VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

METODY KLASIFIKACE SÍŤOVÉHO PROVOZU

METHODS FOR NETWORK TRAFFIC CLASSIFICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MICHAL JACKO**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MAROŠ BARABAS, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Jacko Michal, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Metody klasifikace síťového provozu**
Methods for Network Traffic Classification
Kategorie: Bezpečnost

Pokyny:

1. Prostudujte metody a nástroje (zejména IDS a ADS systémy) pro detekci anomálií v síťovém provozu a klasifikaci síťového toku. Diskutujte jejich výhody a nevýhody s ohledem na jejich úspěšnost detekce, náročnost na výpočetní prostředky a charakteristiku vstupních dat.
2. Dle analýzy navrhněte kombinaci metod pro klasifikaci síťového toku. Zaměřte se na využití platformy CUDA pro zpracování síťových dat. Navrhněte nástroj pro automatickou klasifikaci toku použitím zvolených metod.
3. Navržený nástroj implementujte a otestujte na poskytnutém testovacím vzorku dat. Porovnejte účinnost použitých metod s existujícími nástroji.
4. Vyhodnoťte kvalitu implementovaných metod, diskutujte možná rozšíření a další vývoj.

Literatura:

- Dle doporučení vedoucího

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Barabas Maroš, Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá problémom detekcie anomálií v sieťovej prevádzke a klasifikácie sieťových tokov. Na základe existujúcich metód popisuje práca návrh a implementáciu nástroja, pomocou ktorého je možné automaticky klasifikovať sieťové toky. Nástroj využíva platformu CUDA na spracovanie sieťových dát a výpočet metrík sieťových tokov pomocou grafickej karty. Spracované toky sú následne klasifikované navrhnutými metódami pre detekciu sieťových anomálii.

Abstract

This paper deals with a problem of detection of network traffic anomaly and classification of network flows. Based on existing methods, paper describes proposal and implementaion of a tool, which can automatically classify network flows. The tool uses CUDA platform for network data processing and computation of network flow metrics using graphics processing unit. Processed flows are subsequently classified by proposed methods for network anomaly detection.

Kľúčové slová

detekcia anomálií, IDS, bezpečnosť sietí, klasifikácia, CUDA, sieťové útoky

Keywords

anomaly detection, IDS, network security, classification, CUDA, network attacks

Citácia

JACKO, Michal. *Metody klasifikace síťového provozu.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Maroš Barabas, Ph.D.

Metody klasifikace síťového provozu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracovalsom samostatne pod vedením pána Ing. Maroša Barabasa, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Michal Jacko
24. mája 2017

Poďakovanie

Rád by som poďakoval vedúcemu práce Ing. Marošovi Barabasovi za vedenie tejto diplomovej práce a odbornú pomoc pri jej riešení.

Obsah

Kapitola 1

Úvod

Detekcia útokov na počítačové siete je v posledných rokoch často rozoberanou témou. Kvôli útokom sú často spôsobené veľké finančné škody v korporátnych alebo súkromných počítačových sietiach. Mnohé z týchto útokov nie je možné detekovať, pretože bezpečnostné systémy sú často založené na signatúrach známych typov útokov. Aktuálne detekčné systémy využívajú na detekciu nežiadaného správania rôzne metódy, niektoré hľadajú vzory charakteristické pre jednotlivé útoky, iné hľadajú odchýlky od normálnej prevádzky, prípadne využívajú hybridné riešenia kombinujúce rôzne postupy.

1.1 Obsah a ciele práce

Prvým z cieľov práce je štúdium existujúcich metód na detekciu anomálií v sieťových tokoch. Práca bude popisovať prehľad o metódach na detekciu útokov v počítačových sietiach a nástrojoch na klasifikáciu sieťovej prevádzky. Táto problematika je rozoberaná v kapitole 2. V podkapitole 2.6 je uvedený prehľad vybraných existujúcich nástrojov a výskumov na tému detekcie anomálii v sieťových tokoch s popisom použitých metód, charakteristiky vstupných dát a úspešnosti detekcie.

Kapitola 3 obsahuje popis platformy CUDA, ktorá umožňuje tvorbu a spustenie programu na vybraných grafických kartách. Z dôvodu hardwarového usporiadania grafického akcelerátora je použitie GPU vhodné na spracovanie sieťových tokov.

Hlavným zameraním práce je implementácia vlastného nástroja na automatické spracovanie sieťovej prevádzky a klasifikáciu sieťových tokov. V kapitole 4 je popísaný návrh tohto nástroja s popisom zvolených metód na klasifikáciu sieťových tokov. Nasledujúca kapitola 5 popisuje spôsob implementácie nástroja. Zdokumentovaný je spôsob spracovania sieťových tokov s vyuťitím platformy CUDA na výpočet navrhnutých metrík pre jednotlivé sieťové toky. Ďalej je popísaný spôsob uloženia spracovaných tokov pomocou open-source nástroja Elasticsearch a spôsob klasifikácie sieťových tokov pomocou niekoľkých detekčných metód (5.3).

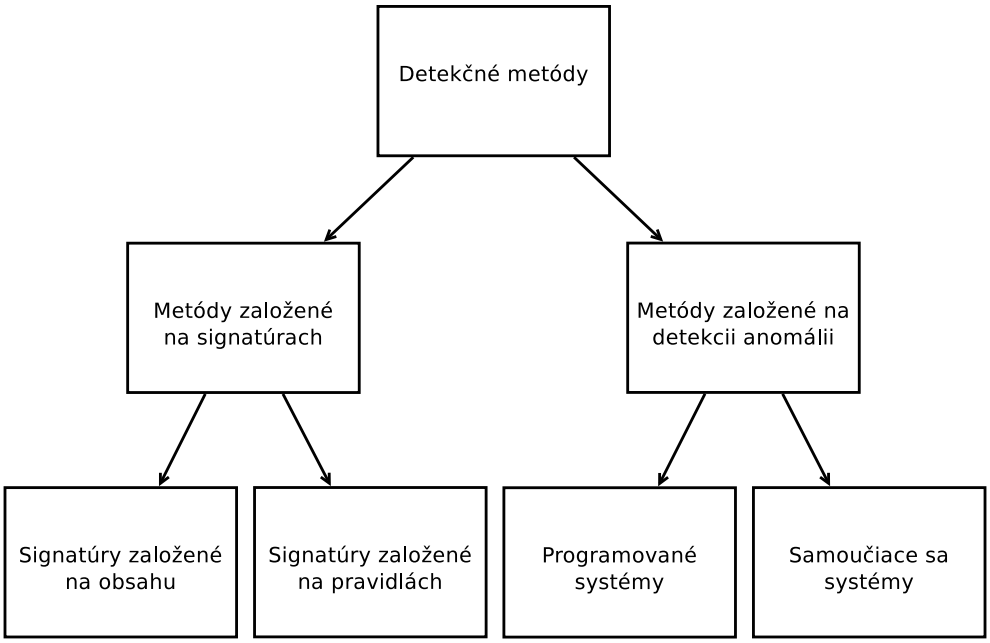
V kapitole 6 sú popísané výsledky testovania navrhnutého nástroja. Nachádza sa tu porovnanie implementácie spracovania sieťovej komunikácie pomocou platformy CUDA v porovnaní s implementáciou bez použitia GPU. Kapitola ďalej obsahuje vyhodnotenie úspešnosti klasifikácie sieťových tokov pomocou vytvorených detekčných metód.

Kapitola 2

Nástroje na detekciu útokov v počítačových sietiach

Táto kapitola popisuje existujúce metódy na detekciu útokov na počítačové siete, ich rozdelenie, výhody jednotlivých metód a spôsob, akým jednotlivé metódy fungujú.

Vo všeobecnosti môžeme metódy na detekciu sieťových útokov môžeme rozdeliť na dva základné typy – metódy založené na signatúrach a metódy založené na detekcii anomálií (ako je vidieť na obrázku 2.1).



Obr. 2.1: Rozdelenie detekčných metód

2.1 Metódy založené na signatúrach

Signatúra je vzor zodpovedajúci priebehu známeho útoku [19]. Systémy využívajúce túto metódu detekujú útoky pomocou porovnávania aktuálnej sieťovej prevádzky so signatúrami, ktoré sú uložené v databáze systému. Tieto signatúry sú vytvárané väčšinou manuálne na základe analýzy sieťovej komunikácie a vzorkov malware, avšak v posledných rokoch

dochádza k snahe o automatizáciu tohto procesu napríklad pomocou honeypot systémov [6].

Výhodou tejto metódy je veľká efektivita pri detekcii známych útokov a nízka miera false-positive. Naopak, nevýhodou týchto systémov je, že majú zlú účinnosť pri detekcii nových útokov, pretože doba od vzniku malware po vytvorenie a distribúciu signatúry môže trvať niekoľko dní.

Metódy založené na signatúrach sa rozdeľujú do 2 kategórii podľa spôsobu tvorby signatúr:

- **Signatúry založené na obsahu** - útoky sú identifikované na základe porovnávania hlavičiek a obsahu paketov. Príkladom je detekcia exploitu, ktorý je možné identifikovať špecifickou postupnosťou bytov.
- **Signatúry založené na pravidlách** - Systémy tohto typu modelujú útoky pomocou množiny pravidiel, ktoré sú potom porovnávané s aktuálnym dátovým tokom. Príklad takejto signatúry je detekcia komunikácie s IP adresou, ktorá sa nachádza na blackliste.

2.2 Metódy založené na detekcii anomálií

Hľadanie útokov pomocou týchto metód vychádza z predpokladu, že nežiadúca sieťová aktivita je vždy abnomálna. Na vytvorenie takýchto detektorov je potrebná definícia profilu normálnej aktivity sledovaním charakteristík bežnej dátovej prevádzky v určitom časovom období. Detekcia anomálii predstavuje štúdium hodnôt rôznych atribútov (napr. množstvo dát odoslaných užívateľom, počet neúspešných pokusov o prihlásenie a iné) a určenie hranice, kedy je možné ich považovať za normálne.

Hlavnou výhodou týchto metód oproti detekcii útokov na základe signatúr je schopnosť detekovať nové a neznáme útoky. Nevýhodou je však väčšie množstvo false-positive, pretože detekcia je závislá na presnosti určenia normálneho stavu systému.

Systémy hľadajúce útoky na základe detekcie anomálií je možné rozdeliť do dvoch základných kategórii - pragramovateľné a samoučiace sa systémy.

Programované systémy

Táto kategória systémov je programovo naučená detekovať anomálie. Predpoklady, čo je považované za abnormálne sú do systému vkladané programátorom, napríklad ak počet neúspešných pokusov o prihlásenie prekročí určitý počet, systém to bude považovať za porušenie bezpečnosti.

Samoučiace sa systémy

Tieto systémy vytvárajú model sieťovej prevádzky na základe jej dlhodobého pozorovania. Ak dôjde k vyhodnoteniu, že sa aktuálna prevádzka na sieti líši od naučeného modelu, systém je alarmovaný.

2.3 Hybridné metódy

Postupy kombinujúce detekciu založenú na signatúrach a detekciu anomálií sú označované ako hybridné. Tento spôsob spája výhody oboch prístupov, kde na odhalenie známych úto-

kov slúži detekcia pomocou signatúr a nové útoky sú odhaľované pomocou detekcie anomálii. Kombináciou oboch prístupov je možné udržať nízke množstvo false-positive poplachov pri zachovaní úspešnosti detekcie. Tento prístup je využitý v systéme popísanom v publikácii *A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic* [22].

2.4 Honeypoty

Ďalšou možnosťou detekcie sieťových útokov je využitie honeypot systémov [15]. Honeypot je informačný systém, ktorého účelom je pritiahnuť potenciálnych útočníkov (preto pomenovanie honeypot - medový hrniec).

Honeypoty sú používané hlavne pre včasné detekovanie útoku a následnú analýzu chovania. Tieto systémy bývajú často úmyselne zraniteľné, pôsobia tak ako vhodný cieľ pre potencionálny útok, bývajú však upravené tak, aby bol útok zaznamenaný v čo najvačšej miere pre ďalšiu analýzu. Dôležitá je nerozoznateľnosť nastraženého honeypotu od bežných systémov, pretože v prípade, ak útočník rozozná, že sa jedná o honeypot, jeho správanie sa mení.

Tieto systémy sú používané hlavne vo výskumnom prostredí ako nástroj na analýzu nových postupov útokov, zraniteľností a nových druhov malware.

Honeypoty je možné rozdeliť do dvoch skupín podľa miery interakcie s útočníkom:

- **Honeypoty s vysokou mierou interakcie** - zobrazujú kompletný reálny systém so všetkými službami a funkciami akoby sa nejednalo o honeypot. Ich účelom je čo najlepšia možnosť detekcie a analýzy chovania útočníka na systéme. Nevýhodou tohto spôsobu implementácie je možnosť napadnutia celého systému vrátane zdrojov honeypotu.

- **Honeypoty s nízkou mierou interakcie** - simulú iba obmedzenú funkcionalitu emulovanej služby alebo operačného systému. Ich účelom je detekcia pokusu o útok a následné zastavenie útočníka pomocou iných technológii (napríklad blokácia IP adresy útočníka na FW).

2.5 Dátové sady na testovanie účinnosti detekčných metód

V posledných rokoch vzniká množstvo výskumných prác na tému detekčných metód na identifikáciu podozrivých sieťových komunikácii. Vzniklo niekoľko dátových sád sieťových záznamov útokov a valídnej komunikácie, ktoré slúžia na testovanie účinnosti detekčných metód a porovnanie výsledkov týchto výskumných prác.

KDD Cup 99

V roku 1999 došlo k vzniku dátovej kolekcie KDD Cup [2] , ktorá je v súčastnosti jedna z najpoužívanejších kolekcii dát súvisiaca so sieťovými útokmi. Množina obsahuje takmer 5 miliónov vzorkov spojení, ktoré sú klasifikované do 24 tried predstavujúcich iný typ útoku. Každé spojenie v množine obsahuje 41 vyextrahovaných prvkov.

Útoky v množine je možné rozdeliť do 4 kategórií [20]:

- **Denial of Service** – útoky, pri ktorých dochádza k zahlteniu výpočetných prostriedkov stroja tak, že nie je schopný obsluhovať legitímne požiadavky, alebo odmieta prístup na stroj pre legitímnych používateľov (napr. syn flood).

- **Remote to local** – útočník s možnosťou komunikácie so strojom využije zraniteľnosť na prístup k lokálnemu účtu daného stroja (napr. hádanie hesla, vzdialený buffer overflow útok).
- **User to root** – zneužitie zraniteľnosti na eskaláciu privilégií (napr. rôzne lokálne buffer overflow útoky).
- **Probing** – pokusy o získanie informácii o sieti pre účel objavenia zraniteľností (napr. port scanning pre nájdenie zraniteľných služieb).

DARPA Intrusion Detection Evaluation

Dátová sada DARPA [1] bola vytvorená s účelom testovania IDS systémov. Bola vytvorená v roku 1999 a v roku 2000 rozšírená o ďlašie známe útoky. Používaná bola do roku 2015 a slúžila na porovnanie účinnosti rôznych detekčných metód.

Komunikácie v dátovej sade boli vytvorené pomocou neverejného software, nie je teda možnosť zistiť autenticitu a presnosť vygenerovaných sieťových tokov.

CDX 2009

Dátová sada CDX 2009 (Cyber Defense Excercise) [18] bola vytvorená počas súťaže tímov útočiacich na pripravenú infraštruktúru, na ktorej boli umiestnené zraniteľné systémy. Cieľom súťaže bolo vytvoriť popísanú sadu útokov zo záznamov TCP spojení a vzniknutých incidentov IDS systému SNORT [5].

2.6 Prehľad existujúcich metód

V nasledujúcej kapitole sú uvedené vybrané existujúce nástroje na detekciu anomálii v sieťových tokoch a metódy, ktoré nástroje používajú.

ADAM

Audit Data Analysis and Mining (ADAM) [7] bol navrhnutý a implementovaný ako systém na detekciu anomálií, ale využíva modul pre klasifikáciu podozrivých udalostí do nepravdivých alarmov alebo reálnych útokov. ADAM je výnimočný v dvoch smeroch. V prvom smere ADAM používa data mining pre budovanie upraviteľných profilov pravidiel normálneho správania a následne klasifikátor klasifikuje podozrivú aktivitu na reálny útok alebo nepravdivý alarm. V druhom smere je ADAM navrhnutý pre použítie v reálnom čase, čo je dosiahnuté využitím algoritmu pre inkrementálne dátové dolovanie, ktoré používa kĺzajúce časové okno pre nájdenie podozrivých udalostí.

ADAM používa kombináciu asociatívnych pravidiel, dátového dolovania a klasifikácie pre odhalenie útočníka v TCP dump [21] auditnom zázname. Najskôr si ADAM vytvorí zoznam normálnych frekventovaných setov položiek, ktoré udržuje počas bezútokových období na základe dolovania dát, ktoré sú voľné od útokov. Potom ADAM spustí kĺzajúce okno ako algoritmus v reálnom čase, ktorý hľadá frekventované sety položiek v posledných XY pripojení a porovnáva ich s normálnymi setmi položiek vo svojom zozname a zároveň vyraduje tie položky, ktoré považuje za normálne. Pre roztriedenie zbytku položiek, ADAM používa klasifikátor, ktorý bol predčasne vytrénovaný pre klasifikáciu podozrivých pripojení ako známy typ útoku, neznámy typ útoku alebo nepravdivý alarm.

Výkon ADAMa bol vyhodnotený na DARPA IDE datasete [1] a získal celkovú presnosť približne 45% pre detekcie DoS útokov a presnosť 28% pre sondové detekcie, avšak v prípade útokov *User to Root* a *Remote to Local* získal 0% presnosti.

Detekcia útokov s neoznačenými dátami pomocou zhlukovania

Autori príspevku *Intrusion Detection with Unlabeled Data Using Clustering* [16] prezentujú intrusion detection algoritmus, ktorý je založený na nedohliadanej detekcii anomálií. Tento algoritmus berie ako vstup sady neoznačených dát a pokúša sa nájsť prienik dát. Po detekcií prienikov je nad dátami aplikovaný tréning algoritmu na detekciu anomálií. Prístup instancií clusterov dát do spoločných clusterov s využitím jednoduchej vzdialenostnej metriky je referovaný ako single-linkage clustering. Tento clustering sa vykonáva nad neoznačenými dátami, vyžadujúc iba vlastnosti vektorov bez označení. Po tom ako sú dáta v spoločných clusteroch, všetky inštancie ktoré sa nachádzajú v malých clusteroch sú označené ako anomálie.

Tréning aj testovanie bolo uskutočnené s použitím rozličných podsietí z KDD Cup 99 datasetu [2]. Autori využili 10-fold cross validačnú metódu pre vyhodnotenie výkonnosti. V priemere bola senzitivita detekcie okolo 40% –55% s mierou false positive 1.3% –2.3%.

Využitie Bayessovských sietí

Autori príspevku *Bayesian event classification for intrusion detection* [12] prezentujú metódu pre vykonávanie Bayessovskej klasifikácie vstupných udalostí zameraných na detekciu narušení. V tejto metóde boli zlepšené naivné základné prahové hodnoty schém, tradične používané na kombináciu výstupov modelu s využívaním Bayesovych sietí. Tento prístup umožnuje autorom prirodzene včleniť dôveru modelu a závislosti medzi modelmi do klasifikačného procesu.

Vyhodnotenie tejto schémy bolo vykonané na *MIT Lincoln Labs 1999* dátovom sete, ktorý obsahoval záznam sieťového prenosu tak isto ako informáciu o sekvenciách systémových volaní. Experimentálne výsledky ukázali, že bolo dosiahnuté značné zníženie miery false positive. Po detekovaní všetkých útokov z testovacieho dátového setu, Bayessova klasifikácia hlásila iba polovicu false pozitívov ako tradičný prístup, založený na rovnakom modelovom výstupe.

Využitie neurónovej siete

Ibrahim so spoluautormi analyzuje výkon strojového učenia anomálií IDS systému, ktorý je založený na *Self Organization Map (SOM) Artificial Neural Network (ANN)*[9]. Systém používa SOM ANN pre detekciu a separáciu bežného sieťového prenosu od nežiadúceho. Navrhované IDS je validované na detekčnej schopnosti anomálií v KDD Cup 99 a NSL–KDD sadách dát.

Výsledky ukazujú, že SOM ANN dosiahol 92.37% presnosť s KDD Cup 99 a 75.49% presnosť v prípade NSL KDD CUP 99 sady dát.

Holistická ochrana siete

Ji Jenny s kolektívom prezentuje hybridnú sieťovú stratégiu na monitorovanie hostov [10], ktorá skúma dáta zo siete a hostov pre rozpoznanie malware infekcií. Zameriava sa na tri kategórie: normálne, skenujúce a infikované.

Prvá časť informácií od hosta je získaná extrahovaním 248 vlastností sieťového prenosu využívajúc verejne dostupný nástroj pre extrakciu A. Moora [13]. Ďalšia časť informácie je získaná z hosta použitím textového dolovania, pozerajúc sa na frekvenciu 500 najviac častých reťazcov a ich analýzov v podobe vektorov slov. Zlepšenia výkonu schopnosti detekcie sú dosiahnuté neustálým zlučovaním sieťových vlastností získaných z IP paketov sieťového toku a vlastnsťami hosta získaných z textového dolovania.

Presnosť hybridnej metódy vykonanej iba za pomoci klasifikácie hostov je 31.7% a klasifikácie iba za pomoci sieťového prenosu je 25%.

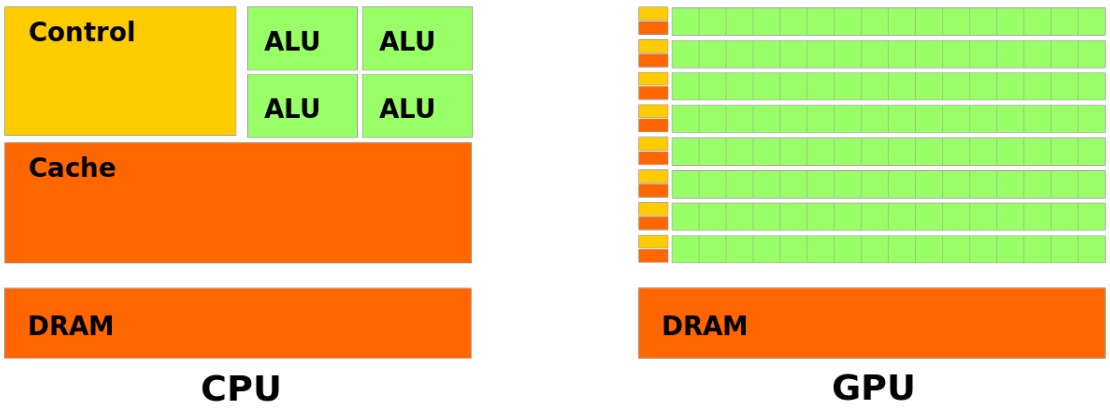
Kapitola 3

Platforma CUDA

CUDA (akronym z angl. Compute Unified Device Architecture) je hardwarová a softwarová architektúra, ktorá umožnuje na vybraných GPU spúšťať programy napísané v jazykoch C/C++, Fotran a iných. Použitie tejto architektúry je obmedzené iba na grafické akcelerátory spoločnosti nVIDIA, ktorá ich vyvinula. Nasledujúca kapitola čerpá informácie z oficiálnej dokumentácie CUDA SDK [14].

3.1 Mikroarchitektura GPU

Väčšinu plochy čipu grafického akcelerátoru od nVIDIA zaberá veľké množstvo relatívne jednoduchých skalárnych procesorov, ktoré sú organizované do väčších celkov zvaných streaming multiprocesory (SM). Jedná sa o architektúru typu SIMT (Single Instruction Multiple Threads), takže riadenie jednotiek a plánovanie inštrukcí je jednoduché a zaberá malé množstvo plochy GPU čipu.



Obr. 3.1: GPU má v porovnaní s CPU vačší podiel tranzistorov na spracovanie dát [14]

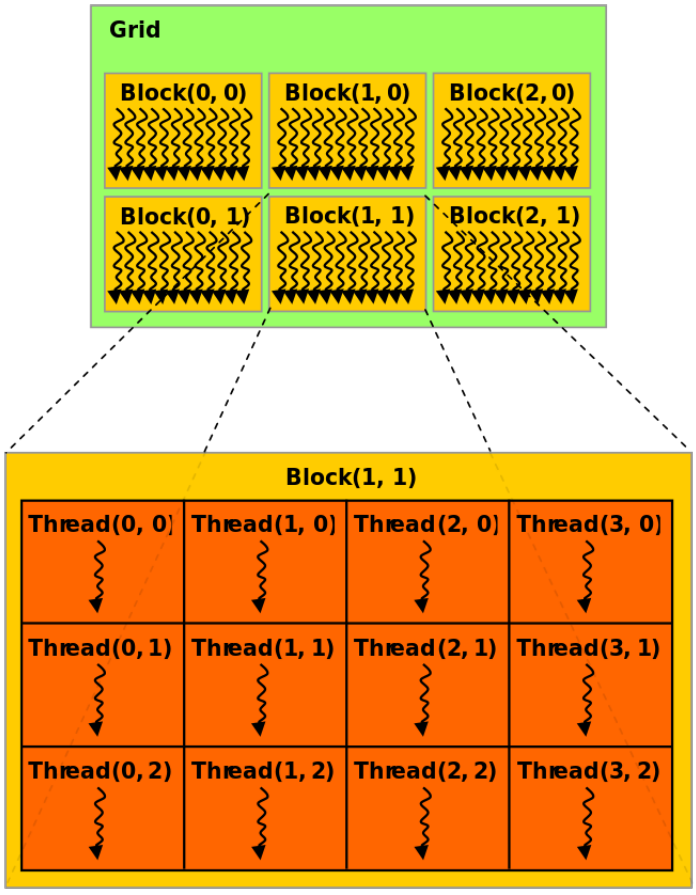
Z dôvodu usporiadania grafického akceleráta je použitie GPU vhodné na adresovanie problémov, ktoré môžu byť riešené pomocou paralelných výpočtov - rovnaký program je vykonávaný na mnohých dátových elementoch súčastne. Spracovanie sieťových paketov a maticové operácie pri aproximácii polynómov sú typickými príkladmi takýchto problémov.

3.2 Programovací model

CUDA aplikácia je zložená z častí, ktore bežia buď na host (CPU) alebo na CUDA zariadení (GPU). Časti aplikácie bežiace na GPU sú spúštané hostom zavolaním kernelu, čo je funkcia vykonávaná každým spusteným vláknom.

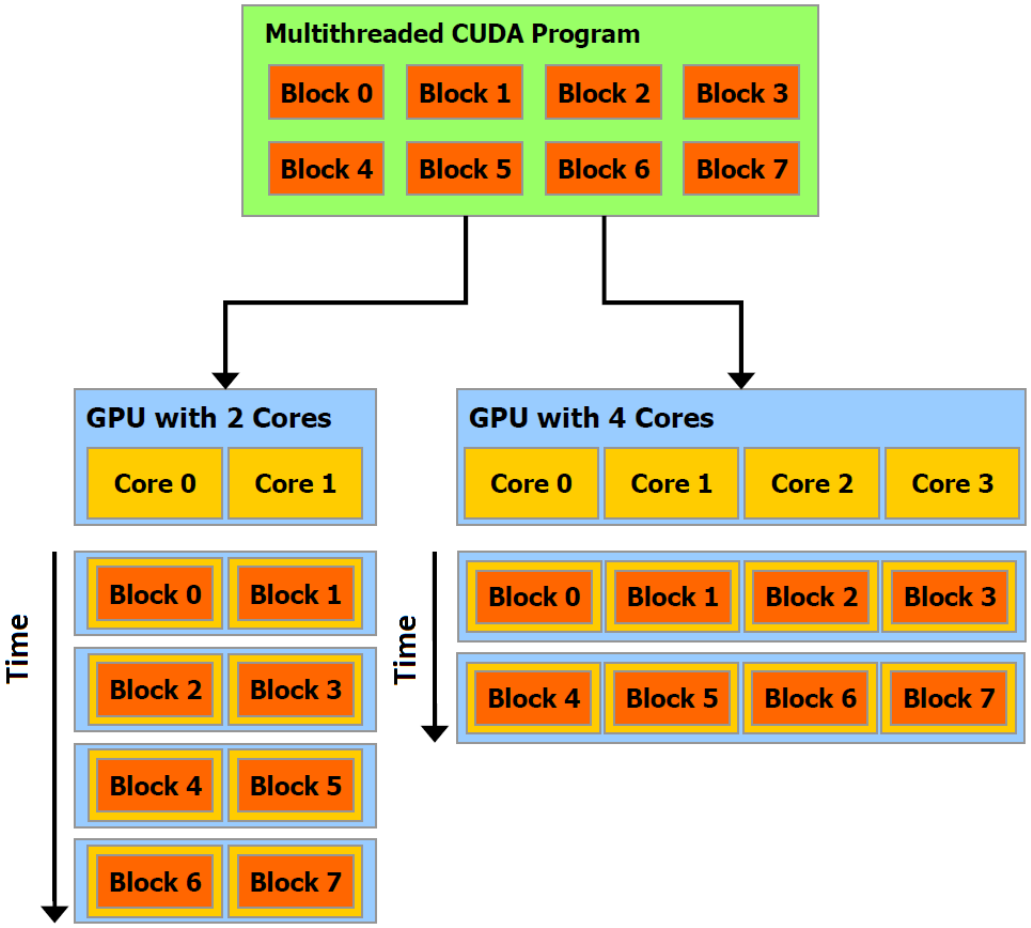
Vlákna sú organizované do 1- až 3-rozmerného bloku, v ktorom môžu zdieľať data a je možné synchronizovať ich beh. Počet vlákien na blok je závislý na výpočetných možnostiach daného zariadenia.

Bloky vláken sú organizované do mriežky, ktorá môže mať taktiež 1 až 3 rozmery.Každý blok musí byť schopný pracovať nezávisle na ostatných.



Obr. 3.2: Usporiadanie 2-rozmernej mriežka blokov vlákien [14]

Na jednom streaming multiprocesore beží vždy jeden blok vlákien. Rôzne druhy grafických kariet môžu obsahovať odlišné počty jednotiek SM, programovací model CUDA je preto automaticky škálovateľný, čo umožňuje spustenie rovnakého kódu na širokom spektre GPU. Jednotlivé bloky vláken sú na streaming multiproesoroch vykonávané sekvenčne za sebou, takže grafická karta s väčším množstvom SM vykoná program v kratšom čase, ako karta s menej SM (obrázok 3.3).



Obr. 3.3: Automatická škálovateľnosť [14]

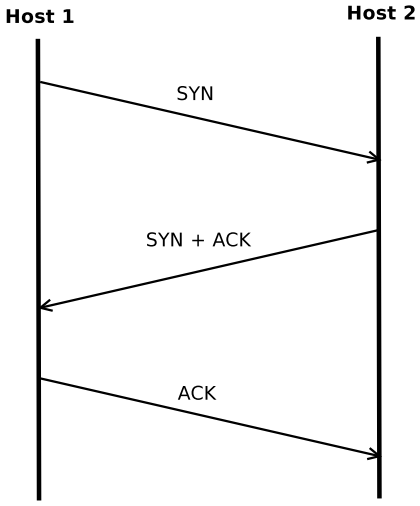
Kapitola 4

Návrh nástroja na klasifikáciu sieťového toku

Táto kapitola sa zaoberá návrhom nástroja, ktorý je schopný klasifikovať rôzne druhy útokov v sieťovej prevádzke.

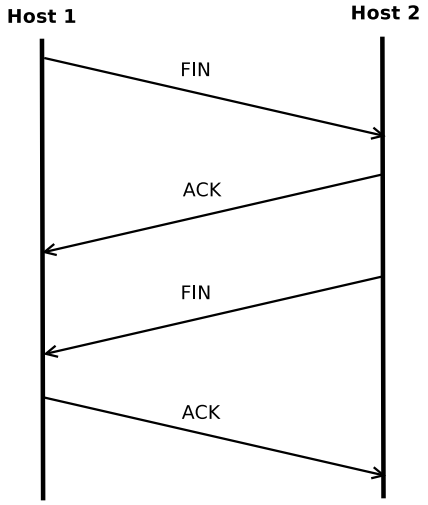
4.1 Spracovanie sieťových tokov

Nástroj na klasifikáciu sieťových tokov bude pracovať s paketmi využívajúcimi protokol TCP. Pri nadviazaní TCP spojenia musí vždy dôjsť k 3-way handshake (4.1) a spojenie prebieha na konštantných IP adresách a portoch, takže nie je problém s priradením paketov k jednotlivým tokom.



Obr. 4.1: TCP handshake

Ukončenie TCP spojenia je možné detekovať po prebehnutí štandardnej procedúry ukončenia spojenia zobrazenej na obrázku 4.2, V prípade, ak procedúra neprebehla alebo ak spojenie trvá dlhú dobu, je potrebné určiť veľkosť časového okna, v rámci ktorého budú pakety patriť k danému spojeniu. Po uplynutí tejto doby bude tok označený ako ukončený a je možné nad ním vykonať analýzu. Špecifikácia protokolu TCP [17] definuje štandardný interval 5 minút, po ktorom bude spojenie zatvorené ak nedôjde k prijatiu paketu.



Obr. 4.2: Ukončenie TCP spojenia

4.2 Metriky sieťových tokov

Hodnoty atribútov paketov protokolu TCP/IP nie sú vhodné na ďalšiu analýzu, je preto potrebná definícia metrík, podľa ktorých bude prebiehať klasifikícia sieťových spojení.

Definícia 4.2.1. Metrika je funkcia, ktorej vstupom je číslo alebo vektor a výstupom je číslo alebo postupnosť čísiel konštantnej dĺžky.

Atribúty toku, ktoré majú v priebehu komunikácie konštantnú hodnotu je možné použiť priamo ako metriky pre jednotlivé klasifikátory. Pre atribúty, ktoré môžu mať v každom pakete spojenia odlišnú hodnotu (sú určené vektorom hodnôt), je potrebné zvoliť spôsob transformácie na metriky.

Štatistické metriky

Pre každý vektor hodnôt je možné spočítať štatistické hodnoty:

- maximum - najväčšia hodnota z vektoru hodnôt
- minimum - najmenšia hodnota z vektoru hodnôt
- medián - medián z hodnôt vektoru
- priemerná hodnota - hodnota vypočítaná z hodnôt vektoru podľa vzorca 4.1.

$$average = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{4.1}$$

- štandardná odchýlka- hodnota vypočítaná z hodnôt vektoru podľa vzorca 4.2.

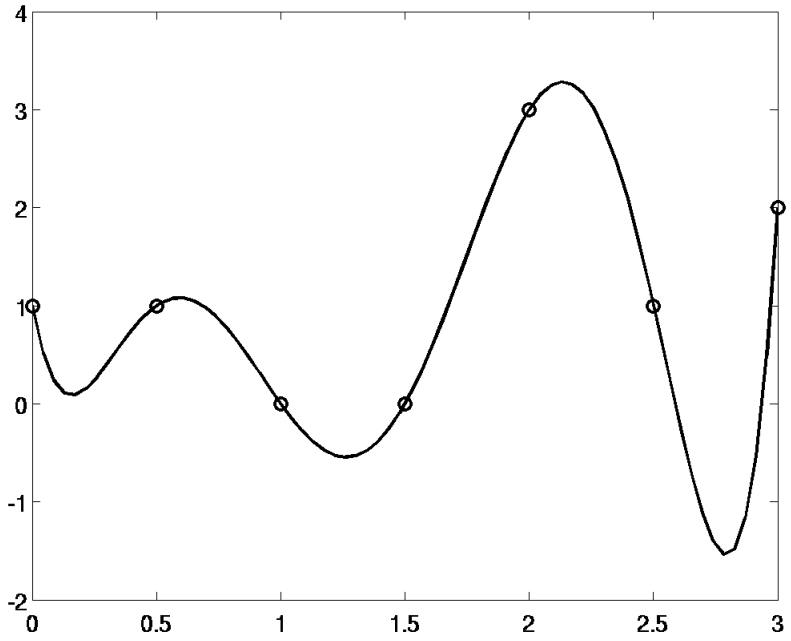
$$std = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - average)^2} \tag{4.2}$$

- počet zložiek - pre parameter príznakov flags je možné spočítať pre každý príznakový bit počet jeho výskytov, čím sú vytvorené ďalšie metriky.

Aproximácia pomocou polynómov

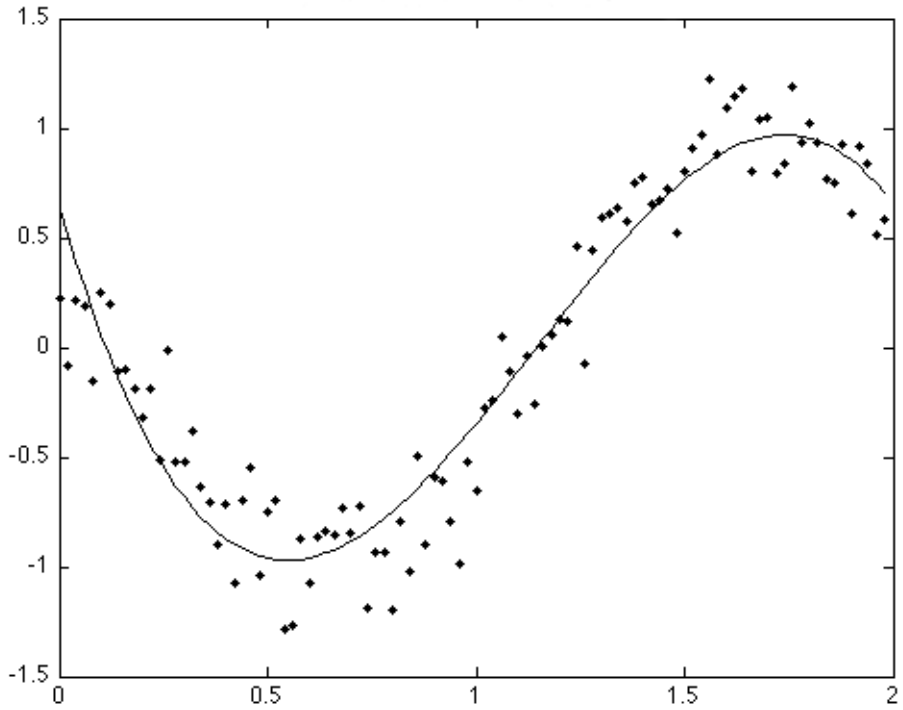
Vektory atribútov paketov sieťového toku je možné transformovať na koeficienty polynómov. Existujú 2 spôsoby transformácie vektoru na polynóm - interpolácia hodnôt alebo aproximácia.

V prípade interpolácie výsledný polynóm prechádza zadanými bodmi (ako je zobrazené na obrázku 4.3). Výstupný počet koeficientov je závislý na veľkosti vstupného vektora (definícia metriky 4.2.1 požaduje postupnosť konštantnej dĺžky), táto metóda teda nie je vhodná.



Obr. 4.3: Interpolácia hodnôt polynómom

Pri aproximácii výsledná funkcia nemusí prechádzať vstupnými bodmi (obrázok 4.4), musí ale spĺňať určité špecifické vlastnosti takéhoto priebehu. Jednou z metód aproximácie je metóda najmenších štvorcov, pri ktorej má výsledná funkcia minimálnu sumu štvorcov vzdialeností od daných bodov. Pri n vstupných bodoch $(x_i, y_i), i \in 1...n$ a stupni polynómu k je možné vypočítať výsledné koeficienty $a_0...a_k$ vyriešením sústavy rovníc 4.3 zapísanej v maticovom tvare 4.4.



Obr. 4.4: Polynomická aproximácia 3. stupňa

$$
\begin{aligned}
a_0 n + a_1 \sum_{i=1}^{n} x_i + \ldots + a_k \sum_{i=1}^{n} x_i^k &= \sum_{i=1}^{n} y_i \\
a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \ldots + a_k \sum_{i=1}^{n} x_i^{k+1} &= \sum_{i=1}^{n} x_i y_i \\
&\vdots \\
a_0 \sum_{i=1}^{n} x_i^k + a_1 \sum_{i=1}^{n} x_i^{k+1} + \ldots + a_k \sum_{i=1}^{n} x_i^{2k} &= \sum_{i=1}^{n} x_i^k y_i
\end{aligned}
\tag{4.3}
$$

$$
\begin{bmatrix}
n & \sum_{i=1}^{n} x_i & \ldots & \sum_{i=1}^{n} x_i^k \\
\sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \ldots & \sum_{i=1}^{n} x_i^{k+1} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \ldots & \sum_{i=1}^{n} x_i^{2k}
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix}
=
\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^k y_i \end{bmatrix}
\tag{4.4}
$$

Problémom pri analýze môže byť voľba použitého stupňa polynómu. Na výpočet aproximácie polynómom n-tého musí mať spojenie minimálne *n+1* paketov.

Diskrétna Fourierova transformácia

Ďalším spôsobom transformácie vstupných hodnôt je využitie Diskrétnej Fourierovej transformácie [24]. Pomocou DFT je možné transformovať vektor hodnôt na postupnosť koeficientov konečnej kombinácie komplexných exponenciál zoradených podľa ich frekvencií. Jednotlivé hodnoty koeficientov je možné získať pomocou rovnice 4.5, kde *X(k)* je k-ty koeficient Fourrierovej transformácie a N je počet vstupných atribútov funkcie x.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i k n / N}, \qquad k = 0 \dots N-1 \tag{4.5}$$

Výstupné koeficienty sú komplexné čísla, preto treba porovnávať okrem ich reálnej hodnoty aj hodnotu imaginárnu. Vyššie koeficienty patria ku komplexným exponenciálam s vysokými frekvenciami, preto vo výsledku nemajú veľký vplyv a je možné ich zanedbať.

4.3 Klasifikačné nástroje

Na klasifikáciu sieťovej komunikácie je možné použiť rôzne algoritmy a matematické metódy. Na experimentovanie s vypočítanými hodnotami metrík bolo zvolených niekoľko klasifikačných algoritmov.

Naivný Bayesovský klasifikátor

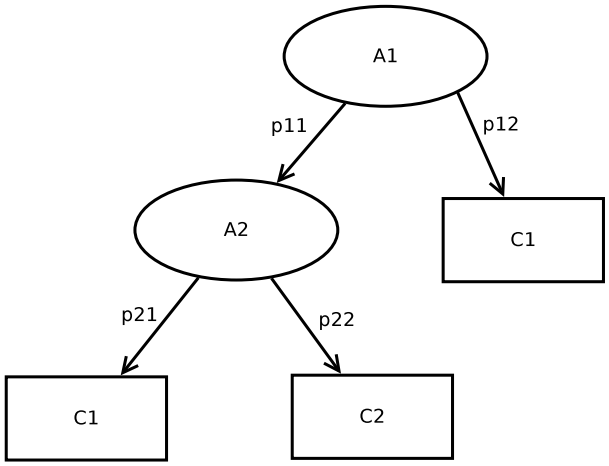
Bayesovské klasifikátory sú štatistické klasifikátory, ktoré predikujú pravdepodobnosti, s ktorými daný príklad patrí do tej – ktorej triedy. Vychádzajú pritom z určenia podmienených pravdepodobností jednotlivých hodnôt atribútov pre rôzne triedy. Naivný Bayesovský klasifikátor vychádza z predpokladu nezávislosti atribútov medzi sebou. To znamená, že efekt, ktorý má hodnota každého atribútu na danú triedu, nie je ovplyvnený hodnotami ostatných atribútov. Kvôli tomuto zjednodušeniu je tento klasifikátor nazývaný ako „naivný“ [11].

Rozhodovací strom

Rozhodovací strom [23] je klasifikátor so stromovou štruktúrou. Na zatriedenie príkladu začne v koreni stromu a prechádza cez jednotlivé uzly až k listu, ktorý poskytuje klasifikáciu inštancie. Dôležitým kritériom v algoritme rozhodovacieho stromu je výber atribútu na testovanie v každom rozhodovacom uzle stromu.

Pre rozhodovací strom, ktorý klasifikuje vstupné atribúty $A_1 \dots A_m, m \geq 1$ do tried $C_1 \dots A_n, n \geq 2$ platí:

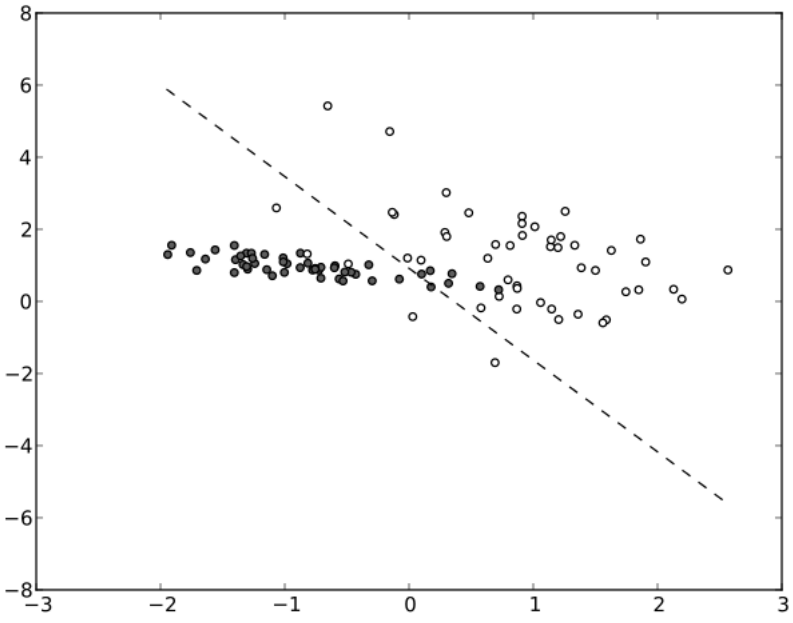
- každý uzol stromu (nie list) je označený niektorým z atribútov A_i
- každý list je označený niektorou z tried C_i
- každá hrana je označená predikátom aplikovateľným na atribút rodičovského uzlu



Obr. 4.5: Príklad rozhodovacieho stromu

Support Vector Machine

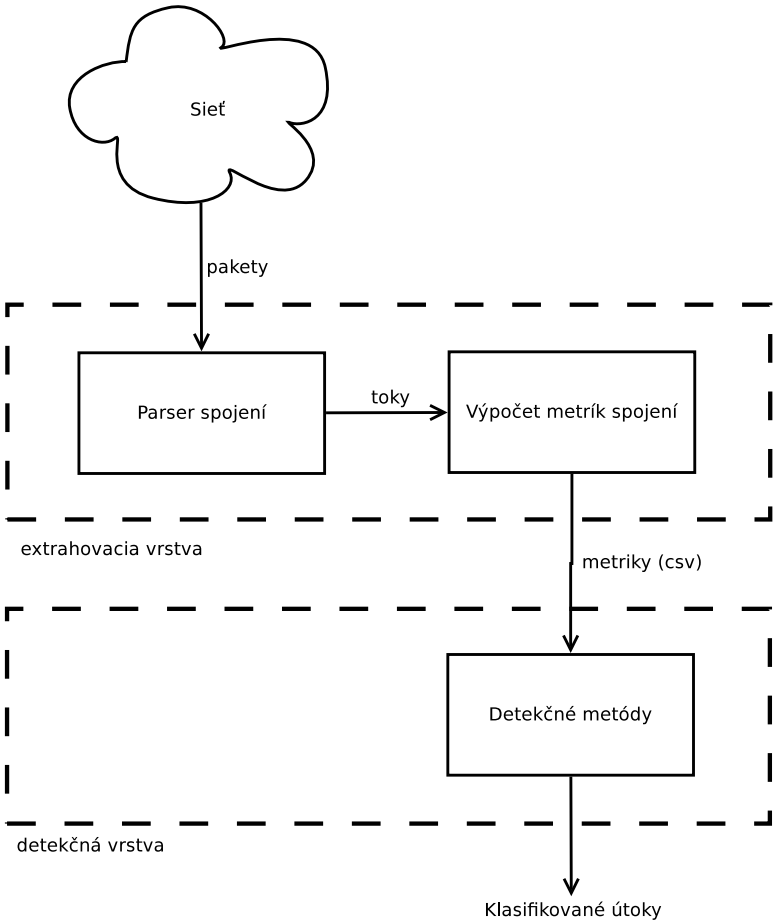
SVM [8] je optimalizačná metóda slúžiaca na analýzu a binárnu klasifikáciu rozličných typov dát. Binárna klasifikácia znamená, že dáta sú triedené do dvoch skupín alebo kategórii, pričom každá skupina obsahuje dáta s rovnakými vlastnosťami. Na začiatku je daný set tzv. tréningových vzoriek, z ktorého každá vzorka prináleží do jednej z dvoch skupín. Tréningový algoritmus použitím týchto vzoriek vytvorí separátor, ktorý v priestore oddelí vzorky tak, že jedna skupina sa nachádza na jednej jeho strane, druhá na druhej. Klasifikačný algoritmus potom každú novú vzorku zaradí do jednej zo skupín, na jednu alebo druhú stranu separátora.



Obr. 4.6: Zobrazenie separátora oddeľujúceho vzorky do dvoch tried

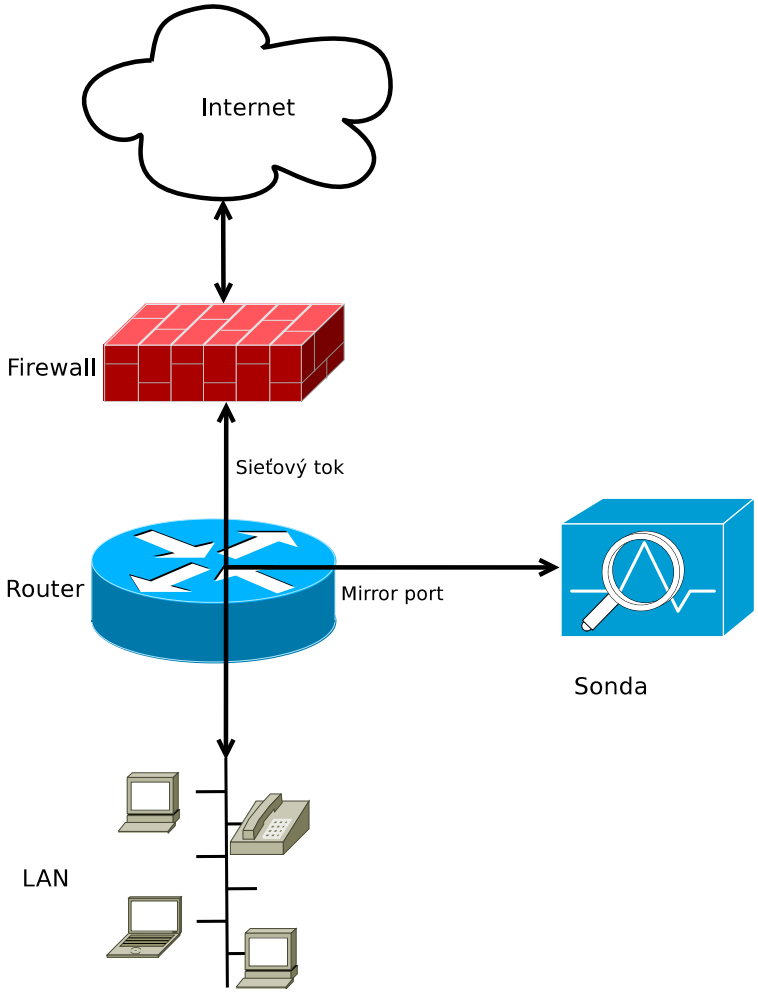
4.4 Architektúra systému

Navrhovaný nástroj je zložený z niekoľkých modulov. Vstupom sú pakety sieťovej prevádzky, ktorú chceme analyzovať. Pakety sú rozparsované do jednotlivých tokov, z ktorých sú následne vypočítané zvolené metriky. Výstupné dáta vo formáte tvoria vstup do detekčného modulu, ktorý v sieťových spojeniach vyhľadáva anomálie s využitím klasifikačných metód. Celý tento proces je zobrazený na obrázku 4.7.



Obr. 4.7: Architektúra systému

Vstupom systému je záznam sieťovej komunikácie, ktorý je predmetom analýzy. Tento záznam je možné získať zo sieťových zariadení pomocou zrkadliaceho portu (obrázok 4.8), ktorý kopíruje toky na vstupe sieťového zariadenia do nakonfigurovaného portu, čím sa zrkadlí všetok tok do výstupného portu (tzv. mirror port). Sieťovú komunikáciu je taktiež možné zachytávať aj na koncovom zariadení.



Obr. 4.8: Príklad odchytávania komunikácie z internej siete do Internetu pomocou mirror portu

Kapitola 5

Implementácia nástroja

Táto kapitola popisuje spôsob implementácie nástroja na klasifikáciu sieťových tokov.

5.1 Spracovanie sieťových tokov

Na spracovanie sieťových tokov bola použitá knižnica Libpcap [3]. Program dokáže čítať dáta dvoma spôsobmi - buď je možné načítať vstupné pakety priamo zo sieťového rozhrania, alebo je vstup načítaný zo súboru vo formáte pcap.

Na uloženie tokov je využitý štandardný kontajner *unordered_map* využívajúci hashovaciu tabuľku. Zloženým kľúčom tabuľky je štvorica obsahujúca položky:

- srcIp – zdrojová IP adresa toku
- dstIp – cieľová IP adresa toku
- srcPort – zdrojový TCP port
- dstPort – cieľový TCP port

Hodnoty v tabuľke sú reprezentované triedou *Flow*, ktorá obsahuje potrebné informácie o danom toku. Zo vstupných dát sú ku každému toku uložené vektory nasledujúcich hodnôt:

- ihls - hodnoty Internet Header Length
- lengths - veľkosti paketov
- ttls - hodnoty TTL
- seqs - sekvenčné čísla paketov
- acks - čísla potvrdených paketov
- winsizes - hodnoty veľkosti okna
- delays - časy medzi jednotlivými paketmi toku

Pri načítaní vstupných dát zo súboru sú toky najskôr uložené do tabuľky, následne bežia výpočty nad uloženými hodnotami. V prípade spracovania tokov zo sieťovej karty beží program v dvoch vláknach - jedno má na starosti ukladanie tokov do tabuľky, druhé vykonáva výpočty nad ukončenými tokmi. Toky sú označené ako ukončené buď v prípade, ak správne prebehlo ukončenie TCP spojenia alebo ak k toku nepribudne žiadny paket po dobu 5 minút.

5.2 Výpočet metrík zo sieťových tokov

Po uložení ukončeného toku sa spočítajú hodnoty jednotlivých metrík daného toku. Pre aproximáciu som zvolil polynómy 3. a 7. stupňa. V prípade menšieho počtu paketov spojenia ako je stupeň aproximačného polynómu sú chýbajúce hodnoty doplnené nulami.

Bola zvolená nasledujúca množina metrík:

- **cntPkt** - počet paketov toku
- **maxLength** - maximum dĺžok paketov
- **minLength** - minimum dĺžok paketov
- **avgLength** - priemer dĺžok paketov
- **stdLength** - štandardná odchýlka dĺžok paketov
- **approx3Length**[**4**] - keoficienty aproximácie 3. rádu dĺžok paketov
- **approx7Length**[**8**] - keoficienty aproximácie 7. rádu dĺžok paketov
- **sumLength** - celková veľkosť toku
- **maxIhl** - maximum hodnôt IHL
- **minIhl** - minimum hodnôt IHL
- **avgIhl** - priemer hodnôt IHL
- **stdIhl** - štandardná odchýlka hodnôt IHL
- **approx3Ihl**[**4**] - keoficienty aproximácie 3. rádu hodnôt IHL
- **approx7Ihl**[**8**] - keoficienty aproximácie 7. rádu hodnôt IHL
- **maxTtl** - maximum hodnôt IHL
- **minTtl** - minimum hodnôt IHL
- **avgTtl** - priemer hodnôt IHL
- **stdTtl** - štandardná odchýlka
- **approx3Ttl**[**4**] - keoficienty aproximácie 3. rádu hodnôt TTL
- **approx7Ttl**[**8**] - keoficienty aproximácie 7. rádu hodnôt TTL
- **maxSeq** - maximum hodnôt sekvenčných čísel
- **minSeq** - minimum hodnôt sekvenčných čísel
- **avgSeq** - priemer hodnôt sekvenčných čísel
- **stdSeq** - štandardná odchýlka hodnôt sekvenčných čísel
- **approx3Seq**[**4**] - keoficienty aproximácie 3. rádu hodnôt sekvenčných čísel
- **approx7Seq**[**8**] - keoficienty aproximácie 7. rádu hodnôt sekvenčných čísel

- **maxAck** - maximum hodnôt potvrdených paketov
- **minAck** - minimum hodnôt potvrdených paketov
- **avgAck** - priemer hodnôt potvrdených paketov
- **stdAck** - štandardná odchýlka
- **approx3Ack**[**4**] - keoficienty aproximácie 3. rádu hodnôt potvrdených paketov
- **approx7Ack**[**8**] - keoficienty aproximácie 7. rádu hodnôt potvrdených paketov
- **maxWinsize** - maximum hodnôt veľkosti okna
- **minWinsize** - minimum hodnôt veľkosti okna
- **avgWinsize** - priemer hodnôt veľkosti okna
- **stdWinsize** - štandardná odchýlka hodnôt veľkosti okna
- **approx3Winsize**[**4**] - keoficienty aproximácie 3. rádu hodnôt veľkosti okna
- **approx7Winsize**[**8**] - keoficienty aproximácie 7. rádu hodnôt veľkosti okna
- **maxDelay** - maximum hodnôt zdržania paketov
- **minDelay** - minimum hodnôt zdržania paketov
- **avgDelay** - priemer hodnôt zdržania paketov
- **stdDelay** - štandardná odchýlka
- **approx7Delay**[**4**] - keoficienty aproximácie 3. rádu hodnôt zdržania paketov
- **approx7Delay**[**8**] - keoficienty aproximácie 7. rádu hodnôt zdržania paketov
- **sumDelay** - celkové trvanie toku

Koeficienty aproximácie polynómov sú získané pomocou metódy najmenších štvorcov 4.3. Na výpočet koeficientov z matice je použitá Gaussova eliminačná metóda.

Využitie platformy CUDA

Na výpočet metrík zo sieťových tokov bola využitá platforma CUDA, pomocou ktorej je možné presunúť výpočty na grafickú kartu a dobre ich tak paralelizovať.

Architektúra každej grafickej karty je odlišná, preto je potrebné kód optimalizovať na konkrétny hardware, na ktorom bude aplikácia bežať. Základný výpočetný kernel je spustený v lineárnej mriežke blokov, ktorá má veľkosť rovnakú, ako je počet analyzovaných tokov. Jeden blok vláken počíta metriky pre jeden sieťový tok, jednotlivé bloky sú teda medzi sebou nezávislé. Testovaná architektúra má 6 streamovacích multiprocesorov so 192 výpočetnými jadrami na každom SM, bola preto zvolená mriežka vlákien na blok o rozmeroch 12 x 16 vlakien. Výpočet rovníc 4.3 Gaussovou elimináciou prebieha paralelne s uložením matice v zdieľanej pamäti bloku vláken.

Najdlhšie trvajúcou časťou výpočtu, ktorá vyžadovala najviac optimalizácii pre urýchlenie je výpočet samotných prvkov matice 4.4. Pakety jednotlivých tokov sú vždy číslované

celými číslami vzostupne začínajúc hodoutou 1, prvky matice sústavy sú preto konštantné pre všetky metriky využívajúce aproximáciu. Tieto prvky sú predpočítané a pred výpočtom samotnej Gaussovej eliminácie sú dopočítané iba prvky rozšírenej matice, ktoré obsahujú sumy mocnín vstupných hodnôt. Sumy mocnín sú počítané paralelne 16-timi vláknami príslušného riadku bloku vláken.

5.3 Detekčné metódy

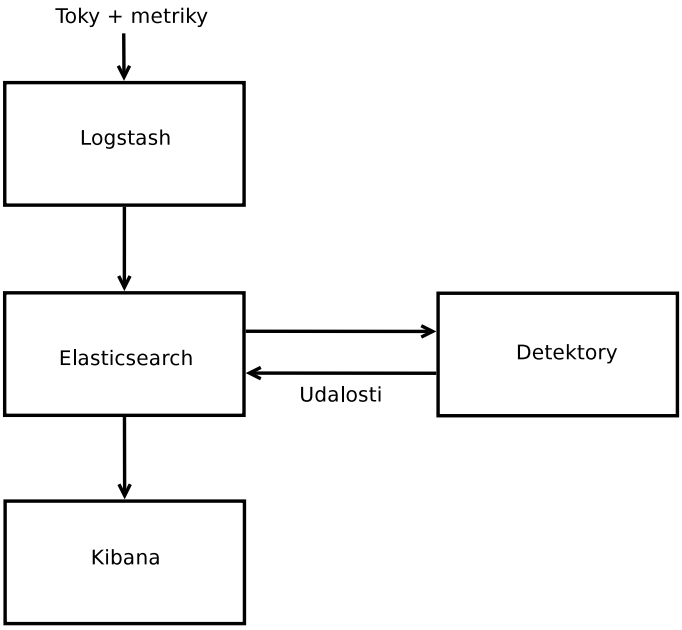
Na uloženie a analýzu tokov sú použité open-source nástroje Logstash, Elasticsearch a Kibana, ktoré vyvíja skupina Elastic [4].

Logstash je aplikácia na zber, spracovanie a preposielanie udalostí a logov. Pomocou tohto nástroja je možné zbierať dáta v mnoho rôznych formátoch, pomocou filtrov ich upravovať a následne poslať výstup do množstva externých nástrojov vrátane Elasticsearch.

Elasticsearch je bezschémová databáza a fulltextový vyhľadávač. Disponuje RESTful rozhraním, pomocou ktorého môže byť každá akcia vykonaná použitím dokumentu vo formáte JSON, ktorý je zasielaný cez HTTP.

Kibana je webové grafické užívateľské rozhranie, ktoré umožňuje vizualizovať data z Elasticsearch.

K nástrojom je naprogramovaná aplikácia v programovacom jazyku Java, ktorá pomocou detekčných metód v uložených tokoch vyhľadáva udalosti. Schéma procesu je zobrazená na obrázku 5.1.



Obr. 5.1: Schéma komunikácie detekčných metód a balíka ELK

Boli implementované 4 analyzátory - portscan detektor, detektor periodickej komunikácie, detektor blacklistovaných spojení a multicast detektor.

Analyzátory sú spúštané pravidelne každých 60 sekúnd. Základom je trieda *Analyser*, z ktorej dedia všetky odvodené triedy jednotlivých detektorov. Spôsob volania je je pre všetky detektory rovnaký - najskôr sa zostaví dotaz pre Elasticserach vo formáte JSON, ktorým sa z databázy vyberú potrebné dáta. Nad získanými dátami prebehne analýza

a v prípade pozitívnej klasifikácie incidentu je do databázy vložený záznam označujúci príslušnú udalosť.

Portscan detektor

Detektor v sieťových tokoch vyhľadáva horizontálny portscan. Skenovanie portov je metóda zisťovania, či sú na vzdialenom počítači v sieti otvorené porty, čím je možné zistiť aké služby sú na počítači spustené. Pod pojmom horizontálny portscan sa rozumie skenovanie jedného špecifického portu naprieč viacerými stanicami.

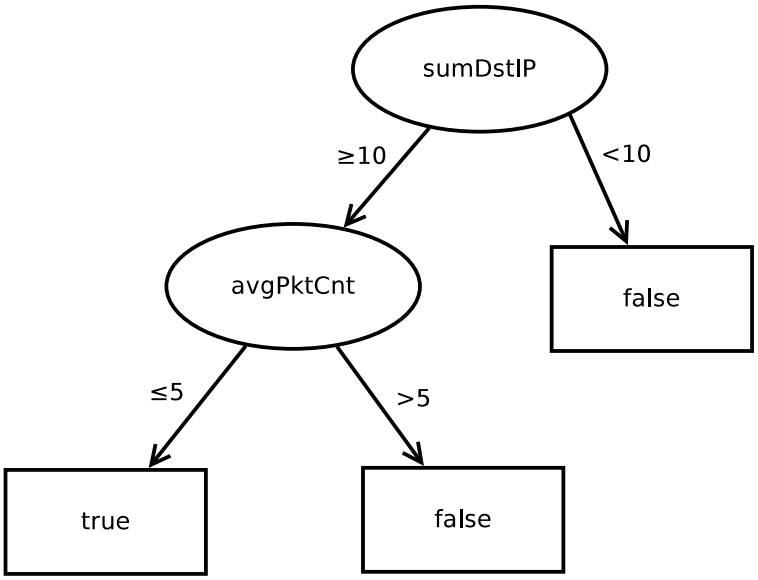
Analyzátor z databázy vyberie kombinácie všetkých zdrojových adries a cieľových TCP portov v sieťových tokoch, ktoré dobehli počas analyzovaného intervalu. Dotaz je zobrazený v úseku kódu 5.1. Ak pre dvojicu (zdrojová IP adresa a cieľový port) počet cieľových IP adries prekročí daný limit a priemerný počet paketov daných spojení je menší ako určitá hodnota, toky sú klasifikované ako portscan a je ohlásená udalosť.

```
{ "query": {
    "bool": {
      "must": [
        {"query_string": {"query": "*", "analyze_wildcard": true } },
        {"range": { "startTime": { "gte": 1495105740, "lte": 1495105800, "
            format": "epoch_millis"}}}
      ]
    }
  },
  "size": 0,
  "aggs": {
    "ip_addr": {
      "terms": { "field": "srcIpAddr" },
      "aggs": {
        "dst_port": {
          "terms": {"field": "dstPort" },
          "aggs": {
            "scanned_hosts": { "cardinality": {"field": "dstIpAddr"} },
            "sum_pkt": { "sum": {"field": "cntPkt" } },
            "avg_pkt": {
              "bucket_script": {
                  "buckets_path": {"pkts": "sum_pkt", "hosts": "
                      scanned_hosts" }, "script": "params.pkts/params.hosts
                      "}
            },
            "max_avg": {
              "bucket_selector": { "buckets_path": { "pkts": "avg_pkt" }, "
                  script": "params.pkts < 5" }
            },
            "min_hosts": {
              "bucket_selector": { "buckets_path": { "hosts": "scanned_hosts
                  " }, "script": "params.hosts > 10" }
          } } },
        "no_empty": {
          "bucket_selector": { "buckets_path": { "cnt": "dst_port.
              _bucket_count" }, "script": "params.cnt > 0" }
  } } } } }
```

Zdrojový kód 5.1: Elasticsearch dotaz na získanie dát pre portscan detektor

Experimentami bola zvolená hodnota 10 pre minimálny počet rôznych cieľových IP adries a hodnota 5 pre maximálny priemerný počet paketov spojení. V prípade voľby voľnejších limitov mal detektor množstvo false positive klasifikácii štanderdnej sieťovej prevádzky.

Spôsob klasifikácie detektorom je možné popísať pomocou rozhodovacieho stromu zobrazeného na obrázku 5.2.



Obr. 5.2: Rozhodovací strom popisujúci portscan detektor

Detektor periodickej komunikácie

Analyzátor pri každom spustení uloží do štruktúry počty výskytov jednotlivých komunikácii. Pod komunikáciou sa rozumie trojica, zdrojová IP adresa, cieľová IP adresa a cieľový port. Dotaz na získanie dát z Elasticsearch je uvedený v časti kódu 5.2.

```
{
  "query": {
    "bool": {
      "must": [
        { "query_string": { "query": "*", "analyze_wildcard": true } },
        {
          "range": {
            "startTime": {"gte": 1495105740,"lte": 1495105800,"format": "
                epoch_millis"}
          }
        }
      ]
    }
  },
  "size": 0,
  "aggs": {
    "src_addr": {
      "terms": { "field": "srcIpAddr" },
      "aggs": {
        "dst_addr": {
          "terms": { "field": "dstIpAddr" },
          "aggs": {
            "dst_port": {
```

```
                "terms": { "field": "idstPort" },
                "aggs": {
                  "sum_pkt": { "sum": { "field": "cntPkt"} }
} } } } } } } }
```

Zdrojový kód 5.2: Elasticsearch dotaz na získanie dát pre detektor periodických spojení

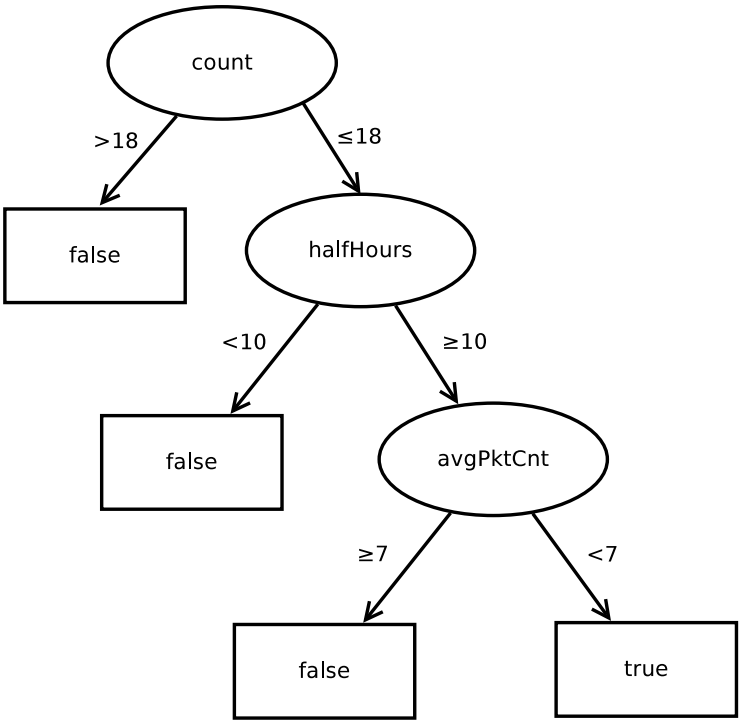
Na uloženie komunikácií je použitá trieda *org.apache.commons.collections.map.MultiKeyMap*. Použitým zloženým kľúčom tejto triedy je trojica zdrojová IP adresa, cieľová IP adresa a cieľový port, hodnotou je trieda *FlowAggregation*, ktorá reprezentuje výskyty jednotlivých komunikácii v neprekrývajúcich sa polhodinových intervaloch. Ide o hashovaciu tabuľku, ktorej kľúčom je identifikácia intervalu a hodnotou je trieda *FlowStats*, ktorá reprezentuje štatistiky danej komunikácie vo zvolenom intervale.

Raz za hodinu je vykonaná funkcia, ktorá v uložených komunikáciách hľadá periodicky sa opakujúci výskyt danej komunikácie za posledných 6 hodín. Počas behu funkcie sú taktiež mazané uložené štatistiky v starších intervaloch. Pre pozitívnu klasifikáciu komunikácie musí spĺňať nasledujúce parametre, ktoré boli zvolené na základe experimentov:

- počet výskytov za posledných 6 hodín musí byť menší ako 18
- musí sa vyskytovať minimálne v 10 neprekrývajúcich sa polhodinových intervaloch za posledných 6 hodín
- priemerný počet paketov spojenia musí byť menší ako 7

V prípade nálezu komunikácie, ktorá bola na základe splnenia podmienok klasifikovaná pozitívne je do databázy vložený záznam označujúci klasifikovanú udalosť.

Spôsob klasifikácie detektorom je možné popísať pomocou rozhodovacieho stromu zobrazeného na obrázku 5.3.



Obr. 5.3: Rozhodovací strom popisujúci detektor periodických tokov

Detektor blacklistovanej komunikácie

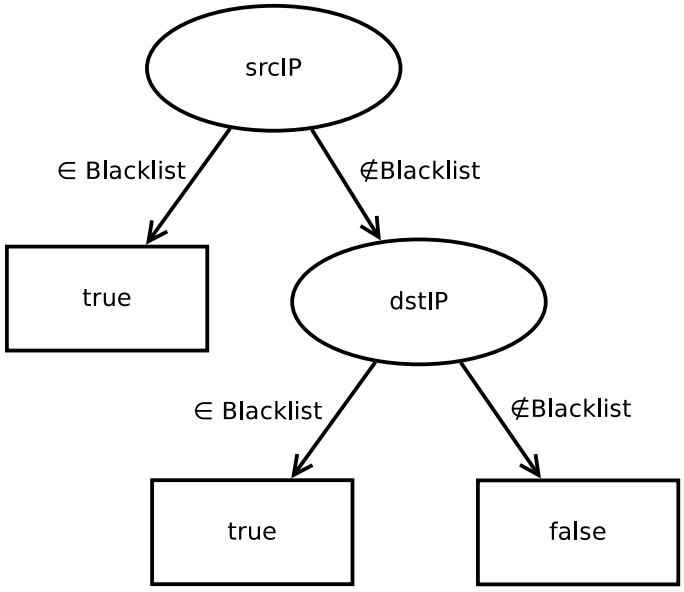
Blacklist je vopred definovaná nmožina IP adries. Ak zdrojová alebo cieľová IP adresa toku odpovedá jednej z IP adries patriacich do blacklistu, je tok možné klasifikovať už pred vložením do databázy pomocou nástroja Logstash s využitím filtra *translate*. Spôsob použitia filtra je popísaný v zdroji 5.3.

```
filter {
  translate {
     field => "[dstIpAddr]"
      destination => "blacklisted"
      dictionary_path => '/tmp/blacklist.yaml'
  }

  translate {
     field => "[srcIpAddr]"
      destination => "blacklisted"
      dictionary_path => '/tmp/blacklist.yaml'
  }
}
```

Zdrojový kód 5.3: Použitie logstash filtra translate pre klasifikáciu blacklistu

Spôsob klasifikácie detektorom je možné popísať pomocou rozhodovacieho stromu zobrazeného na obrázku 5.4.



Obr. 5.4: Rozhodovací strom popisujúci detektor blacklistovanej komunikácie

Implementovaný analyzátor vyberá toky, ktoré sú klasifikované ako blacklist a v prípade nálezu vloží do databázy príslušnú udalosť.

Multicast detektor

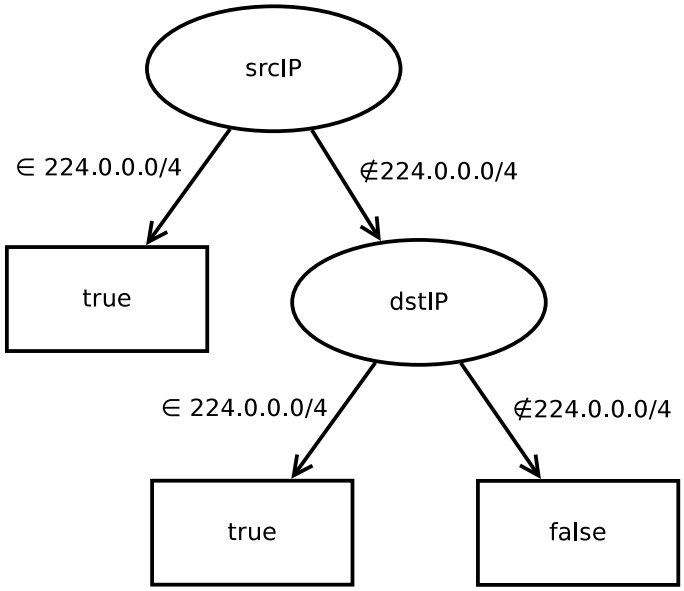
K identifikácii multicastových spojení sa používají IP adresy triedy D (rozsah 224.0.0.0 – 239.255.255.255). Podobne ako pri detekcii blacklistovanej komunikácie je možné toky klasifikovať pomozou nástroja Logstash pred vložením do databázy. Na klasifikáciu je využitý

filter *cidr* so spôsobom použitia popísanom v zdroji 5.4. Taktiež rovnako ako pri detekcii blacklistu sú implementovaným analyzátorom do databázy ukladané udalosti v prípade klasifikácie multicastu.

```
filter {
  cidr {
    add_field => { "multicast" => "true" }
    address => ["%{[srcIpAddr]}", "%{[dstIpAddr]}"]
    network => ["224.0.0.0/4"]
  }
}
```

Zdrojový kód 5.4: Použitie logstash filtra cidr pre klasifikáciu multicastu

Spôsob klasifikácie detektorom je možné popísať pomocou rozhodovacieho stromu zobrazeného na obrázku 5.4.



Obr. 5.5: Rozhodovací strom popisujúci multicast detektor

Kapitola 6

Dosiahnuté výsledky

Táto kapitola popisuje výledky testovania implementovaného nástroja na spracovanie a klasifikáciu sieťových tokov.

6.1 Spracovanie dát a výpočet metrík

Boli implementované dve verzie programu na spracovanie sieťovej prevádzky a výpočet metrík spojení - jedna verzia používa na výpočet iba CPU, druhá verzia s využitím platformy CUDA vykonáva výpočty metrík na grafickej karte.

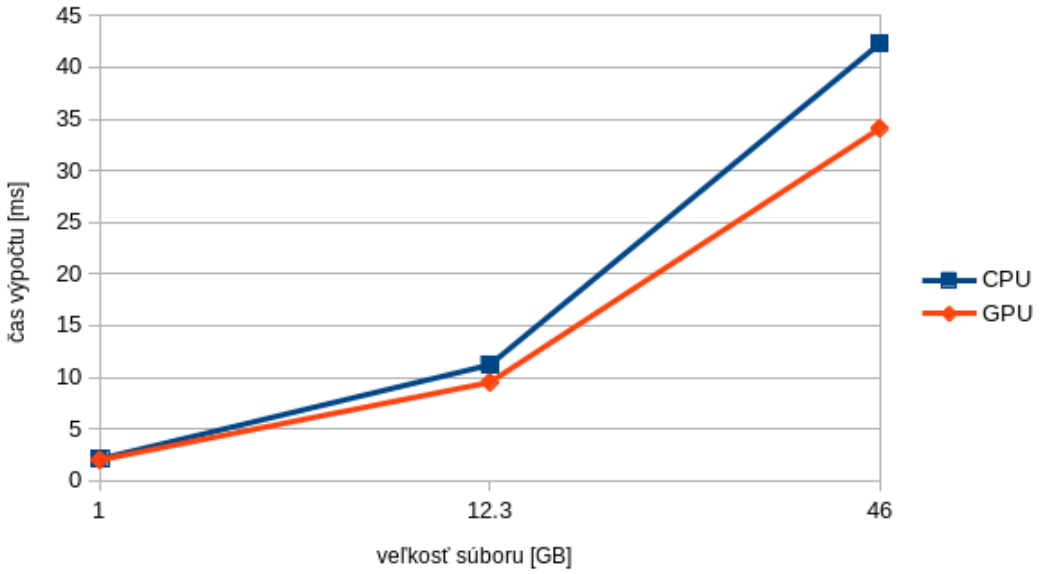
Implementácia bola testovaná na hardwarovej konfigurácii:

- Intel Core i7-4930K CPU
- nVIDIA GeForce GTX 760
- 32GB RAM

Obe verzie boli testované na plne vyťaženom sieťovom rozhraní s množstvom prenesených dát 1 gbps a sú schopné spracovávať sieťové toky a vykonávať výpočet metrík v reálnom čase.

Správnosť výpočtov metrík bola overená porovnaním výstupu s referenčným výstupom programu MATLAB. Vo všetkých prípadoch boli výstupné koeficienty pri porovnaní s presnosťou na 7 rádov rovnaké s referenčnými výstupmi.

Efektivita oboch verzií bola porovnaná pomocou troch nahraných pcap súborov o veľkostiach 1 GB, 12.3 GB a 46 GB. Meranie výpočtov prebehlo pre každý pokus 10x a výsledný čas bol spriemerovaný. Súbor o veľkosti 1 GB bol spracovaný verziou využívajúcou grafickú kartu v priemere o 4.8% rýchlejšie. Pre súbor o veľkosti 12.3 GB bolo zrýchlenie 15.2% a pri veľkosti súboru 46 GB 19.4% v prospech implementácie využívajúcej GPU. Graf nameraných hodnôt je vidieť na obrázku 6.1.



Obr. 6.1: Porovnanie doby trvania výpočtu dvoch verzií implementácie

Spracované dáta sú ukladané do databázy Elasticsearch. Pri generovanej nepretržitej sieťovej prevádzke o rýchlosti 1 gbps má denný index veľkosť v priemere 21 GB.

6.2 Detekčné metódy

Boli implementované 4 metódy na detekciu sieťových anomálii.

Portscan detektor

Detektor bol testovaný pomocou nástroja *nmap*. Správne sú klasifikované všetky scany, ktoré smerujú minimálne na 10 cieľových staníc. Analyzátor je schopný detekovať všetky skenovanie techniky nástroja nmap - SYN scan, ACK scan, null scan, FIN scan a Xmas scan.

Detektor periodických spojení

Implementovaný analyzátor detekuje výskyty previdelnej komunikácie na sieti, ktoré odpovedajú definovanému vzoru. K detekovaným spojeniam vačšinou patrí bežná sieťová prevádzka, ktorá má periodický charakter ako napríklad kontrola dostupnosti aktualizácii pre software alebo synchronizácia pomocou NTP. Z nežiadúcich sieťových tokov bola detekovaná prítomnosť malware *W32.Sality*, ktorý v pravidelných intervaloch posiela informáciu o svojej prítomnosti do externej siete.

Detektory multicast a blacklistovaných tokov

Tieto relatívne jednoduché detektory detekujú multicast komunikáciu a komunikáciu s blacklistovanými adresami. V oboch prípadoch je úspešnosť detekcie 100%.

Kapitola 7

Záver

Práca sa zaoberá problematikou detekcie anomálii v sieťovej prevádzke a klasifikácie sieťových tokov. Cieľom bolo preštudovať existujúce metódy a a ich základe navrhnúť a implementovať nástroj pre automatickú klasifikáciu sieťových tokov.

Jadro práce popisuje vlastný návrh a implementáciu nástroja na spracovanie a analýzu sieťových tokov. Bol implementovaný systém schopný spracovávať sieťovú prevádzku o rýchlosti 1 gbps v reálnom čase. Spracovanie tokov bolo optimalizované presunutím výpočtu metrík na grafickú kartu s využitím platformy CUDA, čím došlo k zrýchleniu výpočtov o viac ako 15%. Výsledné toky s vypočítanými metrikami sú uložené do databázy Elasticsearch, nad ktorou pracujú štyri klasifikátory sieťových tokov.

7.1 Vlastný prínos

Počas riešenia tejto diplomovej práce som získal prehľad o existujúcich metódach klasifikácie a nástrojoch pre detekciu anomálií v sieťovej prevádzke. Popísal som metódy na detekciu útokov v počítačových sietiach a preštudoval výskumy zaoberajúce sa touto tematikou. Na základe analýzy som navrhol vlastný nástroj na klasifikáciu sieťových tokov.

Na spracovanie sieťových tokov som musel preštudovať problematiku programovania na grafických kartách, ich architektúru a spôsob použitia frameworku CUDA. Tieto znalosti som využil na optimalizáciu výpočtov navrhnutých metrík, najmä výpočtov aproximácie vstupných hodnôt polynómami.

Pre prácu so spracovanými dátami som použil open-source nástroje Logstash a Elasticsearch, npreštudoval možnosti komunikácie s týmito nástrojmi a naučil sa pracovať s dátami uloženými v tejto NoSQL databáze.

Na základe poznatkov som v programovacom jazyku Java implementoval štyri funkčné analyzátory slúžiace na klasifikáciu sieťových tokov.

7.2 Možnosti ďalšieho rozšírenia

Navrhnutý nástroj poskytuje množstvo možností ďalšieho rozšírenia.

V kapitole 5.2 sú uvedené metriky, ktoré sú počítané ku každému sieťovému toku. Väčšina týchto vypočítaných metrík však nie je využitá implementovanými detektormi. Architektúra nástroja umožňuje jednoduché pridanie ďalších detekčných metód, ktoré môžu na klasifikáciu využiť pripravené metriky.

Ďalšou možnosťou rozšírenia je pridanie výpočtu koeficientov Diskrétnej Fourierovej transformácie do množiny navrhnutých metrík, čím by sa rozšírili možnosti klasifikácie pomocou nových detekčných metód.

Implementovaný nástroj bol testovaný na sieťovej prevádzke o rýchlosti 1 gbps. Systém má pri tejto prevádzke veľké rezervy výpočetného výkonu, implementáciu by preto bolo možné otestovať na výšších rýchlostiach sieťovaj komunikácie.

Literatúra

[1] DARPA Intrusion Detection Evaluation [omline].
URL
http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/

[2] KDD Cup 99. [online].
URL http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[3] Libpcap C/C++ library [online].
URL http://http://www.tcpdump.org/

[4] The Open Source Elastic Stack [online].
URL https://www.elastic.co/products

[5] THE SNORT PROJECT: Snort, The OpenSource Network Intrusion Detection System [online].
URL http://www.snort.org/

[6] Barabas, M.; Drozd, M.; Hanáček, P.: Behavioral signature generation using shadow honeypot. *World Academy of Science, Engineering and Technology*, ročník 2012, č. 65, 2012: s. 829–833, ISSN 2010-376X.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=9852

[7] Barbará, D.; Couto, J.; Jajodia, S.; aj.: ADAM: Detecting Intrusions by Data Mining. In *In Proceedings of the IEEE Workshop on Information Assurance and Security*, 2001, s. 11–16.

[8] Burges, C. J.: A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, ročník 2, č. 2, 1998: s. 121–167.

[9] Ibrahim, L. M.; Basheer, D. T.; Mahmod, M. S.: A Comparison Study for Intrusion Database (KDD'99, NSL-KDD) Based on Self Organization Map (SOM) Artificial Neural Network. *Journal of Engineering Science and Technology*, ročník 8, č. 1, 2013: s. 107–119.

[10] Ji, J.: Holistic Network Defense: Fusing Host and Network Features for Attack Classification. 2011.

[11] Kantardzic, M.: *Data Mining: Concepts, Models, Methods and Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2002, ISBN 0471228524.

[12] Kruegel, C.; Mutz, D.; Robertson, W.; aj.: Bayesian event classification for intrusion detection. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, Dec 2003, s. 14–23, doi:10.1109/CSAC.2003.1254306.

[13] Moore, A.; Crogan, M.; Moore, A. W.; aj.: Discriminators for use in flow-based classification. Technická zpráva, 2005.

[14] nVIDIA: CUDA Toolkit documentation v8.0.61 [online]. [cit. 2017-05-15].
URL http://docs.nvidia.com/cuda/

[15] Peter, E.; Schiller, T.: *A practical guide to honeypots*. Washington Univerity, 2011.

[16] Portnoy, L.; Eskin, E.; Stolfo, S.: Intrusion Detection with Unlabeled Data Using Clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, 2001, s. 5–8.

[17] Postel, J.: Transmission Control Protocol. RFC 793 [online]. 1981.

[18] Sangster, B.; O'Connor, T.; Cook, T.: Toward Instrumenting Network Warfare Competitions to Generate Labeled Datasets. In CSET, 2009.

[19] Scarfone, K.; Mell, P.: *Guide to Intrusion Detection and Prevention Systems (IDPS)*. National Institute of Standards and Technology, 2007.

[20] Tavallaee, M.; Bagheri, E.; Lu, W.: A detailed analysis of the kdd cup 99 data set. *Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.

[21] Tcpdump: Tcpdump/Libpcap public repository [online].
URL http://www.tcpdump.org/

[22] Tombini, E.; Debar, H.; Mé, L.; aj.: A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. In *20th Annual Computer Security Applications Conference (ACSAC 2004), 6-10 December 2004, Tucson, AZ, USA*, 2004, s. 428–437, doi:10.1109/CSAC.2004.4.
URL http://dx.doi.org/10.1109/CSAC.2004.4

[23] Utgoff, P. E.: Incremental induction of decision trees. *Machine learning*, ročník 4, č. 2, 1989: s. 161–186.

[24] Winigrad, S.: On computing the discrete Fourier transform. 1976.
URL http://www.ams.org/journals/mcom/1978-32-141/S0025-5718-1978-0468306-4/S0025-5718-1978-0468306-4.pdf