sa spracujú už iba pakety TLS, konkrétne tie s obsahom *Application Data*. V prípade prijatia takéhoto paketu sa uloží veľkosť aplikačných dát a nastaví sa príznak `store_ipt` v štruktúre `plugin_record_t`, ktorý indikuje nutnosť výpočtu a uloženia medzipaketovej medzery pri prijatí ďalšieho paketu toku. Ďalej inkrementuje sa počet paketov TLS a pričíta sa veľkosť aplikačných dát paketu k celkovej veľkosti TLS dát toku.

V prípade prijatia dalšieho paketu prebehne výpočet medzipaketovej medzery ak je nastavený príznak `store_ipt`. Neskorší výpočet je potrebný kvôli tomu, že premenná obsahujúca časové razítko posledného paketu je aktualizovaná až po jeho spracovaní. Časové razítko posledného TLS paketu je teda dostupné až pri príchode nasledujúceho paketu toku. Táto hodnota pre posledný paket toku je obsadená v premennej `record->biflow[0].end` alebo `record->biflow[1].end`. Tieto premenné obsahujú časové razítka posledných paketov pre oba smery toku. Časové razítko posledného paketu obojsmerného toku je tá väčšia hodnota.

V prípade ak sa jedná o prvý paket s obsahom *Application Data* tak sa iba uloží časové razítko paketu. Inak sa vypočíta medzera a nastaví sa príslušná premenná privátnej štruktúry. Výpočet prebieha pomocou rozdielu časového razítka posledného a predposledného paketu s obsahom TLS *Application Data* v toku. Výpočet zobrazuje ukážka 6.4.

```
static void calc_ipt(flow_record_t *record, plugin_record_t *rd){
  // po príchode prvého paketu sa nepočíta medzipaketová medzera, iba sa uloží časové razítko
  if(rd->packets == 1){
    // uloženie časového razítka, v prípade prvej medzery sa jedná o posledné časové razítko
    rd->last_timestamp = record->biflow[0].end > record->biflow[1].end ? record->biflow[0].end :
   record->biflow[1].end;
  }
  else{
    // uloženie aktuálneho časového razítka
    uint64_t actual_timestamp = record->biflow[0].end > record->biflow[1].end ?
   record->biflow[0].end : record->biflow[1].end;
    // výpočet medzipaketovej medzery
    uint64_t ipt = actual_timestamp - rd->last_timestamp;
    rd->ipts[rd->packets-2] = ipt;
    rd->last_timestamp = actual_timestamp;
  }
  // vynulovanie príznaku, ktorý určuje výpočet medzery
  rd->store_ipt = 0;
}
```

Výpis 6.4: Výpočet medzipaketových medzier.

Takýmto spôsobom sa spracujú prijaté TLS pakety až po požadovaný počet paketov udávaný makrom `PACKETS_STORED`. V prípade ak je nastavené makro `FLOW_DATA`, ktorý určuje exportovanie údajov na úrovni toku sa spracuje celý tok. Ináč po spracovaní požadovaného počtu paketov sa spracovanie toku pluginom zruší pomocou funkcie `flowmon_hook_update_remove` a záznam sa môže exportovať.

Pred exportovaním záznamov sa volá ešte funkcia `process_eta_postprocess`. V závislosti na módu exportéru sa nastavia vybrané atribúty. Ak je nastavené makro `CLASSIFIER` na nenulovú hodnotu tak sa vykoná klasifikácia podobne ako na kolektoru a uložia sa tri protokoly s najvyššími skóremi a ich dosiahnuté percentá, ostatné atribúty sa nenaplnia.

V prípade ak dané makro nie je nastavené tak sa naplnia atribúty obsahujúce veľkosti paketov po hodnotu uvedenej v makre `PACKETS_STORED` a medzipaketové medzery po `PACKETS_STORED-1`. V prípade ak makro `FLOW_DATA` obsahuje nenulovú hodnotu tak sa nastavia aj hodnoty na úrovni toku. Ak sa nejednalo o TLS tok tak ani jedna hodnota sa nenastaví. Nenastavené atribúty v záznamoch majú hodnotu NIL.

```
static int process_eta_postprocess(void *base_data, flow_record_t *record, void *plugin_data, int
      switch_needed) {
  eta_private_t *private = (eta_private_t*) base_data;
  plugin_record_t *pd = (plugin_record_t*) plugin_data;

  // naplnenie hodnôt iba v prípade ak išlo o tok TLS
  if(pd->is_tls){
       // nastavenie veľkostí paketov a medzipaketových medzier
    #if !(CLASSIFIER)
    for (int i=0; i < PACKETS_STORED; i++){
      flowmon_getter_set(private->g_sizes[i], record,(char*)&pd->sizes[i], sizeof(uint16_t), 0);
    }
    for (int i=0; i < PACKETS_STORED-1; i++){
      flowmon_getter_set(private->g_ipts[i], record, (char*)&pd->ipts[i], sizeof(uint64_t), 0);
    }

    // nastavenie počtu paketov TLS a veľkostí dát TLS
    #if FLOW_DATA
    flowmon_getter_set(private->g_tls_packets, record, (char*)&pd->packets, sizeof(uint32_t), 0);
    flowmon_getter_set(private->g_tls_octets, record, (char*)&pd->octets, sizeof(uint64_t), 0);
    #endif
    #endif

    #if CLASSIFIER
    // volanie klasifikačnej funkcie, ktorá aj naplní atribúty
    classifier(private, pd, record);
    #endif
  }
  return (FLOW_FILTER_PASS);
}
```

Výpis 6.5: Funkcia `process_eta_postprocess`.

Kapitola 7

Testovanie a vyhodnotenie

Po implementácií pluginu a klasifikačného algoritmu nasledovalo testovanie. Nemal som možnosť nasadiť vytvorený plugin na sieťové zariadenie, funkčnosť exportéru bola otestovaná v rámci virtuálneho obrazu. Prebiehalo lokálne testovanie pomocou súborov PCAP ako aj na prevádzke v reálnom čase. Bola overená aj funkčnosť spolupráce exportéru a kolektoru. Testovanie klasifikačného algoritmu bolo vykonané pomocou extrahovaniu údajov z testovacích súborov PCAP, ktoré obsahovali jednotlivé toky protokolov. K tomuto typu vyhodnoteniu bez exportéru bolo pristúpené kvôli tomu, že exportér `flowmonexp5` pri vstupnom súbore PCAP pri každom spustení exportuje rozdielne medzipaketové medzery, čo by mohlo viesť k rozdielnym výsledkom testovania pri každom spustení. Pri nasadení exportéru v prevádzke exportovaná medzera reprezentuje reálny časový rozdiel.

7.1 Testovacie dáta

Testovacie údaje pochádzajú z dvoch zdrojov. Prvým sú online zdroje, druhým sú vlastné vytvorené súbory PCAP. Pri niektorých protokoloch bolo menej dostupných online zdrojov, kvôli tomu sa použili vlastne vytvorené toky (podobne ako pri modelu). Vlastne vytvorené súbory PCAP boli zachytené pomocou programu `Wireshark` na koncových zariadeniach. Nie je to úplne ideálne, keďže oproti hodnotám na monitorovacom zariadení sa môžu líšiť medzipaketové medzery, aj veľkosti paketov kvôli MTU. Toky boli vytvárané manuálne.

7.2 Vyhodnotenie

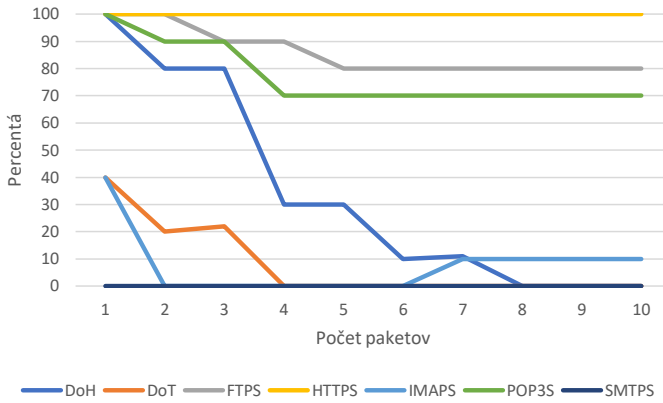
Z jednotlivých testovacích súborov PCAP boli vytvorené textové súbory obsahujúce požadované vlastnosti, konkrétne veľkosti aplikačných dát paketov TLS, medzipaketové medzery, počet paketov TLS a veľkosť dát TLS v toku. Tieto súbory sa potom použili ako vstup pre klasifikátor. Bolo použitých 70 testovacích tokov, 10 tokov pre každý protokol. Postupne sa zvyšoval počet použitých paketov až kým sa nedosiahlo maximum podporované aktuálnym modelom, teda 10 paketov. Vykonalo sa testovanie bez vlastností na úrovni toku, aj s nimi. Tabuľka 7.1 obsahuje výsledky testovania pri klasifikácií bez atribútov na úrovni paketov, tabuľka 7.2 obsahuje výsledky pri použití atribútov na úrovni toku. Každé políčko v tabuľkách obsahuje štyri hodnoty. Prvou je počet testovacích tokov daného protokolu použitých pri klasifikácií. Druhá hodnota predstavuje počet jednoznačne klasifikovaných protokolov, tj. že výstupom klasifikátoru bol daný protokol. Tretia hodnota reprezentuje počet nejednoznačne klasifikovaných tokov, tj. ak výstupom boli okrem daného protokolu aj ďalšie, pri

ktorých sa tiež dosiahlo rovnaké skóre. Štvrtá hodnota udáva počet nesprávne klasifikovaných tokov. Obrázok 7.1 zobrazuje úspešnosť klasifikácie jednotlivých protokolov bez údajov na úrovni toku v závislosti na počte paketov, obrázok 7.2 zobrazuje úspešnosť aj s údajmi na úrovni toku. Úspešnosť bola vypočítaná ako podiel súčtu jednoznačne a nejednoznačne klasifikovaných tokov k počtu klasifikovaných tokov a je uvedená v percentách.

Poč. pak.	Typ[1]	DoH	DoT	FTPS	HTTPS	IMAPS	POP3S	SMTPS	∑
1	A	10	10	10	10	10	10	10	70
	B	0	0	2	0	0	0	0	2
	C	10	4	8	10	4	10	0	46
	D	0	6	0	0	6	0	10	22
2	A	10	10	10	10	10	10	10	70
	B	0	0	2	5	0	0	0	7
	C	8	2	8	5	0	9	0	32
	D	2	8	0	0	10	1	10	31
3	A	10	9	10	10	10	10	10	69
	B	0	0	4	9	0	0	0	13
	C	8	2	5	1	0	9	0	25
	D	2	7	1	0	10	1	10	41
4	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	3	0	5	1	0	7	0	16
	D	7	1	1	0	10	3	10	32
5	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	3	0	4	1	0	7	0	15
	D	7	1	2	0	10	3	10	33
6	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	1	0	4	1	0	7	0	13
	D	9	1	2	0	10	3	10	35
7	A	9	1	10	10	10	10	10	60
	B	0	0	4	10	1	7	0	22
	C	1	0	4	0	0	0	0	5
	D	8	1	2	0	9	3	10	33
8	A	8	1	10	10	10	10	9	58
	B	0	0	4	10	1	7	0	22
	C	0	0	4	0	0	0	0	4
	D	8	1	2	0	9	3	9	32
9	A	8	1	10	10	10	10	9	58
	B	0	0	4	10	1	7	0	22
	C	0	0	4	0	0	0	0	4
	D	8	1	2	0	9	3	9	32
10	A	7	1	10	10	10	10	9	57
	B	0	0	4	10	1	7	0	22
	C	0	0	4	0	0	0	0	4
	D	7	1	2	0	9	3	9	31

[1] A – počet B – jednoznačne klasifikované C – nejednoznačne klasifikované D – nesprávne klasifikované

Tabuľka 7.1: Počet správne a nesprávne klasifikovaných tokov pri použití údajov na úrovni paketov a modelu so štandardnou odchýlkou.

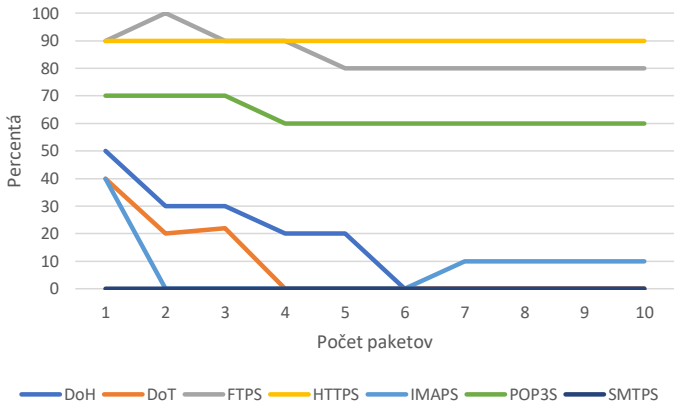


Obr. 7.1: Úspešnosť pri použití údajov na úrovni paketov a modelu so štandardnou odchýlkou.

Protokol	Typ[1]	DoH	DoT	FTPS	HTTPS	IMAPS	POP3S	SMTPS	∑
1	A	10	10	10	10	10	10	10	70
	B	0	0	1	2	0	0	0	3
	C	5	4	8	7	4	7	0	38
	D	5	6	1	1	6	3	10	29
2	A	10	10	10	10	10	10	10	70
	B	0	0	2	5	0	0	0	7
	C	3	2	8	4	0	7	0	24
	D	7	8	0	1	0	3	10	39
3	A	10	9	10	10	10	10	10	69
	B	0	0	4	9	0	0	0	13
	C	3	2	5	0	0	7	0	17
	D	7	7	1	1	0	3	10	39
4	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	2	0	5	0	0	6	0	13
	D	8	1	1	1	0	4	10	35
5	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	2	0	4	0	0	6	0	12
	D	8	1	2	1	0	4	10	36
6	A	10	1	10	10	10	10	10	61
	B	0	0	4	9	0	0	0	13
	C	0	0	4	0	0	6	0	10
	D	10	1	2	1	0	4	10	38
7	A	9	1	10	10	10	10	10	60
	B	0	0	4	9	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	9	1	2	1	9	4	10	36
8	A	8	1	10	10	10	10	9	58
	B	0	0	4	9	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	8	1	2	1	9	4	9	34
9	A	8	1	10	10	10	10	9	58
	B	0	0	4	9	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	8	1	2	1	9	4	9	34
10	A	7	1	10	10	10	10	9	57
	B	0	0	4	9	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	7	1	2	1	9	4	9	33

[1] A – počet B – jednoznačne klasifikované C – nejednoznačne klasifikované D – nesprávne klasifikované

Tabuľka 7.2: Počet správne a nesprávne klasifikovaných tokov pri použití údajov na úrovni paketov aj toku a modelu so štandardnou odchýlkou.



Obr. 7.2: Úspešnosť pri použití údajov na úrovni paketov aj toku a modelu so štandardnou odchýlkou.

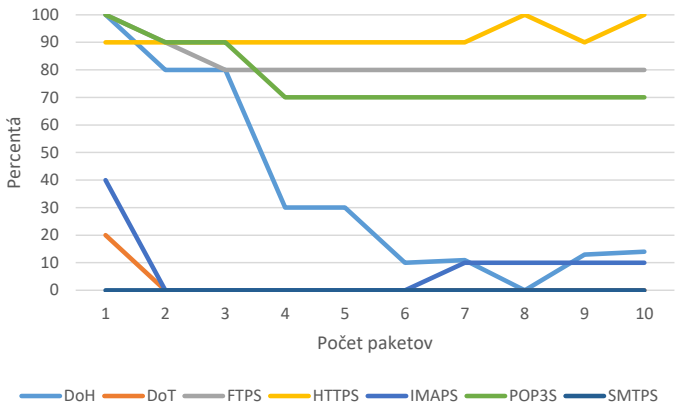
Okrem modelu popísaného v kapitole 5 bol pri testovaní vytvorený aj iný model. Tento model podobne používal minimálne a maximálne hodnoty, ale namiesto štandardnej odchýlky sa použilo 10% veľkosti intervalu $((max - min) * 0,1)$. Intervaly (min, max) boli rozšírené o túto hodnotu. Pri tomto modelu sa dosiahli podobné výsledky, s tým rozdielom že pri dosiahnutí rovnakého počtu správne klasifikovaných tokov v tomto prípade bol vyšší

počet jednoznačne klasifikovaných tokov. Výsledky testovania pri tomto modelu obsahujú tabuľky 7.3 a 7.4, úspešnosť podobne ako pri predošlom modelu interpretujú obrázky 7.3 a 7.4.

Poč. pak.	Typ[1]	DoH	DoT	FTPS	HTTPS	IMAPS	POP3S	SMTPS	∑
1	A	10	10	10	10	10	10	10	70
	B	0	0	2	0	0	0	0	2
	C	10	2	8	9	4	10	0	43
	D	0	8	0	1	6	0	10	25
2	A	10	10	10	10	10	10	10	70
	B	0	0	2	3	0	0	0	5
	C	8	0	7	6	0	9	0	30
	D	2	10	1	1	10	1	10	35
3	A	10	9	10	10	10	10	10	69
	B	0	0	4	7	0	8	0	19
	C	8	0	4	2	0	1	0	15
	D	2	9	2	1	10	1	10	35
4	A	10	1	10	10	10	10	10	61
	B	0	0	4	7	0	6	0	17
	C	3	0	4	2	0	1	0	10
	D	7	1	2	1	10	3	10	34
5	A	10	1	10	10	10	10	10	61
	B	0	0	4	8	0	6	0	18
	C	3	0	4	1	0	1	0	9
	D	7	1	2	1	10	3	10	34
6	A	10	1	10	10	10	10	10	61
	B	0	0	4	8	0	6	0	18
	C	1	0	4	1	0	1	0	7
	D	9	1	2	1	10	3	10	36
7	A	9	1	10	10	10	10	10	60
	B	0	0	4	9	1	7	0	21
	C	1	0	4	0	0	0	0	5
	D	8	1	2	1	9	3	10	34
8	A	8	1	10	10	10	10	9	58
	B	0	0	4	10	1	7	0	22
	C	0	0	4	0	0	0	0	4
	D	8	1	2	0	9	3	9	32
9	A	8	1	10	10	10	10	9	58
	B	1	0	8	9	1	7	0	26
	C	0	0	0	0	0	0	0	0
	D	7	1	2	1	9	3	9	32
10	A	7	1	10	10	10	10	9	57
	B	1	0	8	9	1	7	0	26
	C	0	0	0	1	0	0	0	1
	D	6	1	2	0	9	3	9	30

[1] A – počet B – jednoznačne klasifikované C – nejednoznačne klasifikované D – nesprávne klasifikované

Tabuľka 7.3: Počet správne a nesprávne klasifikovaných tokov pri použití údajov na úrovni paketov a modelu s percentami.

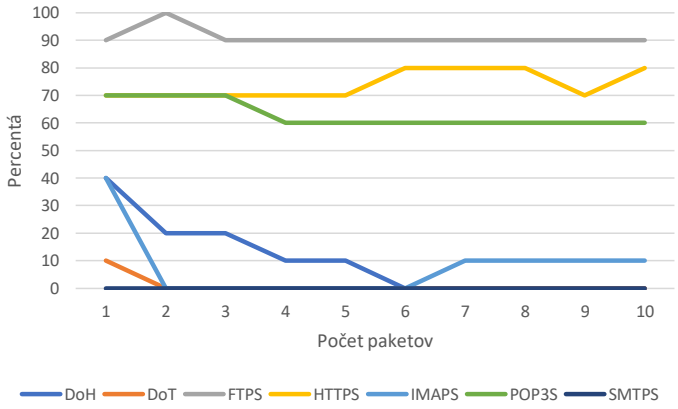


Obr. 7.3: Úspešnosť pri použití údajov na úrovni paketov a modelu s percentami.

Poč. pak.	Typ	DoH	DoT	FTPS	HTTPS	IMAPS	POP3S	SMTPS	∑
1	A	10	10	10	10	10	10	10	70
	B	0	0	2	1	0	0	0	3
	C	4	1	7	6	4	7	0	29
	D	6	9	1	3	6	3	10	38
2	A	10	10	10	10	10	10	10	70
	B	0	0	4	3	0	0	0	7
	C	2	0	6	4	0	7	0	19
	D	8	10	0	3	10	3	10	44
3	A	10	9	10	10	10	10	10	69
	B	0	0	5	7	0	6	0	18
	C	2	0	4	0	0	1	0	7
	D	8	9	1	3	10	3	10	44
4	A	10	1	10	10	10	10	10	61
	B	0	0	5	7	0	5	0	17
	C	1	0	4	0	0	1	0	6
	D	9	1	1	3	10	4	10	38
5	A	10	1	10	10	10	10	10	61
	B	0	0	5	7	0	5	0	17
	C	1	0	4	0	0	1	0	6
	D	9	1	1	3	10	4	10	38
6	A	10	1	10	10	10	10	10	61
	B	0	0	5	8	0	5	0	18
	C	0	0	4	0	0	1	0	5
	D	10	1	1	2	10	4	10	38
7	A	9	1	10	10	10	10	10	60
	B	0	0	5	8	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	9	1	1	2	9	4	10	36
8	A	8	1	10	10	10	10	9	58
	B	0	0	5	8	1	6	0	20
	C	0	0	4	0	0	0	0	4
	D	8	1	1	2	9	4	9	34
9	A	8	1	10	10	10	10	9	58
	B	0	0	9	7	1	6	0	23
	C	0	0	0	0	0	0	0	0
	D	8	1	1	3	9	4	9	35
10	A	7	1	10	10	10	10	9	57
	B	0	0	9	7	1	6	0	23
	C	0	0	0	1	0	0	0	1
	D	7	1	1	2	9	4	9	33

[1] A – počet B – jednoznačne klasifikované C – nejednoznačne klasifikované D – nesprávne klasifikované

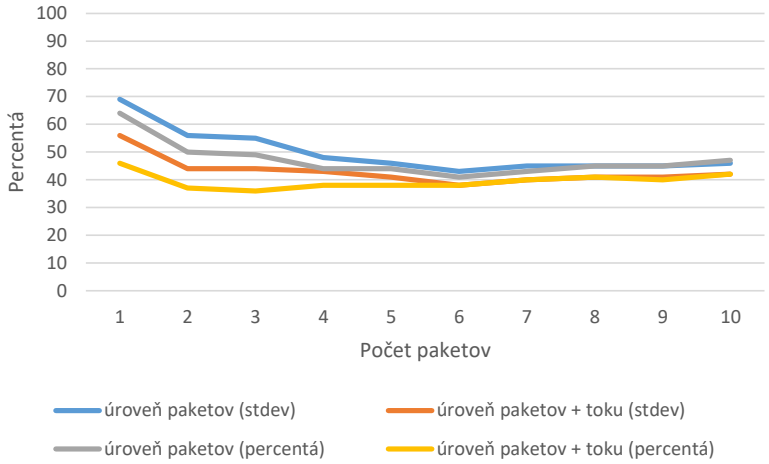
Tabuľka 7.4: Počet správne a nesprávne klasifikovaných tokov pri použití údajov na úrovni paketov aj toku a modelu s percentami.



Obr. 7.4: Úspešnosť pri použití údajov na úrovni paketov aj toku a modelu s percentami.

Ak berieme v úvahu iba jednoznačne klasifikované toky, tak najlepším scenárom bolo použitie 9 paketov bez údajov na úrovni toku a s modelom, kde sa používali percentá. Z výsledkov vidíme, že pridávaním vlastností na úrovni toku klasifikácia sa nezlepšila, ale trochu zhoršila. Taktiež vidíme že zvyšovaním počtu paketov sa postupne zvyšuje jednoznačne klasifikovaných protokolov oproti nejednoznačne klasifikovaných, keďže sa zvyšuje rozptyl možných skóre. Najlepšia úspešnosť sa dosiahla pri použití 10 paketov, konkrétne 47%. Celkovú úspešnosť jednotlivých prístupov zobrazuje obrázok 7.5. Úspešnosť bola vy-

počítaná ako podiel súčtu jednoznačne a nejednoznačne klasifikovaných tokov pri danej metóde k počtu klasifikovaných tokov a je uvedená v percentách.



Obr. 7.5: Úspešnosť jednotlivých prístupov.

Celkovo nízka úspešnosť môže byť pôsobená viacerými faktormi. Prvým je nižšie množstvo trénovacích dát. Ďalším faktorom je rôznorodosť dát, štatistická klasifikácia funguje najlepšie ak trénovacie dáta sú zachytené na tom istom monitorovacom zariadení kde sa monitoruje prevádzka vybranej množiny strojov. Taktiež výsledok ovplyvňujú jednotlivé priradené skóre. Administrátor si môže vytvoriť nové trénovacie dáta pomocou dodaných skriptov, alebo v prípade potreby môže upraviť klasifikátor a vytvoriť nový algoritmus v ktorom použije exportované údaje.

7.2.1 Rýchlosť

Po testovaní úspešnosti bola ešte otestovaná rýchlosť klasifikátoru, ktorý je určený pre kolektor. Pre testovanie boli vytvorené tri vstupné súbory vo formáte CSV pomocou exportéru. Prvý súbor obsahoval 1 tok, druhý 10 a tretí 100 tokov. V každom prípade boli naplnené všetky veľkosti paketov a medzipaketové medzery, taktiež sa naplnili údaje na úrovni toku. Testy boli vykonané na operačnom systéme Linux Mint 19.2 s použitím procesoru Intel i7-3520M. Hodnoty reprezentujú čas od spustenia klasifikátoru po získanie výsledkov, nezahŕňajú zápis, resp. výpis. Výsledky testovania zobrazuje tabuľka 7.5.

Počet tokov / **Testy**	1	10	100
1.	0,0015	0,0023	0,132
2.	0,0015	0,0024	0,132
3.	0,0013	0,0024	0,129
4.	0,0017	0,0024	0,132
5.	0,0014	0,0025	0,132
Priemer	0,0015	0,0024	0,1314

Tabuľka 7.5: Doby spracovania vstupných súborov v sekundách.

Kapitola 8

Záver

Cieľom diplomovej práce bolo rozšíriť záznamy NetFlow pre zlepšenie klasifikácie šifrovanej sieťovej prevádzky a vytvoriť algoritmus na klasifikáciu pomocou hodnôt zo záznamov NetFlow.

V rámci práce boli preštudované rôzne typy algoritmov ktoré boli aplikované v rôznych výskumoch na klasifikáciu šifrovanej komunikácie. Ďalej boli naštudované atribúty vhodné pre klasifikáciu šifrovanej prevádzky a boli zvolené atribúty pre použitie v rámci tejto práce. Konkrétne ide o veľkosti aplikačných dát paketov TLS, medzipaketové medzery, počet paketov TLS a veľkosť TLS dát v toku. Taktiež bol naštudovaný protokol NetFlow na monitorovanie sieťovej komunikácie.

Po získaní znalostí a výberu atribútov nasledoval návrh klasifikačného algoritmu, ktorý používa vybrané vlastnosti sieťových tokov a klasifikuje ich do tried na základe aplikačného protokolu. Ďalej nasledoval návrh modulu pre exportér od firmy Flowmon Networks a. s., ktorý poskytuje exportovanie zvolených atribútov.

Nasledovala implementácia modulu pre exportér v jazyku C. Záznamy NetFlow boli rozšírené o nové atribúty. Modul spracuje toky po paketoch, identifikuje toky TLS a ukladá potrebné hodnoty. Po implementácii exportovania údajov nasledovala implementácia klasifikátoru pre kolektor v jazyku Python. Boli vytvorené profily podporovaných protokolov z trénovacích dát, ktoré tvoria model. Konkrétne ide o protokoly DNS over HTTP, DNS over TLS, FTPS, HTTPS, IMAPS, POP3S, SMTPS. Klasifikátor prideľuje skóre na základe hodnôt vybraných vlastností tokov, výstupom je protokol s najvyšším skóre. Algoritmus bol implementovaný aj v modulu pre exportér, užívateľ môže zvoliť kde chce vykonať klasifikáciu.

Po implementácií nasledovalo testovanie úspešnosti klasifikačného algoritmu. Testovanie bolo vykonané na 70 tokoch, bolo použitých 10 tokov pre každý aplikačný protokol. Najlepšia úspešnosť sa dosiahla pri použití 10 paketov a absencie údajov na úrovni toku. Bolo vykonané ešte testovanie rýchlosti klasifikátoru pre kolektor na vstupných súboroch s rôznym počtom záznamov.

Podarilo sa rozšíriť záznamy NetFlow o vybrané atribúty šifrovanej prevádzky a implementovať klasifikačný algoritmus, ktorý bol vyhodnotený. Tým ciele práce boli naplnené. Úspešnosť klasifikátoru bola v najlepšom prípade 47%. Zvýšenie úspešnosti dáva priestor pre ďalšie rozšírenie práce, kedy by sa mohli pridať ďalšie atribúty alebo zlepšiť algoritmus. Taktiež by sa dal klasifikátor rozšíriť o ďalšie aplikačné protokoly.

Literatúra

[1] *Chytré řešení pro monitorování a bezpečnost počítačových sítí* [online]. [cit. 2020-02-10]. Dostupné z: https://www.flowmon.com/cs/products/flowmon.

[2] *Cisco IOS NetFlow Version 9 Flow-Record Format* [online]. [cit. 2020-01-17]. Dostupné z: https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.pdf.

[3] *Flow-based approaches in network management* [online]. [cit. 2020-03-01]. Dostupné z: https://media.plixer.com/resources/whitepapers/flowBased-Approaches-in-Network-Management.pdf.

[4] *Flowmon Probe* [online]. [cit. 2020-03-17]. Dostupné z: https://www.flowmon.com/en/products/appliances/probe.

[5] *Network Test Access Point (TAP) and Port Mirroring (SPAN)* [online]. [cit. 2020-02-08]. Dostupné z: https://www.garlandtechnology.com/tap-vs-span.

[6] *Overfitting in Machine Learning: What It Is and How to Prevent It* [online]. [cit. 2020-03-01]. Dostupné z: https://elitedatascience.com/overfitting-in-machine-learning.

[7] *What is Machine Learning?* [online]. [cit. 2020-03-01]. Dostupné z: https://deepai.org/machine-learning-glossary-and-terms/machine-learning.

[8] *What Is Machine Learning?* [online]. [cit. 2019-12-22]. Dostupné z: https://www.mathworks.com/discovery/machine-learning.html.

[9] *What is machine learning?* [online]. [cit. 2019-12-21]. Dostupné z: https://www.ibm.com/topics/machine-learning.

[10] IEEE Standard for Ethernet. *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*. 2018, s. 1–5600. Dostupné z: https://ieeexplore.ieee.org/document/8457469.

[11] Adibi, S. Traffic Classification – Packet-, Flow-, and Application-based Approaches. *International Journal of Advanced Computer Science and Applications*. The Science and Information Organization. 2010, roč. 1, č. 1. Dostupné z: http://dx.doi.org/10.14569/IJACSA.2010.010102.

[12] Azzana, Y., Chabchoub, Y., Fricker, C., Guillemin, F. a Robert, P. Adaptive algorithms for identifying large flows in IP traffic. Január 2009. Dostupné z: https://www.researchgate.net/publication/23962181_Adaptive_algorithms_for_identifying_large_flows_in_IP_traffic.

[13] B. Claise, E. *Cisco Systems NetFlow Services Export Version 9* [Internet Requests for Comments]. RFC 5321. RFC Editor, October 2008. Dostupné z: https://www.rfc-editor.org/rfc/rfc3954.txt.

[14] B. Claise, E. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information* [Internet Requests for Comments]. RFC 5101. RFC Editor, January 2008. Dostupné z: https://www.rfc-editor.org/rfc/rfc5101.txt.

[15] B. Claise, E. a B. Trammell, E. *Information Model for IP Flow Information Export (IPFIX)* [Internet Requests for Comments]. RFC 7012. RFC Editor, September 2013. Dostupné z: https://www.rfc-editor.org/rfc/rfc7012.txt.

[16] B. Claise, E., B. Trammell, E. a Aitken, P. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [Internet Requests for Comments]. RFC 7011. RFC Editor, September 2013. Dostupné z: https://www.rfc-editor.org/rfc/rfc7011.txt.

[17] Bakhshi, T. a Ghita, B. On Internet Traffic Classification: A Two-Phased Machine Learning Approach. *J. Comput. Netw. Commun.* jún 2016, roč. 2016. Dostupné z: https://doi.org/10.1155/2016/2048302.

[18] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A. a Salamatian, K. Traffic Classification on the Fly. *SIGCOMM Comput. Commun. Rev.* Association for Computing Machinery. apríl 2006, roč. 36, č. 2, s. 23–26. Dostupné z: https://doi.org/10.1145/1129582.1129589. ISSN 0146-4833.

[19] Cao, Z., Xiong, G., Zhao, Y., Li, Z. a Guo, L. A Survey on Encrypted Traffic Classification. In: *Applications and Techniques in Information Security*. Springer Berlin Heidelberg, 2014, s. 73–81.

[20] Carela Español, V. *Network traffic classification: from theory to practice*. 2014. Dizertačná práca. Universitat Politècnica de Catalunya. Dostupné z: http://hdl.handle.net/10803/283573.

[21] Carela Español, V., Barlet Ros, P. a Solé Pareta, J. Traffic Classification with Sampled NetFlow. Február 2020. Dostupné z: https://www.researchgate.net/publication/264873851_Traffic_Classification_with_Sampled_NetFlow.

[22] Claise, B. a Wolter, R. *Network Management: Accounting and Performance Strategies*. Cisco Press, 2007. ISBN 1587051982.

[23] Crotti, M., Dusi, M., Gringoli, F. a Salgarelli, L. Traffic Classification Through Simple Statistical Fingerprinting. *SIGCOMM Comput. Commun. Rev.* ACM. 2007, roč. 37, č. 1, s. 5–16. Dostupné z: http://doi.acm.org/10.1145/1198255.1198257. ISSN 0146-4833.

[24] Deebalakshmi, R. a Jyothi, V. L. A survey of classification algorithms for network traffic. In: *2016 Second International Conference on Science Technology Engineering and Management (ICONSTEM)*. March 2016, s. 151–156.

[25] Erman, J., Arlitt, M. a Mahanti, A. Traffic Classification Using Clustering Algorithms. In:. Association for Computing Machinery, 2006. Dostupné z: https://doi.org/10.1145/1162678.1162679.

[26] Erman, J., Mahanti, A., Arlitt, M. a Williamson, C. Identifying and Discriminating between Web and Peer-to-Peer Traffic in the Network Core. In:. Association for Computing Machinery, 2007. Dostupné z: https://doi.org/10.1145/1242572.1242692.

[27] Jaber, M., Cascella, R. G. a Barakat, C. Can We Trust the Inter-Packet Time for Traffic Classification? In: *2011 IEEE International Conference on Communications (ICC)*. 2011, s. 1–5. Dostupné z: https://ieeexplore.ieee.org/document/5963024.

[28] Jaber, M. a Barakat, C. Enhancing Application Identification by Means of Sequential Testing. In: *NETWORKING 2009*. Springer Berlin Heidelberg, 2009, s. 287–300. ISBN 978-3-642-01399-7.

[29] John, G. a Langley, P. Estimating Continuous Distributions in Bayesian Classifiers. *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*. Február 2013, roč. 1.

[30] Lan, K.-c. a Heidemann, J. A Measurement Study of Correlations of Internet Flow Characteristics. *Comput. Netw.* Elsevier North-Holland, Inc. 2006, roč. 50, č. 1, s. 46–62. Dostupné z: https://doi.org/10.1016/j.comnet.2005.02.008. ISSN 1389-1286.

[31] Lotfollahi, M., zade, R. Shirali hossein, Jafari Siavoshani, M. a Saberian, M. Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning. *Soft Computing*. September 2017. Dostupné z: https://www.researchgate.net/publication/319622754_Deep_Packet_A_Novel_Approach_For_Encrypted_Traffic_Classification_Using_Deep_Learning.

[32] Mori, T., Uchida, M., Kawahara, R., Pan, J. a Goto, S. Identifying Elephant Flows Through Periodically Sampled Packets. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2004, s. 115–120. IMC '04. Dostupné z: http://doi.acm.org/10.1145/1028788.1028803. ISBN 1-58113-821-0.

[33] Nguyen, T. T. T. a Armitage, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*. 2008, roč. 10, č. 4, s. 56–76. Dostupné z: https://ieeexplore.ieee.org/document/4738466.

[34] Nguyen, T. a Armitage, G. Training on multiple sub-flows to optimise the use of Machine Learning classifiers in real-world IP networks. In:. November 2006. Dostupné z: https://ieeexplore.ieee.org/document/4116573.

[35] Nguyen, T. a Armitage, G. Synthetic Sub-flow pairs for timely and stable IP traffic identification. *IEEE/ACM Transactions on Networking*. Január 2007, roč. 20. Dostupné z: https://www.researchgate.net/publication/228658200_Synthetic_Sub-flow_pairs_for_timely_and_stable_IP_traffic_identification.

[36] Pietrzyk, M. *Methods and algorithms for network traffic classification*. 2011. Dizertačná práca. Télécom ParisTech. Dostupné z: http://www.eurecom.fr/publication/3366.

[37] Quittek, J., Bryant, S., Claise, B., Aitken, P. a Meyer, J. *Information Model for IP Flow Information Export* [Internet Requests for Comments]. RFC 5102. RFC Editor, January 2008. Dostupné z: https://www.rfc-editor.org/rfc/rfc5102.txt.

[38] Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.3* [Internet Requests for Comments]. RFC 8446. RFC Editor, August 2018. Dostupné z: https://www.rfc-editor.org/rfc/rfc8446.txt.

[39] Rychlý, M. *Klasifikace a predikce* [online]. [cit. 2019-12-22]. Dostupné z: http://www.fit.vutbr.cz/~rychly/public/docs/classification-and-prediction/classification-and-prediction.pdf.

[40] Shaikh, Z. A. a Harkut, D. An Overview of Network Traffic Classification Methods. In:. February 2015. Dostupné z: https://pdfs.semanticscholar.org/8efd/03df47062a376fbd1e8710a10940296643a6.pdf.

[41] Trammell, B. a Boschi, E. *Bidirectional Flow Export Using IP Flow Information Export (IPFIX)* [Internet Requests for Comments]. RFC 5103. RFC Editor, January 2008. Dostupné z: https://www.rfc-editor.org/rfc/rfc5103.txt.

[42] Vlăduţu, A., Comăneci, D. a Dobre, C. Internet traffic classification based on flows' statistical properties with machine learning. *International Journal of Network Management*. 2017, roč. 27, č. 3. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.1929.

[43] Wang, X. a Parish, D. J. Optimised Multi-stage TCP Traffic Classifier Based on Packet Size Distributions. In: *2010 Third International Conference on Communication Theory, Reliability, and Quality of Service*. June 2010, s. 98–103.

[44] Williams, N., Zander, S. a Armitage, G. A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification. *Computer Communication Review*. Október 2006, roč. 36.

[45] Yang, Y. a Webb, G. I. Discretization for naive-Bayes learning: managing discretization bias and variance. *Machine Learning*. 2009, roč. 74, č. 1, s. 39–74. Dostupné z: https://doi.org/10.1007/s10994-008-5083-5.

[46] Yingqiu, L., Wei, L. a Yunchun, L. Network Traffic Classification Using K-means Clustering. September 2007.

[47] Zuev, D. a Moore, A. Traffic Classification Using a Statistical Approach. *Lecture Notes in Computer Science*. Marec 2005.

Príloha A

Obsah pamäťového média

Pamäťové médium obsahuje nasledujúce priečinky:

- **tex/** – Zdrojové súbory technickej správy vo formáte LaTeX, potrebné obrázky a `Makefile` na preklad.
- **pdf/** – Technická správa vo formáte PDF, obsahuje verziu pre WIS a verziu pre tlač.
- **src/** – Zdrojové kódy. Súčasťou obsahu priečinka **src/** sú aj nasledujúce podpriečinky:
  - **plugin/** – Kód pluginu, `Makefile` na preklad, konfiguračné súbory pre spustenie, súbor `README.txt` obsahujúci návod na spustenie a `SETUP.txt` obsahujúci návod na konfiguráciu.
  - **klasifikator/** – Kód klasifikátoru pre Python 2 a 3, súbor `README.txt` obsahujúci návod na spustenie.
  - **skripty/** – Pomocné skripty.
- **data/** – Súbory PCAP a z nich vygenerované textové súbory použité pri práci. Súčasťou obsahu priečinka **data/** sú nasledujúce podpriečinky:
  - **trenovacie/** – Súbory použité pri vytváraní modelu.
  - **testovacie/** – Súbory použité pri vyhodnocovaní.
- **vysledky/** – Výsledky vyhodnotenia testovacích tokov vo formáte CSV.

Príloha B

Profily jednotlivých protokolov

```
#min. velkost paketu
[[[21.364916635990518, 46.872425957386085, 74.33593774815768], [0, 0, 0],
   [0, 0, 20.837722339831622], [0, 0, 0], [0, 0, 113.0], [0, 0, 875.0],
   [0, 0, 270.0], [32.0, 32.0, 32.0], [39.0, 39.0, 39.0], [24.0, 24.0,
   24.0]],
#max. velkost paketu
[[310.63508336400946, 186.1275740426139, 172.66406225184232],
   [1776.7672286341842, 1238.5963675998705, 1109.650185039395],
   [342.74035855914934, 212.6903183560416, 34.16227766016838],
   [1262.9809885145103, 875.9825833926122, 894.3550359870912],
   [369.15980368408697, 369.15980368408697, 113.0], [1295.2692901406915,
   1295.2692901406915, 875.0], [1292.1247169827893, 1292.1247169827893,
   270.0], [32.0, 32.0, 32.0], [39.0, 39.0, 39.0], [24.0, 24.0, 24.0]],
#min. medzip. medzera
[[0, 5.566509708258527, 5.986159004400905], [0, 0, 0], [0, 0, 0], [0, 0,
   3.0], [0, 0, 53.0], [0, 0, 263.0], [0.0, 0.0, 0.0], [53684.0, 53684.0,
   53684.0], [17.0, 17.0, 17.0]],
#max. medzip. medzera
[[544.340302563647, 16.433490291741474, 16.013840995599097],
   [386.9550244044887, 45.921205394677244, 1.3499271061118825],
   [4614.20007586102, 1.5, 1.4714045207910318], [439.2481926997833,
   439.2481926997833, 3.0], [18110.83766504455, 18110.83766504455, 53.0],
   [44065.80090905355, 44065.80090905355, 263.0], [0.0, 0.0, 0.0],
   [53684.0, 53684.0, 53684.0], [17.0, 17.0, 17.0]],
#min. pocet paketov
[0.3606403689244999, 1.4733656391764776, 1.5938835689662931],
#max. pocet paketov
[11.6393596310755, 4.526634360823523, 3.406116431033707],
#min. velkost toku
[0, 0, 0],
#max. velkost toku
[4461.987544928582, 1370.7897567788455, 1305.066068808011]]
```

Výpis B.1: Profil protokolu DNS over TLS.

```
#min. velkost paketu
[[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
    10.743485272908153], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
    90.34452264662848]],
#max. velkost paketu
[[320.3778149107167, 306.55926076130726, 294.06300586186273],
    [774.820390093678, 760.4081772283756, 730.6508781759479],
    [182.94263195127363, 172.9880223721899, 134.99344947612076],
    [795.3820388193375, 779.5048897145859, 706.3029374407499],
    [1015.7920856638798, 787.850610791357, 759.4799097027067],
    [892.6995871679198, 573.1684124277358, 119.25651472709185],
    [1670.7657401738104, 602.2683378405897, 387.67288146682677],
    [549.5144929304876, 441.4841740813628, 333.65133590786326],
    [349.26033272052854, 136.62176579359675, 131.647883507521],
    [1758.5127659498303, 784.9896429286657, 114.65547735337152]],
#min. medzip. medzera
[[0, 0, 0], [0, 0.0, 0.0], [0, 0, 0], [0, 0, 0], [0, 0, 0.0], [0, 0, 0],
    [0, 0, 0], [0, 0.0, 0.0], [0, 0, 0]],
#max. medzip. medzera
[[9.90592068026641, 9.572199630180364, 7.670308521068954],
    [33.52631196450267, 0.0, 0.0], [66.53895575864645, 25.83329613083375,
    16.144660377352203], [50.217643313436874, 7.042516829796024,
    6.818118685772619], [17.62571418855899, 2.484122918275927, 0.0],
    [203.7516386103354, 23.606499124843925, 2.595119035711904],
    [35765.75805238471, 43.42920993509, 25.299985259413464],
    [7.464255551920374, 0.0, 0.0], [2258.9719208048427, 164.76325869781738,
     20.11101251999952]],
#min. pocet paketov
[0, 0, 5.477597841129574],
#max. pocet paketov
[950.0822087224199, 51.101661337321005, 14.522402158870426],
#min. velkost toku
[0, 0, 1148.6707147763975],
#max. velkost toku
[108834.270037996, 12328.989277181452, 2008.3292852236025]]
```

Výpis B.2: Profil protokolu DNS over HTTPS.

```
#min. velkost paketu
[[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
   0], [0, 0, 0], [0, 0, 42.801230038066365], [0, 0, 0]],
#max. velkost paketu
[[1877.9070272407814, 1541.5896928361624, 1210.2194930889223],
   [2756.0177947813618, 1923.790118614698, 1782.8655541351795],
   [3494.1617198404792, 1913.9071832127886, 1717.6579135728516],
   [3856.696022625918, 1995.092245925285, 1867.3664553800807],
   [6858.931704975062, 2008.7139027811968, 1913.7772623081514],
   [6947.176073546532, 2040.339137374874, 1965.1787270111806],
   [7111.8040813636235, 2013.6668808398117, 1913.2233283420603],
   [10424.271643616372, 2000.3170531811966, 1887.688714168106],
   [13318.478610557144, 1922.4279216301861, 1810.1987699619335],
   [10025.242832301861, 2045.932358629295, 1994.617457549573]],
#min. medzip. medzera
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
   0], [0, 0, 0], [0, 0, 0]],
#max. medzip. medzera
[[743.9722754437331, 147.0507499310669, 100.03563257605117],
   [39542.32613881321, 5980.125109940287, 109.23984670111616],
   [5724.130075954719, 209.28246086683174, 44.33364126824959],
   [5206.662937147203, 553.0688018746768, 94.62483084291436],
   [67297.9603957435, 226.86965155703126, 125.54733709397615],
   [32194.77979941329, 149.54866470659408, 62.05761174525403],
   [72327.78850278804, 309.6813262001666, 86.78185530500005],
   [46234.985736707175, 458.9021264374824, 149.67315186422576],
   [33298.584403807, 186.41871971697336, 79.91476046016196]],
#min. pocet paketov
[0, 0, 0],
#max. pocet paketov
[7559.916824162467, 167.00026415767195, 76.17742369183499],
#min. velkost toku
[0, 0, 0],
#max. velkost toku
[9588705.412492244, 201319.17188111506, 107301.43631663336]]
```

Výpis B.3: Profil protokolu HTTPS.

```
#min. velkost paketu
[[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
   0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
#max. velkost paketu
[[2045.1962974910402, 2007.3259750748998, 1910.449554356987],
   [2105.0896888008847, 2063.331016763993, 1956.528545945523],
   [2095.6750881608727, 2054.545356342073, 1949.3190186788331],
   [2105.0896888008847, 2063.331016763993, 1956.528545945523],
   [2103.375850009153, 2061.7441546877135, 1955.2396915038378],
   [2107.197317713505, 2065.2745244928124, 1958.0926425999935],
   [2100.698239715859, 2058.588934581887, 1951.8920364370663],
   [2105.0589443350636, 2062.152144958955, 1952.2697068535847],
   [2060.539466681139, 2007.4864226298118, 1873.3110255322626],
   [2091.9898465470187, 2040.6861989498632, 1945.5648046843562]],
#min. medzip. medzera
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0,
   0], [0, 0, 0], [0, 0, 0]],
#max. medzip. medzera
[[4760.133481516386, 11.548694739537527, 7.559614682921289],
   [195.4979222880656, 2.5813347903782766, 1.4330127018922192],
   [4160.802245314526, 5.022259189450265, 2.6175577309210922],
   [156.97671901059783, 2.550119604220181, 1.4987228587060337],
   [17.432063024585414, 2.901291778397037, 2.847415036978905],
   [214.5640882438677, 2.7859570801614777, 2.5890150893739516],
   [18.269970693038903, 2.5954583420518293, 1.4883855118277622],
   [195.6423047794683, 1.48237638894272, 1.4670248868079294],
   [28.91581195993539, 15.745073281209649, 2.7756704744277236]],
#min. pocet paketov
[0, 0, 9.74722737455495],
#max. pocet paketov
[10025.700297096553, 1682.1191049058343, 98.25277262544505],
#min. velkost toku
[0, 0, 0],
#max. velkost toku
[14338072.36142669, 2416734.356566647, 61007.494852345975]]
```

Výpis B.4: Profil protokolu FTPS.

```
#min. velkost paketu
[[[0, 22.977403467830484, 57.219066250929004], [0, 49.182299212600334,
   53.0], [0, 0, 9.505812048570661], [0, 39.082230211483804,
   40.756079059386245], [0, 0, 4.566766339101065], [0, 21.35283436304977,
   36.60808164115469], [0, 0, 9.969153423048944], [0, 0,
   26.000773355936072], [0, 0, 69.0], [0, 39.22795967574056,
   47.61150143136359]],
#max. velkost paketu
[[1659.0759242691308, 117.02259653216952, 112.780933749071],
   [1663.0121671086524, 72.81770078739967, 53.0], [1659.7264743495994,
   315.72568874651205, 288.49418795142935], [1660.5910967561626,
   98.9177697885162, 97.24392094061375], [1660.3700400442299,
   320.9581345227992, 309.43323366089896], [1665.1542079485089,
   100.64716563695023, 39.39191835884531], [1671.6203199622073,
   231.38560125144085, 144.03084657695106], [1676.7948215070764,
   292.70378902963756, 79.99922664406392], [1744.0695650969985,
   796.7999183612283, 69.0], [1666.3511307620422, 114.77204032425944,
   74.38849856863641]],
#min. medzip. medzera
[[0, 0, 1.0], [0, 0, 0], [0, 0, 1.0], [0, 0, 0], [0, 0, 1.0], [0, 0, 0],
   [0, 0, 0], [0, 0, 0], [0, 0, 0]],
#max. medzip. medzera
[[1313.0174641931899, 75.0732701589227, 1.0], [3377.8649695138092,
   64.26785636612189, 52.40273651366197], [213.092642010916,
   45.504219157617726, 1.0], [7094.335992168683, 1709.099440214098,
   1588.27695599077], [274.94464835577105, 44.34108558459705, 1.0],
   [735688.7806565163, 447.06783375425897, 433.2870335675341],
   [30106.04788659235, 203.28354107328835, 12.835769366803536],
   [443.6922970616996, 428.83060183579676, 328.8286543679558],
   [417.05772765914287, 205.2644240641835, 67.82332543256544]],
#min. pocet paketov
[0, 0, 0],
#max. pocet paketov
[8992.695258707427, 247.38460037894612, 204.72599127201266],
#min. velkost toku
[0, 0, 0],
#max. velkost toku
[13024808.728425816, 21424.999237263346, 19038.467173751396]]
```

Výpis B.5: Profil protokolu IMAPS.

```
#min. velkost paketu
[[[0, 0, 0], [0, 16.545855386202227, 24.005203384401153], [0,
    11.715860608591392, 32.76079366120864], [0, 20.009054482493717,
    30.759259343915346], [0, 0, 0.5953075200662994], [0,
    17.286280019539582, 65.30654633168358], [0, 9.868220674026126,
    32.61113957206851], [0, 4.980486041342662, 18.199147446520417], [0,
    19.427492993384547, 31.067230964514774], [0, 11.097960074989913,
    24.612599794646528]],
#max. velkost paketu
[[1748.0682027118091, 831.1741937012416, 824.6296148148143],
    [1722.977650071979, 80.45414461379777, 44.994796615598844],
    [1699.8175436342804, 205.28413939140862, 194.23920633879138],
    [1714.7447822526458, 98.99094551750629, 44.240740656084654],
    [1736.8906037070692, 432.906857869345, 182.40469247993371],
    [1709.2541324357753, 96.71371998046043, 86.69345366831642],
    [1717.6910742748273, 141.13177932597387, 46.38886042793149],
    [1731.0732459516519, 124.01951395865734, 85.80085255347959],
    [1721.7796551704268, 99.57250700661545, 44.93276903548523],
    [1730.665037793308, 101.90203992501009, 38.38740020535347]],
#min. medzip. medzera
[[0, 0, 0], [0, 9.818627467043862, 13.341073084926101], [0, 0, 0], [0,
    13.985172896313976, 17.144652432626476], [0, 0, 0], [0, 0, 0], [0, 0,
    0], [0, 0, 0], [0, 0, 1.9294120174277118]],
#max. medzip. medzera
[[6156.232668756535, 19.33718785404514, 3.8908422980528035],
    [116.7413042234742, 43.18137253295614, 37.6589269150739],
    [219.63101082315515, 4.985663998017611, 2.587877538267963],
    [743.1987519722745, 45.014827103686024, 41.85534756737353],
    [326.69999694451604, 23.559800037718404, 3.8944271909999157],
    [2263.821967337564, 1769.5606404272671, 1689.4135904281084],
    [198.33020620010342, 9.872635519187096, 1.476280484787101],
    [1572.7378906787828, 245.8688731399359, 213.93539205707592],
    [326.89418826040855, 15.70376493571209, 12.070587982572288]],
#min. pocet paketov
[0, 0, 3.1630618223696567],
#max. pocet paketov
[3229.411487523941, 131.58929799481254, 54.836938177630344],
#min. velkost toku
[0, 0, 0],
#max. velkost toku
[4578136.734909786, 122053.24104494255, 43353.45971065797]]
```

Výpis B.6: Profil protokolu POP3S.

```
#min. velkost paketu
[[[68.83450429409078, 77.11189362253384, 85.0], [46.400464013561646,
   47.466711289782786, 50.03077004416764], [231.98089233237363,
   238.4680273525782, 261.0], [42.90779746611885, 45.39781545366444,
   49.102885682970026], [42.223002862727185, 47.74126241502256, 53.0],
   [52.57695381103848, 55.95627013125123, 62.133655727137665],
   [36.83450429409078, 45.11189362253385, 53.0], [33.655349552261754,
   40.367370292241866, 46.039103388739846], [29.706037356169436,
   45.11189362253385, 53.0], [32.01661472814935, 35.10466386022449,
   52.97853002998788]],
#max. velkost paketu
[[149.16549570590922, 140.88810637746616, 85.0], [75.59953598643835,
   74.53328871021722, 71.96922995583236], [306.0191076676264,
   299.5319726474218, 261.0], [95.09220253388115, 92.60218454633556,
   77.89711431702997], [95.77699713727282, 90.25873758497744, 53.0],
   [133.42304618896154, 130.04372986874878, 112.86634427286234],
   [117.16549570590922, 108.88810637746616, 53.0], [184.34465044773825,
   102.63262970775813, 96.96089661126015], [204.29396264383055,
   108.88810637746616, 53.0], [126.98338527185065, 123.89533613977551,
   122.02146997001212]],
#min. medzip. medzera
[[0, 0, 0], [0, 0, 43.352491057904174], [0, 0, 0], [0, 0.47663105737106726,
    36.8729751543817], [0, 0, 0], [0, 0.5994765784047242,
   39.05037941943483], [0, 0, 0], [0, 0, 150.0915317996259], [0, 0, 0]],
#max. medzip. medzera
[[113.6624017857667, 9.46011795566127, 2.515078753637713],
   [72.32369866164588, 61.37816643660808, 53.647508942095826],
   [33.230425845649435, 2.598351645237167, 1.479157423749955],
   [69.34698614142201, 57.523368942628935, 52.1270248456183],
   [20.885414071971244, 2.5270462766947297, 1.4330127018922192],
   [69.30712276506863, 57.400523421595274, 51.94962058056517],
   [8.455710351546365, 1.372677996249965, 1.2575393768188565],
   [1804.6856277373693, 550.2770538747147, 311.9084682003741],
   [28.820648308489602, 3.74535599249993, 1.4948716593053935]],
#min. pocet paketov
[0, 17.55991723561792, 19.0],
#max. pocet paketov
[139.63150809642562, 21.44008276438208, 19.0],
#min. velkost toku
[0, 1268.3121551227077, 1835.59088440805],
#max. velkost toku
[167887.81154363783, 5228.687844877292, 2630.40911559195]]
```

Výpis B.7: Profil protokolu SMTPS.

Príloha C

Zdroje súborov PCAP

Súbory PCAP použité v rámci práce boli prevzaté z nasledujúcich zdrojov:

1. https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys?path=%2FUNSW-NB15%20-%20pcap%20files
2. https://digitalcorpora.org/corpora/scenarios/m57-patents-scenario
3. https://digitalcorpora.org/corpora/scenarios/nitroba-university-harassment-scenario
4. https://github.com/markofu/hackeire/tree/master/2011/pcap
5. https://github.com/elcabezzonn/Pcaps
6. https://drive.google.com/drive/folders/0B9TXiR9NkjmpOHNMRTl6VVA2RnM
7. https://iscxdownloads.cs.unb.ca/iscxdownloads/CIC-IDS-2017/PCAPs/
8. https://www.netresec.com/?page=PCAP4SICS
9. https://download.netresec.com/pcap/FIRST-2015/FIRST-2015_Hands-on_Network_Forensics_PCAP.zip
10. https://www.netresec.com/?page=MACCDC
11. https://www.stratosphereips.org/datasets-normal
12. http://tcpreplay.appneta.com/wiki/captures.html
13. https://westpoint.edu/centers-and-research/cyber-research-center/data-sets
14. https://www.pcapr.net/home