VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE SÍŤOVÝCH ANOMÁLIÍ NA ZÁKLADĚ NETFLOW DAT

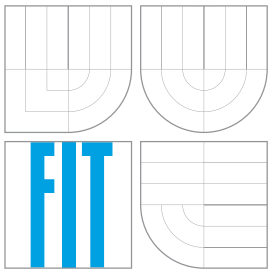
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MAREK CZUDEK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE SÍŤOVÝCH ANOMÁLIÍ NA ZÁKLADĚ NETFLOW DAT

DETECTION OF NETWORK ANOMALIES BASED ON NETFLOW DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MAREK CZUDEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN KOŘENEK, Ph.D.
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se zabývá využitím NetFlow dat v systémech pro detekci narušení nebo anomálií v síťovém provozu. Práce zahrnuje popis způsobů, které se využívají pro sběr dat ze síťě. Rozsáhleji je zde popsán protokol NetFlow. Následně se práce zaměřuje na analýzu a popis různých metod, které se využívají pro detekci anomálií v síťovém provozu, se zhodnocením jejich výhod a nevýhod. Na základě analýzy těchto metod je v další části práce vybrána konkrétní metoda. Následně je provedena analýza datové sady s využitím vybrané metody. Na základě výsledků je navržen algoritmus pro detekci anomálií v reálném čase. Tato konkrétní metoda byla vybrána na základě toho, že je schopna detekovat anomálie v neoznačeném síťovém provozu. V poslední části práce je algoritmus implementován a jsou provedeny experimenty s výslednou aplikací nad reálnými NetFlow daty.

Abstract

This thesis describes the use of NetFlow data in the systems for detection of disruptions or anomalies in computer network traffic. Various methods for network data collection are described, focusing especially on the NetFlow protocol. Further, various methods for anomaly detection in network traffic are discussed and evaluated, and their advantages as well as disadvantages are listed. Based on this analysis one method is chosen. Further, test data set is analyzed using the method. Algorithm for real-time network traffic anomaly detection is designed based on the analysis outcomes. This method was chosen mainly because it enables detection of anomalies even in an unlabelled network traffic. The last part of the thesis describes implementation of the algorithm, as well as experiments performed using the resulting application on real NetFlow data.

Klíčová slova

NetFlow, detekce anomálií, dolování dat, statistické metody, strojové učení, K-means

Keywords

NetFlow, anomaly detection, data mining, machine learning, statistical methods, K-means

Citace

Marek Czudek: Detekce síťových anomálií na základě NetFlow dat, diplomová práce, Brno, FIT VUT v Brně, 2013

Detekce síťových anomálií na základě NetFlow dat

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Jana Kořenka, Ph.D.

.....
Marek Czudek
15. května 2013

Poděkování

Děkuji panu Ing. Janu Kořenkovi, Ph.D. za odborné vedení mé diplomové práce a panu Ing. Martinu Žádníkovi za jeho cenné rady a připomínky. Rovněž děkuji svým blízkým a přátelům za jejich podporu.

Obsah

Kapitola 1

Úvod

V poslední době zaznamenal Internet velký rozmach a stal se nedílnou součástí každodenního života obyčejných lidí. Většina firem a domácností ze západního světa má stálé internetové připojení a tak mnoho lidí tráví na internetu nemálo času. Internet v dnešní době již neslouží pouze jako zdroj zábavy a informací, ale je rovněž prostředkem pro komunikaci, sdílení objemných dat, a také jej firmy využívají pro podnikání. Právě využívání všech těchto služeb je důvodem k neustálému zvyšování přenosových rychlostí na páteřních linkách a rovněž u lokálních připojení.

Nicméně s rostoucím počtem těchto služeb roste rovněž počet bezpečnostních hrozeb na tyto služby jako jsou Denial of service (DoS), Distributed denial of service (DDoS), počítačové viry nebo červi, krádeže dat atp., které ve výsledku mohou vést k nemalým finančním ztrátám. Samozřejmě se nemusí jednat pouze o útoky, ale rovněž o problémy způsobené špatným nastavením sítě atp. Z toho důvodu je kladen velký důraz na monitorování sítě, aby bylo možné případné anomálie detekovat a předejít tak těmto problémům.

Abychom byli schopni monitorovat sítě, je nezbytné, aby existoval způsob, kterým by bylo možné dané informace o síti sbírat. Aktuálně existuje několik způsobů, jak lze informace posílané po síti odchytávat a následně analyzovat. Pro sběr dat je možné využít například odchytávání raw paketů, protokol SNMP nebo NetFlow protokol. Všechny tyto způsoby získávají informace o tom, co se děje na monitorované síti, a ty pak lze následně ukládat pro další zpracování. Uložená data můžeme tedy dále analyzovat a snažit se z nich získat co nejvíce informací. Pro analýzu těchto dat je možné využít řadu metod. Mezi tyto metody můžeme zařadit statistické metody, které ale ne vždy dokáží modelovat požadovaná chování. Dále to mohou být metody založené na strojovém učení, které ale často vyžadují pro svou funkčnost učitele, což v neoznačeném síťovém provozu bývá problémem. K dalším metodám patří dolování dat, které už je možné využít při analýze dat, kde nemáme trénovací data označená. Do této skupiny dat se často řadí data ze síťových provozů, kde tyto informace často dostupné nejsou. Mezi další metody můžeme rovněž zahrnout hybridní systémy, které kombinují více metod dohromady.

V této práci se budu detailněji zabývat vybranými detekčními metodami, které jsou často využívané v systémech pro detekci útoků či anomálií. Jednotlivé metody se budu snažit popsat z hlediska jejich funkčnosti a efektivnosti a také z hlediska jejich výpočetní náročnosti atp. Následně ze získaných informací o jednotlivých metodách vyberu konkrétní metodu, která bude schopna detekovat anomálie v síti CESNET, kde nemáme dostupná označená trénovací data. Dále s využitím vybrané metody navrhnu a implementuju algoritmus, který bude schopen provádět detekci anomálií v reálném čase na uvedené síti.

Struktura této práce bude následující. Ve druhé kapitole budou popsány různé způsoby,

kterými lze data ze sítě odchytávat. V další kapitole blíže popíšu vybrané metody využívané v detekci anomálií. Ve čtvrté kapitole bude provedena analýza datového souboru a v páté kapitole bude navržen algoritmus, který bude schopen detekovat anomálie v síti CESNET, kde nejsou dostupná označená trénovací data. V předposlední kapitole popíšu implementaci algoritmu a provedu experimenty s výslednou aplikací.

Kapitola 2

Sběr informací o síťovém provozu

2.1 Odchytávání raw paketů

Jde o způsob odchytávání raw[1] paketů, které obsahují hlavičky nižších vrstev. Jedná se tedy o hierarchickou strukturu. Na obrázku 2.1 lze tedy vidět, že k uživatelským datům je přidána aplikační hlavička, ke které je následně na transportní vrstvě přidána například TCP hlavička a na síťové vrstvě se ještě zabalí s IP hlavičkou. Následně k celému paketu se připojí Ethernetová hlavička. Všechny programy běžící pod operačním systémem přijímají a odesílají data prostřednictvím protokolů aplikační vrstvy. Na této úrovni se tedy pracuje pouze s aplikačními daty, kde zbylé vrstvy jsou zcela odstíněny. Pro odchytávání raw paketů je tedy nutné použít nějaký nástroj, což může být program nebo pouze nějaká knihovna, která nám umožní přístup na linkovou vrstvu odkud jsme schopni odchytit tyto pakety. Takovým nástrojem může být například TCPDUMP, který využívá knihovnu libpcap [34]. Tento způsob odchytávání paketů se využívá například v případě, kdy již víme o konkrétních tocích, které způsobují nějaké problémy, a chceme se podívat, co se v nich konkrétně posílá.

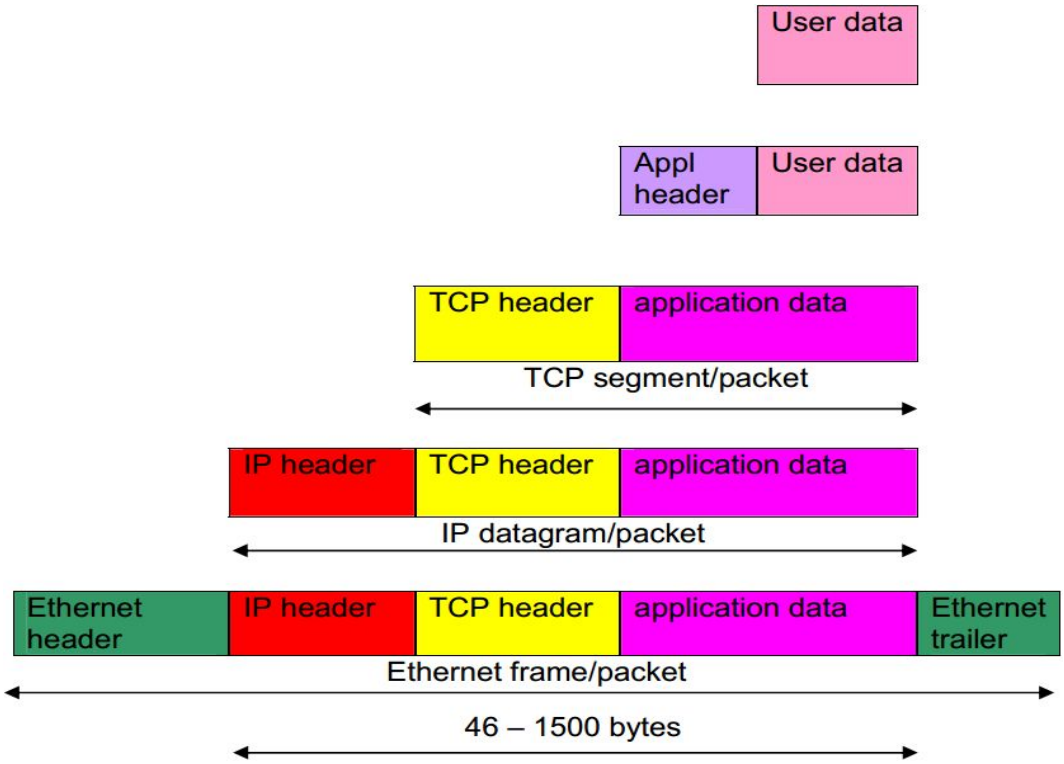
2.2 SNMP

SNMP je standardizovaný protokol, který je jednoduchý a široce rozšířený. Slouží k získávání a nastavování hodnot požadovaných zařízení. Celá řada zařízení má podporu právě tohoto protokolu. Jsou to například aktivní síťové prvky, tiskárny, přístupové body atp. Rovněž pomocí softwaru a ovladačů je možné tento protokol zpřístupnit osobním počítačům a serverům. S využitím tohoto protokolu můžeme pravidelně získávat různé informace z požadovaných zařízení a tyto informace následně vizualizovat, a zobrazit si tak například graf vytížení procesoru nebo datový tok na portu přepínače [33, 31].

Tento protokol je založen na modelu klient/server. V SNMP je server označován jako SNMP agent (monitorovaná strana) a klient je označován jako SNMP manažer (monitorovací strana).

- **SNMP manažer** - posílá dotazy agentovi a přijímá odpovědi. Tedy informace z agenta může získávat více manažerů.
- **SNMP agent** - výhodou SNMP agenta je to, že je schopen asynchronně zasílat různá oznámení (tzv. trap) manažerovi, pokud nastala nějaká nadefinovaná situace. To může

[1] raw - "syrový", nezpracovaný



Obrázek 2.1: Struktura IP datagramu [19].

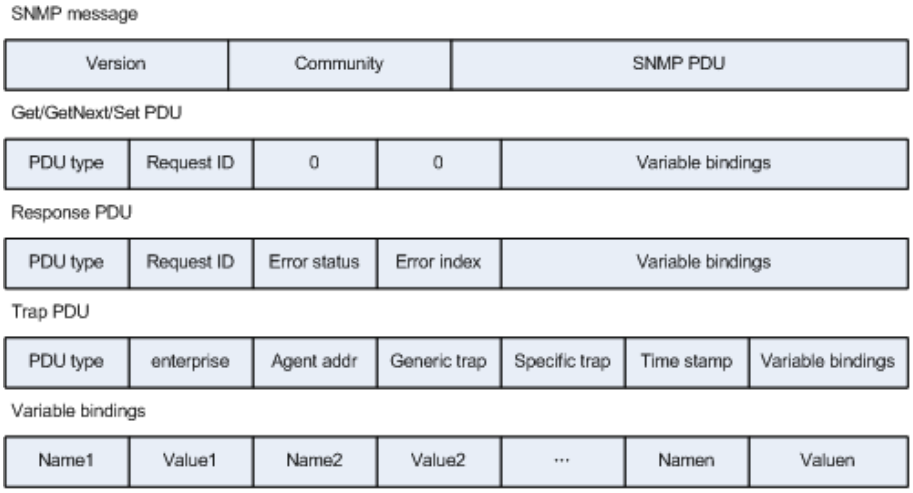
být například překročení nastaveného hraničního prahu nějaké hodnoty nebo agent může posílat předem nadefinované oznámení v pravidelných časových intervalech.

Aktuálně existují tři verze SNMP protokolu. Jmenovitě se jedná o SNMP v1, SNMP v2c a SNMP v3. Ve verzích v1 a v2c byla autentizace prováděna pomocí Community String, tedy textové heslo. V poslední verzi již byla přidána autentizace pomocí jména, hesla a šifrování.

Protokol SNMP využívá pro přenos dat UDP protokol, díky kterému je velmi rychlý. Nevýhodou UDP je možná ztráta paketu při přenosu, ale verze v2 již obsahuje kontrolu doručení a nemělo by tedy docházet k těmto ztrátám. Standardní porty pro komunikaci mezi agentem a manažerem jsou port 161 na straně agenta a port 162 na straně manažera.

Formáty SNMP paketů lze vidět na obrázku 2.2. Formáty paketů se mohou lišit v závislosti na tom, zda se jedná o paket typu dotaz/odpověď nebo se jedná o paket typu trap. SNMP zpráva se sestává ze dvou částí. První částí je hlavička zprávy následovaná vlastní PDU (Protocol Data Unit) částí. Hlavička obsahuje verzi protokolu a Community String. Datová část, při použití zprávy typu dotaz/odpověď, je složená z typu a ID dotazu pro svázání mezi dotazem a odpovědí. Dále obsahuje položku Error status, která indikuje chybu a její typ. Další položkou je Error index, který přiřazuje chybu dané proměnné z pole Variable bindings. Pole Variable bindings obsahuje vlastní data. Přiřazuje proměnným jejich aktuální hodnoty. Při použití zprávy typu trap má datová část odlišný formát. Položka Enterprise identifikuje typ objektu, který vygeneroval daný trap. Dále Agent address označuje adresu objektu, který daný trap vygeneroval. Následující položky identifikují typ a kód trapu. Variable bindings opět obsahuje seznam proměnných, které obsahují informace k danému trapu.

Komunikace dotaz/odpověď probíhá následovně:



Obrázek 2.2: SNMP dotaz/odpověď [35].

- **dotaz** - typ se nastaví na GET, vyplní se pole OID požadované hodnoty a hodnota se nastaví na NULL.
- **odpověď** - typ se nastaví na RESPONSE, vyplní se pole OID požadované hodnoty a nastaví se hodnota.

Všechny komunikační příkazy lze nalézt v článku [35].

Každá hodnota je v SNMP jednoznačně identifikována pomocí číselného OID - Object Identifier. OID tvoří posloupnost čísel, které jsou od sebe odděleny tečkou a tvoří tak stromovou strukturu. Celá tato struktura je uložená v MIB databázi, která zároveň obsahuje jména a popisy jednotlivých hodnot [31].

Příkladem takového OID může být například:
1.3.6.1.2.1.2.2.1.6.1
což v textové verzi bude odpovídat
iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifPhysAddress

2.3 IPFIX

Protokol IPFIX (Internet Protocol Flow Information eXport) byl vytvořen organizací IETF (Internet Engeneering Task Force) a je využíván pro přenos informací o IP tocích z exportérů do kolektorů. Vychází z prioprietárního protokolu NetFlow, který je popsán v kapitole 2.4, konkrétně z verze v9, ze které přebírá používání šablon. Nicméně definici položek záznamů převzatou z NetFlow v9 zpřesňuje a definuje další hodnoty, které lze pomocí tohoto protokolu měřit. Také přináší vylepšení v oblasti spolehlivosti přenosu dat, kdy je vyžadován v protokolu IPFIX, oproti NetFlow, protokol RS-SCTP, který zajišťuje spolehlivost a předchází zahlcení. IPFIX definuje formát a protokol pro export informací o IP tocích ze směrovače nebo jiného zařízení, které podporuje tento protokol. Jedná se o protokol pro přenos dat, který je lehce rozšířitelný pro potřeby různých aplikací. Informace o tocích procházejících daným zařízením jsou exportovány ve dvou formátech a jedná se konkrétně o data, která informace obsahují a šablony, které nám definují jaká data nás z daného toku

zajímají. Šablony jsou jednoznačně identifikovatelné podle konkrétního ID šablony, na jehož základě jsou spojeny s konkrétními daty. Očekává se, že protokol IPFIX v budoucnu nahradí NetFlow [22].

2.4 NetFlow

Netflow je otevřeným, ale proprietárním síťovým protokolem, který byl vyvinutý firmou Cisco Systems, aby shromažďoval informace o IP tocích. Protokol slouží k monitorování sítě na základě IP toků, což umožňuje v reálném čase získávat informace o provozu na síti. Tyto informace mohou být využity například v bezpečnostních systémech IDS, IPS, ale rovněž pro ISP (Internet Service Provider) k účtování konektivity [12, 13].

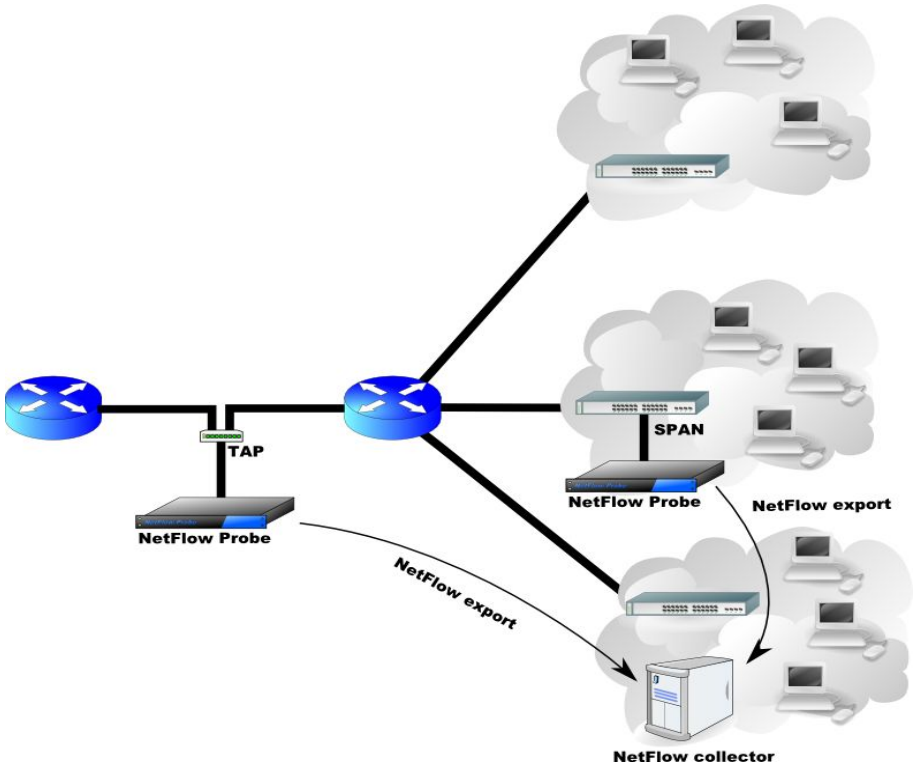
Základním prvkem v NetFlow je IP tok[2], který je definován jako nesměrový proud paketů mezi zdrojem a cílem. Konkrétně je tok identifikován jako kombinace následujících sedmi klíčových vlastností:

- zdrojová IP adresa,
- cílová IP adresa,
- zdrojový port,
- cílový port,
- číslo protokolu,
- rozhraní (interface),
- typ služby (ToS).

Všechny pakety, které mají stejnou výše uvedenou sedmici vlastností, jsou seskupovány do toku, který je zpracováván v NetFlow cache. IP tok může rovněž obsahovat další informace jako jsou například doba vzniku, délka trvání, počet přenesených paketů a bajtů a další údaje. Získávané informace závisí na verzi NetFlow protokolu. Nejrozšířenější verzí je NetFlow verze 5. Tato verze definuje pětici údajů, které jsou do toku zahrnuty. Jmenovitě to je doba vzniku toku, doba jeho trvání, počet přenesených bytů a paketů a TCP příznaky [13].

Typicky se NetFlow architektura skládá z několika exportérů jednoho kolektoru a vizualizace viz obrázek 2.3.

1. **Exportér** - nejčastěji se jedná o různá síťová zařízení routery, switche nebo to můžou být rovněž specializovaná zařízení jako NetFlow sondy. Exportéry monitorují toky a nasbírané informace následně odesílají do kolektoru.
2. **Kolektor** - tento prvek přijímá data zaslané exportérem nebo exportéry. Je schopen přijatá data různě filtrovat či agregovat. Tyto informace jsou nejčastěji ukládány do databáze.
3. **Vizualizace** - uložená data můžeme analyzovat a následně zobrazovat v podobě grafů atp.

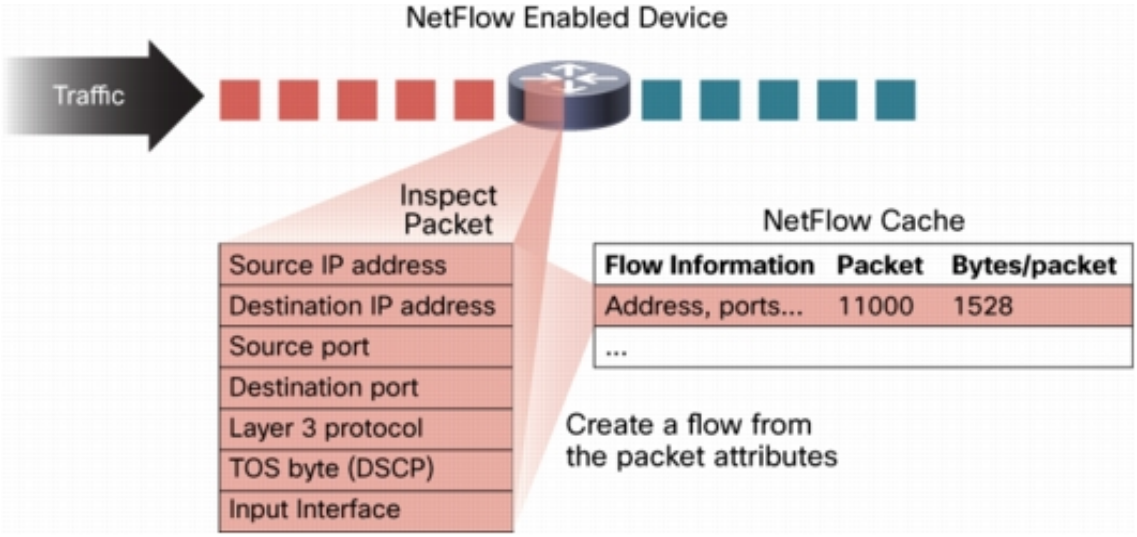


Obrázek 2.3: NetFlow architektura [12].

V NetFlow záznamech viz obrázek 2.4, jsou uchovávány informace týkající se pouze toku. Tyto záznamy tedy naobsahují žádná data přenášených, monitorovaných paketů. Z tohoto důvodu je velikost NetFlow záznamu nesrovnatelně menší v porovnání s daty přenášenými v rámci daného toku. Všechny důležité informace se o daném toku pouze agregují do jednoho záznamu. V tomto záznamu se následně zvyšuje počet přenesených bajtů a paketů atp. Rovněž TCP příznaky se agregují tak, že do záznamu se uloží všechny příznaky, které se vyskytly v průběhu celého toku. Tyto záznamy jsou následně uloženy na kolektoru. Informace, které se získávají o toku, jsou závislé na verzi NetFlow protokolu. V NetFlow verzi 5 jsou následující informace [12]:

- SNMP index vstupního a výstupního rozhraní,
- sekvenční číslo,
- číslo verze,
- čas začátku a konce IP toku,
- údaje z L3 hlavičky:
  - zdrojová a cílová IP adresa,
  - zdrojový a cílový port,
  - IP protokol,
  - typ služby (ToS).

[2]tok - pochází z anglického slova flow



Obrázek 2.4: NetFlow záznam [12].

- u TCP toků je obsažena množina, tvořena sjednocením všech TCP flagů, které se v toku vyskytly,
- směrovací informace:
  - IP adresa příštího hopu,
  - maska cílové a zdrojové IP adresy.

Mezi kvalitní open-source nástroje pro práci s NetFlow protokolem patří sada NFDUMP tools [26].

- **nfcapd (netflow capture daemon)** - nástroj pro sběr NetFlow dat z jednoho exportéru. Pokud chceme odchytávat data z více exportérů zároveň, je třeba spustit tolik instancí tohoto nástroje kolik máme exportérů, ze kterých chceme data odchytávat. Toky z jednotlivých exportérů jsou ukládány do oddělených složek a lze je tak zpracovávat dohromady nebo odděleně.

  Adresářová struktura uložených dat je snadno konfigurovatelná. Vlastní data jsou následně ukládána v binárním formátu. Názvy jednotlivých souborů odpovídají aktuálnímu datu a času. Defaultně jsou ukládány pětiminitové úseky, poté je soubor uzavřen a otevře se nový.

- **nfdump (netflow dump)** - tento nástroj slouží ke zpracovávání dat (souborů), které získáme využitím nástroje nfcapd, ale i jiných nástrojů, které používají pro ukládání toků stejný formát. Umí zobrazovat data po jednotlivých souborech nebo za určité časové období. Je schopen zpracovat i soubory, které byly pomocí nfcapd zkomprimovány. Rovněž zvládá jednoduché statistické úlohy jako N statistiky nebo průměrný tok atp. Užitečná funkce tohoto nástroje je také možnost využití filtrů. Tyto filtry mají podobný formát, který používá nástroj tcpdump.

- **nfprofile (netflow profiler)** - další nástroj, který umí zpracovávat data (soubory) z nfcapd. Umožňuje ukládat specifická filtrovací kritéria do tzv. profilů, podle kterých může následně zpracovávané data filtrovat a opět ukládat do souboru pro další zpracování.

- **nfreplay (netflow replay)** - nástroj pro přeposílání dat z nfcapd do jiného NetFlow kolektoru v síti.

Dalšími nástroji mohou být:

- **NfSen** - grafická webová nadstavba nad NFDUMP tools [3],
- **fprobe** - softwarové řešení NetFlow sondy [1],
- **Softflowd** - jedná se o exportér netflow záznamů [6],
- a další.

Kapitola 3

Metody detekce anomálií

Hlavním předpokladem pro detekci anomálií je to, že intrusivní aktivita je podmnožinou anomální aktivity. Pokud uvažujeme útočníka, který o legitimním chování uživatelů nic neví, je velká pravděpodobnost, že v době útoku na systém bude útočníková aktivita detekována jako anomální. V ideálním případě bude množina anomálních aktivit stejná jako množina útočníkových aktivit. V takovém případě označení všech anomálních aktivit jako útoků vede k tomu, že nejsou vygenerovaná žádná falešně pozitivním ani žádná falešně negativním hlášení. Nicméně intrusivní aktivity se ne vždy kryjí s anomální aktivitou. Máme čtyři možnosti, kde každá má nenulovou pravděpodobnost.

- Intrusivní, ale není anomální: Tyto jsou falešně negativní. IDS nedokáže detekovat tento typ aktivity, protože tato aktivita není anomální. Jsou nazývány falešně negativními, protože IDS nenahlásilo intrusi.
- Není intrusivní, ale je anomální: Tyto jsou falešně pozitivní. Aktivita není intrusivní, ale je anomální. IDS ji ale nahlásí jako intrusivní. Jsou tedy nazývány falešně pozitivními, protože je IDS falešně nahlásilo jako instrusivní.
- Není intrusivní a není anomální: Tyto jsou skutečně negativní. Aktivita není intrusivní a rovněž není nahlášena jako intrusivní.
- Intrusivní a anomální: Tyto jsou skutečně pozitivní. Aktivita je intrusivní a rovněž je nahlášena jako intrusivní.

Pokud mají být falešně negativní hlášení minimalizována, je třeba nastavit prahy, které definují anomálii nízko.

V této podkapitole probereme několik různých architektur a metod, které se využívají nebo využívaly pro detekci anomálií. Do těchto metod patří statistické metody detekce anomálií, metody založené na dolování dat a také metody založené na strojovém učení [27].

3.1 Statistické metody detekce anomálií

Ve statistických metodách detekce anomálií je síťový provoz odchytáván a následně je vytvářen profil, který reprezentuje jeho chování. Tento profil je založen na různých parametrech jako jsou počet toků, paketů či bajtů nebo počet unikátních adres atp. Typicky jsou využívány dva profily. Jeden profil reprezentuje aktuálně sledované parametry v průběhu

času a druhý profil reprezentuje dříve vytrénovaný statistický profil. Jakmile jsou zpracovávány příchozí síťové události, IDS systém aktualizuje původní profil a periodicky vypočítává anomální skóre tím, že porovnává tento profil s uloženým profilem. Anomální skóre obvykle označuje stupeň iregulárnosti konkrétní události. Pokud anomální skóre překročí nastavený práh, IDS systém vygeneruje poplach.

Statistický přístup k detekci anomálií má několik výhod. Za prvé tyto systémy nevyžadují dopředu znalost bezpečnostních chyb nebo konkrétních útoků. Výsledkem je, že tento typ systémů má schopnost detekovat "zero day" útoky nebo velmi nové útoky. Kromě toho statistické metody rovněž poskytují přesnou notifikaci škodlivých činností, které se obvykle vyskytují po delší dobu a jsou dobrými ukazateli na hrozící útoky typu Denial of Service (DoS). Dalším příkladem takové činnosti může být rovněž skenování portů.

Nicméně statistické metody detekce anomálií mají také své nevýhody. Tyto metody detekce jsou náchylné na to, že dobří útočníci si jsou schopni vytrénovat statistické metody detekce tak, aby provoz vygenerovaný během útoku, který je anomální, byl považován za normální. Rovněž může být obtížné určit hodnoty různých parametrů, které vyvažují pravděpodobnost falešných poplachů s pravděpodobností falešně negativních poplachů. Kromě toho potřebují statistické metody přesné statistické rozdělení. Bohužel ne všechna chování mohou být modelována použitím čistě statistických metod. Ve skutečnosti většina navrhovaných statistických metod detekce anomálií vyžaduje předpoklad kvazistacionárního procesu, který nelze předpokládat u většiny údajů zpracovávaných v systémech detekce anomálií [17].

Normální data patří do oblastí stochastického modelu, které mají vysokou pravděpodobnost, narozdíl od anomálií, které patří do oblastí stochastického modelu s nízkou pravděpodobností. V následujícím odstavci si uvedeme příklad.

V diplomové práci [28] byl sledován síťový provoz na Casablanca INT. V odchycených datech bylo sledováno několik statistik. Konkrétně se jednalo o počet toků, paketů a bytů. Níže na grafech lze vidět, že v době normálního provozu jsou sledované statistiky korelované[1] a jsou na sobě do určité míry lineárně závislé, což je vidět na obrázku 3.1. Odhady korelačních koeficientů přesahovaly ve všech případech 0.9. Bylo zjištěno, že poměry statistik, které lze vidět v pravém sloupci, se za normálních okolností mění pouze v určitém rozsahu, ale v době útoku se některé z techto poměrů výrazně změní, čímž dochází k porušení lineární závislosti. V době útoku klesly korelační koeficienty na 0.61-0.75, lze je pozorovat v grafech na obrázku 3.2.

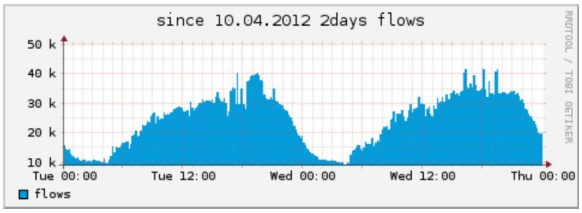
Výhody statistických metod:

- Není nutná předchozí znalost bezpečnostních chyb nebo konkrétních útoků.
- Schopnost detekce zero-day útoků.

Nevýhody statistických metod:

- Náchylná na zneužití schopnými útočníky.
- Ne všechna chování lze modelovat pomocí statistických metod.

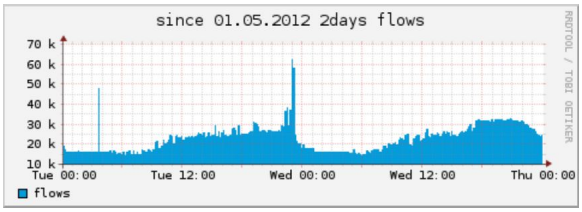
[1] Korelace se zabývá mírou závislosti mezi dvěmi veličinami nebo procesy. Tzn. že při změně jedné veličiny se mění i veličina druhá. Výstupem korelace je korelační koeficient, který může nabývat hodnot od -1 do $+1$.



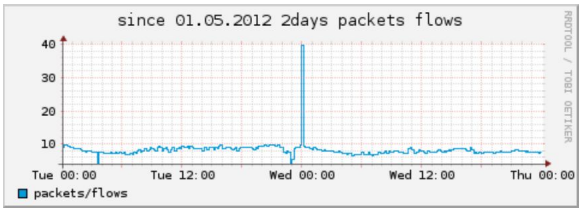
(a) Počet toků za sekundu naměřených v průběhu dvou dnů.

(b) Poměr počtu paketů k počtu toků. Odhad korelačního koeficientu $r = 0.92$.

Obrázek 3.1: Na grafech jsou zobrazené sledované statistiky a jejich poměry v době, kdy nedošlo k žádnému většímu útoku [28].



(a) Počet toků za sekundu naměřených v průběhu dvou dnů.



(b) Poměr počtu paketů k počtu toků. Odhad korelačního koeficientu $r = 0.61$.

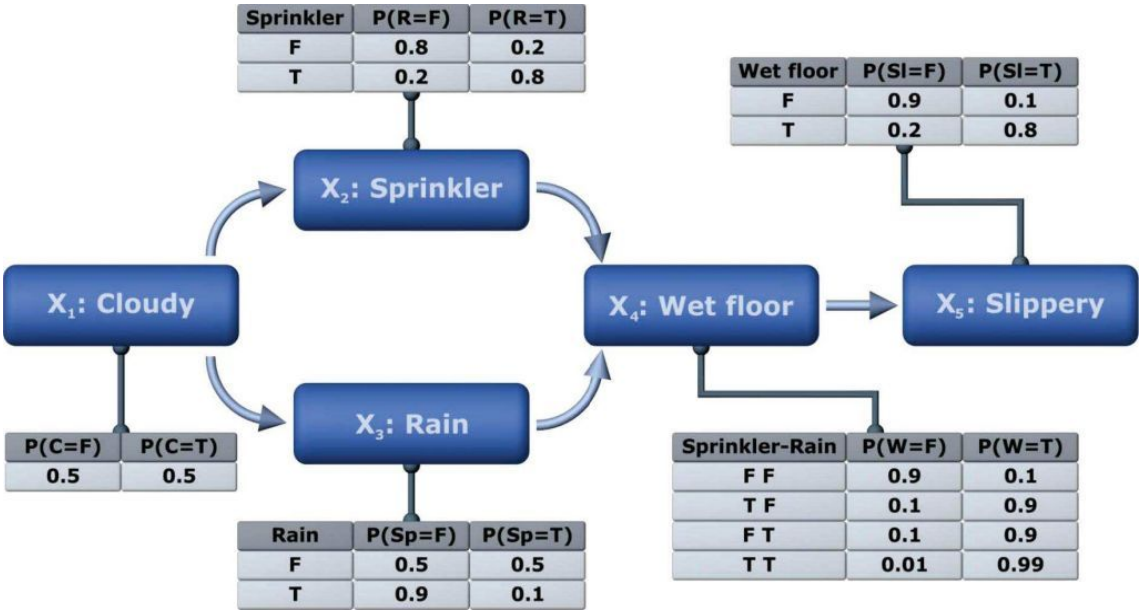
Obrázek 3.2: Na grafech jsou zobrazené sledované statistiky a jejich poměry v době, kdy došlo k několika útokům a jednomu většímu DoS útoku [28].

3.2 Metody detekce anomálií založené na strojovém učení

Strojové učení může být definováno jako schopnost programu či systému učit se a zlepšovat svoji činnost vůči nějakému úkolu nebo skupině úkolů v závislosti na čase. Strojové učení si klade podobné cíle jako statistické metody nebo dolování dat. Nicméně technika strojového učení se soustředí na výstavbu systému, který je schopen zlepšovat svoji činnost na základě předchozích výsledků narozdíl od statistických metod, které se zaměřují na pochopení procesu, který data generoval. Jinými slovy systémy založené na strojovém učení mají schopnost měnit svoji strategii založenou na nových informacích během zpracovávání dat [27]. Existuje mnoho metod, které jsou založené na strojovém učení. Zde se budu zabývat pouze třemi vybranými metodami.

3.2.1 Bayesovské sítě

Bayesovské sítě [9, 36] jsou definovány jako grafický pravděpodobnostní model pro analýzu dat s velkým množstvím proměnných. Konkrétně se jedná o orientovaný acyklický graf, který má přidruženou pravděpodobnostní funkci vyjádřenou například pomocí tabulky podmíněných pravděpodobností (Conditional probability table - CPT). Uzly v těchto orientovaných grafech reprezentují proměnné a hrany reprezentují podmíněnou závislost mezi proměnnými viz obrázek 3.3. Proměnné, které nejsou spojeny s žádným jiným uzlem, reprezentují proměnné, které nemají žádnou podmíněnou závislost. Navíc pravděpodobnostní funkce ilustruje sílu těchto vztahů v grafu. Bayesovské sítě jsou založeny na Bayesovu teorému [36]. Tento teorém se zabývá podmíněnými pravděpodobnostmi událostí. Formálním zápisem pro pravděpodobnost události A za předpokladu výskytu události B je $P(A|B)$. Thomas Bayes (1702-1761) byl anglickým duchovním, který objevil, že podmíněná pravdě-



Obrázek 3.3: Jednoduchá Bayesovská síť [9].

podobnost $P(A|B)$ souvisí s opačně podmíněnou pravděpodobností $P(B|A)$ takto:

Máme-li dvě náhodné události A a B, jejichž pravděpodobnosti jsou $P(A)$ a $P(B)$, a pokud $P(B) > 0$, potom bude platit

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{3.1}$$

Většina přístupů pro detekci anomálií vyžadují schopnost učení se. Bayesovské sítě toto umožňují rovněž. Existují dva typy učení se v Bayesovských sítích. Konkrétně se jedná o strukturální učení nebo parametrické učení. U strukturálního jde o vytvoření struktury grafu, který popisuje závislosti (a nezávislosti) mezi proměnnými. U druhého typu je vyžadována již hotová struktura grafu a úkolem je nalézt správné parametry předchozích a podmíněných pravděpodobností (CPT u diskrétních Bayesovských sítí). Tento typ učení je možné využít i při nedostatku dat. Je také velmi důležité, že tento způsob umožňuje měnit hodnoty stávajících parametrů a tím způsobem dosáhnout inkrementálního učení [39].

Příkladem jednoduché Bayesovské sítě může být znázornění příčin mokré podlahy. Tento příklad byl převzatý z práce [9]. Příčinou může být déšť nebo použití rozprašovače. To znamená, že pravděpodobnost mokré podlahy závisí na tom, zda pršelo nebo byl zapnutý rozprašovač. Celou tuto situaci je možné modelovat jednoduchou Bayesovskou sítí s několika binárními pravděpodobnostními proměnnými. Závislosti pravděpodobností na předchozích událostech znázorňují orientované hrany mezi nimi. Tato síť i s ukázkovými pravděpodobnostmi výskytu jednotlivých událostí je zobrazena na obrázku 3.3.

Pokud chceme zjistit, proč je podlaha mokrá, naskytují se nám pouze dvě odpovědi a to, že buď byl zapnutý rozprašovač (uzel Sprinkler) nebo pršelo (uzel Rain). Pokud tedy chceme zjistit, co je pravděpodobnější, zda pršelo nebo byl zapnutý rozprašovač, můžeme využít Bayesův teorém a odvodit pravděpodobnosti obou příčin, které mohly způsobit, že je podlaha mokrá.

V následujícím příkladu použijeme notaci:

- Sp(Sprinkler) = rozprašovač,
- W(Wet Floor) = mokrá podlaha,
- C(Cloudy) = zamračeno,
- R(Rain) = déšť,
- T = TRUE,
- F = FALSE.

V tabulkách na obrázku 3.3 jsou uvedeny základní pravděpodobnosti rozprašovače, deště, zamračení a mokré podlahy, které byly v příkladu použity.

Nyní vypočítáme s jakou pravděpodobností je příčinou mokré podlahy rozprašovač a déšť, a zjistíme, která z těchto příčin je pravděpodobnější.

Pro rozprašovač:

$$\begin{aligned} P(Sp=T|W=T) &= \frac{P(Sp=T)*P(W=T|Sp=T)}{P(W=T)} \\ &= \frac{P(Sp=T,W=T)}{P(W=T)} \\ &= \frac{\Sigma_{C,R\in\{T,F\}}P(C,Sp=T,R,W=T)}{\Sigma_{C,R,Sp\in\{T,F\}}P(C,Sp,R,W=T)} \\ &= \frac{\Sigma_{C,R\in\{T,F\}}P(C)*P(Sp=T|C)*P(R|C)*P(W=T|Sp=T,R)}{\Sigma_{C,R,Sp\in\{T,F\}}P(C)*P(Sp|C)*P(R|C)*P(W=T,Sp,R)} \\ &= \frac{0,1692}{0,3712} = 0,4558 \end{aligned} \tag{3.2}$$

Pro déšť:

$$\begin{aligned} P(R=T|W=T) &= \frac{P(R=T)*P(W=T|R=T)}{P(W=T)} \\ &= \frac{P(R=T,W=T)}{P(W=T)} \\ &= \frac{0,2412}{0,3712} = 0,6498 \end{aligned} \tag{3.3}$$

Zjistili jsme, že příčinou byl pravděpodobněji déšť. Poměr pravděpodobností je $\frac{0,6498}{0,4558} = 1,4256$ z čehož vyplývá, že déšť je 1,4x pravděpodobnější příčinou mokré podlahy než rozprašovač.

3.2.2 PCA

Typické soubory dat pro detekci anomálií a narušení jsou obvykle velmi rozsáhlé a multidimenzionální. S růstem vysokorychlostních sítí a distribuovaných sítí je kladen důraz na velmi intenzivní ukládání aplikačních dat, zpracování, přenos, vizualizaci. Tedy i pochopení dat se stává mnohem komplexnější a dražší. Pro vypořádání se s tímto problémem v multidimenzionálních souborech dat byla vyvinuta technika pro redukci rozměrnosti dat známá jako Analýza hlavních komponent (PCA). V matematice je PCA technikou, kde se n korelovaných proměnných transformuje do $d \leq n$ nekorelovaných proměnných. Nekorelované proměnné jsou lineární kombinací původních proměnných a mohou být pro vyjádření dat v redukované formě. Typicky první hlavní komponentou transformace je lineární kombinace původních proměnných s největší proměnlivostí čili rozptylem. Jinými slovy první hlavní komponentou je projekce do směru, ve kterém je rozptyl projekce maximalizován. Druhou hlavní komponentou je lineární kombinace původních proměnných s největším rozptylem, který není obsažen v první hlavní komponentě atd. V mnoha datových souborech nejvíce přispívá k rozptylu původních dat několik prvních hlavních komponent, tím pádem ostatní můžeme vyloučit s minimální ztrátou rozptylu, čímž dosáhneme redukce dimenzí. PCA byla široce používáná v oblasti komprese snímků, rozpoznávání vzorů a detekce narušení. Shyu et al. [32] navrhli schéma detekce anomálií, kde PCA bylo použito jako schéma pro detekci outsiderů a bylo aplikováno pro redukci rozměrnosti auditních dat. U jednotlivých pozorování měřili Mahalanobisovu vzdálenost[2] od centra dat pro detekci anomálií. Shyu et al. zhodnotili jejich metodu na KDD CUP99 datech a prokázali, že tato metoda vykazuje lepší detekční výsledky než jiné známé algoritmy pro detekci anomálií jako například Local Outlier Factor "LOF", nejbližší soused, a k-tý nejbližší soused [27].

3.2.3 Markovovy modely

Markovovy řetězce byly rovněž hojně využívány pro detekci anomálií. V článku Ye et al. [40] autoři prezentovali metodu detekce anomálií založenou právě na Markovových řetězcích. V tomto dokumentu byly sekvence nedávných událostí sledovány pomocí monitorovacího okna o velikosti N. Typ auditních události $E_{t-N+1}, \ldots, E_t$ v tomto okně a v čase t byl zkoumán a byla získána sekvence stavů $X_{t-N+1}, \ldots, X_t$. Následně byla získána pravděpodobnost, s jakou je sekvence stavů $X_{t-N+1}, \ldots, X_t$ normální. Čím větší pravděpodobnost, tím spíše sekvence stavů pochází z normální aktivity. Předpokládá se, že pravděpodobnost sekvence stavů z anomální aktivity získána z normálního modelu Markovových řetězců bude nízká [27].

3.2.4 Zhodnocení metod strojového učení

Obecně metody strojového učení vyžadují označená trénovací data, podle kterých se mohou učit. Jedná se tedy nejčastěji o učení se s učitelem. Tyto metody jsou z toho důvodu nevhodné pro využití v situacích, kde potřebná trénovací data nemáme řádně označena. Do této kategorie se často řadí data, která pocházejí ze síťového provozu. Další nevýhodou metod strojového učení je možný stav přeučení se (tzv. overfitting). Jedná se o stav, kdy je model příliš přizpůsoben množině trénovacích dat, ale již není schopen generalizace a tím

[2] Mahalanobisova vzdálenost je míra vzdálenosti mezi každým pozorováním v multidimenzionálním shluku bodů a těžištěm daného shluku. Jinými slovy Mahalanobisova vzdálenost mezi konkrétním bodem x a středem μ "normálních dat"je vypočítána jako $D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-}1(x-\mu)}$ kde Σ je inverzní kovarianční matice[21].

pádem selhává na množině testovacích dat. Přeučení obvykle nastane, pokud je množina trénovacích dat příliš malá ve srovnání s komplexitou daného modelu [18]. Metody strojového učení jsou rovněž často výpočetně náročné. Je tedy třeba s těmito metodami často experimentovat a hledat tak optimální nastavení.

Metoda PCA se využívá v detekci anomálií pro redukci rozměrnosti dat. To znamená, že pokud si ze síťového provozu vytvoříme matici o n vektorech $\vec{v_1}, \ldots, \vec{v_n}$, které reprezentují časové intervaly a každý interval i obahuje m parametrů (např. $|\vec{v_i}| = m$), kde $v_{i,j}$ může být například počet bytů nebo počet paketů, nebo i komplexnější statistiky jako například entropie rozdělení zdrojových IP adres, které byly sledovány v časovém intervalu i na pozici j, můžeme s využitím metody PCA rozdělit sledovaný provoz na normální část a reziduální část. Reziduální část bude následně obsahovat potenciální anomálie. Nevýhodou metody PCA v detekci anomálií je to, že pokud se vyskytne nějaká větší anomálie na sledované síti, může tato anomálie vnést tolik rozptylu do sledovaného provozu, že se vyskytne v několika prvních hlavních komponentách a bude považována za normální. V práci [30] došli k závěru, že využití metody PCA v detekci anomálií je mnohem těžší, než se na první pohled zdá. Tedy před použitím PCA metody je třeba využít ještě jinou metodu, která bude řešit některé problémy, které PCA není schopna sama o sobě vyřešit.

Bayesovké sítě v detekci anomálií mohou být využity při klasifikaci nebo při potlačení negativních alarmů. Výhodou Bayesovských sítí oproti jiným prediktivním modelům je to, že struktura v těchto sítích reprezentuje vztahy mezi atributy, které se v datech nacházejí. Bayesovské sítě jsou rovněž jednoduše čitelné a pokud je to zapotřebí, je možné je modifikovat a vytvořit tím lepší predikční modely [11].

Metody detekce anomálií s využitím Markovových řetězců jsou často velmi výpočetně náročné a to z toho důvodu, že využívají techniku pro odhad parametrů, která je založena na Bayesovu algoritmu. Pokud vezmeme v úvahu velké množství dat s relativně vysokou frekvencí jejich výskytu v dnešních počítačových sítích, není tato technika příliš vhodná pro detekci anomálií v reálném čase [27].

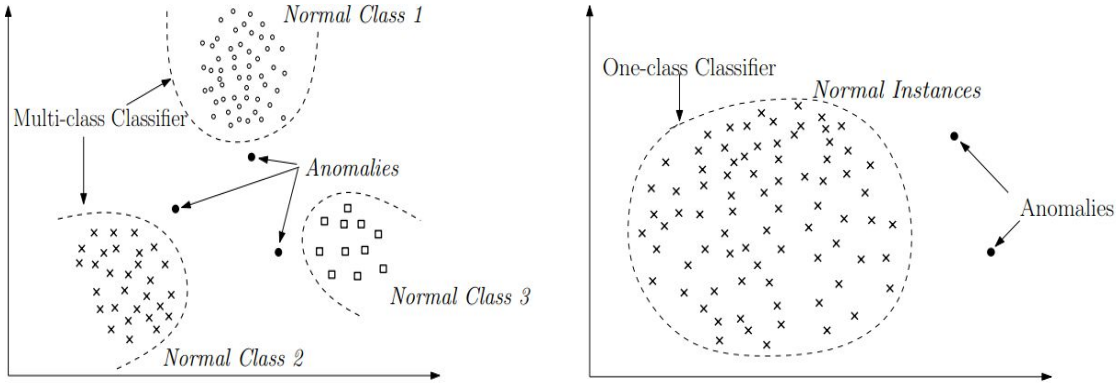
3.3 Metody detekce anomálií založené na dolování dat

V následující podkapitole bylo čerpáno hlavně ze zdrojů [10, 27]. Dolování dat (Data mining) je proces vyhledávání informací a znalostí ve velkém objemu dat. Slouží tedy k odhalení dříve neznámých vztahů mezi daty. Tímto způsobem je rovněž možné odhalit různé anomálie a proto se tyto techniky využívají v systémech IDS.

3.3.1 Detekce narušení založená na klasifikaci

Detekční systém, který klasifikuje auditní data jako normální nebo anomální na základě souboru pravidel, vzorů nebo jiných přidružených technik, lze obecně definovat jako systém detekce narušení založený na klasifikaci. Klasifikační proces obvykle zahrnuje následující kroky:

- Indentifikace atributů a tříd z trénovacích dat.
- Indentifikace atributů pro klasifikaci.
- Vytvoření modelu, který pro učení používá trénovací data.
- Použít model pro klasifikaci neznámých dat.



(a) Multi-class. (b) One-class.

Obrázek 3.4: Typy klasifikace [10].

Klasifikace je využita pro to, aby učila model (klasifikátor) ze souboru trénovacích dat (tréning), a pak klasifikovala testovací data do jedné ze tříd s využitím naučeného modelu (testování). Technika detekce anomálií založena na klasifikaci funguje ve dvoufázovém módu. Nejdříve musí projít trénovací fází, ve které se klasifikátor učí ze souboru označených trénovacích dat. Následně přichází testovací fáze, ve které klasifikátor obdrží testovací data a musí je klasifikovat jako normální nebo anomální. Technika detekce anomálií založena na klasifikaci funguje na základě obecného předpokladu:

Klasifikátor, který je schopen rozlišovat mezi normálními a anomálními třídami, se toto může naučit.

Na základě označených dat dostupných pro trénovací fázi, může být technika detekce anomálií založená na klasifikaci rozdělená do dvou širokých kategorií: technika detekce anomálií multi-class a one-class.

Multi-class[3] technika detekce anomálií založena na klasifikaci předpokládá, že trénovací data obsahují označené instance, které patří do více tříd viz obrázek 3.4a. Na tomto obrázku lze vidět tři různé normální třídy a pak tři body, které jsou mimo tyto třídy a jsou tedy označeny jako anomálie. Tyto techniky detekce anomálií učí klasifikátor rozlišovat mezi jednotlivými normálními třídami proti zbytku, který není klasifikován jako normální. Některé techniky v této podkategorii spojují důvěrné skóre s predikcí, která byla provedena pomocí klasifikátoru. Pokud si žáden z klasifikátorů není jist při klasifikaci testované instance, zda je tato instance normální, je tato instance označena jako anomální.

One-class[4] technika detekce anomálií založena na klasifikaci předpokládá, že všechna trénovací data mají pouze označení pro jednu třídu viz obrázek 3.4b. Na tomto obrázku vidíme pouze jednu normální třídu. Body, které se nacházejí mimo tuto třídu, jsou označeny jako anomálie. Tyto techniky se učí diskriminační hranici kolem normálních dat s využitím algoritmu one-class pro klasifikaci. Každá testovací instance, která nespadá do naučených hranic je označena jako anomální.

Již bylo navrženo několik různých technik pro klasifikaci. Patří mezi ně například technika induktivního generování pravidel, techniky založené na fuzzy logice, genetických algoritmech či neuronových sítích.

[3] multi-class - použití většího počtu tříd
[4] one-class - použití jedné třídy

Algoritmus pro induktivní generování pravidel obvykle zahrnuje používání souboru asociačních pravidel a časté vzory pro klasifikaci auditních dat. Pokud pravidlo stanoví, že "pokud nastane událost X, je pravděpodobné, že nastane i událost Y", pak události X a Y mohou popisovat soubor dvojic (proměnná, hodnota), kde je cílem najít soubor X a Y, kde $X \implies Y$. V oblasti klasifikace stanovíme Y a pokusíme se najít soubor X, který dobře predikuje pravou klasifikaci. Během klasifikace s učitelem se obvykle pouze odvozuje pravidla, která souvisí s jedním atributem. Obecné techniky pro indukci pravidel, které jsou v přírodě obvykle bez učitele, odvozují pravidla, která souvisí s jakýmikoliv nebo se všemi atributy. Výhodou používání pravidel je, že mají tendenci být jednoduché a intuitivní, rovněž nestrukturované a méně rigidní. Mezi nevýhody patří to, že jsou těžce udržovatelné a v některých případech jsou neadekvátní pro reprezentaci velkého množství typů informací. Bylo navrženo mnoho algoritmů pro induktivní generování pravidel, které lze nalézt v literatuře. Některé z nich nejdříve sestavují rozhodovací strom a poté zpracovávají soubor klasifikačních pravidel z rozhodovacího stromu. Další algoritmy RIPPER, C4.5 přímo odvozují pravidla z dat použitím přístupu rozděl a panuj. RIPPER byl úspěšně použit v mnoha algoritmech pro detekci anomálií, založených na dolování dat, pro klasifikaci příchozích auditních dat a následnou detekci narušení. Jednou z hlavních výhod používání RIPPER je to, že generovaná pravidla jsou jednoduchá na použití a jsou ověřená. RIPPER byl použit pro vyznačení sekvencí, které se vyskytují v normálních datech pomocí menší sady pravidel, které zachycují společné prvky v těchto sekvencích. Sekvence, které porušují tato pravidla jsou označeny jako anomálie.

Technika využívající Fuzzy logiku byla využita v oblasti počítačů a bezpečnosti sítí již od roku 1990. Tato technika byla využita pro systémy IDS z několika důvodů. Za prvé kvantitativní parametry, které se využívají v systémech IDS jako například využití CPU, délka trvání spojení atp. mohou být vnímány jako fuzzy proměnné. Za druhé koncept bezpečnosti jako takový je rovněž fuzzy. Jinými slovy "fuzzines" pomáhá rozdělit normální chování od anomálního chování. To znamená, že daný bod, který spadá do vnitřního/vnějšího "normálního intervalu", bude považován za normání/anomální bez ohledu na jeho vzdálenost od/v intervalu. Dickerson et al. [15] vyvinuli systém Fuzzy Intrusion Recognision Engine (FIRE), ve kterém používají fuzzy množiny a fuzzy pravidla. FIRE používá zjednodušené techniky dolování dat pro zpracování vstupních dat a generování fuzzy množin pro každou sledovanou vlastnost. Fuzzy množiny jsou potom využity pro definování fuzzy pravidel pro detekci jednodlitých útoků. FIRE nesestavuje žáden model, který by reprezentoval aktuální stav systému, ale namísto toho spoléhá na pravidla pro detekci specifických útoků. FIRE namísto toho vytváří a aplikuje logické fuzzy pravidla pro auditní data, aby je mohl klasifikovat jako normální či anomální. Dickerson et al. zjistili, že tento přístup je částečně efektivní proti skenování portů. Hlavní nevýhodou tohoto přístupu je náročný proces generování fuzzy pravidel.

Genetické algoritmy jsou řazeny do heuristických postupů, které se snaží pomocí aplikace evoluční biologie nalézt řešení složitých problémů, na které neexistuje exaktní algoritmus. Tato technika byla rovněž široce využita v oblasti detekce narušení pro odlišení normálního síťového provozu od anomálního. Hlavní výhodou genetických algorimů je jejich flexibilita a robusnost jako vyhledávací metody. Rovněž vyhledávací genetický algoritmus konverguje k řešení z několika různých směrů a je založen na pravděpodobnostních pravidlech namísto deterministických. V oblasti síťové detekce narušení byly genetické algoritmy použity různě. Některé přístupy využily genetické algoritmy přímo pro klasifikaci pravidel, jiné zase využily genetické algoritmy pro vybrání vhodné vlastnosti nebo k určení optimálních parametrů funkcí, zatímco jiné techniky dolování dat jsou využity pro získávání

pravidel. Prvními, kteří se pokusili o aplikaci genetických algoritmů v oblasti detekce narušení, byli Crosbie a Spafford [14] v roce 1995, kdy aplikovali technologii více agentů, aby detekovali síťové anomálie.

Výpočetní náročnost technik založených na klasifikaci záleží na použitém klasifikačním algoritmu. Obecně trénovací rozhodovací stromy jsou rychlejší než techniky využívající kvadratickou optimalizaci. Testovací fáze u technik pro klasifikaci bývá velice rychlá, protože se pro ni využívá již naučený klasifikační model.

Výhody technik založených na klasifikaci:

- Techniky založené na klasifikaci, zvláště techniky multi-class, mohou využívat algoritmy, které rozlišují instance patřící do různých tříd.
- Testovací fáze u techniky založené na klasifikaci je rychlá, protože každá testovaná instance je porovnávána s předem vypočítaným modelem.

Nevýhody technik založených na klasifikaci:

- U multi-class techniky založené na klasifikaci se spoléhá na dostupnost štítků, které by označovaly různé normální třídy, což často není možné.

Pokud máme dostupná data, která jsou dobře označená, je vhodné využít nějakou z metod klasifikace pro detekci anomálií, protože po výpočtu modelu z testovacích dat je následné porovnávání nových instancí velmi rychlé. Rovněž pokud jsou dostupná data označena pomocí multi-class, je možné tuto skutečnost využít pro zlepšení detekční schopnosti. Bohužel pro data, která označená nejsou, je používání klasifikačních metod nemožné. V této práci budou data získávána ze sítě CESNET, kde označení dat není dostupné a tím pádem není možné využít metody klasifikace pro naše účely.

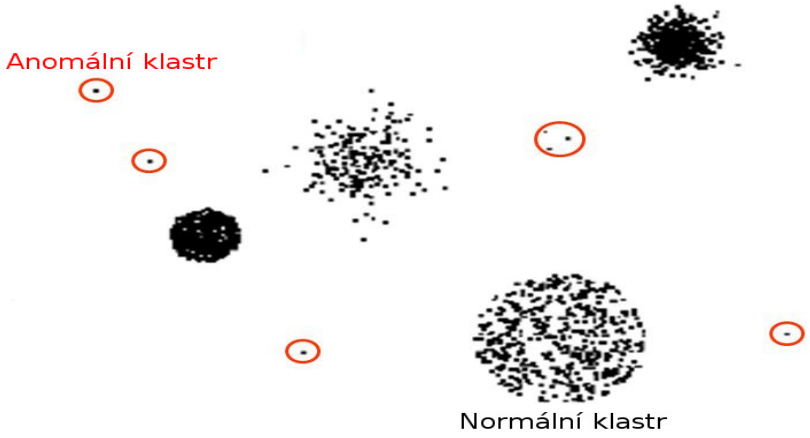
3.3.2 Klastrování a detekce outsiderů

Klastrování je technika využívaná při seskupování podobných instancí dat do klastrů viz obrázek 3.5. Na obrázku vidíme čtyři relativně velké a husté klastry, které jsou považovány za normální. Klastry, které jsou malé a řídké jsou tedy označeny jako anomálie. Klastrování je v první řadě "unsupervised", ale rovněž jej lze použít v "semi-supervised" módu. Klastrování je zajímavé v kontextu detekce narušení. Hlavní výhodou klastrování je schopnost učit se a detekovat narušení z auditních dat, kdy není vyžadováno po administrátorovi, aby poskytl explicitní popis tříd útoků či jejich typů. To vede k tomu, že je redukováno množství trénovacích dat, které je třeba poskytnout systému pro detekci anomálií. Klastrování a detekce outsiderů jsou se sebou spjaty. Z pohledu klastrovacích algoritmů jsou outsideři objekty, které nespadají do žádného klastru dat a v kontextu detekce anomálií mohou být považovány za narušení či útok. Techniky detekce anomálií založené na klastrování mohou být rozděleny do tří kategorií.

První kategorie technik založených na klastrování se spoléhá na předpoklad:

Normální datové instance patří do klastru v datech, zatímco anomálie nepatří do žádného klastru.

Techniky založené na tomto předpokladu používají známé algoritmy založené na klastrování a všechny instance, které nepatří do žádného klastru, označí jako anomálie. Příkladem



Obrázek 3.5: Klastry.

mohou být DBSCAN,ROCK,SNN. Nevýhodou této techniky je, že není optimalizována pro vyhledávání anomálií, poněvadž hlavním cílem těchto klastrovacích algoritmů je vyhledávání klastrů.

Druhá kategorie technik založených na klastrování se spoléhá na předpoklad:

Normální datové instance leží v blízkosti těžiště jejich nejbližšího klastru, zatímco anomálie jsou daleko od težiště jejich nejbližšího klastru.

Techniky založené na tomto předpokladu se skládají ze dvou kroků. Prním krokem je zařazení dat do klastru pomocí klastrovacího algoritmu. Ve druhém kroku je pro každou datovou instanci vypočítána vzdálenost k těžišti nejbližšího klastru jako anomální skóre. Příkladem takovýchto algoritmů mohou být Self-Organizing Maps (SOM), K-means Clustering a Expectation Maximization (EM).

Problémem tohoto přístupu je, že pokud anomálie v datech vytvoří vlastní klastr, výše zmíněné techniky nebudou schopny tyto anomálie detekovat. Z tohoto důvodu byla vytvořena třetí kategorie technik založených na klastrování, které se spoléhají na předpoklad:

Normální datové instance patří do velkých a hustých klastrů, zatímco anomálie patří do malých a řídkých klastrů.

Techniky založené na tomto předpokladu považují instance patřící do klastrů, které mají velikost a/nebo hustotu pod nastaveným prahem, za anomální. Příkladem takovýchto algoritmů může být FindCBLOF.

Vzdálenosti mezi body hrají v klastrování velkou roli. Nejpopulárnější metrikou vzdálenosti je Euklidovská vzdálenost. Vzhledem k tomu, že všechny vlastnosti přispívají stejnou mírou k výpočtu Euklidovské vzdálenosti, je tato vzdálenost nežádoucí v mnoha aplikacích. Toto platí zvláště, když mají vlastnosti velmi rozdílnou variabilitu nebo jsou rozdílné vlastnosti měřeny s různou váhou. Vlastnosti, které mají při měření větší váhu nebo vysokou variabilitu budou mít větší vliv než vlastnosti s nižší váhou nebo menší variabilitou. Přístup založený na vzdálenosti, který zahrnuje měření variability je nazýván Mahalanobis distance[2] založený na schématu detekce outsiderů. V tomto schématu všechny body v au-

ditních datech, které mají vzdálenost k průměru (vypočítán během tréhovací fáze) vyšší než je práh, budou detekovány jako outsideři.

Výpočetní náročnost trénování u technik detekce anomálií založených na klastrování závisí na použitém klastrovacím algoritmu. Tyto techniky mohou mít kvadratickou náročnost, pokud klastrovací technika vyžaduje výpočet vzdáleností pro všechny datové instance nebo lineární, pokud se použijí heuristické techniky jako například k-means nebo aproximační klastrovací techniky. Testovací fáze u technik založených na klastrování je rychlá a to z toho důvodu, že se testovací instance porovnává pouze s malým počtem klastrů.

Výhody technik založených na klastrování:

- Techniky založené na klastrování mohou být použity v "unsupervised"módu.
- Testovací fáze technik založených na klastrování je rychlá z důvodu malého počtu klastrů, se kterými musí být každá testovací instance porovnána.

Nevýhody technik založených na klastrování:

- Výkonnost technik založených na klastrování je velmi závislá na efektivnosti klastrovacího algoritmu.
- Mnoho technik detekce anomálií je pouze vedlejším produktem klastrování a tudíž nejsou optimalizována pro detekci anomálií.
- Některé klastrovací algoritmy nutí každou instanci, aby byla zařazena do nějakého klastru. To může mít za následek, že anomálie bude přiřazena do nějakého velkého klastru a tím bude považována za normální technikami, které pracují s předpokladem, že anomálie nepatří do žádného klastru.
- Některé techniky založené na klastrování jsou účinné pouze tehdy, pokud anomálie samy netvoří významnější klastry.
- Výpočetní náročnost klastrování dat je často úzkym hrdlem, zvláště když je použit klastrovací algoritmus s $O(N^2d)$.

Metody založené na klastrování nepředpokládají označená data a jsou tedy vhodné pro použití tam, kde tyto informace v datech dostupné nejsou. Problémem je větší výpočetní náročnost, kterou ale můžeme částečně redukovat využitím například metody K-means. V této práci budeme využívat data získaná ze sítě CESNET, kde označení dat nemáme, a proto využijeme metodu klastrování pro naše účely.

3.3.3 Vyhledávání asociačních pravidel

Technika dolování dat pomocí asociačních pravidel začala jako vyhledávání zajímavých pravidel z transakčních databází. Původně se tato technika využívala pro vyhledávání asociací v komerčních datech z databází transakcí, kde jednotlivé transakce představovaly soubor položek, které si zákazník nakoupil. Jinými slovy jde o techniku popisující události, které mají tendenci vyskytovat se společně. Dolování pomocí asociačních pravidel tedy vyhledává korelace mezi atributy. Cílem je pomocí asociačních pravidel odvodit korelaci mezi atributy z databáze. Z pohledu detekce anomálií je tato technika využita pro hledání korelace mezi atributy, které se nacházejí v síťových datech [38].

Asociační pravidla můžeme tedy chápat následujícím způsobem. Z dané databáze transakcí D, kde každá transakce $T \in D$ popisuje soubor položek v databázi a asociační pravidlo je následující implikací $X \implies Y$, kde $X \subset D$, $Y \subset D$ a $X \cap Y = \emptyset$. Hlavními koncepty u asociačních pravidel jsou "confidencial" a "support". Implikační pravidlo $X \implies Y$ je v dané databázi D s "confidencial (s jakou pravděpodobností je tato implikace pravdivá)" $c = \frac{support(X \cup Y)}{support(X)}$ a "support (kolikrát se vyskytla)" $s = support(X \cup Y)$. Asociační pravidla byla úspěšně použita při dolování auditních dat s cílem nalézt normální šablony pro detekci anomálií. Jsou důležitá v oblasti detekce anomálií, protože mohou být využita pro konstrukci zhodnocení anomálií, které byly detekovány nějakým systémem detekce narušení [24].

3.4 Hybridní systémy

Hybridní systémy jsou využívány pro zlepšení detekční schopnosti využitím jak systémů pro detekci anomálií, tak rovněž systémů pro detekci vzorů. V těchto hybridních systémech jsou techniky detekce anomálií využity pro detekci nových či neznámých útoků, zatímco techniky detekce vzorů jsou využity tak, aby detekovaly známé útoky. Techniky pro detekci vzorů mohou být rovněž schopny detekovat útoky, které vytvářejí útočníci ve snaze změnit vzor chování modulu pro detekci anomálií tak, aby akceptoval útočníkovo chování jako normální. Tombini et al. [37] využili tohoto přístupu, přičemž technika detekce anomálií byla použita pro vytváření seznamu podezdřelých záznamů. Následně klasifikační modul, který využíval detekci vzorů klasifikoval tyto podezdřelé záznamy jako falešný alarm, útok nebo neznámý útok. Tento přístup funguje za předpokladu, že komponenta pro detekci anomálií bude mít vysokou míru detekce, protože útoky které nezachytí nemohou být detekovány následující komponentou pro detekci vzorů. Rovněž se předpokládá, že komponenta pro detekci vzorů bude schopna identifikovat falešné poplachy. Nicméně hybridní systémy nejsou schopny zachytit některé útoky, ale redukovaná míra falešných poplachů zvyšuje pravděpodobnost zkoumání většiny alarmů.

Je sice pravdou, že vytvoření jednoho systému kombinací několika metod pro detekci narušení, může teoreticky vést k vytvoření silnějšího detekčního systému, ale často výsledné hybridní systémy pro detekci narušení nejsou efektivnější než dané metody použity samostatně. Rozdílné systémy pro detekci narušení zkoumají síťový provoz a vyhledávají narušení a anomálie různými způsoby. Tedy hlavní výzvou při tvorbě hybridního detekčního systému je schopnost propojení těchto různých metod tak, aby byly schopné spolu efektivně spolupracovat.

3.5 Útoky v NetFlow datech

V této podkapitole si stručně popíšeme útoky typu skenování, útok hrubou silou a slovníkové útoky. Informace k těmto typům útoků byly čerpány ze zdrojů [23, 20]. Jedná se o útoky, které se na síti CESNET vyskytují nejčastěji. V této práci budou data pocházet právě ze zmíněné sítě. Rovněž si ukážeme několik příkladů, jak se konkrétní útoky projevují v NetFlow datech. Všechny výpisy jsou anonymizovány nástrojem nfanon [2]. Významy jednotlivých sloupců u výstupů z nástroje nfdump se nacházejí v příloze F.

Skenování

Skenování je jedním z nejčastěji používaných útoků, které často předchází následnému zneužití sítě. Tento typ útoků je využíván útočníky ke zmapování sítě a stanic, které se na dané síti nacházejí. Cílem je zjistit informace o skenovaných stanicích, na které může být následně veden konkrétní útok, nebo mohou být tyto stanice použité například při útoku DoS (Denial of Service).

Skenování můžeme rozdělit následovně:

- **skenování portů** - skenování, které má za cíl zjistit, které služby jsou na skenované stanici dostupné,
- **skenování IP adres** - skenování, které má za cíl zjistit, které stanice jsou v dané podsíti dostupné.

Skenování portů můžeme ještě dále rozdělit na:

- **vertikální skenování portů** - má za cíl zjistit, které služby jsou dostupné na konkrétní stanici,
- **horizontální skenování portů** - má za cíl zjistit, na kterých stanicích je dostupný konkrétní port,
- **blokové skenování portů** - jedná se o kombinaci výše uvedených skenování portů.

Skenování portů využívá architektury TCP, kdy se v paketu posílají různé příznaky a při navazování spojení probíhá tzv. třícestné navazování spojení. Existuje tedy několik možných technik, které můžeme využít při zjišťování, zda je port na dané stanici otevřen, či nikoliv. Zde si uvedeme pouze několik technik.

SYN skenování

Tento typ skenování, známý rovněž jako tzv. napůl otevřené skenování, generuje pouze SYN pakety, a pokud je port na dané stanici otevřen, vrátí se nám paket s příznaky SYN-ACK. Následně by měl útočník odpovědět paketem s příznakem ACK, ale toto již útočník neudělá, případně zašle paket s příznakem RST, čímž ukončí spojení. Výhodou této techniky je to, že spojení, které nebylo kompletně navázáno, nebude nijak logováno, a tím pádem bude hůře vypátratelné. Níže přiložený výstup nám ukazuje, jak se toto skenování projevuje v NetFlow datech.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP      242.14.34.60  4756 ->   247.81.164.65  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  1734 ->   247.81.179.62  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  3097 ->  247.81.161.250  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  4645 ->  247.81.164.227  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  4268 ->  247.81.166.145  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  3098 ->  247.81.161.251  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  2857 ->   247.81.160.24  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  1671 ->  247.81.178.253  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  4324 ->  247.81.166.209  135 ....S.       1     48     1    48       0
TCP      242.14.34.60  1641 ->  247.81.178.197  135 ....S.       1     48     1    48       0
IP addresses anonymised
Summary: total flows: 3291, total bytes: 157968, total packets: 3291, avg bps: 4089, avg pps: 10, avg
     bpp: 48
Time window: <unknown>
```

```
Total flows processed: 3291, Blocks skipped: 0, Bytes read: 158076
Sys: 0.013s flows/second: 253153.8 Wall: 0.011s flows/second: 292091.9
```

Výpis 3.1: Ukázka SYN skenování.

ACK skenování

Tento typ skenování je poměrně unikátní. U tohoto skenování nelze přesně určit, zda je port otevřený nebo uzavřený, ale lze zjistit je-li port filtrovaný nebo nefiltrovaný. Je to obzvláště výhodné ke zjišťování, zda se na dané stanici nachází firewall.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP     238.16.34.151  1224 -> 150.223.254.163 1433 .A....       2     82     2    41       0
TCP     238.16.34.151  1621 ->   95.115.124.78 1433 .A....       2     82     2    41       0
TCP     238.16.34.151  2730 ->   95.115.124.52 1433 .A....       2     82     2    41       0
TCP     238.16.34.151  2318 -> 150.223.252.134 1433 .A....       3    123     3    41       0
TCP     238.16.34.151  1293 ->  247.81.132.136 1433 .A....       2     82     2    41       0
TCP     238.16.34.151  2267 -> 150.223.252.134 1433 .A....       2     82     2    41       0
TCP     238.16.34.151  2353 ->  247.81.131.121 1433 .A....       3    123     3    41       0
TCP     238.16.34.151  3783 ->   247.81.133.50 1433 .A....       3    123     3    41       0
TCP     238.16.34.151  3892 ->  247.81.131.120 1433 .A....       3    123     3    41       0
TCP     238.16.34.151  2307 ->  247.81.135.159 1433 .A....       3    123     3    41       0
IP addresses anonymised
Summary: total flows: 44925, total bytes: 1.8 M, total packets: 44953, avg bps: 47023, avg pps: 142,
     avg bpp: 41
Time window: <unknown>
Total flows processed: 44925, Blocks skipped: 0, Bytes read: 2156532
Sys: 0.020s flows/second: 2246250.0 Wall: 0.019s flows/second: 2358143.9
```

Výpis 3.2: Ukázka ACK skenování.

TCP skenování

Jedná se o nejzákladnější typ skenování. Využívá se celého procesu třícestného navazování spojení, tedy pokud je port na dané stanici otevřen, dojde ke kompletnímu navázání spojení s následným ukončením. Tato technika je bohužel velmi nápadná, protože často bývá kompletně navázané spojení logováno.

Jiné typy skenování

- NULL a X-mas skenování,
- FIN skenování,
- UDP skenování.

Útok hrubou silou

Tento útok funguje na principu zkoušení všech možných kombinací (slov) nad zvolenou množinou znaků, která může mít různá omezení. Zde je vhodné si uvědomit, jaký dopad má délka hesla a velikost použité množiny znaků na množství variant, které bude nutné prozkoumat. Pokud si vytvoříme heslo o m znacích nad množinou obsahující n znaků, výsledný počet všech možných variant bude m^n. Příkladem může být například šestimístné heslo vytvořené nad anglickou abecedou malých písmen. To nám ve výsledku dá 26^6 různých kombinací, což je cca 308 miliónů. Pokud použijeme osmimístné heslo a abecedu rozšíříme i o velká písmena, dosáhneme tímto počtu 62^8 různých kombinací, což je asi sedm set tisíc krát více než u předchozího příkladu. Proto se této metodě říká útok hrubou silou.

Slovníkový útok

Slovníkový útok sází na nepříliš ostražité uživatele, kteří často používají jednoduchá, lehce zapamatovatelná hesla. Tito uživatelé většinou používají běžná slova, která lze nalézt v obyčejných slovnících. Tento typ útoku se oproti útoku hrubou silou liší tím, že má nesrovnatelně menší počet variant pro uhádnutí hesla. Testovaná hesla jsou tedy uložená v externím souboru, ze kterého jsou postupně načítána nějakým specializovaným programem, který je využívá k uhádnutí hesla. Tyto slovníky je možné nalézt různě na internetu nebo si je můžeme vytvořit taky sami.

Níže uvedený příklad nám ukazuje, jak se tyto dva výše zmíněné útoky mohou projevovat v NetFlow datech.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP      150.41.46.50 18746 ->   82.82.78.125  3389 .APRSF      24     2557     1    106       8
TCP      150.41.46.50 21457 ->  82.84.126.121  3389 .APRSF      44     3359     1     76      24
TCP      150.41.46.50 32019 ->  82.84.114.178  3389 .APRS.      35     3117     1     89      14
TCP      150.41.46.50 62651 ->   82.82.77.147  3389 .APRSF      25     2607     1    104       8
TCP      150.41.46.50  9376 ->   82.82.88.238  3389 .APRSF      27     2689     1     99       5
TCP      150.41.46.50 14868 ->    82.82.64.77  3389 .APRSF      28     2725     1     97      10
TCP      150.41.46.50 30516 ->  82.83.231.129  3389 .APRS.      39     3353     1     85      20
TCP      150.41.46.50 48966 ->  82.84.114.134  3389 .APRS.      22     2497     1    113       5
TCP      150.41.46.50 65028 ->    82.84.243.4  3389 .APRS.      65     4664     1     71      12
TCP      150.41.46.50 59276 ->   82.84.92.232  3389 .APRSF      26     2639     1    101      13
IP addresses anonymised
Summary: total flows: 82416, total bytes: 260.3 M, total packets: 2.9 M, avg bps: 5.4 M, avg pps:
    7628, avg bpp: 88
Time window: <unknown>
Total flows processed: 82416, Blocks skipped: 0, Bytes read: 3956112
Sys: 0.030s flows/second: 2747200.0 Wall: 0.030s flows/second: 2740895.9
```

Výpis 3.3: Ukázka hádání hesla.

Kapitola 4

Analýza dat

V této kapitole se budeme věnovat analýze dat a návrhu algoritmu pro detekci anomálií s využitím NetFlow dat. Pro analýzu dat a následný návrh detekčního algoritmu byla na základě výsledků analýzy jednotlivých detekčních metod vybrána metoda klastrování pro její schopnost zpracovávat i neoznačená trénovací data, která budou ze sítě CESNET k dispozici.

4.1 Rozbor datové sady

Nejdříve bylo zapotřebí provést analýzu dat a zjistit, jak se během síťového provozu projevují jednotlivé charakteristiky a jak je možné tyto charakteristiky použít ve spojení s klastrovacími metodami. Jedná se například o počet přenesených paketů, bajtů, toků, TCP příznaků atp., které je možné získat pomocí NetFlow protokolu. Pro tyto účely bylo nutné se seznámit s nástroji pro práci s NetFlow. Následně bylo potřeba získat data z reálného provozu, nad kterými by bylo možné provádět testování klastrovacích algoritmů.

V této práci se využívá datová sada, která pochází ze sítě CESNET. V tabulce 4.1 jsou uvedeny specifikace této sítě. Jednalo se o datovou sadu, která byla vytvořena ze všech jedenácti linek jednoho pětiminutového intervalu. Tabulka 4.3 uvádí objemy provozu na jednotlivých linkách za vteřinu. Z důvodu velikosti tabulky jsou zobrazeny pouze celkové statistiky ze všech protokolů, dále pak v oddělených sloupcích statistiky pro TCP a UDP provoz. Celková velikost datové sady obsahující všechny linky je uvedena v tabulce 4.2. NetFlow záznamy z daného intervalu, pocházející z jednotlivých linek, byly v aplikaci HostStats sloučeny a následně z nich byly vypočítané statistiky pro jednotlivé IP adresy. Aplikace HostStats je vyvíjena v rámci projektu traffic-analysis. Statistiky, které jsou výsledkem HostStats, nebyly v původní datové sadě děleny podle konkrétních služeb. V rámci této práce mi byl umožněn přístup ke zdrojovým kódům této aplikace, kterou jsem mohl využít k předzpracování NetFlow dat z reálného provozu.

Pro efektivní analýzu získaných dat byla použita aplikace WEKA tools (Waikato Environment for Knowledge Analysis)[8]. Tato aplikace je napsána v jazyce Java a je sbírkou algoritmů strojového učení a dolování dat. Tento nástroj umožňuje použít tyto algoritmy

Síť	Počet linek	Agregace	Interval	Objem provozu
CESNET	11	5 min	06.03.2013 - 08:50	19 Gb/s

Tabulka 4.1: Specifikace sítě, ze které pocházel zkoumaný interval.

Síť	Počet toků	Počet bajtů	Počet paketů
CESNET	27202747	712.9 G	813.4 M

Tabulka 4.2: Celková velikost datové sady.

Statistiky intervalu 06-03-2013 - 08:50									
Linka	Toky			Pakety			Provoz		
	vše:	tcp:	udp:	vše:	tcp:	udp:	vše:	tcp:	udp:
1	14.2 k/s	10.1 k/s	3.9 k/s	460.4 k/s	410.1 k/s	44.8 k/s	3.6 Gb/s	3.3 Gb/s	235.8 Mb/s
2	7.8 k/s	6.4 k/s	1.4 k/s	359.3 k/s	341.6 k/s	15.5 k/s	2.3 Gb/s	2.2 Gb/s	83.5 Mb/s
3	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s
4	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s	0 /s
5	23.2 k/s	15.8 k/s	7.0 k/s	299.8 k/s	229.2 k/s	69.3 k/s	1.8 Gb/s	1.4 Gb/s	391.6 Mb/s
6	8.0 k/s	6.1 k/s	1.8 k/s	354.6 k/s	289.5 k/s	64.4 k/s	2.9 Gb/s	2.4 Gb/s	476.0 Mb/s
7	3.3 k/s	2.4 k/s	914.4 /s	93.0 k/s	83.3 k/s	9.5 k/s	721.8 Mb/s	665.3 Mb/s	56.3 Mb/s
8	13.8 k/s	10.0 k/s	3.6 k/s	472.5 k/s	421.4 k/s	42.3 k/s	3.7 Gb/s	3.4 Gb/s	229.7 Mb/s
9	7.4 k/s	6.2 k/s	1.2 k/s	178.3 k/s	167.5 k/s	9.8 k/s	384.2 Mb/s	348.2 Mb/s	34.8 Mb/s
10	1.6 k/s	1.2 k/s	330.7 /s	195.5 k/s	191.0 k/s	4.3 k/s	1.6 Gb/s	1.5 Gb/s	18.2 Mb/s
11	11.5 k/s	8.1 k/s	3.3 k/s	300.7 k/s	264.6 k/s	35.8 k/s	2.2 Gb/s	1.9 Gb/s	290.4 Mb/s
CELKEM	vše:	tcp:	udp:	vše:	tcp:	udp:	vše:	tcp:	udp:
	90.8 k/s	66.3 k/s	23.5 k/s	2714 k/s	2398 k/s	296 k/s	19.0 Gb/s	17.1 Gb/s	96.1 Mb/s

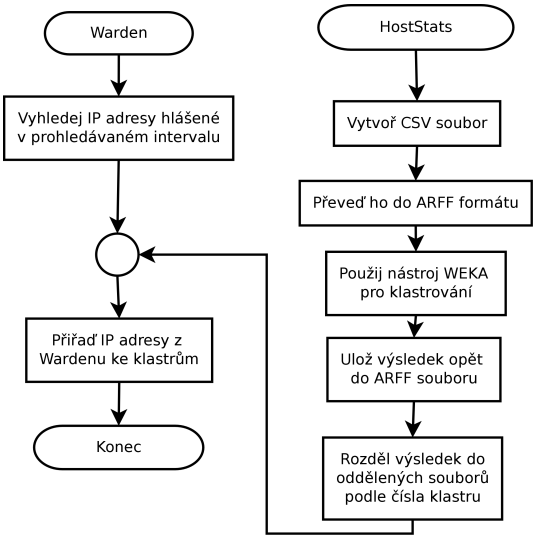
Tabulka 4.3: Objemy provozu na jednotlivých linkách.

přímo na vytvořené soubory dat, pro které je připraveno jednoduché uživatelské GUI prostředí, nebo je možné volat tyto algoritmy z vlastního Java kódu. WEKA obsahuje nástroje pro předzpracování dat, klasifikaci, klastrování, asociační pravidla a vizualizaci. Jedná se o volně šiřitelný software vydávaný pod GNU General Public License.

4.2 Normalizace dat

Aby bylo možné použít nástroj WEKA a výsledky následně analyzovat, byl vytvořen systém, jehož schéma je zobrazeno na obrázku 4.1.

Nejprve bylo zapotřebí uložit vypočítané host statistiky z aplikace HostStats do CSV souboru. To vyžadovalo úpravu aplikace HostStats. Před uložením do CSV souboru byla provedena normalizace dat do rozsahu <0, 1>. Následujícím krokem byla konverze souboru z formátu CSV do formátu ARFF, který je podporovaným formátem nástroje WEKA. V tomto nástroji byl na výsledný datový soubor použit klastrovací algoritmus, jehož výsledek byl opět uložen do ARFF souboru. Výsledek klastrovacího algoritmu již obsahoval příznak, do jakého klastru daná IP adresa spadá. Tento výsledný soubor byl následně rozdělen podle čísla klastru do oddělených souborů. Pro efektivní zjišťování anomálních klastrů byly využity informace od firem nacházejících se uvnitř sítě CESNET. Tyto firmy provádějí vlastní detekci útoků a informace o těchto útocích následně odesílají do Wardenu. Warden, jakožto systém založený na architektuře klient-server, umožňuje jednotlivým klientům zasílat detekované útoky jako události, jež se na straně serveru uloží do logů. Zasílané události obsahují například informace o tom, jaká služba zaslala danou událost, kdy byl útok detekován, jakého typu byl útok, jeho anomální skóre atp. Díky těmto informacím bylo možné vyhledat útoky (a k nim přislušející IP adresy), které proběhly v námi analyzovaném intervalu. V posledním kroce byly všechny nalezené IP adresy z Wardenu přiřazeny do výsledných klastrů získaných z nástroje WEKA. Pomocí tohoto systému bylo možné dosáhnout efektivního vyhodnocování výsledků získaných z jednotlivých testů. Nicméně procházení jednotlivých výsledků a zkoumání, zda výstup z klastrovací metody odpovídá nalezeným anomáliím z Wardenu, bylo časově velmi náročné.



Obrázek 4.1: Schéma popisující zpracování a analýzu dat.

4.3 Experimentální práce s klastrovacími algoritmy

Během načítání souboru arff nástrojem WEKA došlo k selhání tohoto nástroje. Důvodem byla velikost vstupního souboru, který obsahoval kompletní statistiky hostů z daného pětiminutového intervalu, ve kterém se jich nacházelo cca od 2,5 do 3 milionů. Objem dat byl v tomto souboru příliš velký na to, aby ho byl nástroj WEKA schopen zpracovat. Pro redukci objemu vstupního souboru byl vytvořen filtr, který odfiltrovával neužitečná data. Pravidla[1] vytvořeného filtru se nacházejí v příloze E.2. Použitím vytvořeného filtru došlo ke snížení počtu výstupních host statistik na cca 200 - 300 tisíc. Tento objem dat již bylo možné zpracovat nástrojem WEKA.

Během počátečního testování bylo použito několik klastrovacích algoritmů. Jmenovitě se jednalo o algoritmy Cobweb, DBSCAN, HierarchicalClusterer a K-means. Všechny algoritmy byly použity nad stejným datovým souborem a prováděly klastrování nad čtyřmi zvolenými dimenzemi (odchozí počet toků, SYN příznaků, ACK příznaků a unikátních IP adres). První tři výše zmíněné algoritmy nebyly schopny podat výsledek v reálném čase. Jako jediný K-means algoritmus umožňující nastavit maximální počet klastrů, byl schopen podat výsledek během několika minut. Tímto byl tento algoritmus zvolen pro využití v tomto projektu. Detailnější popis zmíněné metody se nachází v kapitole 5.4.

V první fázi bylo provedeno několik testů, které zahrnovaly všechny dimenze[2] a experimentálně byl zvolen maximální počet klastrů 15. Výsledky získané z těchto testů nebyly pozitivní. Většina detekovaných útoků, které byly k dispozici z Wardenu, se nacházela v těch největších klastrech. Z tohoto důvodu byl výsledek klastrovací metody nepoužitelný. Útoky, které se na síti vyskytly, byly přiřazeny do klastrů, které obsahovaly z větší části pouze normální provoz, a proto nebylo možné takovýto klastr označit za anomální. Z výsledků těchto testů bylo možné odvodit, že klastrovací metody nejsou schopny korektně rozpoznat, jakou váhu mají přikládat jednotlivým dimenzím. Z tohoto důvodu bylo nutné klastrovací metodě předávat určitou znalost o daném problému, který by měl být v datech vyhledáván.

[1] Jednotlivé statistiky použité v pravidlech se nacházejí v příloze E.1.

[2] Dimenzemi jsou myšleny jednotlivé statistiky, které jsou počítány aplikací HostStats. Všechny statistiky jsou popsány v příloze E.1.

Dimenze	**Počet klastrů**							
	15 (Série 1)		30 (Série 2)		75 (Série 3)		150 (Série 4)	
	AC	DIP [%]	AC	DIP [%]	AC	DIP [%]	AC	DIP [%]
out_syn_ack	0	0	2	4.1	3	6	-	-
out_fin	0	0	2	0.8	-	-	-	-
out_rst	0	0	1	0.8	5	7.2	6	24.2
in_syn	0	0	1	0.8	-	-	-	-
in_ack	0	0	0	0	4	6.4	-	-
in_fin	0	0	0	0	-	-	-	-
in_rst	0	0	1	0.8	4	6.4	6	25

Tabulka 4.4: Výsledky testů nad datovým souborem obsahujícím celý provoz.

Aby bylo možné tyto znalosti předat klastrovacímu algoritmu, bylo zapotřebí zjistit jaké útoky se v datech nacházejí a jak se v NetFlow datech projevují. Po prozkoumání dostupných informací z Wardenu se zjistilo, že nejčastější anomálií, která byla hlášena do Wardenu, bylo skenování portů. Dále se také jednalo například o útoky hrubou silou či slovníkové útoky. Nejdříve bylo nutné tyto typy útoků nastudovat a zjistit, jak se projevují v NetFlow datech. Tyto útoky jsou popsány v kapitole 3.5. Po detailnějším prozkoumání NetFlow záznamů, které obsahovaly zmiňované útoky, bylo objeveno několik hlavních ukazatelů na tyto útoky. Patří mezi ně příznaky, které se během komunikace posílají, a také počty toků a počty unikátních IP adres, které se v daném pětimutovém intervalu zaznamenaly pro daného hosta.

Ve statistikách, které byly získány po zpracování NetFlow záznamů aplikací HostStats, bylo možné pozorovat zmíněné ukazatele pro skenování portů. Tento útok se v host statistikách vyznačoval vysokým počtem odchozích SYN příznaků a malým počtem odchozích ACK příznaků a nízkým počtem odchozích FIN příznaků. Co se týče příchozích příznaků, bylo možné sledovat zvýšený počet ACK příznaků a zvýšený počet RST příznaků. Za hlavní ukazatele na skenování je tedy možné považovat počty různých příznaků vyskytujících se v době komunikace. Dimenze odpovídající jednotlivým příznakům byly používány v následujících testech.

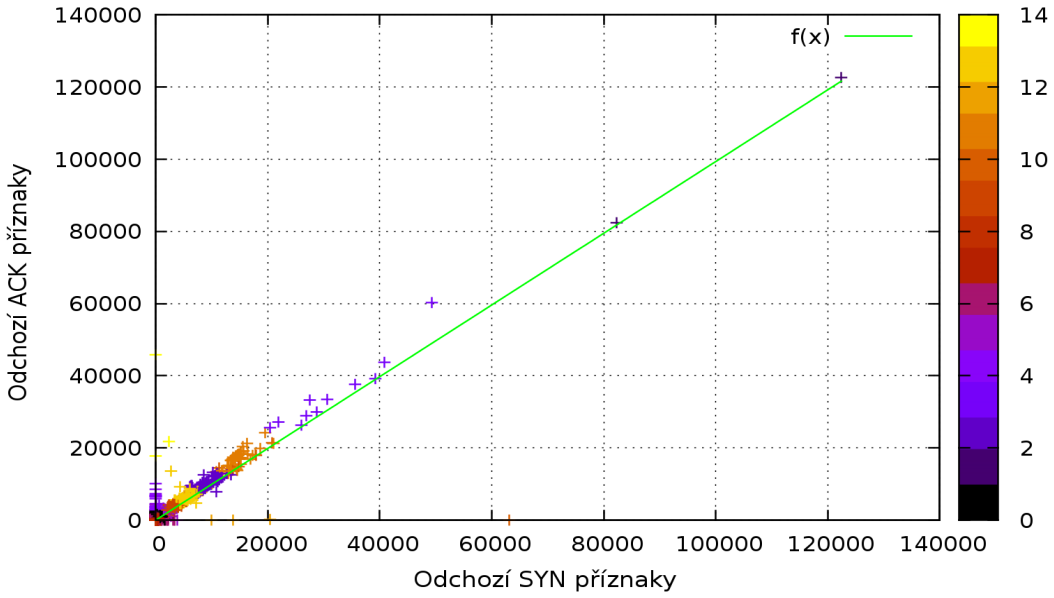
V uvedené tabulce 4.4[3] se nacházejí výsledky jednotlivých sérií testů. Sloupec AC odpovídá počtu klastrů, které obsahovaly alespoň 80% anomálních záznamů[4]. Ve sloupci DIP je zobrazena hodnota vyjádřena vzorcem 4.1. Tato hodnota odpovídá procentuálnímu zastoupení IP adres z Wardenu, které se nacházejí v nalezených anomálních klastrech (AIP), vůči celkovému počtu IP adres hlášených z Wardenu (WIP).

$$DIP = \frac{AIP}{WIP} * 100 \tag{4.1}$$

První série testů probíhala nad konstantním nastavením maximálního počtu klastrů na 15 a byl rozšiřován pouze počet dimenzí. Jednalo se tedy o 7 testů. Po prozkoumání jednotlivých testů byl výsledek negativní. V žádném z případů nebyl nalezen ani jeden klastr,

[3]Dimenze uvedené v jednotlivých řádcích jsou vždy rozšířením dimenzí předchozího řádku. Například řádek 3. bude obsahovat dimenze out_syn_ack_fin_rst.

[4]Anomální záznam je IP adresa, která představovala útočníka nebo oběť během konkrétního útoku.



Obrázek 4.2: Vizualizace výsledků klastrovacího algoritmu K-means nad statistikami vypočítanými z celkového provozu a zařazenými do 15-ti klastrů.

který by obsahoval alespoň 80% anomálních záznamů. Většina IP adres dostupných z Wardenu se nacházela v těch největších klastrech. Po detailnějším prozkoumání vizualizace 4.2, která znázorňuje rozmístění jednotlivých bodů do klastrů, je možné vypozorovat, že těžiště klastrů se nejvíce nacházejí na přímce, která je lineárně závislá na počtu odchozích SYN příznaků a počtu odchozích ACK příznaků. Z grafu je také patrné, že se na ose X nacházejí rovněž body, které nemají žádné odchozí ACK příznaky. S největší pravděpodobností se bude jednat o skenování portů. Obdobná situace se vyskytuje na ose Y, kde je možné vidět body mající pouze odchozí ACK příznaky a žádné odchozí SYN příznaky. Abychom dosáhli větší variace rozložení jednotlivých klastrů, bylo potřeba provést další sérií testů se zvýšeným maximálním počtem klastrů.

Druhá série testů probíhala s konstantním nastavením maximálního počtu klastrů na 30 a rozšiřován byl pouze počet dimenzí. Ze získaných výsledků bylo možné odvodit, že zvětšením maximálního počtu klastrů se již v nalezených klastrech nacházely takové, které obsahovaly alespoň 80% anomálních záznamů. Většina IP adres z Wardenu byla ale stále rozmístěna do větších klastrů a pouze zlomek procenta z těchto adres se nacházel v nalezených anomálních klastrech. Za zmíňku ještě stojí, že nalezené anomální klastry byly velice malé. Řádově obsahovaly jednotky záznamů. Opět bylo potřeba provést další sérií testů se zvýšeným maximálním počtem klastrů.

Třetí série testů probíhala s konstantním nastavením maximálního počtu klastrů na 75 a rozšiřován byl pouze počet dimenzí. Vzhledem k narůstajícímu času výpočtu modelu, bylo nutné zredukovat počet testů ze 7 na 4. U testů, které nebyly prováděny, je v kolonkách v tabulce 4.4 u těchto testů pomlčka. Po provedení těchto testů a po prozkoumání výsledků se oproti druhé sérii testů počet klastrů, obsahujících anomální záznamy, opět zvýšil. Nicméně se stále velký počet IP adres z Wardenu objevoval ve větších klastrech. Došlo ale k mírnému zvýšení procentuálního zastoupení těchto adres v nalezených anomálních klastrech. Z tohoto důvodu byla provedena čtvrtá série testů s opětovným navýšením maximálního počtu klastrů.

Ve čtvrté sérii testů byl maximální počet klastrů zdvojnásoben oproti předchozí sérii na 150. Díky této změně bylo ve výsledku nalezeno relativně velké množství správně označených anomálních klastrů. Rovněž IP adresy dostupné z Wardenu byly rozloženy do menších klastrů, a tímto vypomáhaly při zkoumání, zda je daný klastr opravdu anomální, či nikoliv. Také se zvýšila hodnota DIP, která poukazuje na zvyšení počtu IP adres z Wardenu nacházejících se v anomálních klastrech.

Nicméně procento IP adres z Wardenu v anomálních klastrech bylo poměrně nízké, přestože bylo dosaženo výrazného zvýšení této hodnoty v porovnání s předchozí sérií testů. Po dalším prozkoumání anomálních klastrů se zjistilo, že se v těchto klastrech často nacházely i záznamy, které odpovídaly normálnímu provozu. Tyto záznamy se ve většině případů vztahovaly na provoz směřující na známější porty. Jednalo se například o provoz na portu 80, 443 nebo 22. Služby běžící na těchto portech patří k těm významnějším, které se na síti využívají. To se rovněž promítlo do výsledků, které byly získány použitím klastrovacího algoritmu. Například, pokud se v síti nachází větší webový server, je velmi pravděpodobné, že objem provozu i počty zasílaných příznaků v době komunikace s tímto serverem budou poměrně výrazné a může nastat situace, kdy příznaky známé aplikace budou odpovídat anomálním příznakům jiné aplikace. Toto zjištění dalo podnět k tomu, abychom prošli jednotlivé IP adresy dostupné z Wardenu a zjistili, kvůli jakým útokům byly tyto IP adresy nahlášeny a na jaké porty směřovaly. Po provedení této analýzy se zjistilo, že 21% všech IP adres hlášených do Wardenu směřovalo právě na porty 80, 443 nebo 22. Dále několik nahlášených útoků bylo velmi malých a některé byly falešně pozitivní, a proto nebyly klastrovací metodou přiřazeny žádnému anomálnímu klastru.

Za účelem zvýšení přesnosti klastrovacího algoritmu, bylo zapotřebí navrhnout a implementovat zásuvný modul do aplikace HostStats, který bude vytvářet statistiky hostů a zároveň je dělit do skupin podle čísla portu a protokolu. Rozdělením síťového provozu by mělo dojít k odfiltrování nejobjemnějších toků, a tím k odstranění těchto záznamů z celkového TCP provozu. Celý návrh a implementace se nachází v podkapitole 5.

4.4 Testy nad daty rozdělenými podle služeb

Po implementaci navrženého zásuvného modulu byla získaná data rozdělena následovně[5]:

- **<80, 6>** - jedná se o statistiky z TCP provozu na portu 80,
- **<443, 6>** - jedná se o statistiky z TCP provozu na portu 443,
- **<22, 6>** - jedná se o statistiky z TCP provozu na portu 22,
- **<0, 6>** - jedná se o statistiky ze zbylého TCP provozu,
- **<0, 17>** - jedná se o statistiky z UDP provozu.

Po rozdělení statistik do skupin podle <port, protokol> byly provedeny následující testy, které byly prováděny nad jednotlivými skupinami. Z předchozí analýzy, kdy testovaná data nebyla rozdělená do skupin, bylo z výsledků možné vyvodit, že nejlepším nastavením dimenzí je použití odchozích a příchozích SYN, ACK, FIN a RST příznaků. Následující testy byly prováděny nad stejnými počty klastrů, jako u předchozích testů, s vynechaním počtu 15. Tabulka 4.5 zobrazuje výsledky jednotlivých testů pro konkrétní <port, protokol>, které

[5]Zápis vyskytující se v seznamu: <port, protokol>

<port, protokol>	Počet klastrů					
	30		75		150	
	AC	DIP [%]	AC	DIP [%]	AC	DIP [%]
<0, 6>	5	12.7	8	58.2	9	58.2
<443, 6>	0	0	1	83.3	2	87.5
<22, 6>	1	0	1	33.3	1	33.3
<80, 6>	0	0	0	0	1	40

Tabulka 4.5: Výsledky testů nad jednotlivými skupinami.

byly prováděny v nástroji WEKA. Hodnoty AC odpovídají počtu klastrů, které obsahovaly alespoň 80% anomálních záznamů. Hodnota DIP odpovídá procentuálnímu zastoupení IP adres z Wardenu, které se nacházejí v nalezených anomálních klastrech, vůči celkovému počtu IP adres hlášených z Wardenu směřujících na konkrétní port, případně jedná-li se o skupinu <0, 6> , všech adres nesměřujících na odfiltrované porty 80, 443 a 22.

První série testů probíhala nad skupinou <0, 6>. Výsledky těchto testů jsou zobrazeny v prvním řádku tabulky 4.5. Z výsledků je patrné, že odfiltrovaný provoz na porty 80, 443 a 22 měl na výsledky výrazný vliv. Již při 30-ti klastrech bylo nalezeno pět klastrů, které byly anomální a zároveň se v těchto klastrech nacházelo téměř 13% IP adres z Wardenu[6]. Tento výsledek je neuspokojivý, nicméně při zvýšení počtu klastrů u klastrovacího algoritmu na 75 a 150 se situace výrazně změnila.

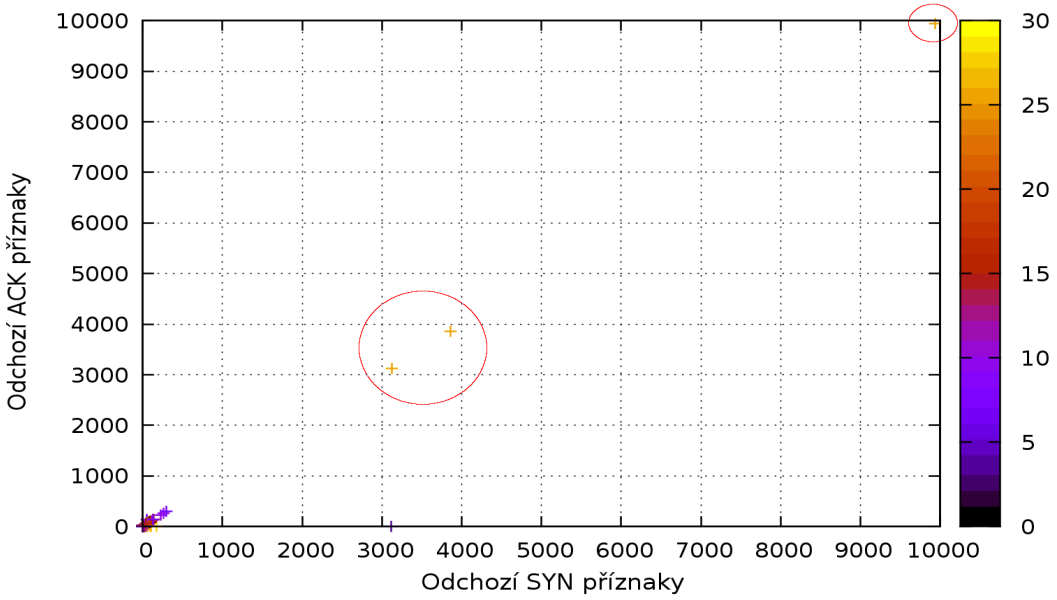
Ze statistik, které jsou zobrazeny v tabulce 4.5 vyplývá, že zvýšením počtu klastrů ze 75 na 150 již nedošlo k výraznějšímu zlepšení. Nicméně po detailnějším prozkoumání bylo možné zjistit, že při zvýšení počtu klastrů na 150 došlo k výraznému pročištění anomálních klastrů od normálního provozu. V nalezených anomálních klastrech se již nacházelo více jak 90% anomálních záznamů a v některých dokonce 100%. Taky bylo možné pozorovat, že zvýšením počtu klastrů došlo zároveň ke zvýšení procentuálního zastoupení IP adres z Wardenu v nalezených anomálních klastrech.

Druhá série testů probíhala nad skupinou <443, 6>. Při nastavení počtu klastrů na 30 nebylo možné ve výsledcích nalézt ani jeden klastr, který by byl anomální. To se změnilo u zvýšeného počtu klastrů. Zde se již objevil jeden klastr, který obsahoval poměrně velký počet anomálních záznamů. Po bližším prozkoumání se tyto záznamy ukázaly jako skenování portů. V tomto klastru se rovněž nacházelo přes 87% IP adres z Wardenu[7].

Třetí série testů probíhala nad skupinou <22, 6>. Počáteční výsledky z těchto testů nebyly uspokojivé. Na první pohled se ve výsledcích nenašly žádné anomální klastry, které by obsahovaly nějakou IP adresu z Wardenu. Takový klastr se objevil pouze v testech, kde byl počet klastrů nastaven na 75 nebo 150. Po prozkoumání výsledků při nastaveném počtu klastrů na 75 a 150 se zjistilo, že maximální počet klastrů, do kterých se provoz rozdělil byl 64. Z toho vyplývá, že nastavovat více než zmíněných 64 klastrů nemá význam.

Při prozkoumání vizualizace z nástroje WEKA, byly nalezeny vyčnívající body mimo normální provoz viz obrázek 4.3. Jednalo se o tři body vyznačené v červených kruzích, které spadají do jednoho klastru. Při prozkoumání tohoto klastru a IP adres v něm obsažených se zjistilo, že jednotlivé IP adresy obsahovaly velký počet toků s příznaky kompletního navázání komunikace. S největší pravděpodobností se tedy jednalo o kontinuální hádání hesla. Pro ověření této domněnky byly prozkoumány i okolní pětiminutové intervaly. Cel-

[6] Jedná se o počet detekovaných IP adres, jejichž útok nesměřoval na porty 80, 443 a 22.
[7] Jedná se o počet detekovaných IP adres, jejichž útok směřoval na port dané skupiny.



Obrázek 4.3: Vizualizace statistik vypočítaných z provozu směřujícího na port 22 a zařazených do 30-ti klastrů.

kový zkoumaný interval byl 40 minut. Rovněž v těchto okolních intervalech bylo možné nalézt zkoumané IP adresy, které obsahovaly velké množství toků směřujících na port 22 a příznaky z kompletního navázání komunikace. Tímto se náš předpoklad potvrdil a pomocí klastrovací metody bylo možné detekovat hádání hesla, které se vyznačovalo poměrně velkým počtem všech odchozích TCP příznaků a velkým počtem toků. Tento útok byl rozdistribuován mezi velký počet cílových IP adres a byl prováděn po dlouhý časový úsek. Proto jej bylo možné nalézt i v okolních pětiminutových intervalech. Delší interval než zmíněných 40 minut nebyl zkoumán, ale s velkou pravděpodobností probíhal tento útok mnohem déle. Podrobný výpis z nástroje nfdump se nachází v příloze F.1.

Hádání hesla se v případě portu 22 projevovalo v NetFlow datech jako velké množství příznaků odpovídajících kompletnímu navázání spojení. Pomocí klastrovací metody, využívající příchozí a odchozí SYN, ACK, FIN a RST příznaky, bylo možné tyto anomálie na portu 22 přiřadit do jednoho klastru. Vzdálenost tohoto klastru od počátku souřadnicového systému byla při použití zmíněných dimenzí výrazně větší než vzdálenost klastrů obsahujících normální provoz. Je tedy možné považovat klastry příliš vzdálené od počátku souřadnicového systému za anomální. Z důvodu tohoto předpokladu byla kontrola vzdálenosti klastrů aplikována i na obecnou skupinu <0, 6> a rovněž na skupinu <443, 6>. V obecné skupině obsahující pouze TCP provoz byla nalezena podobná anomálie, která se vztahovala ke službě RDP běžící na portu 3389. Jednalo se rovněž o hádání hesla k této službě. Útoky směřující na port 22 a na port 3389 se nenacházely v hlášeních z Wardenu. Podobná anomálie se rovněž vyskytla ve skupině <443, 6>. Nicméně během provádění testů výsledné aplikace se zjistilo, že se jedná o falešně pozitivní detekci. Zdůvodnění se nachází v kapitole 6.3.

Poslední čtvrtá série testů probíhala nad skupinou <80, 6>. Z výsledků bylo možné vyvodit, že pro tento provoz by bylo zapotřebí nalézt jiná nastavení, která by se více týkala počtu přenášených bajtů a paketů či počtu toků směřující na tento cílový port. Důvodem jsou útoky na port 80 mající za cíl odepření této služby, ne její prozkoumání či uhád-

nutí hesla. Útočníci mají většinou v tomto případě vytipovaný například webový server, o kterém vědí, že na něm tato služba běží. Jak je z výsledků v tabulkce 4.5 vidět, klastrovací metoda s testovaným nastavením nebyla schopna nalézt téměř žádné anomální klastry v této skupině. Při počtu klastrů 150 byl nalezen jeden klastr, který byl označen jako anomální. Nicméně s velkou pravděpodobností se jednalo o chybu známého serveru, protože nalezený anomální klastr obsahoval v drtivé většině pouze záznamy s malým počtem toků směřujících na tento konkrétní server.

4.5 Zhodnocení provedené analýzy dat

Ve fázi analýzy dat byl zvolen nástroj WEKA, pomocí kterého bylo možné získaná data klastrovat a následně výsledky vizualizovat a ukládat pro další zpracování. Rovněž v rámci analýzy bylo vytvořeno několik skriptů, které výrazně pomohly při automatizaci zpracovávání výsledků obdržených z nástroje WEKA.

V rámci testování byla nejdříve provedena řada testů nad datovým souborem obsahující statistiky z aplikace HostStats, které nebyly děleny do skupin podle služeb. Jednalo se o statistiky jednotlivých hostů, spočítané z celkového provozu. Data pocházející ze sítě CESNET obsahovala poměrně velký počet těchto statistik, takže bylo nutné hned na začátku vytvořit jednoduchý filtr, který odfiltroval nevýznamné informace, a tím snížil počet statistik v daném datovém souboru. Po provedení několika počátečních testů byl zvolen klastrovací algoritmus K-means, který byl jako jediný schopen provádět klastrování v reálném čase. Tento algoritmus byl následně používán ve všech testech.

Výsledky testů nad datovým souborem obsahujícím statistiky z celkového provozu, které se nacházejí v tabulce 4.4, nebyly uspokojivé. Ze získaných výsledků bylo možné vyvodit, že dalším krokem ke zpřesnění detekce anomálií, která využívá klastrovací algoritmus, bude rozdělení získávaných host statistik do skupin podle dvojic <port, protokol>.

Po navržení a implementaci zásuvného modulu do aplikace HostStats byl získán datový soubor, který již měl vypočítané statistiky rozděleny do skupin podle dvojic <port, protokol>.

Parametr	Hodnota
displayStdDevs	false
distanceFunction	EuclideanDistance
dontReplaceMissingValues	false
maxIterations	500
numClusters	XX
preserveInstancesOrder	false
seed	10

Tabulka 4.6: Nastavení parametrů klástrovacího algoritmu SimpleKmeans v nástroji WEKA.

Následovala řada dalších testů již nad konkrétními skupinami. Z výsledků, které se nacházejí v tabulce 4.5, je možné vidět, že rozdělení statistik do skupin podle konkrétních služeb se ukázalo jako správné rozhodnutí. Odfiltrováním provozu směřujícího na porty 80, 443 a 22 bylo dosaženo pročištění zbylého provozu na protokolu TCP. Toto umožnilo nalézt ve výsledcích klastrovacího algoritmu takové klastry, které byly zjevně anomální. Rovněž u jednotlivých skupin, vyjma portu 80, byly nalezeny klastry, které bylo možné označit jako

anomální. Také IP adresy dostupné z Wardenu byly z velké části přiřazeny do klastrů, které byly anomální. Toto zjištění bylo velmi důležité z toho důvodu, že informace z Wardenu byly prozatím jediným možným způsobem jak automaticky určit, zda konkrétní nalezený klastr je anomální, či nikoliv.

Všechny provedené testy byly prováděny minimálně dvakrát. Nastavení parametrů klastrovacího algoritmu K-means se nacházejí v tabulce 4.6. Při opakovaném testu byl výsledek vždy identický s testem, který měl stejně nastavené parametry a byl prováděn nad stejnou datovou sadou se stejnými dimenzemi.

Rovněž byla provedena řada testů, ve kterých byly kombinovány různé dimenze mezi sebou. Výsledkem bylo zjištění, že použitím odchozích a příchozích SYN, ACK, FIN a RST příznaků bylo ve většině případů dosaženo nejlepších výsledků.

Kapitola 5

Návrh aplikace

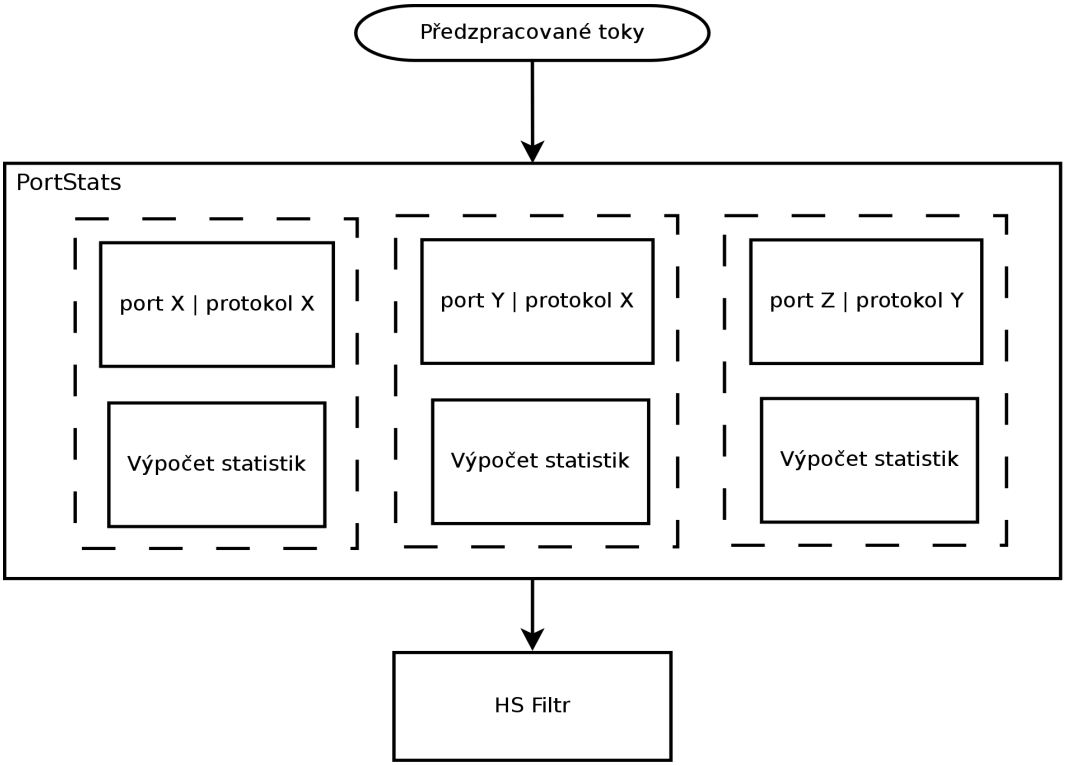
V této podkapitole se budeme věnovat návrhu aplikace pro detekci anomálií s využitím NetFlow dat. Aplikace by měla být schopna vytvořit model, podle kterého bude následně nově příchozí provoz rozdělovat na normální a anomální. V této práci bude použit klastrovací algoritmus K-means, kterým se bude vytvářet zmíněný model. Klastrovací metoda byla zvolena primárně z toho důvodu, že trénovací data, která budou pocházet ze sítě CESNET, nebudou žádným způsobem označená. Jedinými dostupnými informacemi jsou informace z Wardenu, kde jsou hlášeny útoky od spolupracujících firem nacházejících se uvnitř sítě CESNET. Tyto informace budou následně využity pro semi-automatizovaný přístup při určování, zda je nějaký konkrétní nalezený klastr anomální, či nikoliv.

5.1 Raw data a extrahování vlastností

Prvním krokem detekce anomálií je získání a předzpracování dat. Data jsou získávána z NetFlow exportérů, které zasílají data ve standardizovaném formátu blíže popsaném v kapitole 2.4. Dále jsou získaná data předzpracována tak, že jsou pro každou IP adresu vytvořeny požadované statistiky jako například počet toků, bajtů, paketů atp. V této práci využijeme již existující systém HostStats, na který se napojíme. Tento systém získává data ze sítě CESNET, ze kterých následně vytváří statistiky pro každou IP adresu. Využívá rovněž aplikaci NfSen pro grafické zobrazení získaných dat. Úplný seznam všech statistik, které se počítají pro jednotlivé IP adresy se nacházejí v příloze E.1.

5.2 Zásuvný modul PortStats

Ve fázi analýzy dat, popsané v kapitole 4, bylo provedeno několik sérií testů nad daty, která obsahovala souhrnné statistiky. Tyto statistiky byly vypočítány z nerozděleného provozu, pouze s odfiltrovanými nevýznamnými informacemi. Tato data byla získána z aplikace HostStats. Z výsledků bylo možné vyvodit, že bude nutné vytvořit zásuvný modul do aplikace HostStats, který bude provoz dělit podle dvojic <port, protokol>. To znamená, že jednotlivé vypočítané statistiky by měly být zařazeny do skupiny kodle konkrétního protokolu <TCP, UDP> a podle čísla portu. Jak již bylo zmíněno v kapitole 4, bylo vybráno několik skupin, do kterých se budou jednotlivé statistiky zařazovat. Jmenovitě se jednalo o skupiny nad protokolem TCP a porty <80, 443, 22> a dále dvě obecné skupiny pouze podle protokolu <TCP, UDP>. Důvodem je, že normální a anomální provoz vypadá odlišně v závislosti na konkrétní službě.



Obrázek 5.1: Diagram zásuvného modulu portstats.

Bylo tedy nutné vytvořit návrh zásuvného modulu, který bude předzpracovaná data rozdělovat do předem zvolených skupin. Diagram návrhu je možné vidět na obrázku 5.1. Zásuvný modul na vstupu obdrží předzpracovaná data v podobě konkrétního toku, který byl předem zagregován z více linek. Následně proběhne kontrola čísla portu a čísla protokolu, a pokud taková skupina neexistuje, použije se obecná skupina pouze podle čísla protokolu. Po nalezení skupiny se pokusíme ve statistikách hostů přiřazených do této skupiny nalézt záznam s hostem, ke kterému náleží právě zpracovávaný tok. Pokud záznam nalezneme, provedeme aktualizaci statistik tohoto hosta, pokud záznam nenalezneme, vytvoříme záznam nový. Nicméně po nastudování zdrojových kódů aplikace HostStats jsem dospěl k závěru, že koncept, který bych chtěl použít se bohužel neshoduje se stávající podobou aplikace HostStats. Po konzultaci se svým konzultantem jsme se dohodli, že si vytvořím vlastní větev této aplikace a upravím si ji dle libosti.

5.3 Modul pro detekci anomálií

Nyní si popíšeme návrh modulu pro detekci anomálií s využitím klastrování. Tento modul se bude skládat ze čtyř vláken, která budou mezi sebou komunikovat pomocí sdílené paměti viz obrázek 5.2. Rozdělení do vláken je navrženo proto, aby jednotlivé části modulu mohly pracovat nezávisle na sobě.

- **Detektor Anomálií** - jedná se o hlavní vlákno celé aplikace. Toto vlákno bude řídit běh dalších třech vláken a rovněž bude použito jako sdílená pamět mezi jednotlivými vlákny. Bude udržovat ukazatele na jednotlivá vlákna a informace z Wardenu, které budou využívány vláknem "Výpočet Detekčního Modelu". Dále bude udržovat

ukazatele na aktuálně vypočítané statistiky a detekční modely, které budou využívat vlákna "Výpočet Detekčního modelu" a "Detekce Anomálií". Při inicializaci bude probíhat kontrola, zda existuje uložený detekční model. Pokud ano, bude načten a po potvrzení obsluhou se začne používat. Rovněž bude v tomto vlákně probíhat prohledávání souborů nfcapd pro nalezené anomálie. Je to z toho důvodu, abychom mohli případně nahlášenou anomálií rychle ověřit. Toto vlákno bude taky zajišťovat hlášení nalezených anomálií do lokálního souboru.

- **Výpočet Detekčního Modelu** - toto vlákno bude vypočítávat detekční model a bude se skládat ze dvou částí. První část bude pomocí klástrovacího algoritmu K-means, který je blíže popsán v podkapitole 5.4, vypočítávat požadované modely k jednotlivým skupinám <port, protokol>. Nastavení klastrovacího algoritmu bude možné načítat z externího umístění, kde bude možné nastavit jednotlivé parametry pro každou skupinu zvlášť. Po výpočtu jednotlivých modelů se ve druhé části použijí externí informace z Wardenu pro označení klastrů, zda jsou anomální, či nikoliv. Případně bude možné využít UI rozhraní pro manuální označení klastrů. Následně bude potřeba pomocí UI potvrdit vypočítaný model, a tím dojde k výměně stávajícího modelu za nový. Tento nový model se následně začne používat ve vláknu "Detekce Anomálií".

- **Detekce Anomálií** - v tomto vlákně bude probíhat detekce anomálií podle aktuálně vypočítaného modelu. Pro jednotlivé statistiky bude vypočítávána vzdálenost k jednotlivým klastrům. Následně bude provedena kontrola, zda se porovnávaný záznam nachází v normálním či anomálním klastru. Popis klasifikace a detekce outsiderů založené na vzdálenosti naleznete v podkapitole 5.4.

- **UI** - toto vlákno bude zajišťovat komunikaci mezi modulem a uživatelskou aplikací, pomocí které bude možné provádět různá nastavení běžícího modulu.

5.4 K-means klastrování

Jak již bylo zmíněno v textu výše, bude v tomto projektu využit klastrovací algoritmus K-means, který bude nyní blíže popsán.

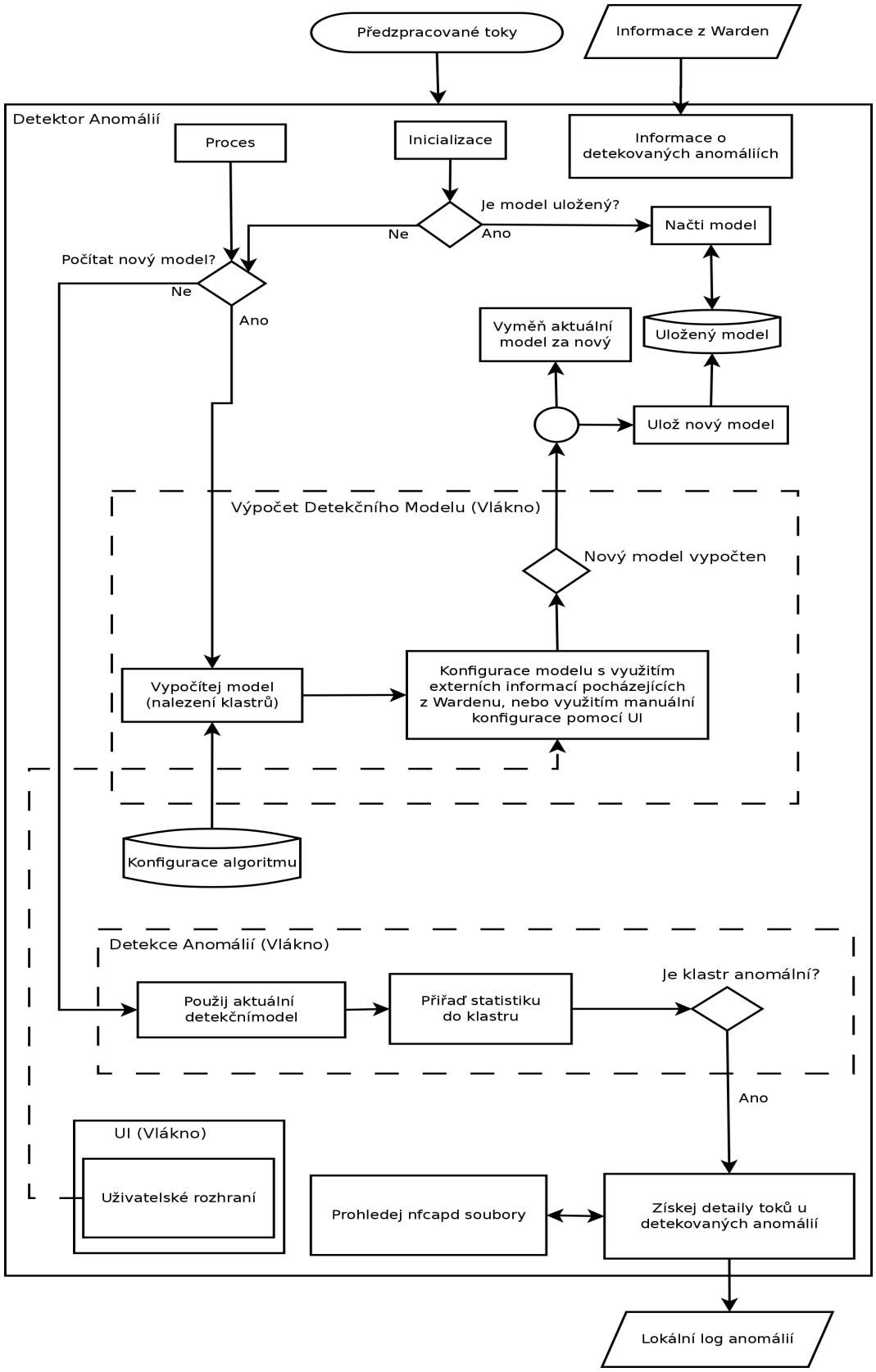
K-means klastrování je algoritmus pro analýzu dat, který seskupuje objekty na základě hodnot jejich vlastností a rozděluje je do K disjunktních klastrů. Objekty, které jsou klasifikovány do stejného klastru mají podobné hodnoty vlastností. K je kladné číslo, které určuje počet klastrů, do kterých se budou objekty zařazovat. Nyní budou vypsány základní kroky klastrovacího algoritmu K-means:

1. Definování počtu klastrů K.

2. Inicializace těžiště pro každý z K klastrů. To může být provedeno libovolně rozdělením všech objektů do K shluků, vypočítáním jejich těžišť a ověřením, že všechna těžiště jsou od sebe odlišná. Alternativou může být, že těžiště jsou inicializována podle K libovolně zvolených odlišných objektů.

3. Iterace přes všechny objekty a výpočet vzdáleností od těžiště všech klastrů. Přiřazení každého objektu do klastru, ke kterému je vypočítána vzdálenost nejkratší.

4. Přepočítání těžiště všech klastrů.



Obrázek 5.2: Diagram detektoru anomálií.

5. Opakování kroku 3., dokud se těžiště nepřestanou měnit.

Nezbytnou součástí je funkce pro výpočet vzdálenosti mezi dvěma objekty. Pro tento výpočet je často využívána Euklidovská vzdálenost, která je definována následovně:

$$d(x, y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{5.1}$$

kde $x = (x_1, \ldots, x_m)$ a $y = (y_1, \ldots, y_m)$ jsou dva vstupní vektory s m vlastnostmi.

V tomto projektu využijeme klastrovací algoritmus K-means na trénovací data z reálného provozu, který bude obsahovat jak normální, tak anomální provoz bez označení, o jaká data se jedná. Důvodem tohoto přístupu je předpoklad, že normální a anomální provoz tvoří různé klastry v prostoru vlastností. Jak již bylo zmíněno výše, klastrování bude použito zvlášť na předdefinované služby, které jsou identifikovány pomocí dvojic <port, protokol> nebo pro defaultní třídy obsahující rozdělení pouze podle protokolu.

Klastrovací algoritmus rozdělí trénovací data do K klastrů, ale neurčuje zda se jedná o normální či anomální provoz. Toto je třeba rozhodnout manuálně. My budeme rovněž používat semi-automatizovaného přístupu, který bude využívat informace o detekovaných anomáliích či útocích od spolupracujících firem, které se nacházejí uvnitř sítě CESNET, jak již bylo v textu zmíněno.

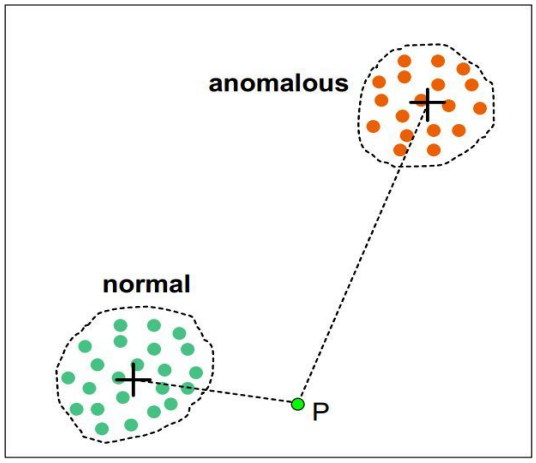
Klasifikace a detekce outsiderů

Výsledkem klastrování pomocí K-means jsou těžiště klastrů pro normální a anomální provoz, které mohou být využity pro detekci anomálií v nových tocích vyskytujících se na stejné síti. Nové toky musí být předzpracovány tak jako trénovací data, aby obsahovaly stejné vlastnosti. Pro detekci anomálií využijeme klasifikaci a detekci outsiderů. Obě metody jsou založené na porovnávání vzdáleností. Mohou být využity individuálně nebo kombinovaně.

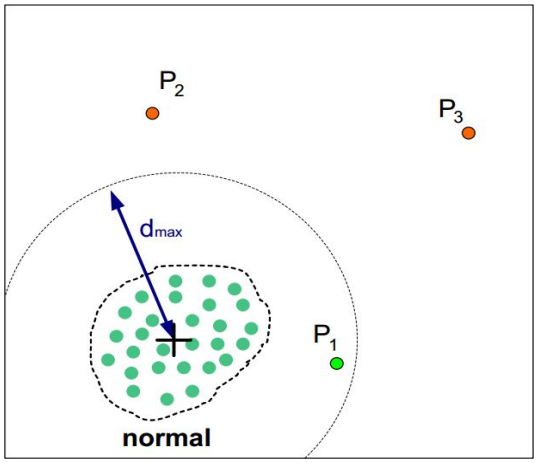
U klasifikace je vzdálenost vypočítána pomocí funkce vážené Euklidovské vzdálenosti. Objekt bude klasifikován jako normální, pokud bude jeho vzdálenost blíž těžišti normálního klastru než k těžišti anomálního klastru a opačně. Toto je možné vidět na obrázku 5.3a, kde jsou pro jednoduchost uvedeny pouze dva klastry, konkrétně pro normální a anomální provoz. Je vidět, že bod P se nachází blíž normálnímu klastru a je tedy označen za normální.

U detekce outsiderů je outsiderem objekt, jehož vzdálenost se významně liší od jiných objektů. Může být tedy označen jako anomální. Pro detekci outsiderů se využívá pouze výpočet vzdálenosti od těžiště normálního klastru. Pokud je vzdálenost mezi objektem a těžištěm větší než nastavený práh d_{max}, bude objekt považován za outsidera, a tím pádem za anomálii. Toto je možné vidět na obrázku 5.3b, kde vzdálenost bodu P_1 je menší než d_{max} a je tedy označen jako normální. Body P_2 a P_3 mají vzdálenost větší než d_{max} a jsou označeny za anomálie.

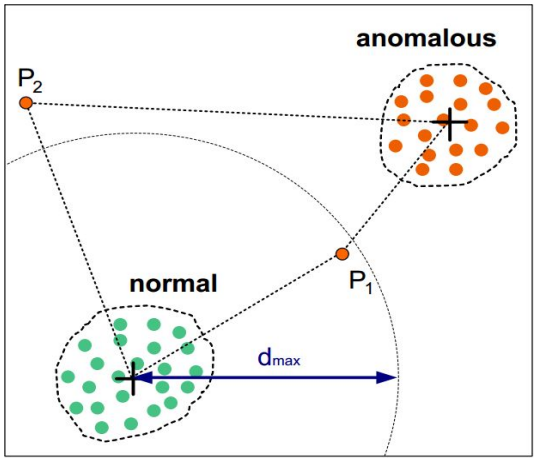
Kombinovaný přístup. Klasifikaci a detekci ousiderů je možné využít kombinovaně a vyhnout se tak omezením, která mají tyto přístupy při samostatném použití. Pokud jsou tyto metody použity zároveň, je objekt považován za anomální, pokud je jeho vzdálenost blíže k anomálnímu klastru, nebo je jeho vzdálenost od normálního klastru větší než nastavený práh. Toto je možné vidět na obrázku 5.3c, kde jsou použity oba přístupy, čímž jsou oba body P_1 i P_2 označeny jako anomálie [25].



(a) Klasifikace.



(b) Detekce outsiderů.



(c) Klasifikace a detekce outsiderů.

Obrázek 5.3: Možnosti detekce anomálií [25].

Kapitola 6

Implementace a výsledky experimentů

V této kapitole bude popsána realizace praktické části. Tato práce byla implementována v rámci aplikace HostStats, která je vyvíjena na Ústavu počítačových systémů UPSY. První částí realizace bylo vytvoření zásuvného modulu, který bude příchozí provoz rozdělovat podle čísla portu a podle protokolu. Následující částí, která je jádrem této práce, je realizace modulu pro detekci anomálií s využitím klastrovací metody. Celá aplikace bude implementována v jazyce C++.

6.1 Zásuvný modul PortStats

Po nastudování zdrojových kódů aplikace HostStats jsem zjistil, jak již bylo zmíněno v podkapitole 5.2, že koncept této aplikace nevyhovuje přímo mým potřebám, proto po dohodě se svým konzultantem jsem vytvořil novou větev této aplikace pro účely této práce.

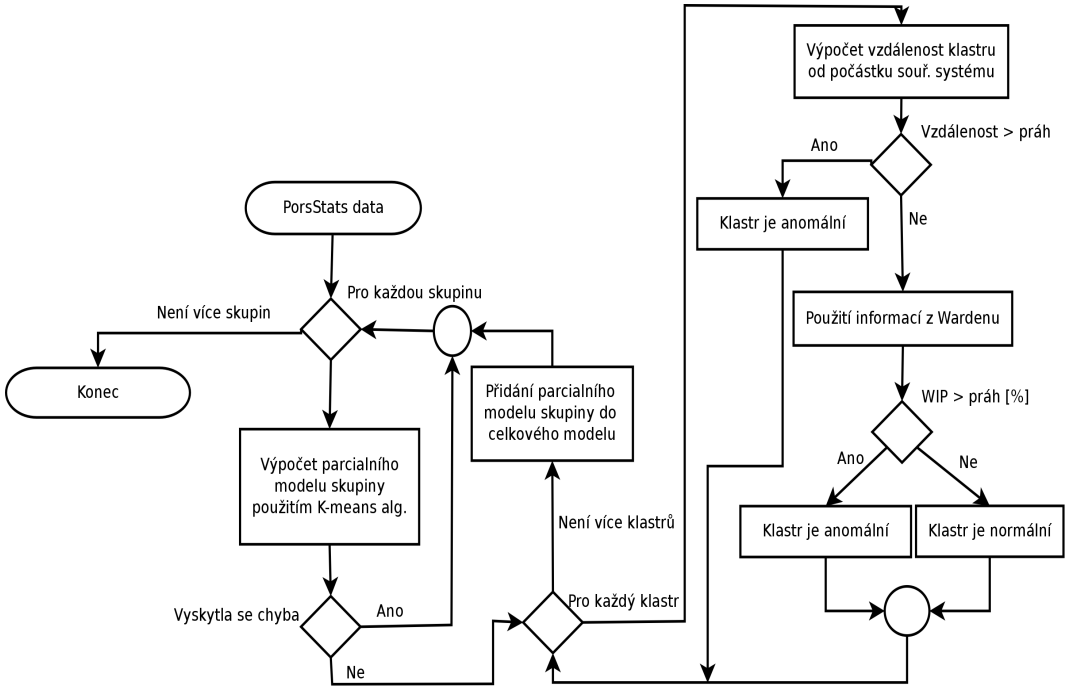
Pro uchovávání statistik je v aplikaci HostStats dostupný datový typ, který mapuje hosty a jejich statistiky. Definice tohoto typu s příslušnými strukturami se nachází v příloze E.3. Aby bylo možné mapovat jednotlivé statistiky do skupin, bylo potřeba vytvořit nový datový typ. Definice tohoto typu se rovněž nachází v příloze E.3. Z aplikace HostStats bylo použito stahování dat z kolektoru. Dále bylo použito čtení souborů, agregace z více linek a počítání statistik jednotlivých hostů. Počítání statistik bylo nutné upravit tak, aby vyhovovalo požadavkům zásuvného modulu, který dělil vypočítávané statistiky do skupin.

Zásuvný modul byl implementován podle návrhnu, který lze nalézt v podkapitole 5 viz obrázek 5.1. V aplikaci HostStats již existovala abstraktní třída `IPlugin`, kterou bylo potřeba upravit dle naších potřeb. Tato třída byla následně použita pro implementaci zásuvného modulu portstats reprezentovaného třídou `PortStats`. Pomocí veřejné metody `init`, která je zděděná z třídy `IPlugin`, dochází k inicializaci konkrétního zásuvného modulu. Následně voláním metody `process_flowmap_record` se zpracuje konkrétní záznam toku.

V zásuvném modulu portstats tedy dochází k aktualizaci či vytvoření nového záznamu konkrétního hosta umístěného do konkrétní skupiny podle <port, protokol>.

6.2 Modul pro detekci anomálií

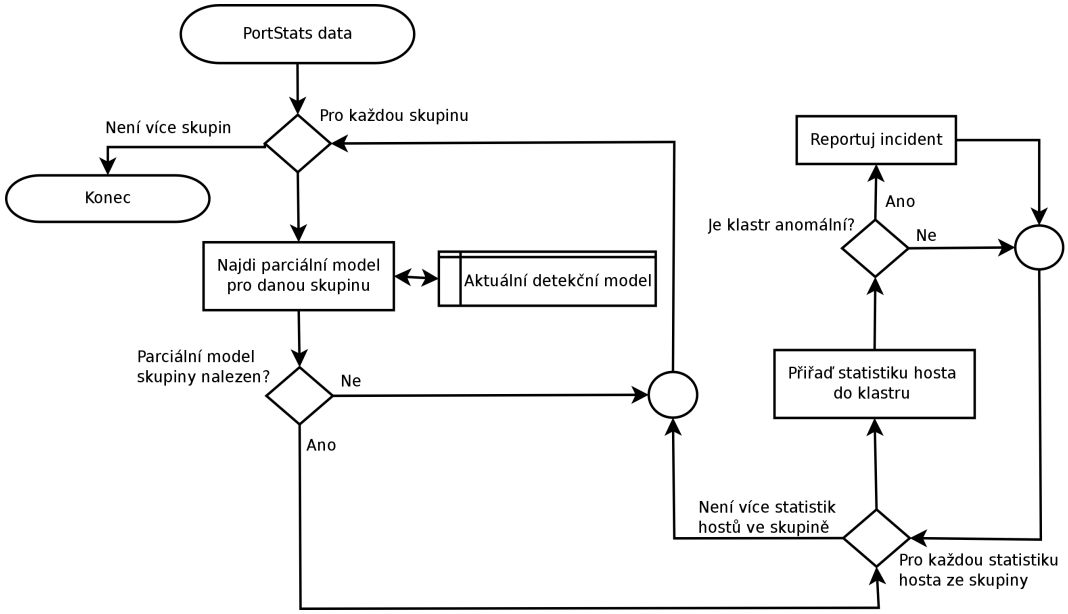
Realizace detekčního modulu je situována do několika tříd, které spravují jeho běh. Hlavní třídou je `AnomalyDetector`, která komunikuje s nadřazeným procesem pomocí veřejných



Obrázek 6.1: Logika výpočtu modelu.

metod. V době inicializace dojde ke kontrole, zda existuje uložený detekční model. Pokud takový model existuje, bude načten do paměti a označen jako nový. Nyní bude aplikace vyčkávat na vstup od obsluhy, která tento nový model potvrdí nebo zahodí. Dále tato třída řídí běh dalších třech vláken, která se starají o zpracování získaných statistik, ke kterým mají přístup právě prostřednictvím této třídy. Voláním veřejné metody `process_port_proto_hs_records`, které je předán ukazatel na aktuálně vypočítané statistiky, se spustí zpracovávání těchto statistik. V této veřejné metodě se spouští jedno vlákno pro detekci anomálií, které využívá pro detekci aktuálně vypočítaný model. Pokud se má počítat nový detekční model, spustí se pro tento účel druhé vlákno pomocí instance třídy `ComputeDetectorModel`. Třetím vláknem je `UI`, které zajišťuje komunikaci s klientskou aplikací.

Třída `ComputeDetectorModel` má za úkol vypočítávat detekční model. Pomocí její instance vytvořené ve třídě `AnomalyDetector`, je voláním veřejné metody `create_thread` vytvořeno nové vlákno. Toto vlákno začne zpracovávat dostupné statistiky a z nich následně vypočítá nový detekční model. Princip fungování tohoto výpočtu se nachází na obrázku 6.1. Na vstupu dostaneme dostupné statistiky jednotlivých hostů, rozdělené do skupin podle <port, protokol>. Tyto statistiky procházíme po jednotlivých skupinách, kde pro každou skupinu vypočítáme parciální model pro danou skupinu. K tomu je využíván klastrovací algoritmus K-means. Pokud nám výpočet proběhne vpořádku, spočítáme vzdálenost pro jednotlivé klastry od počátku souřadnicového systému. Jestliže je vzdálenost větší než nastavený práh, označíme klastr jako anomální a budeme pokračovat dalším klastrem. Pokud bude vzdálenost menší než nastavený práh, přiřadíme nalezenému klastru IP adresy dostupné z Wardenu a zjistíme, zda procentuální zastoupení IP adres z Wardenu v daném klastru je větší než nastavený práh. Pokud ano, označíme daný klastr jako anomální. Jakmile projdeme všechny klastry, uložíme vypočítaný parciální model do celkového modelu a pokračujeme s další skupinou. Po ukončení celého výpočtu bude toto vlákno vyčkávat na vstup od obsluhy, která buďto vypočítaný model potvrdí nebo jej zahodí.



Obrázek 6.2: Logika detekování anomálií.

Během implementace bylo experimentováno s několika klastrovacími knihovnami. Jednalo se o knihovnu AlgLib [29]. Dále byla testována knihovna OpenCV [4]. Poslední knihovna, napsaná v jazyce C, byla vyvíjená v rámci projektu Cluster3 [16]. Tuto knihovnu bylo nutné nejdříve upravit tak, aby byla přeložitelná C++ kompilátorem. Po provedení několika testů s uvedenými knihovnami jsme dospěli k následujícímu závěru. Výsledky z knihoven AlgLib a OpenCV se neshodovaly s výsleky, které nám podával nástroj WEKA, a proto nebylo možné tyto knihovny využít. Knihovna Cluster3 nám podala výsledky, které byly téměř identické z výsledky nástroje WEKA. Pro implementaci byla proto zvolena knihovna Cluster3.

Následující třídou je `DetectAnomalies`. Tato třída s využitím aktuálně vypočítaného modelu vyhledává anomálie v nově příchozích statistikách. Stejně jako třída `ComputeDetectorModel` je využívána třídou `AnomalyDetector`, která pomocí její instance volá veřejnou metodu `create_thread`. Tímto je vytvořeno vlákno pro detekci anomálií v nově příchozích statistikách. Princip fungování se nachází na obrázku 6.2 a nyní si jej stručně popíšeme. Na vstupu dostáváme statistiky jednotlivých hostů rozdělené do skupin podle <port, protokol>. Postupně budeme procházet jednotlivé skupiny, pro které budeme z aktuálního detekčního modelu vyhledávat odpovídající parciální model pro danou skupinu. Pokud tento model nalezneme, začneme procházet jednotlivé statistiky z dané skupiny, a každou z nich přiřadíme nějakému klastru z nalezeného modelu. Přiřazení do klastru je založeno na výpočtu vzdálenosti od jednotlivých klastrů. Následně zkontrolujeme, zda je klastr, do kterého byl daný host přiřazen, anomální, či nikoliv. Podle výsledku určíme, zda daného hosta nahlásíme jako anomálii nebo ne.

Poslední třídou je `UI`. Tato třída zajišťuje komunikaci s uživatelskou aplikací, pomocí které může administrátor zasílat dotazy do běžícího modulu. Tato třída implementuje jednoduchý komunikační protokol. Posílání zpráv je založeno na "Unix domain sockets". Jedná se o prostředek pro komunikaci mezi procesy v rámci jednoho počítače. Na rozdíl od jednosměrných rour je možné pomocí "Unix domain sockets" používat obnousměrnou komunikaci, podobně jak ji známe ze síťových socketů.

Síť	CESNET				
Datum	06.03.2013				
Interval	Počet toků	Počet bajtů	Počet paketů	DZ 1	DZ 2
08:50	27202747	712.9 G	813.4 M	-	-
08:55	27522696	736.8 G	840.0 M	4m13s	3m46s
09:00	28569172	748.4 G	854.1 M	4m16s	3m47s
09:05	29086155	788.3 G	901.0 M	4m22s	3m52s
09:10	29307616	789.7 G	904.8 M	4m29s	3m56s
09:15	30284989	812.5 G	926.2 M	4m39s	4m6s
09:20	31557008	840.0 G	961.4 M	4m55s	4m12s
09:25	31771737	852.3 G	973.8 M	4m54s	4m21s
09:30	31828948	855.2 G	979.6 M	4m49s	4m19s
09:35	31306253	856.8 G	975.9 M	4m45s	4m13s
09:40	31238609	829.0 G	952.3 M	4m45s	4m15s
09:45	31265994	858.4 G	976.0 M	4m49s	4m17s
09:50	31951828	885.9 G	1.0 G	4m50s	4m22s

Tabulka 6.1: Velikosti datových sad použitých při testování.

6.3 Výsledky experimentů

V rámci této práce byla provedena řada experimetů, které měly za úkol ověřit funkčnost implementovaného zásuvného modulu portstats a modulu pro detekci anomálií. Data, nad kterými probíhaly experimenty pocházela ze sítě CESNET stejně jako analyzovaný vzorek z kapitoly 4. Všechny pětiminutové intervaly, které byly použity v testech se nacházejí v tabulce 6.1, kde jsou uvedeny celkové velikosti jednotlivých vzorků. NetFlow data byla získána ze samostatných nfcapd souborů z jednotlivých linek, které se uvnitř aplikace HostStats slučovaly. Z tabulky 6.1 je patrné, že se jedná o poměrně velkou síť, kde jednotlivé intervaly obsahují několik desítek milionů toků. Průměrný objem provozu v testovaném intervalu jedné hodiny byl 23.2 Gb/s. Ve zmíněné tabulce je rovněž uvedena doba zpracování jednotlivých pětiminutových intervalů aplikací HostStats se spuštěným zásuvným modulem PortStats a modulem pro detekci anomálií. Sloupce DZ 1 a DZ 2 zobrazují dobu zpracování se spuštěnou nebo zakázanou funkcí pro vyhledávání detailních informací o tocích z nfcapd souborů. Tato funkce pracuje ve dvou fázích. V první fázi proběhne dotaz nad všemi nfcapd soubory z jednotlivých linek a výsledek se uloží do nového souboru nfcapd. Tento dotaz obsahuje IP adresy všech detekovaných anomálií. Ve druhé fázi se již prochází pouze nově vytvořeným nfcapd souborem, nad kterým probíhají dotazy s jednotlivými detekovanými IP adresami. Konkrétní nastavení detekčního modulu je popsáno v následujícím odstavci.

Na základě výsledků analýzy datové sady, která je popsána v kapitole 4, bylo použito osm dimenzí v nastavení klastrovacího algoritmu. Jednalo se o dimenze obsahující odchozí a příchozí SYN, ACK, FIN a RST příznaky. Dále bylo použito rozdělení do skupin podle dvojic <port, protokol>, kde byly použity dvě obecné skupiny podle čísla protokolu a dále byly použity tři skupiny podle čísla portu <22, 443, 80>. Jednotlivá nastavení skupin sloužící jako vstup klastrovacímu algoritmu se nacházejí v tabulce 6.2. Pokud se v nastavení skupiny ve sloupci "Max. klastrů" vyskytuje hodnota 1, nechceme u této skupiny provádět detekci anomálií, ale chceme odfiltrovat provoz patříci do dané skupiny z celkového provozu. Jak již bylo zdůvodněno v kapitole 4, u portu 80 není možné využít klastrovací metodu ve spojení s dimenzemi obsahující TCP příznaky. Rovněž UDP provoz byl pouze odfiltrován,

Skupina	Max. klastrů	Počet restartů	Dimenze
<0, 6>	150	1	S,A,F,R
<0, 17>	1	1	S,A,F,R
<22, 6>	30	5	S,A,F,R
<443, 6>	150	1	S,A,F,R
<80, 6>	1	1	S,A,F,R

Tabulka 6.2: Nastavení modulu Anomaly Detector

protože neposílá žádné příznaky. Sloupec "Počet restartů" nám určuje kolikrát proběhne výpočet modelu dané skupiny. Klastrovací algoritmus následně vybere nejlepší model. Je třeba mít na vědomí, že čas výpočtu modelu dané skupiny se v tom případě násobně zvyšuje.

Všechny testy probíhaly na hostitelkém počítači s konfigurací AMD FX(tm)-4100 Quad-Core s 8GB operační paměti a OS Fedora 17 (s jádrem 3.8.4).

Doba zpracování testovaných intervalů primárně závisí na velikosti zpracovávaných souborů nfcapd z jednotlivých linek. Celkový součet velikostí těchto souborů z jednoho pětiminutového intervalu se pohyboval cca od 750MB do 900MB. Dalšími faktory jsou počet hostů v jednotlivých skupinách a počty klastrů, se kterými je právě zpracovávaný host porovnáván. Vliv na dobu zpracování má rovněž počet detekovaných anomálií, pokud je povolená funkce pro vyhledávání detailních informací o tocích pro detekované IP adresy. V tabulce 6.1 jsou ve sloupcích DZ 1 a DZ 2 doby zpracování s povolenou a zakázanou touto funkcí. Po analýze algoritmu zpracovávajícího jednotlivé pětiminutové intervaly bylo zjištěno, že nejnáročnější operací je načtení a sloučení všech toků z jednotlivých linek. Tato operace trvá cca od 2,5 do 3 minut v závislosti na velikosti příchozích NetFlow dat. Druhou nejnáročnější operací je spočítání statistik pro jednotlivé hosty a jejich rozdělení do skupin. Tato část měla délku trvání cca od 40 do 50 vteřin. Poslední částí algoritmu je detekce anomálií podle předem vypočítaného modelu. Tato část již probíhala poměrně rychle a trvala cca 30 vteřin, kdy jsme byli schopni označit 4135 hostů za vteřinu. Při nastaveném prohledávání skupin <0, 6>, <22, 6> a <443, 6>byl algoritmus schopen provádět požadovanou detekci anomálií v reálném čase. Všechny intervaly, nad kterými probíhala detekce, byly zpracovány do pěti minut, a tím bylo dosaženo rychlosti, která zvládá průběžně analyzovat příchozí NetFlow data ze všech 11 linek nad uvedenými skupinami. Proces výpočtu detekčního modelu trvá řádově několik desítek minut a probíhá nezávisle na detekci anomálií. Z tohoto důvodu je možné provádět průběžně výpočty nových detekčních modelů aniž by musela být detekce anomálií pozastavena.

V tabulce 6.3 se nacházejí počty detekovaných anomálií v testovaných intervalech. Ve sloupci Warden jsou uvedeny počty detekovaných anomálií, které byly hlášeny do Wardenu. V následujícím sloupci se nacházejí počty anomálií, které byly detekovány modulem vytvořeným v rámci této práce. V dalších sloupcích jsou počty anomálií, které byly manuálně označeny jako falešně pozitivní či pozitivní ze všech detekovaných IP adres modulem pro detekci anomálií. Vyšší počet falešně pozitivních detekcí byl způsoben několika příčinami, které budou popsány níže.

K počtu falešně pozitivních detekcí výrazně přispívá skupina, která provádí detekci anomálií na portu 443. Jedná se o port HTTPS (šifrovaný přenos HTTP protokolu pomocí TLS). Tato skupina tvoří přibližně 40% falešně pozitivních detekcí z nalezených 292. S využitím informací z Wardenu byl v této skupině vytvořen jeden anomální klastr. Po prozkoumání IP adres z Wardenu, které se přiřadily do nalezeného klastru, se zjistilo, že

Síť	CESNET			
Datum	06.03.2013			
Interval	Warden	Detekce Anomálií	Falešně pozitivní	Pozitivní
08:50	112	-	-	-
08:55	109	149	24	125
09:00	210	159	24	135
09:05	112	150	26	124
09:10	106	159	25	134
09:15	103	168	30	138
09:20	90	162	29	133
09:25	98	122	19	103
09:30	98	116	18	98
09:35	103	164	24	140
09:40	106	117	20	97
09:45	97	167	27	140
09:50	100	154	26	128
TOTAL	1332	1787	292	1495

Tabulka 6.3: Detekované anomálie.

všechny tyto IP adresy obsahovaly pouze cca 50-60 toků. Z hlediska jednotlivých IP adres nelze tvrdit, že se jedná o útok či skenování. Nicméně IP adres s tímto počtem toků se ve Wardenu nacházelo přibližně 50 a všechny pocházely z jednoho rozsahu adres. S největší pravděpodobností se v tomto případě jednalo o distribuované skenování portu, případně o podvržené IP adresy, a proto byly tyto útoky detekovány uvnitř sítě CESNET jako anomálie a nahlášeny do Wardenu. Díky těmto nahlášeným událostem došlo k vytvoření anomálního klastru, který obsahoval statistiky hostů s nepříliš výraznými příznaky. Detektor anomálií byl schopen během detekování zachytit tato distribuovaná skenování, ale rovněž docházelo k detekci samostatných IP adres s nízkým počtem toků a příznaků, které nebyly anomální. To mělo za následek nárůst falešně pozitivních detekcí. V této skupině docházelo ještě k dalším falešně pozitivním detekcím, jejichž příčiny budou nyní popsány. Na portu 443 byly v každém intervalu detekovány dvě IP adresy obsahující cca 30000 toků. Jednotlivé toky obsahovaly příznaky z kompletně navázané komunikace. Z toho důvodu byl tento klastr označen jako anomální a bylo možné ho považovat za útok hádání hesla. Nicméně po prozkoumání těchto IP adres se ukázalo, že se jedná o poměrně velké servery v České Republice, ke kterým se neustále přihlašuje spousta uživatelů. Tyto IP adresy byly tímto falešně nahlášeny jako anomální. Další příčinou falešně pozitivních detekcí byla s velkou pravděpodobností P2P komunikace. V každém testovaném intervalu se totiž nacházelo několik IP adres, které komunikovaly z velkého počtu zdrojových portů na velký počet cílových portů.

V dalších odstavcích budou popsány významné anomálie, které byly modulem pro detekci anomálií odhaleny. Bude se jednat o anomálie, které se vyskytovaly nejčastěji, nebo anomálie obsahující velké počty toků. Informace o jednotlivých portech byly získávány ze stránek SpeedGuide [7] a ShieldsUP [5]. Výsledky zkoumaných portů se nacházejí v tabulce 6.4.

Nejčastěji detekovanou anomálií byla detekce skenování portu 443. Celkový počet útoků směřujících na tento port činil 36% ze všech detekovaných anomálií (645 z celkového počtu 1787). Jednalo se o velký počet malých skenování. Nicméně v takto detekovaných anomáliích

Síť	CESNET						
Datum	06.03.2013						
Interval	Porty						
	445	443	1433	3306	3389	22	6000
08:50	-	-	-	-	-	-	-
08:55	31	56	3	3	3	4	0
09:00	36	58	3	1	3	4	0
09:05	37	54	1	1	3	5	0
09:10	38	50	2	0	4	3	0
09:15	42	56	2	3	4	4	1
09:20	39	50	3	2	7	3	2
09:25	8	51	3	2	3	3	3
09:30	9	53	3	4	2	4	2
09:35	39	56	3	3	3	4	3
09:40	8	54	1	0	3	3	1
09:45	45	56	1	1	4	4	1
09:50	39	51	0	1	4	5	1
TOTAL	371	645	25	21	43	46	14

Tabulka 6.4: Počty detekovaných anomálií pro nejčastější porty.

se rovněž nacházelo množství falešně pozitivních detekcí. Zdůvodnění je popsáno výše.

Další často detekovanou anomálií bylo skenování portů 445, 135 a 139. V tabulce 6.4 se nacházejí výsledky pouze k portu 445. Port 135 je přiřazen službě Remote Procedure Call (RPC). Měl by být blokován spolu s porty známými jako "NetBIOS Trio", do kterých patří porty 137, 138 a 139. Otevřením těchto portů se uživatelé vystavují riziku, protože služby běžící na těchto portech jsou nedostatečně zabezpečené. Útoky na port 445 (SMB over IP) tvořily cca 20% všech detekovaných anomálií. Tento port byl vytvořen jako náhrada za "NetBIOS Trio". Společnost Microsoft tímto portem způsobila nemalé problémy, protože tento port obsahoval ještě větší bezpečnostní rizika než zmiňovaná "NetBIOS Trio". Mnoho internetových poskytovatelů začalo blokovat tento port svým uživatelům, aby nedocházelo k jeho zneužívání.

Dalšími výraznějsími anomáliemi byly útoky na porty 1433 a 3306. Tyto porty jsou využívány databazovými službami. 1433 je používán službou Microsoft SQL Server a slouží pro vzdálené přípojení se k databázi. Stejně se využívá port 3306, který patří službě MySQL Database Server. Na tyto porty bylo několikrát útočeno z portu 6000. Jednalo se o výrazné anomálie, kde skenování portu 3306 obsahovalo 330000 toků a skenování portu 1433 obsahovalo 64000 toků. Po ověření portu 6000 se zjistilo, že se jedná o port, který bývá často využíván různými červy. Může se jednat například o červ W32.LoveGate.ak, který masově rozesílá emaily. Ukázky těchto útoků se nacházejí v příloze G.

Dalšími významnými anomáliemi byly útoky na porty 3389 (Remote Desktop Protocol) a 22 (Secure Shell). Hlavním typem útoků na tyto porty bylo hádání hesla. To se vyznačovalo vysokými počty zasílaných SYN, ACK a FIN příznaků. Jednalo se tedy o velké množství toků obsahujících kompletní navázané spojení z čehož bylo možné vyvodit, že se jedná o hádání hesla k těmto službám. Tyto útoky na oba porty, jež byl náš modul schopen detekovat, probíhaly během celého zkoumaného intervalu (08:55 - 09:50). V příloze G.3 a G.4 se nacházejí ukázky těchto útoků. V ukázkach jsou pro jednotlivé IP adresy útočící

na tyto dva porty zobrazeny součty všech toků z celého testovaného intervalu. Jednalo se o velmi rozsáhlé útoky. Z jedné IP adresy bylo během hodiny vygenerováno téměř 1mln toků směřujících na port 3389 a na různé cílové IP adresy. Na port 22 probíhaly během zkoumaného hodinového intervalu tři útoky, kde bylo z jedné adresy vygenerováno cca 125000 toků a z dalších dvou po cca 90000 toků.

Další anomálie, které nahlásil modul pro detekci anomálií, směřovaly například na porty 21 (FTP), 23 (Telnet) a jiné.

Zhodnocení výsledků

V rámci experimentů bylo použito 12 pětiminutových intervalů. Celkový interval byl v rozmezí od 08:55 do 09:50. Pětiminutový interval s časovou značkou 08:50 byl použit pro výpočet detekčního modelu, podle kterého byla v následujících intervalech prováděna detekce anomálií. Pro vytvoření modelu byly použity informace z Wardenu. Ve skupině <22, 6> bylo pomocí těchto informací označeno několik klastrů jako anomálních. Nicméně IP adresy z Wardenu použité pro označení dvou z těchto anomálních klastrů se ukázaly jako falešně pozitivní. Proto byly tyto dva klastry manuálně nastaveny jako normální. Z výsledků je možné odvodit, že modul pro detekci anomálií je schopen provádět detekci anomálií nad zvolenými skupinami v reálném čase. Rovněž dokázal detekovat více anomálií než bylo hlášení z Wardenu. Nicméně bez informací z Wardenu by se neobešel při výpočtu detekčního modelu. Úspěšnost detekce se pohybovala nad hranicí 80%. Tedy celkový počet detekovaných anomálií byl 1787. Z toho 292 bylo označeno jako falešně pozitivních. Detekční modul byl rovněž schopen detekovat útoky hádání hesla ke službám RDP a SSH, které se nenacházely v hlášeních z Wardenu. Vzhledem k tomu, že se nejedná pouze o detekci útoků s využitím konrétních pravidel (vzorů), jsou výsledky poměrně zajímavé. Nicméně cca 16% falešně pozitivních hlášení je stále vysoké procento. Z toho důvodu by bylo potřeba provádět ještě dodatečnou kontrolu již detekovaných anomálií. Další z možností jak zlepšit výsledky, by mohlo být zavedení výjimek, kde by se nastavovaly IP adresy, které mají být z detekce vynechány. V detekovaných anomáliích se totiž nacházely IP adresy větších serverů, které byly falešně označeny jako anomálie.

Kapitola 7

Závěr

Cílem této práce bylo seznámení se s problematikou detekce síťových anomálií, analyzování vybrané detekční metody, následně navržení a implementování algoritmu pro detekci síťových anomálií s využitím zvolené metody. Celá práce byla rozdělená do několika částí.

První část této práce se zaměřila na sběr informací o síťovém provozu, kde byly popsány různé způsoby odchytávání paketů ze sítě. Největší pozornost byla věnována protokolu NetFlow, který byl základem pro tuto práci. Rovněž byly představeny vybrané nástroje pro práci s daty získanými pomocí tohoto protokolu.

Vybrané metody, které se využívají pro detekci anomálií v síťových datech, byly analyzovány ve druhé části práce. Byly popsány statistické metody, metody strojového učení, metody dolování dat a hybrydní systémy. Z výsledků analýzy byla vybrána metoda klastrování. Tato metoda byla zvolena pro její schopnost zpracovávat i neoznačená trénovací data, která pro účely této práce byla k dispozici ze sítě CESNET.

Třetí část již zahrnovala analýzu datové sady pocházející ze zmíněné sítě a následný návrh detekčního algoritmu. Datová sada byla vytvořena pomocí aplikace HostStats, která je vyvíjená v rámci projektu traffic-analysis na Ústavu počítačových systému UPSY. Analýza datové sady byla prováděná s využitím nástroje WEKA. Z výsledků počáteční analýzy bylo možné vyvodit, že klastrovací metody nejsou schopny korektně rozpoznat, jakou váhu mají přiřadit jednotlivým dimenzím během výpočtu. Z tohoto důvodu bylo nutné klastrovacímu algoritmu předat určitou znalost o problému, který se má v datech vyhledávat. Bylo tedy nutné zjistit jaké anomálie se nejčastěji nacházejí v síti CESNET a jak se projevují v NetFlow datech. K tomu nám posloužily informace od spolupracujících firem uvnitř sítě, které jimi detekované útoky hlásí do Wardenu. Po předání těchto znalostí klastrovacímu algoritmu byly výsledky znatelně lepší, avšak stále nedostačující. Dalším krokem pro zlepšení výsledků klastrovací metody bylo rozdělení provozu do skupin podle čísla portu a protokolu. V rámci experimentů byl proto navržen a implementován zásuvný modul do aplikace HostStats, který toto dělení prováděl. Aplikací těchto úprav a provedení řady experimetů bylo dosaženo pozitivních výsledků. Po provedení analýzy byl navržen algoritmus, který s využitím klastrovacího algoritmu K-means vytváří detekční model, s jehož pomocí je prováděna detekce anomálií v reálném čase.

Poslední část práce byla věnována implementaci algoritmu a experimentům s výslednou aplikací. Detekční algorimtus byl vytvořen jako modul do aplikace HostStats. Vytvořený modul prováděl detekci ve dvou fázích. V první fázi byl vypočítán detekční model, který byl ve druhé fázi využit pro detekci anomálií v nově příchozích datech. V rámci experimetů byl použit hodinový interval NetFlow dat pocházejících rovněž ze sítě CESNET. Nad tímto intervalem byl spuštěn detekční algoritmus, který byl schopen detekovat více anomálií než

bylo hlášeno do Wardenu. Nicméně v detekovaných anomáliích se nacházelo poměrně vysoké procento falešně pozitivních detekcí. Z výsledků bylo možné vyvodit, že detekce anomálií, využívající pro výpočet detečního modelu klastrovací metodu, má svá úskalí. Vzhledem k tomu, že se jedná o detekci anomálií založenou na výpočtu detekčního modelu, jsou výsledky poměrně zajímavé. Aplikace je dobrým nástrojem, který dokáže odhalit velký počet anomálií ve vysokorychlostních sítích, kde se objem provozu pohybuje až kolem 23 Gb/s. Může být proto využita jako vstup do dalšího filtru, který bude schopen z detekovaných anomálií odfiltrovat falešně pozitivní detekce. Při počtu detekovaných anomálií pohybujícím se v rozmezí 100-300 detekcí za 5 minut, by tento filtr již mohl provádět detailnější analýzu. Práci by bylo možné rozšířit ještě několika dalšími způsoby. Jedním z nich by mohlo být vytvoření systému vyjímek, který by nastavené IP adresy během detekce vynechával. Dále by bylo možné provést analýzu datové sady se zaměřením na port 80 a zjistit, zda je možné nalézt takové nastavení klastrovacího algoritmu, aby byl schopen detekovat anomálie na tomto portu.

Literatura

[1] fprobe. online, [cit. 2012-12-3].
URL http://fprobe.sourceforge.net

[2] nfanon. online, [cit. 2013-4-30].
URL http://manned.org/nfanon/8cc4ac19

[3] NfSen. online, [cit. 2012-12-3].
URL http://nfsen.sourceforge.net

[4] OpenCV. online, [cit. 2013-4-30].
URL http://opencv.org

[5] ShieldsUp. online, [cit. 2013-4-30].
URL http://www.grc.com/

[6] Softflowd. online, [cit. 2012-12-3].
URL http://www.mindrot.org/projects/softflowd/

[7] SpeedGuide. online, [cit. 2013-4-30].
URL http://www.speedguide.net/

[8] WEKA The University of Waikato. online, [cit. 2013-4-30].
URL http://www.cs.waikato.ac.nz/ml/weka/

[9] Bringas, P. G.; Grueiro, I. S.: Bayessian Networks for Network Intrusion Detection. In *Bayessian Network*, editace A. Rebai, InTech, 2010, s. 229–244.

[10] Chandola, V.; Banerjee, A.; Kumar, V.: Anomaly detection: A survey. *ACM Comput. Surv.*, ročník 41, č. 3, Červenec 2009: s. 15:1–15:58, ISSN 0360-0300.

[11] Chebrolu, S.; Abraham, A.; Thomas, J. P.: Feature deduction and ensemble design of intrusion detection systems. *Computers and Security*, ročník 24, č. 4, 2005: s. 295 – 307, ISSN 0167-4048.

[12] CISCO.Systems: Introduction to Cisco IOS NetFlow - A Technical Overview. online, [cit. 2012-11-14].
URL
http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.pdf.

[13] CISCO.Systems: NetFlow Services Solutions Guide. online, [cit. 2012-11-14].
URL
http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.pdf

[14] Crosbie, M.; Spafford, E. H.; Spafford, P. G.: Applying Genetic Programming to Intrusion Detection. In *IN PROCEEDINGS OF THE AAAI 1995 FALL SYMPOSIUM SERIES*, 1995, s. 1–8.

[15] Dickerson, J.; Dickerson, J.: Fuzzy network profiling for intrusion detection. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, 2000, s. 301 –306.

[16] Eisen, M.: Cluster3. online, [cit. 2013-4-30].
URL http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm

[17] García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; aj.: Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers and Security*, ročník 28, č. 1-2, 2009: s. 18 – 28, ISSN 0167-4048.

[18] Ghahramani, Z.: Hidden Markov models. kapitola An introduction to hidden Markov models and Bayesian networks, River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2002, ISBN 981-02-4564-5, s. 9–42.

[19] Gough, J.: Encapsulation of data. online, [cit. 2012-11-27].
URL
http://www.securitywizard.com/packets/pdf/Encapsulation%20of%20data.pdf

[20] Grégr, M.: *Detekce a izolace útočníků pomocí záznamů NetFlow*. Diplomová práce, FIT VUT v Brně, 2009, diplomová práce.

[21] Hanuš, J.: *Shluková analýza a její aplikace*. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, 2009, bakalářská práce práce.

[22] IETF: IP Flow Information Export (ipfix). online, [cit. 2013-1-3].
URL http://www.datatracker.ietf.org/wg/ipfix/charter

[23] Lee, C. B.; Roedel, C.; Silenok, E.: Detection and Characterization of Port Scan Attacks. online, [cit. 2013-4-30].
URL http://cseweb.ucsd.edu/~clbailey/PortScans.pdf

[24] Lee, W.; Stolfo, S. J.; Mok, K. W.: A Data Mining Framework for Building Intrusion Detection Models. In *In IEEE Symposium on Security and Privacy*, 1999, s. 120–132.

[25] Münz, G.; Li, S.; Carle, G.: Traffic Anomaly Detection Using KMeans Clustering. In *In GI/ITG Workshop MMBnet*, 2007.

[26] NFDUMP: Nfdump tools. online, [cit. 2012-11-14].
URL http://nfdump.sourceforge.net/

[27] Patcha, A.; Park, J.-M.: An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, ročník 51, č. 12, 2007: s. 3448 – 3470, ISSN 1389-1286.

[28] Plch, M.: *Detekce DoS útoků pomocí analýzy síťových toků*. Diplomová práce, ČVUT v Praze, Fakulta informačních technologií, 2012, diplomová práce.

[29] Project, A.: AlgLib. online, [cit. 2013-4-30].
URL http://www.alglib.net

[30] Ringberg, H.; Soule, A.; Rexford, J.; aj.: Sensitivity of PCA for traffic anomaly detection. *SIGMETRICS Perform. Eval. Rev.*, ročník 35, č. 1, Červen 2007: s. 109–120, ISSN 0163-5999.

[31] SAMURAJ: SNMP - Simple Network Management Protocol. online, [cit. 2012-11-22].
URL
http://www.samuraj-cz.com/clanek/snmp-simple-network-management-protocol

[32] Shyu, M.-L.; Chen, S.-C.; Sarinnapakorn, K.; aj.: A novel anomaly detection scheme based on principal component classifier. In *in Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop, in conjunction with the Third IEEE International Conference on Data Mining (ICDM'03*, 2003, s. 171–179.

[33] SNMP: Net-SNMP. online, [cit. 2012-11-22].
URL http://www.net-snmp.org

[34] TCPDUMP: TCPDUMP and Libpcap. online, [cit. 2012-11-27].
URL http://www.tcpdump.org

[35] Telecom, D.: SNMP Tutorial: An Introduction to SNMP. online, [cit. 2012-11-22].
URL http://www.dpstele.com/layers/l2/snmp_tutorials.php

[36] Thakong, M.; Wongthanavasu, S.: Packet Header Anomaly Detection Using Bayessian Belief Network. *ECTI Transaction CIT*, ročník 3, č. 1, 2007: s. 26–30.

[37] Tombini, E.; Debar, H.; Me, L.; aj.: A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic. In *Proceedings of the 20th Annual Computer Security Applications Conference*, ACSAC '04, Washington, DC, USA: IEEE Computer Society, 2004, ISBN 0-7695-2252-1, s. 428–437.

[38] Tsai, F. S.: Network Intrusion Detection Using Association Rules. *International Journal of Recent Trends in Engineering*, ročník 2, č. 2, 2009: s. 43–68.

[39] Tylman, W.: Anomaly-Based Intrusion Detection Using Bayesian Networks. *Dependability of Computer Systems, International Conference on*, ročník 0, 2008: s. 211–218.

[40] Ye, N.; Zhang, Y.; Borror, C. M.: Robustness of the Markov-chain model for cyber-attack detection. *IEEE Transactions on Reliability*, ročník 53, č. 1, 2004: s. 116–123.

Příloha A

Obsah CD

Obsahem CD jsou zdrojové kódy vytvořené aplikace, včetně přiloženého Makefilu. Rovněž jsou přiloženy testovací data v podobě vypočítaných host statistik, které jsou anonymizovány. Dále je na CD soubor README, programová dokumentace k projektu vytvořená pomocí doxygen aplikace a text této diplomové práce, včetně zdrojových souborů z latexu.

Příloha B

Požadavky na systém

Vytvořená aplikace byla testována v prostředí OS Fedora 17 (s jádrem 3.8.4). Všechny zdrojové kódy se nacházejí na CD včetně Makefilu, který využijete pro překlad aplikace. Pro kompilaci je nutné mít nainstalovaný překladač g++. Pro běh některých funkcí je vyžadován nástroj nfdump. Nicméně tato funkce není podmínkou fungování celé aplikace. Jedná se pouze o rozšíření, které lze pomocí konfiguračního souboru vypnout.

Příloha C

Manuál

Vytvořená aplikace je bez grafického rozhraní a ovládá se pomocí konfiguračních souborů a klientské aplikace. Klientská aplikace je rovněž bez grafického rozhraní a příkazy přijímá ze standardního vstupu.

C.1 Aplikace HostStats

Aplikace načítá nastavení ze dvou konfiguračních souborů. Jmenovitě se jedná o soubor hoststats.conf a konfigurační soubor s nastavením pro klastrovací algoritmus, který je definován v hoststats.conf v parametru algorithm-sett-path.

V souboru hoststats.conf lze nastavit parametry, které se nacházejí v tabulce C.3. V souboru s nastavením klastrovacího algortimu je možné nastavit pro jednotlivé skupiny parametry nacházející se v tabulce C.1. Ukázka konfiguračního souboru pro klastrovací algoritmus se nachází ve výpisu C.1.

Parametr	Význam
clusters	Maximální počet klastrů.
restarts	Počet restartů výpočtů (vybírá se nejlepší výsledek).
dimensions	Dimenze použité u výpočtu. Číselný kód dimenzí, viz tabulka C.2.
distanceth	Práh vzdálenosti klastru od počátku souřadnicového systému.
wipth	Procentuální práh počtu IP adres z Wardenu v klastru.

Tabulka C.1: Významy parametrů pro nastavení klastrovacího algoritmu.

Dimenze	Kód	Dimenze	Kód	Dimenze	Kód	Dimenze	Kód
out_flows	0	out_fin_cnt	5	in_flows	10	in_fin_cnt	15
out_packets	1	out_rst_cnt	6	in_packets	11	in_rst_cnt	16
out_bytes	2	out_psh_cnt	7	in_bytes	12	in_psh_cnt	17
out_syn_cnt	3	out_urg_cnt	8	in_syn_cnt	13	in_urg_cnt	18
out_ack_cnt	4	out_uniqueips	9	in_ack_cnt	14	in_uniqueips	19

Tabulka C.2: Mapování dimenze na její číselný kód.

Parametr	Význam
Obecné parametry	
listen-port	Port, na kterém aplikace naslouchá a přijímá nová data.
listen-interface	Interface, na kterém aplikace naslouchá.
link-priority-file	Soubor, ve kterém jsou nastaveny priority jednotlivých linek.
log-upto-level	Úroveň systémových logů.
stats-files-path	Cesta k adresáři, do kterého se ukládají vypočítané statistiky hostů.
store-stats-files	Příznak, zda se mají vypočítané statistiky ukládat či nikoliv (true|false).
detected-log-path	Soubor, do kterého se budou ukládat detekované anomálie.
stored-model-path	Soubor s uloženým detekčním modelem.
warden-detections-path	Hlavní adresář útoků hlášených do Wardenu.
dump-files-path	Adresář, do kterého se budou ukládat soubory z aplikace nfdump detekovaných anomálií.
portstats-pairs	Dvojice <port, protocol> pro inicializaci portstats pluginnu (např. 0,6;22,6).
use-search-nfcapd	Příznak, zda se má vyhledávat detailní informace o detekovaných anomáliích v souboru nfcapd. (true|false)
algorithm-sett-path	Soubor s nastavením pro klastrovací algoritmus.
ui-output-dir-path	Adresář, do kterého se ukládají některé výstupy vyžádány přes UI rozhraní.
cl3-files-path	Adresář, do kterého se uloží výstup z klastrovacího algoritmu s detailními informacemi.
Parametry vyžadované během lokálního testování	
local-testing	Příznak, pro spuětění lokálního testování.
use-raw-data	Příznak, zda budeme používat raw data z nfcapd souboru, nebo použijeme již uložené vypočítané statistiky. (true|false)
compute-new-model-from	Konkrétní soubor se statistikami, který bude použit pro výpočet detekčního modelu (např. nfcapd.201303060850). Pokud tento parametr nebude nastaven, použije se první soubor v pořadí ze seznamu definovaném v testing-data-file-path.
testing-data-file-path	Seznam souborů, které budou během testování použity.
raw-files-path	U lokálního testování je nutné zadat cestu k adresáři s raw daty, pokud máme nastaven parametr use-raw-data na true.

Tabulka C.3: Parametry konfiguračního souboru hoststats.conf.

```
<?xml version="1.0" encoding="UTF-8" ?>
<groups>
  <group port="0" proto="0">
    <clusters>1</clusters>
    <restarts>5</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
```

```
    <distanceth>150000</distanceth>
    <wipth>8</wipth>
  </group>
  <group port="0" proto="6">
    <clusters>150</clusters>
    <restarts>1</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
    <distanceth>150000</distanceth>
    <wipth>10</wipth>
  </group>
  <group port="0" proto="17">
    <clusters>1</clusters>
    <restarts>1</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
    <distanceth>150000</distanceth>
    <wipth>8</wipth>
  </group>
  <group port="22" proto="6">
    <clusters>30</clusters>
    <restarts>5</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
    <distanceth>150000</distanceth>
    <wipth>80</wipth>
  </group>
  <group port="80" proto="6">
    <clusters>1</clusters>
    <restarts>1</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
    <distanceth>150000</distanceth>
    <wipth>8</wipth>
  </group>
  <group port="443" proto="6">
    <clusters>150</clusters>
    <restarts>1</restarts>
    <dimensions>3,4,5,6,13,14,15,16</dimensions>
    <distanceth>150000</distanceth>
    <wipth>18</wipth>
  </group>
</groups>
```

Výpis C.1: Ukázka nastavení parametrů pro K-means algoritmus.

C.2 Klientská aplikace

Klientská aplikace je ovládána z příkazového řádku a reaguje na příkazy zadávané pomocí standardního vstupu. Pomocí této aplikace je možné zasílat dotazy do detekčního modulu, který je součástí aplikace HostStats. Díky tomuto klientovi je možné získávat informace a nastavovat parametry běžícího modulu pomocí sady přepínačů. Klient reaguje na následující příkazy.

- GET - Získávání informací.
- SET - Nastavování parametrů.
- CONFIRM - Potvrzení nově vypočítaného modelu. Po potvrzení se model automaticky uloží a přepíše stávající soubor s uloženým modelem.

- DISCARD - Zahození nově vypočítaného modelu.
- NEWMODEL - Příkaz na započetí výpočtu nového modelu (nový model se začíná počítat nad statistikami z následujícího intervalu).

U příkazů GET a SET jsou k dispozici následující přepínače, viz tabulka C.4, které lze mezi sebou různě kombinovat. U příkazu **set** lze použít u přepínače **o** pouze volbu group. U přepínače **o** je vyžadován rovněž přepínač **l**. Rovněž je třeba mít na vědomí, že při použití přepínače **l** s volbou **advanced** bude vytvořen soubor pro každý klastr ze zvolené skupiny, případně ze všech skupin. Může tedy dojít k vytvoření velkého počtu souborů. Je tedy doporučeno u volby **advanced** používat omezující dotazy s využitím přepínače **t** případně definování konkrétních klastrů pomocí přepínače **c**.

Přepínač	Volby
-o	stats - získávání statistik o skupinách group - získávání detailních informací o konkrétní skupině
-m	new - informace z nového detekčního modelu cur - informace z aktuálně používaného detekčního modelu
-l	basic - základní informace o skupinách zaslané přímo klientovi advanced - detailní informace o skupinách, jejich klastrech a přiřazených hostech, které se uloží do adresáře definovaného v parametru ui-output-dir-path konfiguračního souboru hoststats.conf. Tuto volbu je možné použít pouze u nově vypočítaného modelu!
-g	nastavení konkrétní skupiny ve formátu port:protokol
-t	normal - informace pouze o normálních klastrech anomaly - informace pouze o anomálních klastrech
-c	nastavení konkrétních klastrů ve formátu např. 1,2,3

Tabulka C.4: Seznam přepínačů.

Nyní bude vypsáno několik příkladů použití těchto příkazů a jejich přepínačů.

Získání základních statistik všech skupin a jejich klastrů z nového modelu
get -ostats -mnew -lbasic

Získání základních statistik skupiny 22:6 a jejich klastrů z nového modelu
get -ostats -mnew -lbasic -g22:6

Získání základních statistik skupiny 22:6 a jejich anomálních klastrů z aktuálního modelu
get -ostats -mcur -lbasic -g22:6 -tanomaly

Získání detailních statistik skupiny 0:6 a jejich normálních klastrů z nového modelu. Tyto informace budou uloženy do souborů.
get -ostats -mnew -ladvanced -g0:6 -tnormal

Získání základních statistik skupiny 443:6 a klastrů 1,2,3,4 z nového modelu
get -ostats -mnew -lbasic -g443:6 -c1,2,3,4

Získání informací o anomálních klastrech ze skupiny 22:6 z nového modelu

get -ogroup -mnew -g22:6 -tanomaly

Nastavení klastrů 1,2,3,4 jako anomálních u skupiny 22:6 z nového modelu
set -ogroup -mnew -g22:6 -tanomaly -c1,2,3,4

Příloha D

Seznam použitých zkratek

- IDS - Intrusion Detection System.
- IPS - Intrusion Prevention System.
- WEKA - Waikato Environment for Knowledge Analysis.
- DoS - Denial of Service.
- DDoS - Distrubuted Denial of Service.
- CPT - Conditional probability table
- SSH - Secure Shell.
- RDP - Remote Desktop Protocol.
- AC - Anomální klastry.
- WIP - IP adresy hlášené do Wardenu.
- DIP - Procentuální zastoupení IP adres z Wardenu, které se nacházejí v anomálních klastrech.

Příloha E

HostStats

E.1 Seznam statistik jednotlivých IP adres

- out_flows - počet výstupních toků,
- out_packets - množství výstupních paketů,
- out_bytes - množství výstupních bajtů,
- out_syn_cnt - počet výstupních SYN příznaků,
- out_ack_cnt - počet výstupních ACK příznaků,
- out_fin_cnt - počet výstupních FIN příznaků,
- out_rst_cnt - počet výstupních RST příznaků,
- out_psh_cnt - počet výstupních PSH příznaků,
- out_urg_cnt - počet výstupních URG příznaků,
- out_uniqueips - počet výstupních unikátních IP adres.
- in_flows - počet vstupních toků,
- in_packets - množství vstupních paketů,
- in_bytes - množství vstupních bajtů,
- in_syn_cnt - počet vstupních SYN příznaků,
- in_ack_cnt - počet vstupních ACK příznaků,
- in_fin_cnt - počet vstupních FIN příznaků,
- in_rst_cnt - počet vstupních RST příznaků,
- in_psh_cnt - počet vstupních PSH příznaků,
- in_urg_cnt - počet vstupních URG příznaků,
- in_uniqueips - počet vstupních unikátních IP adres,

E.2 Pravidla použitého filtru

```
if(
  host.out_flows < 10 &&
  host.out_packets < 30 &&
  host.out_bytes < 10000 &&
  host.out_syn_cnt < 10 &&
  host.out_ack_cnt < 10 &&
  host. out_fin_cnt < 10 &&
  host. out_rst_cnt < 10 &&
  host.out_psh_cnt < 10 &&
  host.out_urg_cnt < 10 &&
  host. in_flows < 10 &&
  host.in_packets < 30 &&
  host.in_bytes < 10000 &&
  host.in_syn_cnt < 10 &&
  host.in_ack_cnt < 10 &&
  host. in_fin_cnt < 10 &&
  host. in_rst_cnt < 10 &&
  host.in_psh_cnt < 10 &&
  host.in_urg_cnt < 10
)
```

Výpis E.1: Pravidla použitého filtru.

E.3 Datové typy uchovávající statistiky hostů

```
struct hosts_key_t{
    uint64_t ad[2];

    hosts_key_t() {
        ad[0] = 0;
        ad[1] = 0;
    }

    bool operator<(const hosts_key_t &key2) const {
        return ((ad[1] < key2.ad[1]) ||
                ((ad[1] == key2.ad[1]) && (ad[0] < key2.ad[0])));
    }
};

struct hosts_record_t {
    uint32_t in_flows;
    uint64_t in_packets;
    uint64_t in_bytes;
    uint32_t out_flows;
    uint64_t out_packets;
    uint64_t out_bytes;
    uint32_t in_syn_cnt;
    uint32_t in_ack_cnt;
    uint32_t in_fin_cnt;
    uint32_t in_rst_cnt;
    uint32_t in_psh_cnt;
    uint32_t in_urg_cnt;
    uint32_t out_syn_cnt;
    uint32_t out_ack_cnt;
    uint32_t out_fin_cnt;
```

```
    uint32_t  out_rst_cnt;
    uint32_t  out_psh_cnt;
    uint32_t  out_urg_cnt;
    uint32_t  in_uniqueips;
    uint32_t  out_uniqueips;
    uint64_t   in_linkbitfield ;
    uint64_t   out_linkbitfield ;

    hosts_record_t() { // Constructor sets all values to zeros.
        memset(this, 0, sizeof(hosts_record_t));
    }
};

struct port_proto_key_t{

  uint16_t  port;
  uint8_t  proto;

  port_proto_key_t()
  {
    port = 0;
    proto = 0;
  }

    bool operator<(const port_proto_key_t &key2) const {
        return ((port < key2.port) ||
                ((port == key2.port) && (proto < key2.proto)));
    }
};

typedef map<hosts_key_t, hosts_record_t> stat_map_t;
typedef stat_map_t::iterator  stat_map_iter;

typedef map<port_proto_key_t,stat_map_t> port_proto_stat_map_t;
typedef map<port_proto_key_t,stat_map_t>::iterator port_proto_stat_map_iter;
```

Výpis E.2: Datové typy uchovávající statistiky hostů

Příloha F

Výsledky z testů při analýze dat

Všechny uvedené výpisy z NetFlow dat jsou anonymizovány pomocí nástroje nfanon. Významy jednotlivých sloupců se nacházejí v tabulkce F.1

Sloupec	Význam
Proto	Protokol
Src IP Addr	Zdrojová IP adresa
Src Pt	Zdrojový port
Dst IP Addr	Cílová IP adresa
Dst Pt	Cílový port
Flags	TCP příznaky
Packets	Počet přenesených paketů
Bytes	Počet přenesených bytů
Flows	Počet toků
Bpp	Počet bytů v paketu
pps	Počet přenesených paketů za vteřinu

Tabulka F.1: Významy jednotlivých sloupců u výstupu nástroje nfdump.

F.1 Hádání hesla ke službě SSH

První výpis zobrazuje celkový počet toků, který směřoval na různé cílové IP adresy a konkrétní cílový port 22. Jedná se o službu SSH (Secure Shell). Je možné vidět, že během komunikace se zasílají téměř všechny TCP příznaky. Jedná se tedy o celkové navázání a ukončení komunikace. Nicméně z důvodu vysokého počtu toků z dané konkrétní zdrojové IP adresy na řadu různých cílových IP adres můžeme usoudit, že se jedná o hádání hesla ke službě SSH běžící na portu 22.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP    17.210.41.234 45184 ->  247.81.229.89    22 .AP.SF      13   1305     1   100       2
TCP    17.210.41.234 33219 ->   247.81.97.35    22 .AP.SF      15   1409     1    93       3
TCP    17.210.41.234 35623 ->  247.81.77.149    22 .AP.SF      14   1373     1    98       3
TCP    17.210.41.234 48648 ->   247.81.88.26    22 .AP.SF      13   1321     1   101       2
TCP    17.210.41.234 58575 ->   247.81.77.76    22 .AP.SF      13   1321     1   101       1
TCP    17.210.41.234 36804 ->  247.81.238.56    22 .AP.SF      13   1161     1    89       3
TCP    17.210.41.234 41880 ->  247.81.238.18    22 .AP.SF      14   1425     1   101       3
TCP    17.210.41.234 40996 -> 247.81.123.188    22 .AP.SF      13   1353     1   104       2
TCP    17.210.41.234 59761 ->  247.81.80.253    22 .AP.SF      14   1201     1    85       2
TCP    17.210.41.234 35295 ->  247.81.243.79    22 .AP.SF      14   1425     1   101       1
```

```
TCP     17.210.41.234 58900 ->  247.81.243.79   22 .AP.SF     14     1409    1    100       1
TCP     17.210.41.234 36015 ->  247.81.240.42   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 57848 ->  247.81.229.86   22 .AP.SF     14     1425    1    101       3
TCP     17.210.41.234 60650 ->  247.81.94.235   22 .AP.SF     13     1305    1    100       2
TCP     17.210.41.234 33008 ->  247.81.97.108   22 .AP.SF     13     1321    1    101       2
TCP     17.210.41.234 37934 ->  247.81.240.42   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 49055 -> 247.81.243.114   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 50117 ->  247.81.229.35   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 41963 ->  247.81.70.129   22 .AP.SF     13     1305    1    100       2
TCP     17.210.41.234 42327 ->  247.81.238.56   22 .AP.SF     13     1161    1     89       3
IP addresses anonymised
Summary: total flows: 9957, total bytes: 13.3 M, total packets: 133163, avg bps: 297549, avg pps:
     373, avg bpp: 99
Time window: <unknown>
Total flows processed: 9957, Blocks skipped: 0, Bytes read: 478044
Sys: 0.010s flows/second: 995700.0 Wall: 0.009s flows/second: 1005249.9
```

Výpis F.1: Hádání hesla ke službě SSH.

Tento následující výpis již ukazuje na to, že se opravdu jedná o hádání hesla. V době petiminutového intervalu se vyskytlo 93 toků směřující z konkrétní zdrojové IP adresy na konkrétní cílovou IP adresu a na stále stejný cílový port 22.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP     17.210.41.234 40216 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 58571 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 40691 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 38846 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 35487 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 59503 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 39871 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 35123 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 38020 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 39983 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
IP addresses anonymised
Summary: total flows: 93, total bytes: 121509, total packets: 1209, avg bps: 3184, avg pps: 3, avg
     bpp: 100
Time window: <unknown>
Total flows processed: 93, Blocks skipped: 0, Bytes read: 4572
Sys: 0.007s flows/second: 13285.7 Wall: 0.004s flows/second: 20772.8
```

Výpis F.2: Hádání hesla ke službě SSH. Výpis obsahuje konkrétní cílovou adresu v průběhu pětiminutového intervalu.

Níže uvedený výpis pouze potvrzuje domněnku, že se jednalo o dlouhodobý útok. Výpis byl vytvořen z 40-ti minutového intervalu a jak je vidět, obsahuje 830 toků na konkrétní cílovou stanici a na stále stejný port 22.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP     17.210.41.234 40461 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 52210 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 37350 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 40216 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 57022 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 34219 ->  247.81.229.89   22 .AP.SF     13     1305    1    100       3
TCP     17.210.41.234 38509 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 59119 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 53944 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
TCP     17.210.41.234 37909 ->  247.81.229.89   22 .AP.SF     13     1321    1    101       3
IP addresses anonymised
Summary: total flows: 830, total bytes: 1.1 M, total packets: 10790, avg bps: 3235, avg pps: 4, avg
     bpp: 100
Time window: <unknown>
Total flows processed: 830, Blocks skipped: 0, Bytes read: 39948
Sys: 0.007s flows/second: 118571.4 Wall: 0.006s flows/second: 137645.1
```

Výpis F.3: Hádání hesla ke službě SSH. Výpis obsahuje konkrétní cílovou adresu v průběhu 40-ti minutového intervalu.

F.2 Hádání hesla ke službě RDP

Níže uvedené výpisy mají stejnou strukturu jako výše uvedená analýza hádání hesla ke službě SSH.

```
Proto      Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP      150.41.46.50 18746 ->  82.84.114.220 3389 .APRSF      24     2557     1    106        8
TCP      150.41.46.50 21457 ->  82.84.126.121 3389 .APRSF      44     3359     1     76       24
TCP      150.41.46.50 32019 ->  82.84.114.178 3389 .APRS.      35     3117     1     89       14
TCP      150.41.46.50 62651 ->   82.82.77.147 3389 .APRSF      25     2607     1    104        8
TCP      150.41.46.50  9376 ->   82.82.88.238 3389 .APRSF      27     2689     1     99        5
TCP      150.41.46.50 14868 ->    82.82.64.77 3389 .APRSF      28     2725     1     97       10
TCP      150.41.46.50 30516 ->  82.83.231.129 3389 .APRS.      39     3353     1     85       20
TCP      150.41.46.50 48966 ->  82.84.114.134 3389 .APRS.      22     2497     1    113        5
TCP      150.41.46.50 65028 ->    82.84.243.4 3389 .APRS.      65     4664     1     71       12
TCP      150.41.46.50 59276 ->   82.84.92.232 3389 .APRSF      26     2639     1    101       13
TCP      150.41.46.50 12601 ->   82.84.79.152 3389 .APRS.      29     2833     1     97       12
TCP      150.41.46.50 42981 ->  82.84.249.253 3389 .APRS.      34     3172     1     93        6
TCP      150.41.46.50  1157 ->  82.82.110.115 3389 .APRSF      25     2589     1    103        8
TCP      150.41.46.50 37901 ->    82.82.79.11 3389 .APRSF      25     2599     1    103        8
TCP      150.41.46.50 54725 ->   82.82.70.92 3389 .APRSF      25     2589     1    103        8
TCP      150.41.46.50 37208 ->  82.82.111.115 3389 .APRSF      25     2587     1    103        8
TCP      150.41.46.50 55340 ->  82.84.216.188 3389 .APRS.      38     3317     1     87       14
TCP      150.41.46.50 27407 ->   82.82.65.251 3389 .APRSF      25     2617     1    104        8
TCP      150.41.46.50 11278 ->   82.82.79.218 3389 .APRS.      33     3049     1     92       11
TCP      150.41.46.50  3829 ->  82.83.206.120 3389 .APRSF      71     4915     1     69       20
IP addresses anonymised
Summary: total flows: 82416, total bytes: 260.3 M, total packets: 2.9 M, avg bps: 5.4 M, avg pps:
     7628, avg bpp: 88
Time window: <unknown>
Total flows processed: 82416, Blocks skipped: 0, Bytes read: 3956112
Sys: 0.039s flows/second: 2113230.8 Wall: 0.037s flows/second: 2226617.0
```

```
Proto      Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP      150.41.46.50 59446 ->  82.84.114.220 3389 .APRSF      27     2683     1     99       12
TCP      150.41.46.50 55149 ->  82.84.114.220 3389 .APRSF      28     2723     1     97       12
TCP      150.41.46.50 41169 ->  82.84.114.220 3389 .APRSF      28     2715     1     96       12
TCP      150.41.46.50 61501 ->  82.84.114.220 3389 .APRSF      27     2685     1     99       12
TCP      150.41.46.50 58878 ->  82.84.114.220 3389 .APRSF      27     2712     1    100        5
TCP      150.41.46.50 34390 ->  82.84.114.220 3389 .APRSF      27     2679     1     99       12
TCP      150.41.46.50 58184 ->  82.84.114.220 3389 .APRS.      32     2999     1     93       14
TCP      150.41.46.50 51614 ->  82.84.114.220 3389 .APRSF      28     2723     1     97       12
TCP      150.41.46.50 16948 ->  82.84.114.220 3389 .APRSF      25     2634     1    105        4
TCP      150.41.46.50 12459 ->  82.84.114.220 3389 .APRS.      29     2868     1     98        5
IP addresses anonymised
Summary: total flows: 116, total bytes: 315833, total packets: 3154, avg bps: 8330, avg pps: 10, avg
     bpp: 100
Time window: <unknown>
Total flows processed: 116, Blocks skipped: 0, Bytes read: 5676
Sys: 0.005s flows/second: 23200.0 Wall: 0.004s flows/second: 28884.5
```

```
Proto      Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP      150.41.46.50 30652 ->  82.84.114.220 3389 .APRSF      27     2677     1     99       12
TCP      150.41.46.50 56250 ->  82.84.114.220 3389 .APRS.      25     2613     1    104        5
TCP      150.41.46.50 29599 ->  82.84.114.220 3389 .APRSF      27     2665     1     98       12
TCP      150.41.46.50 17027 ->  82.84.114.220 3389 .APRSF      28     2719     1     97       12
TCP      150.41.46.50  2507 ->  82.84.114.220 3389 .APRSF      26     2660     1    102        5
TCP      150.41.46.50 18192 ->  82.84.114.220 3389 .APRSF      27     2679     1     99       12
TCP      150.41.46.50 19008 ->  82.84.114.220 3389 .APRSF      27     2683     1     99       12
TCP      150.41.46.50 46594 ->  82.84.114.220 3389 .APRS.      26     2665     1    102       11
TCP      150.41.46.50 53787 ->  82.84.114.220 3389 .APRSF      28     2721     1     97       12
TCP      150.41.46.50 60041 ->  82.84.114.220 3389 .APRS.      26     2647     1    101        6
IP addresses anonymised
Summary: total flows: 1004, total bytes: 2.7 M, total packets: 27103, avg bps: 8085, avg pps: 10, avg
      bpp: 100
Time window: <unknown>
```

```
Total flows processed: 1004, Blocks skipped: 0, Bytes read: 48300
Sys: 0.007s flows/second: 143428.6 Wall: 0.005s flows/second: 190983.5
```

Výpis F.4: Hádání hesla ke službě RDP.

Příloha G

Ukázky detekovaných anomálií

Významy jednotlivých sloupců použitých ve výpisech se nacházejí v příloze F.

G.1 Skenování služby MySQL Database Server

Jedná se o velmi velké skenování portu 3306, které během 10-ti minut obsahovalo téměř 330000 toků.

Proto	Src IP Addr	Src Pt		Dst IP Addr	Dst Pt	Flags	Packets	Bytes	Flows	Bpp	pps
TCP	238.113.8.167	6000	->	150.41.227.71	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.43.66.71	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.228.18	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.59.221	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.74.238	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.186.76	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.53.197	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.64.122	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.196.158	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.225.220	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.84.81	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.219.151	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.43.162.216	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.158.40	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.43.193.138	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.169.111	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.220.239	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.203.185	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.41.250.116	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.5.55	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.170.238	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.82.107	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.8.191	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.250.86	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.117.237	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.41.254.247	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.181.115	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.196.222	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.42.116	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.43.129.85	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.81.189	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.200.131	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.242.184	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.43.202.108	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.42.103.227	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.41.14.114	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.8.237	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.40.160.59	3306	....S.	1	40	1	40	0
TCP	238.113.8.167	6000	->	150.62.232.137	3306	....S.	1	40	1	40	0

```
TCP     238.113.8.167  6000 ->   150.40.164.66  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.62.166.46  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.43.240.61  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->    150.42.254.4  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.42.83.139  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.40.80.227  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.41.90.115  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->  150.42.245.110  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->   150.62.14.213  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->  150.62.200.218  3306 ....S.      1       40     1     40        0
TCP     238.113.8.167  6000 ->    150.40.252.1  3306 ....S.      1       40     1     40        0
IP addresses anonymised
Summary: total flows: 327567, total bytes: 13.7 M, total packets: 332510, avg bps: 359884, avg pps:
     1093, avg bpp: 41
Time window: <unknown>
Total flows processed: 327567, Blocks skipped: 0, Bytes read: 15723492
Sys: 0.119s flows/second: 2752663.9 Wall: 0.119s flows/second: 2749221.6
```

Výpis G.1: Útok na službu MySQL Database Server

G.2 Skenování služby Microsoft SQL Server

Rozsáhlého skenování portu se nevyhla ani tato služba. Během jednoho pětiminutového intervalu se vyskytlo 64000 toků směřující na port 1433.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP    244.236.59.108  6000 ->    246.13.60.19  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.118.172  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.58.245  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.124.249  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.87.136  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.46.113  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.119.217  1433 ....S.      1      40     1    40        0
TCP    244.236.59.108  6000 ->    246.13.64.12  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.56.175  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.120.135  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.25.113  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.47.74  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.108.125  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.57.29  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.109.52  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.48.25  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.32.217  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.53.193  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.39.151  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.33.96  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.99.150  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.91.192  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.37.37  1433 ....S.      1      40     1    40        0
TCP    244.236.59.108  6000 ->    246.13.37.53  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.42.178  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.7.238  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->  246.13.118.153  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.20.145  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.1.139  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.102.74  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.14.240  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.17.111  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.59.243  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.47.213  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.30.219  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.43.61  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.82.89  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.6.107  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->    246.13.82.16  1433 ....S.      2      80     2    40        0
TCP    244.236.59.108  6000 ->   246.13.18.236  1433 ....S.      2      80     2    40        0
```

```
TCP    244.236.59.108 6000 ->     246.13.1.27 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->   246.13.116.74 1433 ....S.      1      40    1    40       0
TCP    244.236.59.108 6000 ->   246.13.10.167 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->    246.13.74.17 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->  246.13.111.232 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->    246.13.18.44 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->   246.13.50.211 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->   246.13.87.183 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->    246.13.31.50 1433 ....S.      2      80    2    40       0
TCP    244.236.59.108 6000 ->    246.13.16.99 1433 ....S.      2      80    2    40       0
IP addresses anonymised
Summary: total flows: 64035, total bytes: 2.6 M, total packets: 64036, avg bps: 260342, avg pps: 813,
      avg bpp: 40
Time window: <unknown>
Total flows processed: 64035, Blocks skipped: 0, Bytes read: 3073812
Sys: 0.035s flows/second: 1829571.4 Wall: 0.034s flows/second: 1838079.1
```

Výpis G.2: Útok na službu Microsoft SQL Server

G.3 Hádání hesla ke službě RDP

Ukázka obsahuje celkový součet všech toků směřující z jedné IP adresy na port 3389. Jednalo se o téměř 1mln toků. Co činilo cca. o 270 toků za vteřinu z této konkrétní IP adresy.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp     pps
TCP     150.41.46.50 40275 ->   82.84.87.240 3389 .AP.SF      25    3223     1   128       3
TCP     150.41.46.50  7500 ->   82.82.71.113 3389 .APRSF      28    2721     1    97       9
TCP     150.41.46.50 16776 ->   82.84.92.232 3389 .APRSF      26    2635     1   101      11
TCP     150.41.46.50  3314 ->  82.87.149.132 3389 .APRS.      62    4510     1    72       8
TCP     150.41.46.50  5082 ->  82.83.207.242 3389 .APRS.      47    3475     1    73      17
TCP     150.41.46.50 53667 ->   82.84.87.198 3389 .APRSF      24    2547     1   106       8
TCP     150.41.46.50 64727 ->    82.84.70.28 3389 .APRSF      28    2950     1   105      17
TCP     150.41.46.50 45195 ->  82.84.253.122 3389 .APRS.      59    4172     1    70       9
TCP     150.41.46.50  2538 ->    82.85.50.13 3389 .APRS.      34    3255     1    95      12
TCP     150.41.46.50 60361 ->   82.84.87.201 3389 .AP.SF      25    2659     1   106       9
TCP     150.41.46.50  2191 ->  82.87.149.245 3389 .APRS.      33    3125     1    94      11
TCP     150.41.46.50 64928 ->    82.82.81.70 3389 .APRSF      27    2693     1    99       6
TCP     150.41.46.50 54619 ->  82.82.111.110 3389 .APRSF      25    2993     1   119       4
TCP     150.41.46.50 56480 ->   82.84.91.208 3389 .APRSF      25    2597     1   103       8
TCP     150.41.46.50 46312 ->  82.84.126.112 3389 .APRSF      71    4668     1    65      28
TCP     150.41.46.50 38353 ->   82.84.246.12 3389 .APRS.      40    3401     1    85      20
TCP     150.41.46.50 62202 ->  82.86.238.188 3389 .APRSF      26    2679     1   103       6
TCP     150.41.46.50 36216 ->  82.84.250.187 3389 .APRS.      66    5327     1    80      14
TCP     150.41.46.50 61766 ->   82.84.91.208 3389 .APRSF      25    2627     1   105       6
TCP     150.41.46.50 39097 ->   82.82.108.18 3389 .APRS.      33    3039     1    92       6
TCP     150.41.46.50 18682 ->    82.84.70.26 3389 .APRSF      27    2904     1   107      10
TCP     150.41.46.50 23305 ->  82.84.213.227 3389 .APRS.      61    4235     1    69      13
TCP     150.41.46.50 15735 ->  82.84.253.167 3389 .APRS.      60    4349     1    72      20
TCP     150.41.46.50 25553 ->   82.82.66.187 3389 .APRSF      27    2673     1    99       9
TCP     150.41.46.50 19994 ->  82.82.111.217 3389 .APRS.      44    3549     1    80      19
TCP     150.41.46.50  4028 ->  82.84.109.162 3389 .APRSF      29    2753     1    94      13
TCP     150.41.46.50 45231 ->    82.84.70.43 3389 .APRSF      25    2607     1   104      13
TCP     150.41.46.50  6266 ->   82.84.92.172 3389 .APRSF      27    2675     1    99       9
TCP     150.41.46.50 24217 ->  82.84.247.155 3389 .APRS.      32    3077     1    96      16
TCP     150.41.46.50 19511 ->   82.82.108.23 3389 .APRSF      25    2593     1   103       8
TCP     150.41.46.50  6268 ->    82.82.73.21 3389 .APRSF      25    2599     1   103       9
TCP     150.41.46.50 16036 ->  82.84.114.200 3389 .APRSF      26    2655     1   102       6
TCP     150.41.46.50 34102 ->  82.83.207.242 3389 .APRS.      46    3439     1    74      17
TCP     150.41.46.50 48283 ->  82.84.114.185 3389 .APRS.      29    2869     1    98      13
TCP     150.41.46.50 58046 ->   82.85.42.228 3389 .APRS.      33    3085     1    93      17
TCP     150.41.46.50 51102 ->  82.84.126.120 3389 .APRS.      27    3042     1   112       8
TCP     150.41.46.50 11100 ->    82.86.4.119 3389 .APRS.      62    4357     1    70      24
TCP     150.41.46.50  9204 ->  82.84.114.131 3389 .APRSF      27    2671     1    98       5
TCP     150.41.46.50 52154 ->   82.84.219.53 3389 .APRS.      69    5009     1    72      19
TCP     150.41.46.50 33968 ->   82.84.92.168 3389 .APRSF      25    2603     1   104       9
TCP     150.41.46.50 15247 ->    82.86.13.54 3389 .APRS.      37    3281     1    88      20
```

```
TCP      150.41.46.50 34599 ->  82.84.249.230 3389 .APRS.     66     4565    1     69       17
TCP      150.41.46.50 29121 ->  82.84.114.200 3389 .APRS.     29     2831    1     97        6
TCP      150.41.46.50 56548 ->   82.82.71.226 3389 .APRSF     25     2617    1    104        8
TCP      150.41.46.50 27671 ->   82.84.227.92 3389 .APRS.     33     3206    1     97       10
TCP      150.41.46.50 49948 ->    82.84.91.76 3389 .APRS.     59     4141    1     70       22
TCP      150.41.46.50 19879 ->  82.84.114.176 3389 .APRSF     25     2605    1    104       11
TCP      150.41.46.50 62675 ->   82.84.254.58 3389 .APRSF     63     4305    1     68       22
TCP      150.41.46.50 39432 ->     82.82.76.5 3389 .APRS.     33     3053    1     92       11
TCP      150.41.46.50 50953 ->    82.84.70.43 3389 .APRSF     25     2609    1    104       14
IP addresses anonymised
Summary: total flows: 950647, total bytes: 3.1 G, total packets: 35.0 M, avg bps: 6.1 M, avg pps:
     8752, avg bpp: 87
Time window: <unknown>
Total flows processed: 950647, Blocks skipped: 0, Bytes read: 45631680
Sys: 0.317s flows/second: 2998886.4 Wall: 0.318s flows/second: 2987295.4
```

Výpis G.3: Hádání hesla k RDP

G.4 Hádání hesla ke službě SSH

Níže uvedené ukázky obsahují celkové součty všech toků směřující z jednotlivých IP adresy na port 22.

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP     17.210.41.234 42854 ->  247.81.18.187    22 .AP.SF      14     1425    1   101        0
TCP     17.210.41.234 56737 -> 247.81.116.106    22 .AP.SF      13     1305    1   100        2
TCP     17.210.41.234 50506 ->  247.81.77.149    22 .AP.SF      14     1373    1    98        3
TCP     17.210.41.234 44829 ->  247.81.60.172    22 .AP.SF      13     1321    1   101        2
TCP     17.210.41.234 38196 -> 247.81.242.195    22 .AP.SF      14     1201    1    85        3
TCP     17.210.41.234 50762 ->  247.81.62.154    22 .AP.SF      13     1305    1   100        1
TCP     17.210.41.234 41809 ->  247.81.49.250    22 .AP.SF      14     1217    1    86        2
TCP     17.210.41.234 50051 ->  247.81.238.56    22 .AP.SF      13     1161    1    89        3
TCP     17.210.41.234 33203 ->   247.81.95.18    22 .AP.SF      13     1305    1   100        3
TCP     17.210.41.234 57358 ->  247.81.11.220    22 .AP.SF      13     1321    1   101        1
TCP     17.210.41.234 42919 ->   247.81.88.27    22 .AP.SF      13     1305    1   100        2
TCP     17.210.41.234 55349 ->   247.81.46.27    22 .AP.SF      14     1409    1   100        3
TCP     17.210.41.234 33940 ->   247.81.56.14    22 .AP.SF      13     1305    1   100        2
TCP     17.210.41.234 49116 ->  247.81.65.183    22 .AP.SF      14     1409    1   100        3
TCP     17.210.41.234 48469 ->  247.81.235.83    22 .AP.SF      13     1305    1   100        2
TCP     17.210.41.234 55936 ->  247.81.72.154    22 .AP.SF      14     1409    1   100        2
TCP     17.210.41.234 47176 ->  247.81.65.185    22 .AP.SF      14     1409    1   100        3
TCP     17.210.41.234 55426 ->  247.81.94.213    22 .AP.SF      14     1425    1   101        1
TCP     17.210.41.234 39718 ->   247.81.229.7    22 .AP.SF      13     1305    1   100        2
TCP     17.210.41.234 47789 -> 247.81.247.102    22 .AP.SF      13     1305    1   100        1
IP addresses anonymised
Summary: total flows: 127647, total bytes: 170.0 M, total packets: 1.7 M, avg bps: 344766, avg pps:
     433, avg bpp: 99
Time window: <unknown>
Total flows processed: 127647, Blocks skipped: 0, Bytes read: 6127224
Sys: 0.046s flows/second: 2774934.8 Wall: 0.043s flows/second: 2907080.6
```

```
Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP      87.179.112.7 47405 ->  150.42.12.171    22 .AP.SF      12     1168    1    97        3
TCP      87.179.112.7 44824 -> 150.42.251.185    22 ....S.       2      120    1    60        0
TCP      87.179.112.7 53780 -> 150.43.113.228    22 .AP.SF      12     1152    1    96        2
TCP      87.179.112.7 57581 ->  150.42.142.96    22 ....S.       2      120    1    60        0
TCP      87.179.112.7 54647 -> 150.42.105.143    22 ....S.       2      120    1    60        0
TCP      87.179.112.7 55903 ->   150.43.112.1    22 .AP.SF      12     1168    1    97        2
TCP      87.179.112.7 40390 -> 150.43.113.227    22 .AP.SF      12     1152    1    96        2
TCP      87.179.112.7 43510 ->   150.42.1.149    22 .AP.SF      12     1152    1    96        1
TCP      87.179.112.7 34844 -> 150.42.127.213    22 ....S.       2      120    1    60        0
TCP      87.179.112.7 44727 ->  150.42.63.186    22 ....S.       2      120    1    60        0
TCP      87.179.112.7 53328 ->  150.43.112.39    22 .AP.SF      12     1152    1    96        2
TCP      87.179.112.7 42915 ->  150.42.132.17    22 .A.RSF       6      260    1    43       24
TCP      87.179.112.7 47724 ->  150.43.113.77    22 .AP.SF      12     1152    1    96        2
```

```
TCP      87.179.112.7 53493 ->  150.42.158.114   22 .A..SF      4     180    1     45      13
TCP      87.179.112.7 40962 ->   150.43.113.80   22 .AP.SF     12    1152    1     96       2
TCP      87.179.112.7 41006 ->  150.42.231.139   22 ....S.      2     120    1     60       0
TCP      87.179.112.7 52529 ->    150.43.112.4   22 .AP.SF     12    1152    1     96       2
TCP      87.179.112.7 35696 ->  150.43.113.228   22 .AP.SF     12    1152    1     96       2
TCP      87.179.112.7 33240 ->   150.43.112.12   22 .AP.SF     12    1152    1     96       2
TCP      87.179.112.7 58241 ->   150.43.112.34   22 .AP.SF     12    1152    1     96       2
IP addresses anonymised
Summary: total flows: 87224, total bytes: 65.9 M, total packets: 719049, avg bps: 134634, avg pps:
     183, avg bpp: 91
Time window: <unknown>
Total flows processed: 87224, Blocks skipped: 0, Bytes read: 4186896
Sys: 0.034s flows/second: 2565411.8 Wall: 0.031s flows/second: 2726943.0


Proto     Src IP Addr Src Pt      Dst IP Addr Dst Pt Flags Packets  Bytes Flows   Bpp      pps
TCP    87.243.190.125 55898 ->  150.42.204.94   22 ....S.       1     60     1    60        0
TCP    87.243.190.125 49422 ->  150.43.112.99   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 48371 -> 150.42.224.156   22 ....S.       2    120     1    60        0
TCP    87.243.190.125 42638 ->  150.42.14.145   22 .AP.SF      12   1152     1    96        3
TCP    87.243.190.125 41760 -> 150.42.189.110   22 ....S.       2    120     1    60        0
TCP    87.243.190.125 42543 ->   150.42.8.215   22 .AP.SF      12   1020     1    85        9
TCP    87.243.190.125 56460 ->   150.42.62.96   22 ....S.       2    120     1    60        0
TCP    87.243.190.125 41943 -> 150.42.153.199   22 .A..SF       4    180     1    45       14
TCP    87.243.190.125 35891 ->    150.42.6.21   22 .AP.SF      12   1152     1    96        3
TCP    87.243.190.125 46802 -> 150.42.186.160   22 ....S.       2    120     1    60        0
TCP    87.243.190.125 56480 ->  150.43.112.40   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 46317 -> 150.43.113.181   22 .AP.SF      12   1168     1    97        1
TCP    87.243.190.125 53133 -> 150.42.155.197   22 .A..SF       4    180     1    45       14
TCP    87.243.190.125 54205 ->  150.43.112.12   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 38252 ->  150.42.131.64   22 ....S.       2    120     1    60        0
TCP    87.243.190.125 40395 -> 150.43.113.237   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 57594 ->  150.43.113.65   22 .AP.SF      12   1168     1    97        2
TCP    87.243.190.125 37863 ->  150.43.112.39   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 39106 -> 150.43.112.132   22 .AP.SF      12   1152     1    96        2
TCP    87.243.190.125 43268 ->   150.42.8.192   22 .AP.SF      12   1036     1    86        9
IP addresses anonymised
Summary: total flows: 88008, total bytes: 68.4 M, total packets: 745916, avg bps: 139724, avg pps:
     190, avg bpp: 91
Time window: <unknown>
Total flows processed: 88008, Blocks skipped: 0, Bytes read: 4224540
Sys: 0.038s flows/second: 2316000.0 Wall: 0.036s flows/second: 2443173.6
```

Výpis G.4: Hádání hesla k SSH