Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práce

# Detekce anomálií síťového provozu pomocí data miningové analýzy síťových toků

***Bc. Petr Lessner***

Vedoucí práce: Mgr. Rudolf Bohumil Blažek, Ph.D.

29. června 2015

# Poděkování

Děkuji především rodině za podporu a svému vedoucímu za cenné rady a čas strávený konzultacemi se mnou.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 29. června 2015 .....

### Odkaz na tuto práci

Lessner, Petr. *Detekce anomálií síťového provozu pomocí data miningové analýzy síťových toků.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.

# Abstrakt

Tato práce se zabývá vytvořením zásuvného modulu do systému detekce síťových utoků NfSen pro behaviorální analýzu síťových toků. Cílem modulu je detekovat anomální chování hostů, které mohou znamenat útok či poruchu v síti. Vytvořený modul využívá algoritmu shlukové analýzy DBSCAN v kombinaci s detekcí odlehlých bodů pomocí algoritmu LOF. Modul řadí detekované anomálie podle jejich významnosti, která odpovídá míře odlehlosti LOF. Modul byl otestován na reálných datech z provozu reálné sítě, kde se podařilo odhalit několik různých útoků.

**Klíčová slova** behaviorální analýza, data-mining, shluková analýza, DBSCAN, LOF, NetFlow, IPFIX, NfSen.

# Abstract

This thesis deals with development of a plugin for an intrusion detection system NfSen that implements behavioral analysis of network flows. The purpose of the plugin is to detect anomalous behavior of hosts that may represent an attack or a failure in the network. The developed plugin employs a clustering algorithm DBSCAN in combination with detection of outliers via the LOF algorithm. The plugin orders detected anomalies by their significance that corresponds to the outlier factor LOF. The plugin was tested with data from a real network traffic where it detected several different anomalies.

**Keywords** behavioral analysis, data-mining, clustering, DBSCAN, LOF, NetFlow, IPFIX, NfSen.

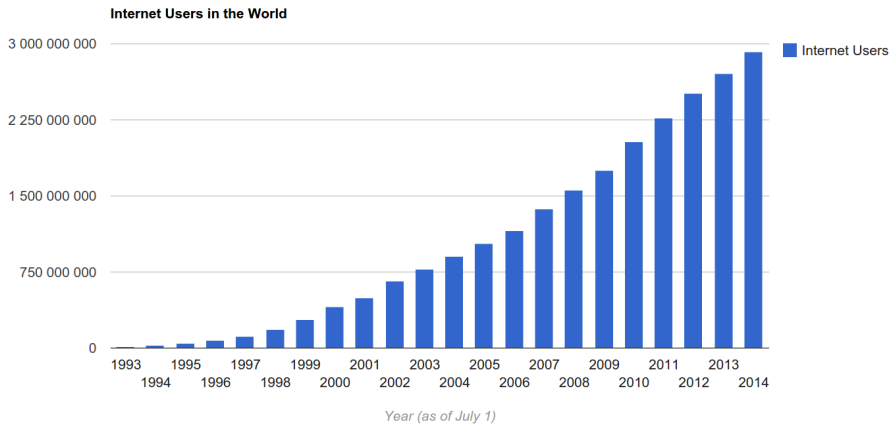
# Obsah

# Seznam obrázků

# Úvod

Internet celosvětově roste a to z několika různých pohledů. Roste počet uživatelů, kteří mají přístup k internetu, jak je vidět na obrázku 0.1. S tím ruku v ruce roste i množství zařízení připojených k internetu. Nejde pouze o klasické počítače a notebooky, ale i o (nejen chytré) mobilní telefony (zvláště v rozvojových zemích), tablety a všemožná zařízení rozvíjejícího se internetu věcí (Internet of Things). Zvětšuje se samozřejmě také celkový objem přenesených dat i rychlost připojení k internetu. Prognózy předpokládají další růst objemů a dalších ukazatelů (například podle zprávy Cisca [1]).

Se zvyšováním provozu roste i počet útoků, které jsou z velké části vedené právě přes internet. Roste nejenom samotný počet, ale přibývá i množství nových útoků [3, 4]. K zabezpečení proti těmto útokům se používá celá škála způsobů obrany – od fyzického zabezpečení přes softwarová řešení až po zvyšování povědomí uživatelů o počítačové a síťové bezpečnosti. Prvním stupněm ochrany vnitřní sítě je firewall, který filtruje provoz z vnější do vnitřní sítě




Obrázek 0.1: Vývoj počtu uživatelů internetu. Zdroj: [2]

a obráceně na základě nastavených pravidel a povolených služeb. Uvnitř sítě pak může být nasazen systém detekce útoků ( IDS – Intrusion Detection System), který monitoruje provoz na vnitřní síti a snaží se odhalit známé útoky na základě signatur, či neznámé anomálie pomocí statistických metod nebo neobvyklé chování počítačů pomocí behaviorální analýzy.

Signatury jsou určité vzory známých útoků. IDS má databázi těchto signatur a porovnává je s aktuálním provozem. V případě shody upozorní správce sítě nebo danou síťovou komunikaci rovnou zastaví, pokud je systém v řežimu protekce – IPS či IDPS (Intrusion Protection System respektive Intrusion Detection and Prevention System).

Toto řešení má nevýhodu v nutnosti neustálé aktualizace databáze signatur, buď vlastním vytvářením pravidel signatur nebo častěji pomocí předplatného u dodavatele systému. Tyto IDS nejsou schopné odhalit neznámé útoky. Než se nový útok odhalí a vytvoří se patřičná signatura, síť je potenciálně napadnutelná.

Tyto problémy se snaží odstranit metody pro detekci statistických anomálií, které již bývají součástí pokročilejších IDS. Pomocí statistických metod lze detekovat i neznámé útoky. Tyto metody však obvykle vyžadují modelování normálního provozu, což je velmi obtížné. Nejpokročilejšími jsou behaviorální systémy, které automaticky hledají neobvyklé (dříve nepozorované) chování počítačů, uživatelů či síťových prvků. Cílem této práce je právě návrh a implementace pluginu do síťové sondy NfSen [5], která bude vyhledávat síťové anomálie pomocí data miningové analýzy síťového provozu.

## Struktura práce

Diplomová práce je rozdělena do čtyř základních kapitol. V první kapitole se věnuji analýze celé problematiky síťových anomálií a způsobů jejich detekce. První sekce se zabývají způsoby sběru dat ze sítě – hlavně tedy síťovými toky. Dále je definován pojem anomálie a rozebrány druhy anomálií. Je zde také popsán způsob zpracování síťových toků pro vytvoření atributů. Následuje popis behaviorální analýzy a v ní používaných shlukovacích algoritmů a algoritmů pro detekci odlehlých bodů.

V druhé kapitole je popsán návrh řešení. Od výběru vhodného programovacího jazyka a jeho knihoven pro zpracování, analýzu a vizualizaci dat, až po navržení samotného způsobu detekce a hodnocení anomálií. Je zde popsáno deset vybraných konkrétních sledovaných atributů a jejich další statistické zpracování, aby mohly být použity ve zvoleném shlukovacím algoritmu DBSCAN. Návrh obsahuje také tři možnosti odhalení anomálií a jejich ohodnocení závažnosti.

V kapitole Implementace jsou stručně popsané použité nástroje (tedy NfSen a nfdump) a případné obtíže s jejich instalací či provozem. Dále je zde rozebráno konkrétní provedení jednotlivých kroků postupu při detekci ano-

málií včetně problémů, na které jsem během vývoje narazil a jejich řešení. Přiložen je také podrobný návod k instalaci vytvořeného pluginu a základní pokyny k užívání.

V rámci testování jsou zdokumentovány nalezené anomálie a je provedena jejich hlubší analýza pro vyhodnocení správnosti fungování pluginu. Jsou zde také uvedeny výsledky měření výkonu pluginu na datech z mého počítače i ze záznamu provozu části fakultní sítě.

Kapitola 1

# Analýza

Tato kapitola v první části pojednává o sledování síťového provozu se zvláštním zaměřením na síťové toky. Vyzkoušel jsem si programy nfdump a NfSen pro práci se síťovými toky (jsou popsané v kapitole Implementace v sekcích 3.1.1.1 respektive 3.1.4). Druhá část kapitoly analyzuje druhy anomálií a jejich detekce zvláště pomocí shlukové analýzy a detekce odlehlých bodů.

## 1.1 Data pro analýzu síťového provozu

Pro analýzu síťového provozu se používají různé druhy dat. Mohou to být celé pakety z libovolné síťové vrstvy či pouze jejich část např. pouze hlavičky paketů nebo objem přenášených dat. K těmto datům lze přidat různá metadata v podobě časové značky, použitého rozhraní apod. Zaznamenaná data lze dále filtrovat (např. podle vybrané služby nebo příjemce) a také agregovat podle různých kritérií (např. podle služby/protokolu) a napočítávat různé statistické údaje (např. počty paketů, celkovou nebo průměrnou velikost apod.). V případě nemožnosti zaznamenávat veškerá požadovaná data ze sítě (nebo pro snížení zátěže zařízení, které data zpracovává), je možné použít vzorkování, tedy zpracovávat pouze zlomek provozu (například každý $n$-tý paket). Záleží na konkrétním účelu pořízování dat a také na hardwarových a softwarových možnostech použitého zařízení. Nejrozšířenější jsou dva přístupy: ukládání celých paketů nebo tzv. síťových toků, případně jejich kombinace. Tyto přístupy jsou dále rozebrány v následujících sekcích.

### 1.1.1 Odchytávání paketů

Jednou z možností zpracování dat ze sítě je odchytávání celých paketů. Ostatní možnosti jsou založené na zpracování těchto paketů a uchování pouze agregovaných informací pro skupiny paketů (např. pro síťové toky, viz sekce 1.1.2). Pakety lze buď rovnou zpracovávat nebo ukládat pro pozdější analýzu. Na rozdíl od síťových toků je poměrně nepraktické a drahé posílat všechny od-

chytané pakety ze síťových prvků na nějaký sběrný centrální server. Bylo by totiž nutné zvýšit kapacitu spojů.

Hlavní nevýhodou zaznamenávání celých paketů je tedy množství dat. Zaznamenávání veškerých dat ze sítě je (alespoň od středně velkých sítí) kapacitně velmi náročné. Z tohoto důvodu se obvykle zaznamenává pouze nějaká část dat pro konkrétní účel (typicky pro ruční nikoliv automatickou analýzu). Může to být krátký časový úsek nebo je použit dostatečně specifický filtr (např. pouze pro odchozí e-maily: pakety z dané IP adresy s cílovým portem 25 SMTP – Simple mail transfer protocol).

Z důvodu ochrany listovního tajemství[1] není legální nahlížet do soukromé komunikace. Výše trestu může být podle trestního zákoníku až pět let odnětí svobody [2]. Oproti tomu má ale zaměstnavatel právo na ochranu svého majetku, případně dalších aktiv, a na kontrolu svých zaměstnanců. Monitorování zaměstnance by mělo být přiměřené a nemělo by nadměrně nenarušovat jeho soukromí. Co znamená přiměřené a nadměrné v zákoně popsáno není a záleží tedy na případném zhodnocení soudem [8]. Z tohoto důvodu může být vhodnější používat pouze hlavičky paketů nebo například pouze adresy navštívených webových stránek a nikoliv samotný obsah komunikace.

### 1.1.2 Síťové toky

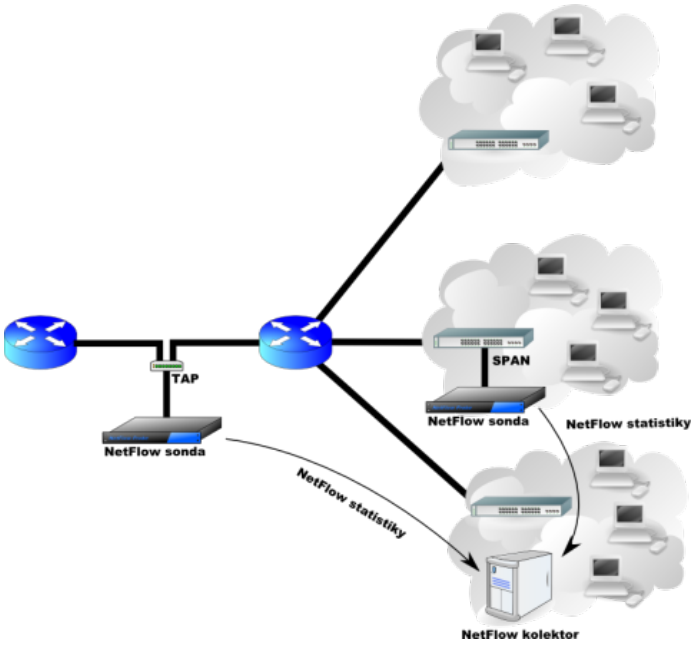
Síťové toky jsou obdobou výpisu telefonních hovorů ve světě počítačových sítí. Jsou to záznamy jednosměrných síťových spojení mezi dvojicí komunikujících prvků. Jeden síťový tok tedy tvoří IP pakety, které se shodují v následujících položkách:

- IP adresa odesilatele a adresáta
- port odesilatele a adresáta (u protokolu TCP či UDP)
- IP protokol
- případně další položky, například:
  - IP ToS (Type of Service)
  - číslo rozhraní (vstupní i výstupní)

Kromě těchto údajů se ukládají také další metadata: čas začátku a doba spojení, počet paketů, objem přenesených dat a množina všech vyskytnuvších se příznaků (u TCP) a případně další údaje v závislosti na použitém protokolu viz 1.1.2.3.

[1] Listina základních práv a svobod, článek 13 [6]

[2] Trestní zákoník § 182: Porušení tajemství dopravovaných zpráv [7]




Obrázek 1.1: Příklad zapojení NetFlow exportérů a kolektoru. Zdroj: [10]

#### 1.1.2.1 Architektura sběru síťových toků

Architektura sběru síťových toků se typicky skládá z několika exportérů a alespoň jednoho kolektoru. Exportéry neboli sondy sledují provoz v síti např. v několika uzlech sítě, sestavují síťové toky a ty pak posílají na kolektor, kde se data ukládají a dále zpracovávají, analyzují a případně prezentují. Exportéry mohou být buď chytřejší (a tedy dražší) síťové prvky (přepínače, směrovače) nebo speciální zařízení, které se připojí přes TAP (Test Access Point) zařízení či pomocí SPAN (Switch Port Analyzer) portu na přepínači [9]. TAP (neboli odposlech) je zařízení, které se vloží inline mezi síťové prvky, jejichž provoz kopíruje na úrovni linkové vrstvy a posílá na zvláštní linku, kam se připojí kolektor. Výhodou je absolutní transparentnost, TAP nemá žádnou adresu a lze na něj tedy jen velmi obtížně útočit. SPAN port na přepínači kopíruje provoz z jednoho či více portů přepínače. Nevýhoda SPAN portu je, že při větším vytížení přepínače nezaručuje zkopírování veškerého provozu (upřednostňuje přepínání jako svojí hlavní funkci). Příklad zapojení exportérů a kolektoru v síti je na obrázku 1.1, síťové toky jsou zde vedeny zvláštní linkou.

#### 1.1.2.2 Průběh měření

Exportér si udržuje tabulku právě probíhajících toků (tzv. flow cache), která ve sloupcích obsahuje pro každý tok čas jeho počátku, počty paketů a bajtů a také množinu všech příznaků a případně další údaje v závislosti na použitém protokolu a nastavení sondy viz sekci 1.1.2.3. V řádcích jsou jednotlivé

právě probíhající (aktivní) toky, které jsou definovány pěticí shodných údajů: zdrojovou a cílovou adresou a porty a identifikačním číslem protokolu. Nově příchozí paket tedy buď způsobí aktualizaci záznamu v tabulce, pokud takový existuje, nebo založení nového záznamu. Při ukončení toku je pak záznam poslán na kolektor. Z důvodu bezpečnosti či kvalitnějšího monitoringu sítě bývá spojení do kolektoru na oddělené síti.

Záznam o toku může být také ukončen po vypršení tzv. pasivního timeoutu, tedy pokud se po dobu timeoutu nevyskytne žádný další paket spadající do tohoto toku. Toky s velmi dlouhým trváním by se však takto v kolektoru objevily až za dlouhou dobu a mohli by se takto vlastně skrývat. Proto se používá také aktivní timeout, který po nějaké době od začátku toku (typicky po pěti minutách) „uřízne“ daný tok a odešle záznam v aktuální podobě do kolektoru, záznam odstraní a v případě pokračování toku vytvoří záznam nový. V kolektoru je pak možné tyto části zase poskládat do jednoho toku. Příklad dat poslaných na kolektor je vidět na obr. 3.2.

#### 1.1.2.3 Protokoly

Existuje několik protokolů pro síťové toky. Jako nejpoužívanější se uvádí protokol NetFlow (verze 5 a 9) od firmy Cisco. Mnoho ostatních firem vytvořilo své protokoly s obdobnou funkčností (například Jflow, CFlow a další) [11].

**NetFlow** je zveřejněný standard firmy Cisco pro síťové toky. Jeho hlavní použití je pro monitoring sítě, sledování zatížení jednotlivých síťových prvků a linek zvláště pro účely plánování a rozvoje sítě a dále také pro účtování využití sítě (například pro operátory) nebo pro odhalení poruchy či špatné konfigurace sítě. Dokument popisující NetFlow protokol verze 9 je RFC 3954 (Request For Comments) [12] (pouze informační dokument, nikoliv standard).

**IPFIX** (IP Flow Information eXport) je otevřený průmyslový standard IETF (Internet Engineering Task Force) pro síťové toky, který vychází z protokolu NetFlow verze 9 a proto někdy bývá označován jako NetFlow verze 10 (X). IPFIX umožňuje snadnou rozšiřitelnost posílaných údajů. Předpokládá se postupný přechod na tento standard. Základní dokument standardu, který popisuje IPFIX je RFC 7011 [13]. Další rozšiřující dokumenty jsou odkazovány z tohoto základního.

### 1.1.3 Kombinace metod

Velmi výhodné je, pokud existuje možnost získávání jak síťových toků, tak i celých paketů. Síťové toky mohou odhalit nějaký problém (ať už bezpečností nebo funkční) a v následně odchytnutých paketech pak lze nalézt podrobné informace. Není už nutné odchytávat všechny pakety a po celou dobu provozu, ale jenom úzce vyfiltrovaná data například konkrétního protokolu, odesilatele

a příjemce. Kombinaci záznamu síťových toků i celých paketů nově umožňuje sonda FlowMon (podrobně viz 3.1.3), konkrétně její (softwarový) plugin FlowMon Traffic Recorder [14].

## 1.2 Anomálie v síťovém provozu

Podle on-line slovníku cizích slov je anomálie nepravidelnost, výjimečnost, odchylka, úchylka od normálu [15]. V oblasti počítačových sítí v tradičním pojetí jde o odchylky především od specifikace protokolu (průmyslového standardu) nebo od akceptovaných pravidel či konvencí pro jejich použití. U dobrého standardu protokolu bývá poměrně jasné, co ho splňuje a co ne, a lze tedy vytvořit způsob, jak kontrolovat jeho dodržování. Mezi standardy jsou samozřejmě rozdíly, některé jsou dosti striktní a některé dávají určitou volnost. Část standardů jsou také brány spíše jako doporučení a jejich implementace se od sebe budou v méně podstatných rysech lišit. Pravidly rozumíme například smluvní podmínky (zvláště SLA – Service Level Agreement) poskytovatele připojení k internetu nebo pravidla použití vnitřní sítě organizace. Některá pravidla lze kontrolovat snadno (maximální rychlost) jiná obtížněji (zákaz P2P sítí, Skypu).

Obecnějším a hůře detekovatelným druhem anomálií jsou anomálie statistické, obvykle chápané buď jako změny parametrů statistického modelu pro normální provoz, nebo jako odchylky od těchto modelů. Problémem není rozlišit standardní provoz od nestandardního, ale samotná definice statistických modelů pro normální provoz. Pro detekci těchto anomálií se používá statistických metod, často neparametrických [16, 17, 18].

Podle [19] se anomálie dělí do tří základních skupin:

1. **Bodové anomálie** – jeden bod je vzdálený/odlehlý od ostatních dat. To odpovídá běžnému významu anomálie.

2. **Kontextové anomálie** – nějaký bod je odlehlý, ale pouze v určitém konkrétním kontextu neboli pohledu. Například nějaký síťový tok mezi ostatními v ničem nevyčnívá, ale při srovnání s toky stejného druhu (například DNS – Domain Name System) už je vidět značný rozdíl a může se jednat například o DNS tunelování [20].

3. **Kolektivní anomálie** – je několik bodů, mezi kterými je nějaká souvislost a které jsou jako celek anomální. Například nějaký časový sled akcí jednoho bodu, který se běžně v datech nevyskytuje. Autoři práce uvádějí příklad typického útoku na webový server jako sekvence ssh provozu, útoku typu přetečení zásobníku následovaném ftp přenosem a to vše z jedné adresy na cílový server. Tento typ anomálie je samozřejmě nejtěžší identifikovat, je tvořen vyšším vzorem než první dva typy anomálií.

Abychom rozeznali anomálie je nutné vytvořit nějaký model normálního provozu. Ten může být vytvořen na základě nějakých důvěryhodných dat. Tedy například na záznamu provozu, který byl ručně zkontrolován nebo který byl vytvořen generováním (tedy v podstatě simulací) běžného provozu. Druhým přístupem je použití záznamu reálného provozu (tedy včetně anomálií) a následné vyhledání anomálií pomocí statistických metod. Oba přístupy mají své zjevné nevýhody. První je nákladný (ruční kontrola) případně může být nedostatečně reprezentativní. Druhý má sice nejspíše reprezentativní data, ale pravděpodobně se nepodaří objevit všechny anomálie.

Normální provoz také samozřejmě záleží na parametrech dané konkrétní sítě. Jde především o účel sítě a o to, kdo (nebo co) ji používá a o jaký druh provozu se jedná. Záleží také jestli jde o malou lokální síť nebo o rozlehlou páteřní síť a také na pravidla nastavená provozovateli sítě a připojených sítí. Model normálního provozu také nemůže být statický, ale potřebuje zohledňovat aktuální čas či období. Záleží například na pracovní době (čily rozdíly během dne) nebo na jiných cyklech vždy v závislosti na konkrétní síti.

Případně nalezená anomálie může znamenat technickou závadu, chybu v konfiguraci sítě nebo síťových prvků, bezpečnostní riziko v případě útoku a nebo také falešný poplach. Může tedy jít o anomálii pouze vzhledem k vytvořenému modelu, ale nikoliv k realitě (daný provoz může být legitimní).

## 1.3 Detekce

Používá se několik způsobů detekce. Některé se soustředí na známé anomálie (tedy například na konkrétní útoky) a další se zaměřují na ještě neobjevené anomálie. Nejjednodušší je detekce podle tzv. signatur. Jde o jedno či několik poměrně jednoduchých pravidel, která se aplikují na samotné pakety určité síťové vrstvy. Takto lze odhalit útoky například na špatně implementované protokoly jako v případě útoku ping smrti (ping of death) [21]. Antivirové systémy pracují mimo jiné i na tomto principu.

Další pokročilejší způsob je pomocí jednoduchých statistik. Lze například sledovat počty SYN a FIN příznaků ve všech TCP spojeních na jednom síťovém prvku, zpracovat je pomocí statistických metod a poměrně rychle odhalit počátek DDoS útoku (Distributed Denial of Service) [22]. Tímto tématem se ve své diplomové práci zabýval Ing. Martin Neumann [23].

Na dosud neznámé anomálie se používá behaviorální analýza. Pomocí různých pokročilejších statistických metod zkoumá chování síťového provozu či účastníků a sleduje jeho vývoj v čase. Pokud se objeví nečekaná prudká změna nebo jinak neobvyklý či nepravidelný provoz, je označen jako anomálie, zaznamená se a může být provedena nějaká akce (například informovat správce sítě). Podrobněji v následující sekci.

## 1.4 Behaviorální analýza

Behaviorální analýza sleduje chování nějakých prvků. Tyto prvky mohou být například jednotlivá síťová spojení (toky), síťové prvky nebo jednotlivé podsítě. U každého prvku jsou sledovány předem vybrané atributy. Ty mohou být několika typů, například číselné hodnoty, řetězce, intervaly a další. Atributy jsou buď nějaké primární údaje (v tocích například čas spojení) anebo nějakým způsobem odvozené/vypočtené hodnoty (průměrný počet bitů za vteřinu, různé poměry apod.). Tyto atributy pak tvoří několika dimenzionální prostor (v závislosti na počtu atributů), ve kterém mohou jednotlivé prvky tvořit shluky. Pro jejich odhalení se používá shluková analýza (cluster analysis), což je jedna z technik data miningu. Jako anomální lze označit prvek, který nepatří do žádného shluku (outlier – viz sekce 1.6), nebo prvek, který v daném prostoru rychle změnil pozici.

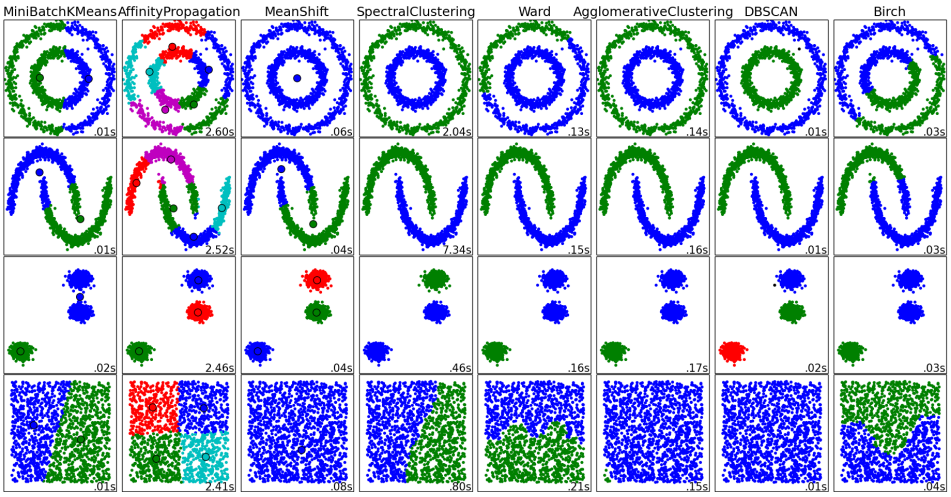
Detekce anomálií založené na behaviorální analýze může trpět problémem častých falešných poplachů. Důležité je tedy vyladit atributy a nastavení algoritmu pro shlukování pro vysokou úspěšnost odhalení anomálie a zároveň nízkou úroveň falešných poplachů (FAR – False Alarm Rate).

## 1.5 Algoritmy shlukové analýzy

Shluková analýza je tak zvané učení bez učitele. Algoritmy tedy nejsou nejdřív trénovány na datech s již označenými shluky, to by bylo učení s učitelem. Algoritmus musí sám nějakým způsobem rozlišit shluky. Hlavním prostředkem je samozřejmě počítání vzdálenosti mezi prvky neboli metrika. Kromě normální euklidovské metriky existuje ještě mnoho dalších [24] nebo lze použít matici vzdáleností (pozor na kvadratickou paměťovou složitost).

Porovnání nejpoužívanějších algoritmů je na obrázku 1.2. Jsou zde vidět jednotlivé algoritmy (sloupce) na čtyřech různých jednoduchých datových sadách (řádky). V pravém dolním rohu každého čtverce je uveden čas výpočtu algoritmu. Jednotlivé shluky, tak jak je vyhodnotil algoritmus, jsou znázorněny barevně. Algoritmy používající koncept středů shluků, mají tyto středy zobrazeny jako černé kroužky. Obrázek je do určité míry pouze ilustrativní. Některé výsledky by šlo zlepšit změnou parametrů.

Každý algoritmus požaduje nastavení nějakých parametrů, to většinou tvoří i jeho nevýhodu. Například K-means (a jeho varianty, na obrázku je to Mini Batch K-Means) potřebuje předem znát počet hledaných shluků. Jeho výsledek také může být ovlivněn počáteční náhodnou inicializací. Většina algoritmů upřednostňuje shluky určitých tvarů. Pro K-Means jsou to tvary konvexního mnohoúhelníku (v rovině), EM algoritmus (Expectation-Maximization) [26] zase normální rozdělení (z pohledu jedné dimenze).




Obrázek 1.2: Porovnání algoritmů pro shlukovou analýzu, podrobný popis je v druhém odstavci sekce 1.5. Zdroj: dokumentace scikit-learn[25]

### 1.5.1 Hierarchické shlukování

Hierarchické shlukování [27] používá hierarchie shluků, tedy že každý shluk se skládá ze dvou či více shluků. Existují dva možná přístupy při hledání shluků. Je možné nejdřív všechny body označit jako jeden shluk a ten poté dělit na menší až na úroveň jednotlivých bodů. Opačným způsobem lze nejdřív všechny body označit jako samostatný shluk a postupně spojovat jednotlivé shluky od nejmenších vzdáleností mezi nimi. Podle způsobu určení vzdálenosti se rozlišují jednotlivé algoritmy. Pro určení vzdálenosti dvou shluků lze použít buď vzdálenost nejbližších bodů nebo nejvzdálenějších bodů nebo vzdálenost mezi středy/těžišti shluků apod.

Hierarchické shlukování umožňuje odhalit i odlehlé body. Mohou to být například ty, které se do výsledného velkého shluku přidaly jako jedny z posledních (při postupu „zespodu nahoru“).

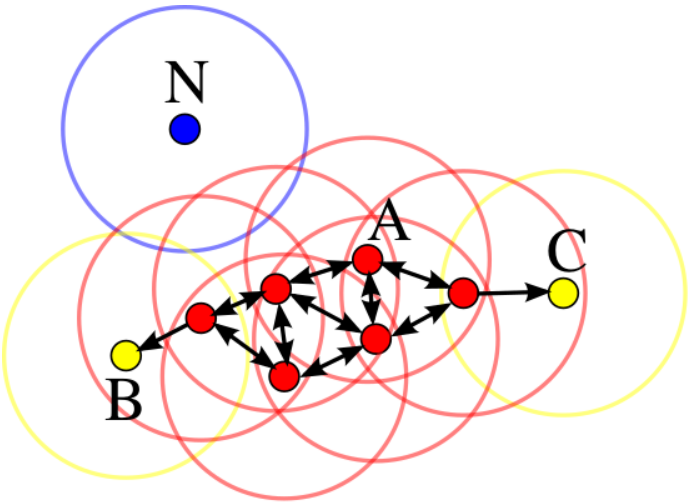
Velkou nevýhodou je bohužel časová složitost $O(n^2)$, která neumožňuje větší škálování.

### 1.5.2 DBSCAN

Algoritmus pro shlukovou analýzu DBSCAN (Density-based spatial clustering of applications with noise) [28] má dva parametry:

- *minPts* – minimální počet bodů tvořících shluk
- $\epsilon$ – maximální vzdálenost dvou sousedních bodů

Shluk je tvořen vnitřními body shluku (core), okrajovými body shluku (edge) a body mimo jakýkoliv shluk (odlehlé body – outliers). Algoritmus




Obrázek 1.3: DBSCAN ilustrační příklad: *minPts*=3, $\epsilon$-okolí znázorněno kružnicemi, červené body jsou vnitřní, žluté vnější a modrý odlehlý. Vysvětlení viz sekce 1.5.2. Zdroj: [29]

postupně prochází jednotlivé body a pokud je v jejich $\epsilon$-okolí alespoň *minPts* bodů, je tento bod označen jako vnitřní bod. Pokud je v jeho $\epsilon$-okolí alespoň jeden bod, je označen jako okrajový bod. Všechny ostatní body jsou označeny jako odlehlé. Jednoduchá vysvětlující ukázka viz obrázek 1.3.

Tento algoritmus je výhodný v tom, že není dopředu nutné znát počet shluků, a tvar shluků může být libovolný. Pro odhalování anomálií se hodí tím, že rovnou jako vedlejší produkt označí odlehlé body, případně lze použít i okrajové body. Algoritmus je také poměrně robustní, vnitřní i odlehlé body jsou označeny deterministicky, pouze u vnějších bodů záleží na pořadí zpracování bodů (mohou být na okraji dvou shluků). Podobně mohou být značky označující jednotlivé shluky v jiném pořadí. DBSCAN nefunguje moc dobře na datech, kde mají shluky příliš odlišné hustoty (tedy řekněme průměrné vzdálenosti od nejbližších sousedů). Parametr $\epsilon$ se totiž při běhu algoritmu nemění a nemůže tedy reagovat na různě husté shluky. Toto částečně řeší kombinace algoritmů DBSCAN a LOF (local outlier factor) navržených v článku [30]. Podrobněji o odlehlých bodech a LOF v sekci 1.6.

DBSCAN algoritmus potřebuje pro každý bod zjistit počet sousedů v jeho $\epsilon$-okolí. Nejjednodušším způsobem vypočtu je zjistit vzdálenost ke všem ostatním bodům, což by vedlo na kvadratickou složitost algoritmu. Výrazně lepší možností je použití vyhledávacích stromů k-d tree, Ball tree či R* tree [31, 32, 33], které vylepší průměrnou časovou složitost DBSCANu na $O(nlog(n))$.

## 1.6 Odlehlé body (outliers)

Odlehlé body bývají při statistickém zpracování zanedbávány, protože výrazně ovlivňují například hodnotu průměru apod. Při hledání anomálií jsou odlehlé body naopak velmi podstatné. Nicméně i odlehlých bodů může být mnoho a je potřeba rozlišovat nakolik jsou odlehlé. Tato *míra odlehlosti* lze měřit různými způsoby. Poměrně sofistikovaný způsob používá LOF (local outlier factor) [34]. Hlavní myšlenka LOFu je, že porovnává hustotu svého okolí s hustotou okolí jeho nejbližších sousedních bodů. Pokud jsou tyto hustoty výrazně odlišné, jde o odlehlý bod. Naopak pokud jsou stejné, jde pravděpodobně o vnitřní bod nějakého shluku.

Tato metoda je použitá v několika odborných pracích zabývajících se právě detekcí síťových anomálií. Ve studii, ve které jsou porovnávány různé metody detekce a anomálií na síťových datech (prakticky síťové toky), je doporučen právě LOF kvůli nejlepším výsledkům [35]. V projektu MINDS (Minnesota Intrusion Detection System) [36] se v modulu pro detekci anomálií také s úspěchem použila metoda LOF.

## 1.7 Atributy

Zřejmě nejdůležitější krok při shlukové analýze je výběr či vytvoření kvalitních atributů (anglicky features). Ze zadání práce vyplývá použití síťových toků jako zdrojových dat. Každý síťový tok nese informace o času počátku a délce trvání spojení, zdroji a cíli (čili IP adresy, čísla portů u TCP a UDP protokolů), počtu paketů a přenesených bajtů a také množinu všech vyskytnuvších se příznaků u TCP protokolu.

Zkoušel jsem prozkoumat, jaké atributy se používají pro detekci anomálií, v necelé desítce vědeckých prací. Většina bohužel uvádí pouze, že atributy tvoří z paketových nebo síťových dat, případně popisují jenom jejich další selekci na ty nejpodstatnější. To je popsáno ve srovnávací studii *Network Anomaly Detection: Methods, Systems and Tools* [37]. Výrazně sdílnější jsou práce z projektu MINDS [36] a studie srovnávající detekce anomálií pro síťové útoky [35]. V obou případech jsou použity IP pakety, většinou ty informace, které jsou obsaženy i v síťových tocích.

Atributy jsou vypočítávány pro jednotlivé IP adresy (nikoliv pro toky), jako podezřelé/anomální chování je tedy případně označeno nějaké koncové zařízení. Atributy jsou rozděleny do dvou skupin podle způsobu výpočtu buď v časovém okénku (tedy za posledních $t$ vteřin) nebo za posledních $n$ spojení neboli toků. Toto dělení umožňuje sledovat jak aktuální (nejnovější) provoz tak i provoz rozložený v čase, který může obsahovat různé pomalé útoky (skenování portů či zkoušení hesel). Velká část atributů lze napočítávat oběma těmito způsoby.

Shodně se v odkazovaných dokumentech vyskytují atributy typu:

- počet toků/paketů/bajtů z dané IP adresy,
- počet toků/paketů/bajtů do dané IP adresy,
- počet toků/paketů/bajtů z různých zdrojových IP adres komunikujících s danou IP adresou,
- počet toků/paketů/bajtů do různých cílových IP adres komunikujících s danou IP adresou,
- počet toků/paketů/bajtů z různých zdrojových portů komunikujících s danou IP adresou,
- počet toků/paketů/bajtů do různých cílových portů komunikujících s danou IP adresou,
- počet různých zdrojových IP adres komunikujících s danou IP adresou,
- počet různých cílových IP adres komunikujících s danou IP adresou,

Snažil jsem se dále přijít s dalšími možnými sledovaný atributy. Šlo by například napočítávat různé jednoduché statistiky jako průměry či standardní odchylky. Napadlo mě také sledovat se kterými zeměmi IP adresy komunikují, případně jejich počty apod. Zajímavý by také mohl být poměr počtu přijatých a odeslaných paketů či bajtů. Použité atributy uvádím v sekci 2.2.

Před použitím atributů je důležité provést normalizaci. Různé atributy totiž mají různé rozsahy hodnot (například jednotky oproti tisícovkám) a při přímém použití ve shlukové analýze (při výpočtu vzdáleností mezi body) by některé atributy mohly mít výrazně vyšší váhu než ostatní, které by se tím staly prakticky nepodstatné. Všechny atributy lze například shodně normalizovat (například lineárně na škálu od 0 do 1) a poté případně vědomě za nějakým účel zvýšit váhu u vybraných atributů.

Kapitola 2

# Návrh

Ve fázi návrhu jsem se zabýval výběrem nástrojů pro implementaci pluginu do NfSenu a samotným návrhem výpočtu atributů a jejich dalším konkrétním zpracováním. Především tedy rozpoznáním shluků, odlehlých bodů a jejich ohodnocení podle stupně odlišnosti.

## 2.1 Výběr nástrojů a knihoven

Ze zadání vyplývá použití opensource sondy NfSen (podrobný popis viz 3.1.4), která je rozdělen na backendovou část psanou v jazyce perl a na frontendovou část napsanou v PHP. Stejně tak jsou rozdělené zásuvné moduly (dále pluginy). Cílem této práce je právě vytvořit takový plugin. Jazyk perl není vhodný na datovou analýzu (postrádá pokročilé knihovny pro data mining a vizualizaci) a je také podle několika statistik (například [38] na základě četnosti vyhledávání jazyků na google.com) poměrně málo používaný a trend jeho popularity klesá. Rozhodl jsem se tedy vybrat vhodnější jazyk a vytvořený program spouštět z pluginu, který tedy bude poměrně jednoduchý. Snahou také bylo, aby vytvořený program byl pokud možno použitelný i samostatně nebo s co nejmenší závislostí na NfSenu.

Hledal jsem tedy nejpoužívanější nekomerční programovací jazyky a jejich knihovny pro data mining. Z několika srovnání (např. [39] jedno z podrobnějších) vyšly jako nejvhodnější jazyky R a python [40, 41]. Z různých porovnání (například [42, 43, 44]) nevyplývá jasný závěr, každý z jazyků má svoje výhody a nevýhody a přitom oba jsou nejspíše perspektivní [38]. Rozhodl jsem se použít python, protože ho umím lépe než jazyk R a protože je to obecný jazyk. V případě potřeby lze z pythonu používat funkcionalitu R pomocí balíčku rpy2 [45].

### 2.1.1 Python balíčky

Pro rychlý a efektivní výpočet atributů, shluků a odlehlých bodů jsou nutné specializované balíčky neboli knihovny. V následujících sekcích krátce popíšu hlavní funkcionalitu vybraných pythonovských balíčků, které jsou veskrze široce doporučované. Ačkoliv jsou jednotlivé balíčky vyvíjeny odděleně, velmi dobře navzájem spolupracují a téměř tedy není nutný žádný kód, který by je dával dohromady.

#### 2.1.1.1 Balíčky NumPy a SciPy

Balíčky NumPy a SciPy [46, 47] (obojí pod BSD licencí) implementují základní funkce pro vědecké výpočty – především s vícerozměrnými maticemi, výpočty lineární algebry, různé numerické výpočty apod. Numpy je postaveno na na knihovně LAPACK (Linear Algebra PACkage) [48], která je postavena na knihovně BLAS (Basic Linear Algebra Subprograms) [49] či na jiné implementaci stejného rozhraní, stejně jako například programy Mathematica, MATLAB a R [50, 51, 40].

Narazil jsem také na návod instalace a porovnání výkonu různých implementací rozhraní BLAS na testech v pythonu a R [52]. Pro práci jsem použil však standardní implementaci.

#### 2.1.1.2 Balíček pandas

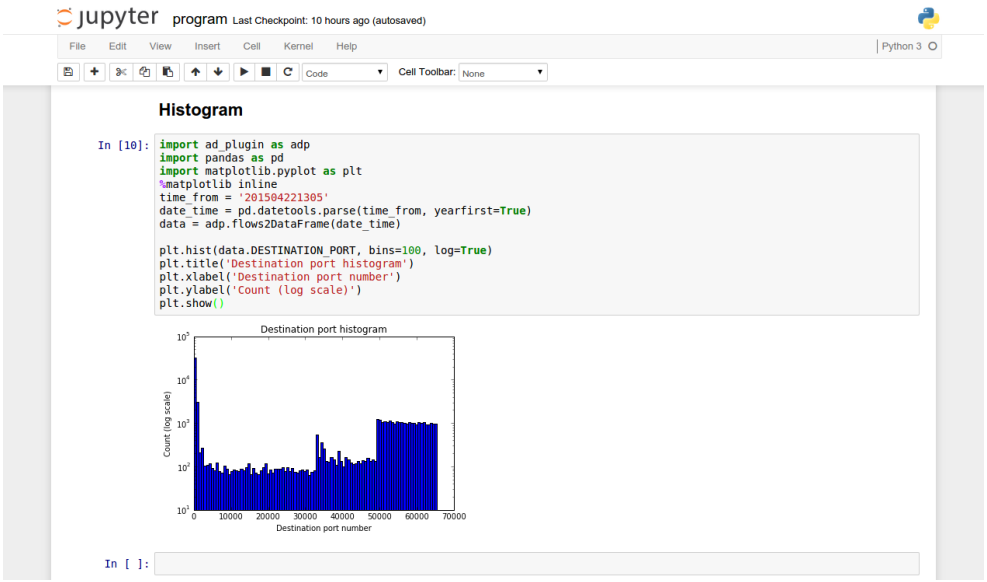
Balíček pandas [53] (BSD licence) umožňuje efektivní práci s velkými tabulkami dat podobně jako databázové stroje, ale v rámci python prostředí. Pandas je postaven na balíčku NumPy (viz předchozí sekce). Kritické části jsou psané v jazyku Cython nebo v C. V práci použiji pandas ke zpracování síťových toků a hlavně výpočet atributů.

#### 2.1.1.3 Balíček scikit-learn

Balíček scikit-learn [54] (BSD licence) je nástrojem pro data-mining. Je postavený na balíčcích NumPy, SciPy a matplotlib (viz okolní sekce). Balíček implementuje mimo jiné algoritmus DBSCAN (viz sekce 1.5.2) a struktury pro vyhledávání nejbližších sousedů v $n$-rozměrných prostorech k-d tree a Ball tree [31, 32], které použiji při implementaci.

#### 2.1.1.4 Balíček matplotlib

Matplotlib [55] (BSD kompatibilní licence) je balíček pro vykreslování 2D grafů mnoha různých typů. Umožňuje jak interaktivní prohlížení, tak i ukládání do množství vektorových i bitmapových formátů. Je přímo využíván balíčky pandas a scikit-learn.




Obrázek 2.1: Příklad práce s IPython notebookem.

### 2.1.2 Prostředí IPython notebooků

Prostředí IPython [56] notebooků spojuje výhody práce v interaktivním stylu příkazové řádky a spouštěním skriptů či programů (podobně jako notebooky v programu Mathematica [50]). Kromě jazyka python je IPython otevřen pro libovolné další jazyky, proto bude v příští verzi oddělena jazykově nezávislá část a nový projekt pojmenován Jupyter [57].

Prostředí notebooků je vhodné právě pro analýzu a vizualizaci dat. Experimentování s daty je rychlé v porovnání se spouštěním skriptů/programů, protože data stačí jednou načíst a pak s nimi lze interaktivně pracovat. Další výhodou je praktické zobrazování tabulek z balíčku pandas či grafů z matplotlibu ve výstupu, který může být zmenšen nebo skryt pro lepší přehlednost (samotný zdrojový kód bohužel sbalit nelze). Prostředí notebooků není tak propracované jako moderní vývojová prostředí, ale nabízí základní nástroje jako zvýraznění syntaxe, zobrazení nápověd k funkcím (i uživatelským) a objektům, doplňování názvů metod, zobrazení zpráv právě uložených do logu atd. Příklad práce s IPython notebookem je vidět na obrázku 2.1.

### 2.1.3 Zeměpisná lokalizace IP adres

Pro výpočet některých atributů jsem chtěl použít zeměpisnou lokalizaci IP adres, tedy zjištění státu nebo oblasti koncového zařízení, které se zúčastňuje síťové komunikace. Hledal jsem tedy řešení, které by bylo bezplatné a pokud možno off-line. Dotazovat se na každou IP po síti by bylo nejspíše velmi pomalé. Narazil jsem na pěkné srovnání různých poskytovatelů zeměpisných

informací o IP adresách [58], ze kterého jsem vybral produkt GeoLite2 [59] firmy MaxMind [60], který splňuje předchozí požadavky a dokonce dodává i python balíček pro čtení databáze. Na výběr je ze dvou možností: lokalizace na úrovni zemí a na úrovni měst. U bezplatné databáze samozřejmě nepočítám s velkou přesností lokalizace, i tak ale mohou být tyto údaje zajímavé.

Při průzkumu jsem narazil na NfSenový plugin SURFmap [61] (stáhnout lze ze stránek SourceForge [62]). Plugin doplňuje data síťových toků o zeměpisné informace a zobrazuje je na interaktivní mapě. Zeměpisná data dokáže získávat z offline databází firem MaxMind [60] i IP2Location [63].

## 2.2 Výběr atributů

Na základě analýzy (sekce 1.7) jsem vybral několik atributů. Některé bylo potřeba přesněji specifikovat, některé jsem převzal přímo z literatury a některé jsem vytvořil sám (ty používající lokalizaci a poměry provozus). U jednotlivých atributů uvádím jaké typy provozu by mohly být schopny rozlišit. V popisu uvedené chování typického serveru a typického klienta je samozřejmě zjednodušené, jde spíše o ilustraci principu. Právě kombinace atributů vytváří (vlastně doslova) méně plochý pohled na koncová zařízení.

Atributy se vztahují k jednotlivým IP adresám, které se vyskytnou v provozu (ať už jako zdrojová nebo cílová adresa).

- Počet unikátních cílových IP adres – rozlišuje mezi serverovým a klientským chováním dané IP adresy, servery mají velký počet cílových adres, klienti méně.
- Počet unikátních zdrojových IP adres – podobně jako předchozí bod. U normální a legitimní komunikace budou hodnoty tohoto atributu vycházet stejně jako počet cílových adres. Naopak větší odchylka mezi těmito dvěma může znamenat anomálii, například když je komunikace jednostranná.
- Počet unikátních cílových portů (u TCP či UDP) – podle počtu portů rozliší například server (např. webový), který bude mít mnoho různých cílových portů, naopak klienti mají méně cílových portů.
- Počet unikátních zdrojových portů (u TCP či UDP) – podobně jako předchozí atribut, rozdíl mezi nimi může zafungovat podobně jako u počtu unikátních zdrojových/cílových adres.
- Počet unikátních cílových zemí – rozliší koncová zařízení podle charakteru provozu na spíše lokální či mezinárodní (s různou mírou). Při velké změně může jít buď o poruchu na síti nebo například o DDoS útok.
- Počet unikátních zdrojových zemí – podobně jako předchozí atribut, rozdíl opět může značit nějakou anomálii

- Průměr poměrů mezi přijatými a odeslanými bajty v obousměrných tocích – rozlišuje mezi koncovými zařízeními, které spíše přijímají data a která spíše odesílají.
- směrodatná odchylka poměrů mezi přijatými a odeslanými bajty v obousměrných tocích – doplňuje předchozí atribut o to, nakolik je poměr typický/častý. Může rozlišit například DNS servery, kde je většina provozu velmi podobná a odchylka tedy bude malá.
- Průměr poměrů mezi přijatými a odeslanými počty paketů v obousměrných tocích – podobně jako v případě s bajty
- směrodatná odchylka poměrů mezi přijatými a odeslanými počty paketů v obousměrných tocích – podobně jako v případě s bajty

Atributy se napočítávají vždy pro nějakou sadu toků. Výběr té sady je samozřejmě velmi podstatný. Nejběžnější je napočítávání vždy na nejnovějších tocích a porovnání se staršími hodnotami – tedy dělit na úseky podle času. Další možností je vytvořit sady podle posledních $n$ toků každé jednotlivé IP adresy – tedy dělit podle počtu spojení. Tento způsob může odhalit pomalé útoky, tedy ty které jsou rozložené v čase, aby byly více nenápadné. V této práci jsem se primárně věnoval časovému dělení.

Konstrukce atributů by v ideálním případě neměla být příliš citlivá na zvýšení samotného objemu provozu. Cílem není odhalit špičky v počtu paketů nebo bajtů paketů, na to jsou jednodušší způsoby detekce. Proto nejsou použity přímo například počty bajtů, ale spíše různé poměry, které by se za normálního provozu neměly příliš měnit.

## 2.3 Výpočet a zpracování atributů

Chtěl jsem porovnávat provoz posledních pěti minut (to vyplývá z pevného nastavení NfSenu) s provozem za nějaký delší časový úsek (neboli za krátkou nejnovější historii) – řekněme například jednu hodinu (pro jednoduchost dál uvádím tuto hodnotu, která ale není fixně určená). Pokud bych napočítal atributy provozu za tyto dva časové úseky, byly by hodnoty nesouměřitelné. Například počet různých cílových IP adres je v jedné hodině provozu pravděpodobně vyšší (u serverů určitě výrazně vyšší) než za posledních 5 minut. Tento problém jsem se rozhodl vyřešit tak, že pro atributy nepočítám pro celou hodinu, ale právě po „pětiminutovkách“, aby byly hodnoty srovnatelné.

Tím ale vznikly další dva problémy. Jak teď 12 „pětiminutovek“ srovnám s jednou aktuální? Nejjednodušší je udělat průměr hodnot, tím se nicméně ztrácí značná část informací. Tak jsem k průměru přidal ještě směrodatnou odchylku, abych zachoval alespoň část informace. Tím jsem si samozřejmě vytvořil další problém – jak porovnat použít odchylky v porovnání s atributy z nejnovějšího provozu. Nejjednodušeji se samozřejmě srovnávají odchylky

s odchylkami, stačí tedy rozdělit nejnovější toky na několik částí a z nich pak spočítat směrodatné odchylky i průměry. Ještě je potřeba poslední úprava a to totiž rozdělit na stejné části i těch 12 „pětiminutovek“.

Výsledný návrh tedy zní takto. Provoz za posledních 5 minut rozdělím na několik částí (například po minutě), pro každou z nich napočítám atributy a z nich ještě vytvořím průměry a směrodatné odchylky. Podobně rozdělím na stejně velké časové úseky (minuty) i provoz za poslední hodinu, pro části napočítám atributy a z nich dále také průměry a odchylky. Výsledné dvě tabulky IP adres s průměry a odchylkami jejich atributů by takto měly být již poměrně dobře souměřitelné.

## 2.4 Normalizace atributů

Většina algoritmů shlukové analýzy či algoritmů pro hledání odlehlých bodů používají nějaký koncept vzdálenosti mezi body. Různé způsoby počítání vzdáleností v libovolně dimenzionálním prostoru se nazývají metriky. Jde například o euklidovu či manhattanskou metriku (tyto a mnoho dalších v [24]). Jelikož mají různé atributy různé rozsahy hodnot, mají pak různý vliv na výpočet vzdálenosti mezi dvěma body ať je metrika jakákoliv. Pokud jsou například hodnoty jednoho atributu ve desítkách a druhého v tisících, na výslednou vzdálenost mezi body bude mít druhý atribut mnohem větší vliv. Aby se předešlo právě takto neúměrným vlivům, používá se normalizace hodnot atributů.

Existuje více způsobů normalizace. Jedním z nich je lineární normalizace na interval $< 0; 1 >$, hodnoty každého jednotlivého atributu se přeškálují do tohoto intervalu. Další možností je normalizace na tzv. z-skóre, tedy takové lineární přeškálování, aby měl normalizovaný atribut průměr rovný 0 a směrodatnou odchylku rovnou 1. Předpokladem je normální rozdělení hodnot atributu.

Normalizace tedy ovlivňuje vzdálenosti mezi body. Toho lze i využít a úmyslně zvýhodnit některé atributy libovolnou měrou.

Vybral jsem lineární normalizaci na interval. Kvalifikovanější výběr lze vytvořit až na základě dat a jeho statistického rozdělení.

## 2.5 Detekce anomálií

Po přípravě atributů a normalizaci jejich hodnot přichází na řadu samotná detekce anomálií. V analýze jsem si prostudoval nejpoužívanější algoritmy pro shlukovou analýzu a pro analýzu síťových anomálií jsem vybral algoritmus DBSCAN (popis viz 1.5.2). Jeho hlavními výhodami jsou dobrá škálovatelnost (časová složitost $O(nlog(n))$), libovolný počet shluků (není nutné zadávat jako parametr) a zabudovaná detekce odlehlých bodů, které je ovšem bez vyhodnocení odlehlosti. Nevýhodami jsou nutnost zadání parametrů *MinPts*

a $\epsilon$. Určení jejich hodnoty při samotném návrhu je velmi obtížné. Hodnoty jsem tedy určil až na základě experimentů.

Algoritmus DBSCANu tedy spustím na normalizované atributy z poslední hodiny spolu s atributy z poslední „pětiminutovky“. Výsledkem je seznam označení shluků a také speciální označení pro body bez přiřazeného shluku. Nejjednodušší možností detekce anomálie je vzít body, které nejsou součástí žádného shluku (ať už jsou z posledních pěti minut nebo starší), a spočítat pro ně míru odlehlosti pomocí LOF (viz 1.6). Ty nejodlehlejší podle LOFu, které přesahují nějakou určitou mez (ang. threshold), lze označit za anomální. Dále je možné zjistit, které IP adresy z poslední „pětiminutovky“ jsou v jiném shluku než stejná adresa ale z poslední hodiny. To by značilo změnu chování dané IP adresy. Co když jsou však tyto dva shluky blízko vedle sebe? Je tedy potřeba vypočítat nějakou míru změny – nejjednodušeji prostě jejich vzdálenost. Nebo může být nějaký shluk poměrně velký a výraznou změnu polohy IP adresy tedy nijak nezaznamenáme. Pak lze vypočítat vzdálenost pohybu pro všechny IP adresy, které jsou v obou skupinách (historické i nové). To by však mohlo být poměrně časově náročné. V prvotní implementaci se tedy zaměřím na odlehlé body a na body, které změnily shluk.

Kapitola 3

# Implementace

## 3.1 Použité nástroje

### 3.1.1 Nástroje nfdump

Balíček nástrojů nfdump [64] obsahuje programy pro příkazovou řádku, které přijímají, ukládají a zobrazují síťové toky. Je poskytován pod velmi svobodnou BSD licencí (Berkeley Software Distribution). Následuje seznam hlavních programů a jejich stručný popis:

- **nfcapd** (*NetFlow capture deamon*) přijímá síťové toky na určeném portu (výchozí 9995) a ukládá je předepsaným způsobem (například do složek ve formátu `zdroj/rok/měsíc/den`). Přijaté toky ukládá do souborů (s možností komprimace) pro daný časový interval (výchozí a typická hodnota je 5 minut). Podporuje NetFlow verze 1, 5, 7 a 9 a IPFIX (rozšíření pouze experimentálně). Při větším vytížení je doporučeno spustit pro každý exportér zvlášť jeden nfcapd proces, tímto způsobem pracuje NfSen 3.1.4.

- **nfdump** čte soubory vytvořené programem nfcapd a zobrazuje jednotlivé toky v tabulce podle nastavení (výběr sloupců a podrobností) a také podle filtrů (například vybraná zdrojová adresa a protokol) podobných programu tcpdump [65]. Program nfdump také zobrazuje statistiky typu: prvních $n$ toků podle počtu bajtů apod. Další podrobnosti, příklad výstupu a zkušeností v následující sekci 3.1.1.1.

- **nfprofile** profiltruje data ze souborů vytvořených programem nfcapd a ukládá je do souborů pro další zpracování například nfdumpem. Filtrovací řetězec může být zadán na příkazové řádce nebo načten ze souboru.

- Dále **nfreplay** pro přeposlání zaznamenaných dat ve formě NetFlow do kolektoru nebo skript **nfclean.pl** na promazávání starých souborů s toky, který je pravidelně volán NfSenem.

#### 3.1.1.1 Program nfdump

Program nfdump je hlavní část balíčku nástrojů nfdump. Zobrazuje síťové toky, které od exportéru přijal, zpracoval a uložil program nfcapd. Umožňuje otevírat jeden soubor či více souborů najednou (typicky časový interval) i napříč složkami (přepínače -r, -R a -M), které reprezentují různé exportéry či profily (viz 3.1.4.2). Dále podporuje agregaci toků podle seznamu zvolených atributů i agregaci do obousměrných toků (přepínače -a a -b či -B). Program nfdump také umožňuje použít kromě několika připravených výstupních formátů i vlastní pomocí formátovacího řetězce (přepínač -o). Dále lze nastavit pořadí výpisu toků (přepínač -O) a mnoho dalších podrobností.

S nfdumpem jsem se během práce dostal do několika problémů. Formátovacím řetězcem jsem si vybral sloupce, které mě zajímají, vypnul jsem výpis statistik na konci a výstup jsem přesměroval do souboru typu csv (comma separated values – čárkou oddělené hodnoty), který jsem dál zpracovával pomocí pythonu. Tento výstup nfdumpu bohužel není vhodný pro další jednoduché strojové zpracování hned z několika důvodů. Nfdump zkracuje různé položky, aby se daly hezky zobrazit a číst uživatelem. Například zkracuje IPv6 adresy uprostřed pomocí dvojice teček nebo objem přenesených dat vypisuje většinou bez jednotek (v bajtech) nebo v MB s výpisem jednotky M atp. Snažil jsem se toto chování pozměnit pomocí přepínačů, ale to se mi podařilo pouze částečně u IPv6 adres (přepínač -6, který je v dokumentaci bez popisu u příkladu). Řešení je tedy pomocí formátu csv (přepínač -o csv), který ale vypíše veškeré atributy (sloupce). Výstup z nfdumpu tedy přesměrovávám rovnou jako vstup do programu cut, pomocí kterého vyfiltruji nepotřebné sloupce. I takto bohužel nastává jedna výjimka. Pokud se v daném souboru nenachází žádný síťový tok, nfdump vypíše hlášku „No matched flows“. Přitom by stačilo nevypisovat vůbec nic.

Po přechodu na vyšší verzi softwaru v sondě FlowMon (viz 3.1.3) docházelo k chybám, které generoval nfdump. Při pokusu o výpis rozšíření v souboru s toky (přepínač -x) byl nfdump dokonce ukončen pro neoprávněný přístup do paměti. Pomocí jednoduchého skriptu jsem zjistil, že se toto stane u všech souborů s toky, které jsem měl k dispozici. S vedoucím práce jsme tedy kontaktovali pana Pavla Minaříka z firmy INVEA-TECH. Dozvěděli jsme se, že síťové toky byly sondou rozšířeny dalšími informacemi, které ale nejsou standardní. INVEA-TECH tedy používá vlastní verzi nfdumpu, který dokáže číst tyto rozšíření. Jelikož nfdump po chybových hláškách dále vypíše síťové toky, rozhodl jsem se pouze filtrovat chybové hlášky pomocí programu grep a cut, aby výsledný program nebyl vázán na sondy firmy INVEA-TECH.

U protokolů jiných než TCP a UDP vypisuje nfdump v polích pro zdrojový a cílový port hodnotu nula. Výjimkou je ovšem protokol ICMP, nfdump zde do cílového portu kóduje typ ICMP zprávy (vyšších 8 bitů) i její kód (nižších 8 bitů).

### 3.1.2 Softwarová sonda fprobe

Program fprobe [66] je softwarová sonda (neboli exportér) postavená na knihovně libpcap [65]. Umožňuje sledovat provoz na určených síťových rozhraních a vytvářet z něj NetFlow síťové toky (verzí 1, 5 nebo 7) a posílat je na kolektor či více kolektorů. Provoz je možné filtrovat (použitá syntaxe tcpdumpu). Samozřejmostí je možnost nastavení aktivního a pasivního timeoutu a mnoha dalších technických parametrů (velikost bufferu, různé limity atd.). Sondu fprobe jsem použil ve spojení s NfSenem pro sledování provozu na svém počítači.

### 3.1.3 Sonda FlowMon

Velká část dat, které jsem analyzoval, pochází z hardwarové sondy FlowMon od firmy INVEA-TECH [67]. Sonda sleduje část provozu fakulty na 1 Gbps lince a slouží zároveň i jako kolektor s vlastním softwarovým balíkem pro vizualizaci a analýzu dat.

### 3.1.4 Program NfSen

Program NfSen [5] je webová aplikace pro síťové toky, která je postavena na balíčku programu nfdump 3.1.1.1 a distribuována pod BSD licencí. Zobrazuje síťová data a vytváří grafy a statistiky síťového provozu, které lze snadno nastavit. Umožňuje vytvářet profily pro různé provozy (např. konkrétní protokol nebo IP adresa), pluginy pro rozšíření funkcionality a alerty, které mohou spustit následné akce (např. informovat správce sítě). Podrobněji v následujících sekcích.

NfSen je složen z frontendu a backendu. Frontend je webové rozhraní napsané v jazyce PHP, backend slouží ke zpracování toků a je v jazyku perl. Tyto části spolu komunikují pomocí socketů, protože je plánováno oddělení těchto částí, aby nemusely být na jednom počítači či serveru, ale mohly být odděleně.

#### 3.1.4.1 Instalace

Instalace NfSenu má mnoho kroků a oficiální dokumentace je bohužel poměrně strohá. Instaloval jsem verzi 1.3.6p1 z ledna 2012 (poslední stabilní verze, existuje i nová verze z prosince 2014). Využil jsem několika návodů na webu [68, 69] a jejich kombinací jsem nakonec dosáhl úspěchu. Základem je instalace nfdumpu s přepínačem `--enable-nfprofile`. Dále jsou potřeba interpretry perlu a PHP (příslušných minimálních verzí) a několika rozšiřujících modulů a také RRDtool (Round Robin Database Tool). Instaloval jsem na Ubuntu, kde jsou tyto programy dostupné přes balíčkovací systém případně přes systém pro instalaci perlovských modulu CPAN (Comprehensive Perl Archive Network) [70].

Samotnou instalaci NfSenu pak ještě předchází úprava konfiguračního souboru `nfsen/etc/nfsen.conf` a to především nastavením proměnných pro umístění NfSenu (`$BASEDIR` pro backend a `$HTMLDIR` pro frontend), uživatelů (`$USER$` pro nfcapd, `$WWWUSER` pro http server), které je potřeba předtím vytvořit a nastavit jim stejnou skupinu uživatelů (`$WWWUSERGROUP`), dále nastavit hash (asociativní pole) zdrojů (`%sources`) pro příjem síťových toků ze sond. Nastavení pluginů je popsáno v části instalace pluginu 3.3.

Kromě NfSenu je nutné také nainstalovat webový server a sondu. Jako sondu jsem použil program fprobe (viz 3.1.2) která odchytává provoz a vytváří síťové toky a posílá je NfSenu. Jako webový server jsem použil Apache HTTP verze 2.4 [71].

#### 3.1.4.2 Profily

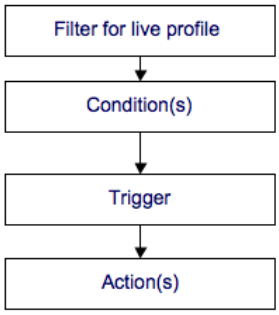
NfSen má zabudovaný jeden profil s názvem *live*, který obsahuje veškerý síťový provoz, ale dovoluje také vytváření vlastních profilů. Profil reprezentuje nějaký výsek provozu. Může to být například provoz na určitém portu (tedy nějaká konkrétní služba) nebo pouze provoz, kterého se účastní určitá IP adresa (typicky server). Profil je tvořen jedním či více kanály, kde každý kanál se skládá z filtru (ve formátu pro nfdump) a dále seznamem zdrojů síťových toků (výběr použitých sond). Každý kanál má také nastavenou barvu a pořadí vykreslování v grafech. Příkladem tedy může být profil pro sledování webových služeb, který obsahuje dva kanály pro filtrování http a https provozu.

Každý profil je „reálný“, tedy že ukládá vyfiltrované toky do souborů a nebo stínový (shadow), který při každém dotazu pouze použije dané filtry na toky z profilu live. Profily jsou určené také časovým intervalem. Buď jsou čistě historické (časový interval v minulosti) anebo od nějakého časového okamžiku do současnosti a prodlužují se s každým novým tokem.

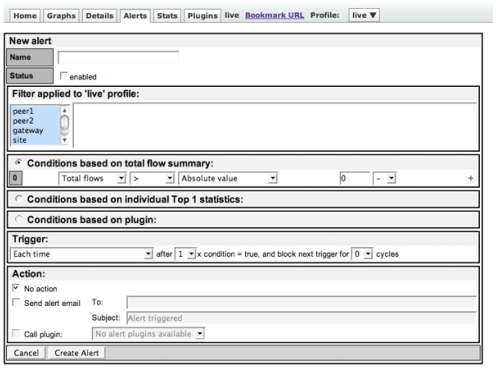
Pomocí profilů (například i v kombinaci s pluginy) lze odhalit různé kontextové anomálie popsané v sekci Anomálie 1.2.

#### 3.1.4.3 Alerty

Alert neboli poplach umožňuje provést nějakou akci na základě splnění určité podmínky. Aby na základě síťových toků za posledních 5 minut byla spuštěna akce, je třeba projít několika kroky (viz obrázek 3.1a). Nejdříve jsou toky zpracovány uživatelským filtrem, dále jsou testovány na jednu či více podmínek, které jsou spojeny logickými operátory konjunkce či disjunkce. Podmínky mohou být na základě jednoduchých statistických ukazatelů (například počet toků se zvýšil přes 20 % nad průměr za poslední hodinu) nebo pomocí výpočtu ve vyhrazené funkci pluginu (tedy libovolně složitá podmínka). Po splnění této podmínky je ještě nutné splnit podmínky spuštění akce. Ta může být spouštěna buď pokaždé nebo třeba až po několikanásobném spouštění podmínky v řadě a další spuštění povoleno až po několika dalších cyklech (tedy pětimi-




(a) Vyhodnocení Alertu. Zdroj: [5]




(b) Formulář pro nastavení nového alertu. Zdroj: [5]

nutovkách). Výsledná akce pak může být poslání e-mailu (například správci sítě) nebo zavolání funkce v určitém pluginu nebo bez akce. Vzhled formuláře je vidět na obrázku 3.1b.

#### 3.1.4.4 Pluginy

NfSen je možné rozšířit pomocí pluginů. Plugin se skládá z části na backendu (v perlu) a z nepovinné části na frontendu (v PHP). Obě části spolu komunikují pomocí příslušných modulů (v perlu respektive v PHP), které umožňují frontendu zavolat funkci na backendu a dostat od něj odpověď. Data mezi těmito částmi se posílají skrze sockety. Frontendová část má na starosti prezentaci výstupů pluginu a zpracování formulářových dat od uživatele. Backendová část zpracovává síťové toky na pětiminutové bázi, vyhodnocuje spuštění alertu a provádí akci alertu. Zpracování dat ve všech pluginech tedy logicky nesmí přesáhnout čas pěti minut, jinak by se zpoždění neustále zvětšovalo.

#### 3.1.4.5 Zkušenosti

Kombinace profilů, pluginů a alertů je poměrně praktická. Během používání NfSenu a programování pluginu jsem však narazil na poměrně dost nepříjemností. Dokumentace je až příliš stručná a ne moc kvalitní. Hodilo by se například řešení nejčastějších chyb. V názvu profilu nemůže být mezera, chybová hláška však oznámí pouze to, že řetězec obsahuje invalidní znaky. Vybraný profil se samovolně přepíná na *live* při přechodu na záložku s alerty. Uživatelské rozhraní obecně není moc přehledné, některá důležitá tlačítka jsou malá a nevýrazná, některé volby jsou podivně pojmenované (například záložka *Stats* uživatele dovede na editaci právě používaného profilu). Několikrát jsem hledal, jak něco funguje, přímo ve zdrojovém kódu. Ten je však bohužel téměř bez komentářů.

Pro každý profil by se měl plugin pustit zvlášť, to ale nefungovalo pokud byl profil v nějaké skupině profilů. Podle dokumentačních komentářu v nfsen.conf by ale symbol hvězdy měl znamenat každý profil. Po vyřazení dvou profilů ze skupiny už se plugin spouštěl alespoň pro jeden z nich a druhý byl přeskočen z neznámého důvodu. Ze zpráv v syslogu jsem akorát zjistil, že ani testovací plugin nebyl s tímto profilem spuštěn.

## 3.2 Plugin do NfSenu ad_plugin

Většinu času vývoje jsem strávil v IPython notebooku (popis v sekci 2.1.2), ze kterého jsem ke konci přesunul veškerou funkcionalitu do obyčejného pythonového zdrojového souboru, který je samostatně spustitelný a jeho funkce lze importovat do IPython notebooku nebo jiného programu v pythonu. Plugin do NfSenu tedy prakticky jenom spouští a prezentuje výsledky programu. Následuje popis vývoje jednotlivých částí spolu s nastalými komplikacemi a jejich řešeními.

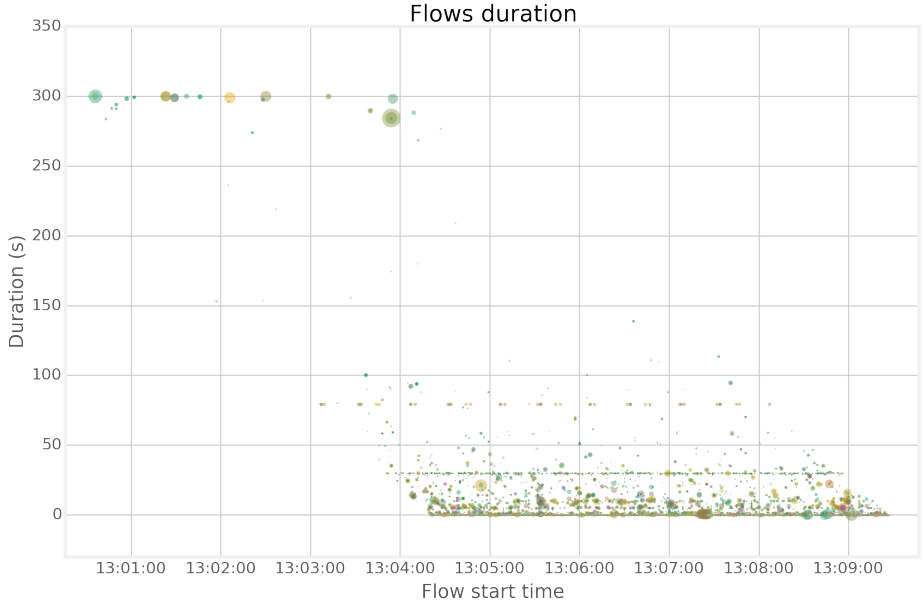
### 3.2.1 Použitá data

Pro účely práce jsem měl k dispozici několikadenní provoz z fakultní sondy FlowMon (více v 3.1.3). Ve špičce provozu je v jedné hodině přibližně 900 000 síťových toků s přibližně 13 000 unikátních IP adres. Toky jsem odchytával i na vlastním počítači pomocí sondy fprobe (viz 3.1.2). Většinu výpočtů jsem tedy prováděl na školních datech, abych zjistil, jestli je výpočet dostatečně efektivní.

### 3.2.2 Zpracování dat z nfdumpu

IPython notebook umožňuje přímé použití shellovských příkazů, takže první verze načítání dat probíhala přesměrováním výstupu nfdumpu do souboru, který jsem pak načetl rovnou jako tabulku (konkrétně objekt třídy DataFrame) pomocí funkce z balíčku pandas. Z důvodů uvedených v sekci 3.1.1.1 jsem se musel několikrát vracet a opravovat načítání dat nfdumpem. Výslednou verzi jsem poté přenesl do čistě pythonovské implementace, která tedy spouští příkazy nfdump, grep i cut s propojenými vstupy a výstupy.

Od začátku jsem procházel načtená data a zjistit z nich nějaké zajímavé informace. Vykresloval jsem si tedy různé grafy, histogramy apod. Na grafu závislosti délky toku na času 3.2 jsem si všiml vodorovných „pásů“ toků. Projevily se i na histogramu. Vyfiltroval jsem si tedy toky s délkou z grafu a zjistil jsem, že jde v jednom případě o IMAPS (Internet Message Access Protocol over TLS/SSL) protokol na portu 993 na adrese, kde běží školní e-mailový server. Nevím proč spojení trvá pokaždé přibližně 30 vteřin, odhaduji že by mohlo jít o nějaký timeout. V druhém případě jde o HTTPS provoz mezi dvěma stále stejnými adresami z rozsahu adres ČVUT. V jednom směru trvá




Obrázek 3.2: Graf délky síťových toků v závislosti na čase. Toky jsou z jednoho nfcapd souboru, který by měl obsahovat toky jež dorazily na kolektor v časovém intervalu od 13.05 do 13.10. Toky tvoří tvar kosodélníku. To je zapříčiněno způsobem sběru toků. Horní hrana (300 s) je způsobená aktivním timeoutem. Šikmé hrany jsou způsobeny odesláním toku ze sondy na kolektor až v čase ukončení toku. Posun doleva na časové ose může být zapříčiněn několika možnostmi: bufferování toků, zpoždění na cestě, pasivní timeout (oba timeouty viz 1.1.2.2). Plocha bodu je úměrná odmocnině počtu odeslaných bajtů v toku, různé barvy bodů značí různá čísla cílových portů.

spojení přibližně 80 vteřin, v opačném pod desetinu vteřiny. Počty paketů a množství bajtu jsou ve všech těchto tocích velmi podobné. Víc se mi podařilo zjistit pouze to, že po zadání IP adresy do prohlížeče jsem se dostal na stránku obsahující pouze nadpis „It works!“. Po připsání `https://` do adresního řádku jsem se dostal na stránku se stejným nadpisem, ale s nedůvěryhodným certifikátem.

### 3.2.3 Výpočet atributů

V programu jsem nakonec implementoval prvních 6 atributů popsaných v návrhu (v sekci 2.2), protože jsem chtěl nejdříve dokončit hlavní část programu a pak se případně vrátit. Ostatně přidání výpočtu dalších atributů je velmi jednoduché, stačí ve funkci `compute_features` do tabulky `ip_table` vložit další sloupec s novým atributem, žádné další změny nejsou potřeba, doporučuji pouze promazat dočasné soubory.

Prvotní implementace byla extrémně pomalá. Výpočet atributů na tocích z 5-ti minut (přibližně 70 000 toků) trval klidně 12 minut. Zjistil jsem, že problém je v geolokaci IP adresy. Uvědomil jsem si, že vyhledání země pro nějakou IP adresu volám pro každý tok dvakrát (zdrojová a cílová adresa), což je určitě zbytečné. Zvláště, když se v tocích vyskytuje pouze přibližně 1000 různých IP adres. Nejprve jsem problém řešil pomocí zvláštní tabulky pouze s IP adresami a zeměmi, kterou jsem pomocí operace `join` připojil k tabulce toků. Poté jsem přišel na elegantnější a praktičtější řešení, pomocí funkce `lru_cache` ze standardní knihovny functools. Ta se pouze připíše před deklaraci funkce jako tzv. dekorátor s nastavením maximální velikosti paměti. Tato funkce následně sleduje vstupy a výstupy z funkce a ukládá si je pro příští zavolání se stejnými parametry až do nastaveného maxima paměti, maže ty nejdéle nepoužité záznamy. Výhoda tohoto přístupu je, že i při dalším spuštění výpočtu atributů se použijí data z této paměti.

Podle návrhu jsem měl jednotlivé „pětiminutovky“ dělit na části po jedné minutě. Toky z jednoho nfcapd souboru však nevypadají tak, jak jsem si je původně představoval (viz obrázek 3.2) a proto jsem se rozhodl pro jiné dělení. Abych neměl v prvním díle většinu toků s největšími délkami toku, rozhodl jsem se pro rozčlenění do skupin podle zbytku po dělení pěti. Do první skupiny se tedy dostanou toky s indexem dělitelným pěti beze zbytku, do druhé se zbytkem jedna a tak dále. Takto by jednotlivé skupiny měli být rovnoměrné.

### 3.2.4 Dočasné soubory

Pro zrychlení výpočetního času jsem implementoval ukládání tabulek s toky a tabulek s atributy do CSV souborů. Soubory vytvářím v dočasném adresáři, který vytvořím na místě, které mi vyhradí operační systém. Mírnou nevýhodou tohoto přístupu je smazání paměti při vypnutí počítače.

Zrychlení jsem změřil pomocí jedné z tzv. magických funkcí IPythonu `%timeit`, která se napíše před libovolný kód. Funkce spustí příkaz alespoň třikrát nebo víckrát, když je kód hodně rychlý. Toto vypíše, když je použita na funkci pro výpočet atributů na jedné hodině toků (při spuštění není žádný dočasný soubor):

```
The slowest run took 182.60 times longer than the fastest.
This could mean that an intermediate result is being cached
1 loops, best of 3: 348 ms per loop
```

Při dalším spuštění pluginu s nejnovějšími pěti minutami toků stačí vypočítat atributy pro tyto nová data, zbytek už je připraven v dočasných souborech, které se načtou již velmi rychle.

V pluginu je nastaveno průběžné promazávání dočasných souborů, které jsou starší než 12 hodin. Při vypnutí NfSenu jsou smazány všechny dočasné soubory. Toto chování lze případně upravit jednoduchým zásahem do kódu.

### 3.2.5 Implementace algoritmu LOF

Původně jsem si myslel, že algoritmus hledání a hodnocení odlehlých bodů LOF [34] (viz sekce 1.6) je implementovaný v balíčku scikit-learn. Tak tomu ale není a tak jsem se pokusil nějakou implementaci vyhledat. Vyzkoušel jsem program pylof [72], který je však velmi pomalý, protože pro vyhledání nejbližších sousedních bodů prostě vypočte vzdálenosti ke všem ostatním. Narazil jsem také na implementaci [73], která používá stromové struktury k-d tree či Ball [31, 32] z balíčku scikit-learn. Nakonec jsem si napsal vlastní implementaci, která také používá uvedené stromové struktury pro vyhledání nejbližších sousedních bodů.

### 3.2.6 Složení částí dohromady

V nejdůležitější funkci programu je `find_anomalies`, která nijak nevybočuje od původního návrhu (viz 2.4 a 2.5). Jako argumenty dostane atributy, z nich vypočítá průměry a směrodatné odchylky (dosud zvlášť pro nejnovější toky a zvlášť pro historické). V dalším kroku tyto dvě tabulky spojí, znormalizuje a aplikuje na ně DBSCAN s nastavenými parametry. Pro nové i staré odlehlé body vypočtu hodnotu odlehlosti LOF. Podobně pro IP adresy, které změnily shluk vypočítám jejich vzdálenost. Výsledně tato funkce vrátí 3 pole s dvojicemi míra anomality a IP adresa.

Plugin je spouštěn pro každý profil a výsledky musí být nezávislé. Veškeré výstupní soubory pluginu tedy mají v názvu právě jméno profilu. Frontend pak podle vybraného profilu zažádá backend o správné soubory.

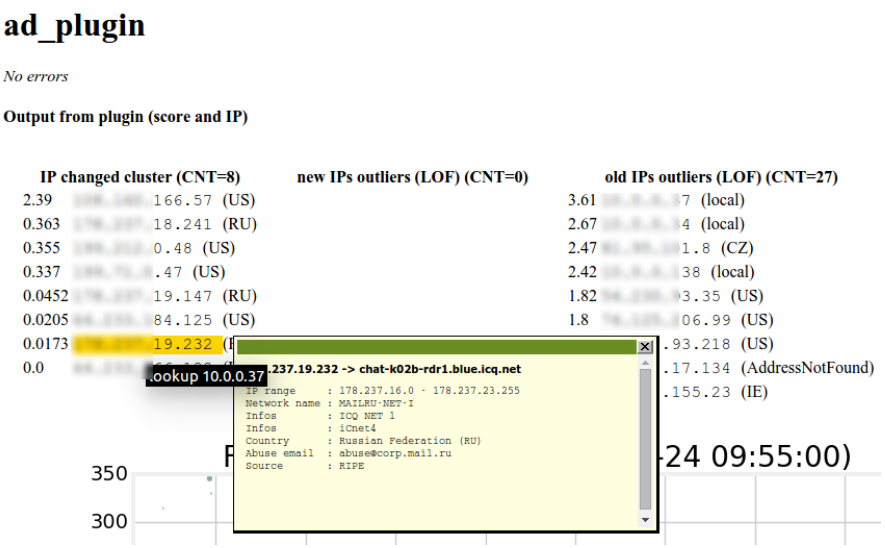
Hlavní kroky postupu jsou zaznamenávány do syslogu včetně použitých parametrů použitých v algoritmech. Různé pomocné funkce zde nebudu zbytečně rozepisovat, vše je vždy alespoň stručně okomentováno v kódu.

### 3.2.7 Podpora profilů

Vytvořený plugin plně podporuje spouštění pro více profilů. NfSen postupně zavolá plugin s nastaveným parametrem profilu. Plugin tedy pro každý profil generuje sadu souborů (dočasné soubory, výstupy, obrázek grafu) zvlášť (soubory jsou označeny názvem profilu).

### 3.2.8 Backend v perlu

Backendová část pluginu má poměrně jednoduchý úkol. Prakticky pouze spustí pythonovský program se všemi potřebnými parametry a uloží jeho výsledky (zvlášť standardní a chybový výstup) do souborů, které na vyžádání pošle serveru. Při ukončení NfSenu ještě zavolá pythonovský program pro odstranění dočasných souborů. Z ladících důvodů zapisuje všechny provedené akce do syslogu.




Obrázek 3.3: Frontend ad_pluginu (výřez).

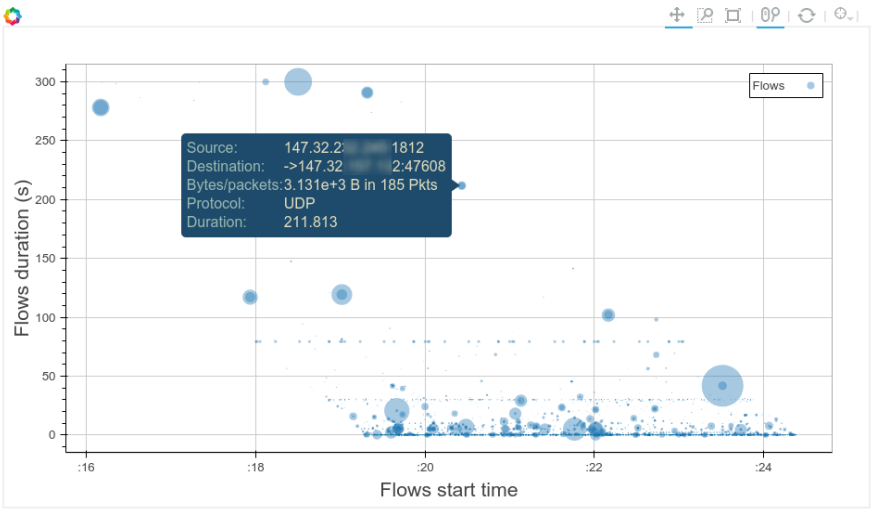
### 3.2.9 Frontend

Frontend je velmi jednoduchý. V zásadě si pouze vyžádá normální i chybový výstup, záznam logu a obrázek grafu toků za posledních pěti minut pro aktuálně zvolený profil z backendové části pluginu a obdržená data zpracuje jednoduchým PHP skriptem do HTML formátu. Vzhled stránky pluginu je vidět na obrázku 3.3. Javascriptovou funkci pro vyhledání dodatečných informací o IP adrese jsem převzal z NfSen stránky *Details*.

Původně jsem uvažoval o interaktivním frontendu, který by umožňoval také ruční analýzu výsledků pluginu. Ideální by bylo vložit do stránky pluginu IPython notebook. K tomu jsou ovšem nenašel žádnou dokumentaci, takže to v současnosti asi ani není možné. Nebo by notebook mohl jít spouštět z pluginu, ale to by bylo zbytečně složité. Jednodušší si je spustit notebook ručně a mít nad ním plnou kontrolu.

Experimentoval jsem také s balíčky bokeh a mpld3 [74, 75], které dokáží z pythonu generovat interaktivní grafy a exportovat je do kódu, jež lze poté vložit do webové stránky. Vyzkoušel jsem je přímo v IPython notebooku. Výsledky byly rozpačité, s grafy sice šlo například přibližovat a u jednotlivých bodů zobrazovat bubliny s popisky při najetí myši (viz obrázek 3.4), vše bylo nicméně poměrně pomalé a paměťově náročné. Nevím, jak přesně IPython notebook ukládá historii výstupů, nicméně jeho paměť se postupně zvyšovala až na úroveň, kdy byly jeho reakce velmi pomalé a bylo nutné ho restartovat.

Frontend neobsahuje žádné nastavení například parametrů DBSCANu proto, že je podle mého názoru praktičtější vyladit parametry v IPython notebooku, kde je možné testovat na libovolných historických tocích. Přímo ve frontendu bych musel vždy čekat dalších 5 minut na nová data.




Obrázek 3.4: Interaktivní HTML graf vygenerovaný balíčkem bokeh se zobrazenou bublinou s detailem toku. Popis viz 3. odstavec sekce 3.2.9.

Podobně v pluginu není implementována funkce pro vyhodnocení alertu. Při vytváření alertu totiž nelze nastavit žádné hodnoty (zde by se hodily například prahové hodnoty pro LOF) a alerty jsou volány na nejnovější data profiltrovaná uživatelským filtrem. Plugin tedy má přístup pouze k této pětiminutovce. Rozumnější je tedy implementovat vlastní alerty, které půjdou aktivovat zvlášť na různé profily a s různými prahovými hodnotami pro různě objevené anomálie.

## 3.3 Instalace ad_pluginu

Instalace se skládá ze tří kroků, které jsou dále popsány v následujících odstavcích. Předpokladem je již nainstalovaný a zprovozněný NfSen. Kroky instalace jsou:

1. Instalace pythonu a potřebných balíčků
   (volitelně také IPythonu)

2. Zkopírování souborů a složky pluginu frontendové i backendové části a nastavení příslušných přístupových práv

3. Registrace a nastavení pluginu v konfiguračním souboru nfsen.conf

Pro běh pluginu je potřeba interpretr pythonu alespoň ve verzi 3.4. Doporučuji standardní interpretr (označovaný jako CPython) [41]. Existuje i rychlejší implementace s názvem PyPy [76], která používá Just In Time (JIT) kompilaci. Třetí verze pythonu a některé použité knihovny však zatím nejsou

plně podporovány. Dále je potřeba nainstalovat balíčky NumPy, SciPy, pandas, geoip2, matplotlib a scikit-learn [46, 47, 53, 77, 55, 54]. Nejjednodušší cestou je instalace pomocí programu pip či pip3 [78] (balíčky stahuje z PyPI – Python Package Index [79]), který je součástí standardní instalace pythonu.

```
sudo pip numpy, scipy, pandas, matplotlib, geoip2
```

Pro případnou instalaci IPythonu [56]:

```
sudo pip install "ipython[notebook]"
```

Jediný balíček, který není dostupný pomocí programu pip je scikit-learn. Pokyny k instalaci pro různé platformy je zde [80]. Pro operační systém Ubuntu je instalace následující:

```
sudo apt-get install build-essential python3-dev \
                python3-setuptools python3-numpy python3-scipy \
                libatlas-dev libatlas3gf-base python-sklearn
```

Dalším krokem je zkopírování souborů pluginu do NfSenu. Do složky `$BACKEND_PLUGINDIR` je třeba zkopírovat soubor `ad_plugin.pm` a složku `ad_plugin` a všem souborům přidat oprávnění na čtení pro skupinu `$WWWGROUP`, pro přidanou složku i právo zápisu. Do složky `$FRONTEND_PLUGINDIR` je třeba zkopírovat soubor `ad_plugin.php`. Všechny tři perlovské proměnné jsou definované v souboru `nfsen.conf`.

Zbývá už jenom zaregistrovat a nastavit plugin v konfiguračním souboru `CONFDIR/$nfsen.conf`. Registrace probíhá pomocí zapsání do pole `@plugins` hodnotu například `['*', 'ad_plugin']`, pro spouštění pluginu pro všechny profily, či místo hvězdičky výčet profilů oddělených čárkou. Dále připište do hashe `%pluginconf` hodnotu `ad_plugin => {ad_plugin_dir=>'ad_plugin'}`.

Konfigurace pluginu se provádí v textovém souboru `ad_plugin` v proměnné *settings*, která je blízko začátku souboru. Lze zde nastavit všechny parametry algoritmů DBSCAN a LOF a také další podrobnosti jako úroveň logování a podobně.

Po instalaci lze plugin samostatně spustit buď pomocí IPython notebooku `program.ipynb` nebo pomocí programu `testPlugin` ze složky `$BINDIR` s libovolným profilem a časovou značkou souboru s toky (doporučuji spouštět jako uživatel `$WWW-USER`). Je vhodné mít připraveno alespoň jednu hodinu síťových toků. V případě potíží doporučuji sledovat syslog.

V případě nutnosti zvýšení efektivity lze použít nejspíše rychlejší implementace BLAS (například z porovnání v [52]) nebo nasadit JIT kompilátor PyPy [76], tedy až bude plně podporovat python verze 3 a použité balíčky.

### 3.3.1 Přiložený IPython notebook

K samotným souborům pluginu přikládám také IPython notebook (ve složce ad_plugin), který umožňuje podrobněji analyzovat výsledky pluginu nebo například vyzkoušet různá nastavení parametrů jednotlivých algoritmů pro určitou specifickou síť. V notebooku je importován ad_plugin jako modul a je tedy možné používat veškeré jeho funkce od načítání toků přes zpracování atributů i po samotnou analýzu odlehlých bodů – tedy detekci anomálií.

V notebooku jsou připravené ukázky použití všech hlavních funkcí pluginu a také příklady kódu pro vytváření grafů či histogramů z načtených dat včetně samotných obrázků grafů.

Přikládám i HTML export notebooku, který není interaktivní, ale také poskytuje určitý návod pro užití funkcí pluginu i v případě, že program IPython není nainstalovaný.

Kapitola 4

# Testování

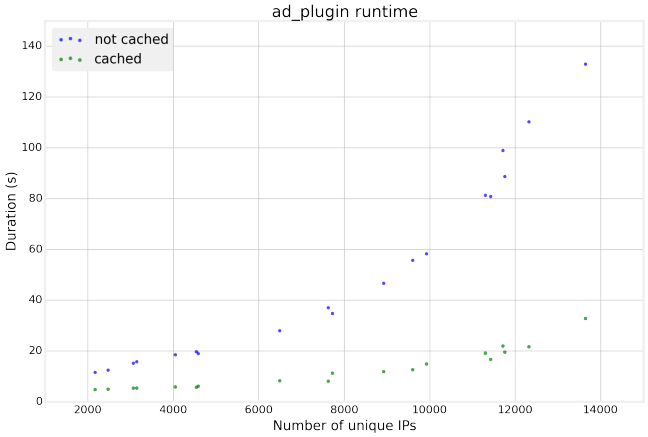
## 4.1 Testování stability a výkonu

Program NfSen spouští pluginy každých 5 minut s nejnovějšími daty a to pro všechny profily nastavené v konfiguraci pro dané pluginy. Zásadní podmínkou tedy je, aby součet časů běhu všech spuštění pluginů bylo menší než oněch 5 minut. Toto bylo na splněno jak na datech z mého počítače, tak na datech z fakultní sítě. Během testování výkonnosti jsem se zaměřil především na rychlost a stabilitu i při různých nastaveních NfSenu, hlavně co se týká právě profilů.

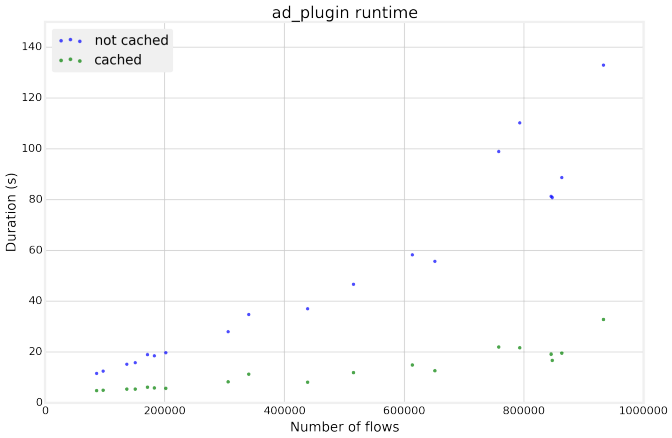
Plugin zapisuje do syslogu v podstatě všechny podstatné informace včetně konfigurace, průběhu výpočtů a výsledků či chybových hlášení. Díky tomu jsem odhalil mnoho chyb. Během zátěžového testu nejdříve s daty naměřenými na mém počítači a poté na datech z fakulty se chyby objevily pouze v případě nedostatku dat. Tedy že nebyla naměřena potřebná hodina dat zpátky vzhledem k časové značce předané při spuštění. Aby po spuštění sondy nebylo potřeba čekat celou hodinu na první výsledky z pluginu, upravil jsem program tak, že toky načítá postupně od současnosti do minulosti a v případě, že existuje alespoň 5 minut toků k porovnání se současnými toky, je do logu zapsáno pouze varování a plugin dále pokračuje s částečnými daty. Plugin tedy vrací výsledky již po 10-ti minutách provozu sondy (porovnáním dvou „pětiminutovek“) a do logu zapíše patřičné varování. Jiné chyby než z nedostatku dat se při prohledání logu již neobjevily.

Samotnou správnost funkcí jsem zkoušel během vývoje v IPython notebooku. Zvláště z počátku, kdy jsem se seznamoval s jednotlivými balíčky, síťovými toky i samotnými daty z fakulty, jsem vždy výsledky jednotlivých funkcí kontroloval ručně na části toků. Prověřoval jsem hlavně výpočty atributů, případné chyby se totiž dále prakticky neprojevily.

Všechna měření probíhala na mém 7 let starém notebooku s 64 bitovým operačním systémem Ubuntu verze 15.04, dvou jádrovým procesorem Intel® Core™2 Duo T8100 s frekvencí 2.10GHz a 3 GB pamětí. Měřil jsem čas běhu pluginu pomocí NfSenového skriptu `testPlugin` ze složky `$BINDIR` s použitím




Obrázek 4.1: Čas běhu ad_pluginu v závislosti na počtu unikátních IP adres s použitím dočasných souborů a bez.




Obrázek 4.2: Čas běhu ad_pluginu v závislosti na počtu toků s použitím dočasných souborů a bez.

dočasných souborů i bez nich. Měření jsem prováděl na datech z fakulty ze všedního dne od pěti hodin ráno do půlnoci. Dočasné soubory pro toky měly celkově 617 MB, pro atributy 67 MB. Prováděl jsem vždy alespoň dvojici stejných měření a zaznamenal lepší výsledek. Výsledky měření jsou vidět na obrázku 4.2, kde je čas v závislosti na celkovém počtu toků, a na obrázku 4.1, kde je čas v závisloti na počtu unikátních IP adres (přibližně odpovídá počtu řádků matice vstupující do DBSCAN algoritmu). Průměrný počet toků na jednu IP adresu není konstantní, ale zvyšuje se lineárně s počtem toků. Dvakrát se mi během měření stalo, že počítač pár desítek vteřin nereagoval.

Měření s daty z mého počítače probíhaly se třemi různými profily, měření

s daty z fakulty bylo prováděno pouze na hlavním profilu *live*, který obsahuje všechny toky.

## 4.2 Testování detekce

Před samotným testováním bylo potřeba určit parametry algoritmu DBSCAN. Postupoval jsem experimentálně a nastavil jsem hodnoty tak, aby počet shluků vycházel alespoň 10 a zároveň počet bodů bez shluku byl spíše pod touto hodnotou. Zvyšováním parametru $\epsilon$ se počet shluků snižuje, naopak to platí pro parametr *minPts*. Více shluků také obvykle znamená více bodů mimo shluky. Parametr $k$ u algoritmu LOF doporučuji zvolit větší než *minPts*. Tato pravidla samozřejmě neplatí univerzálně, jsou to spíše empirická pravidla, se kterými se mi podařilo na datech z fakultního provozu najít nějaké anomálie. Jedna z použitých konfigurací je i v základu nastavena v pluginu.

Funkce *find_anomalies* vrací trojici seznamů, jež obsahují dvojice číslo a IP adresa. Číslo označuje míru změnu nebo odlehlosti. Jednotlivé seznamy obsahují po řadě body, které změnily shluk a nové a staré body mimo shluky. Testoval jsem hlavně na datech z fakultního provozu. Na datech z mého počítače se mi nepodařilo nastavit parametry podle pravidel z předchozího odstavce. Myslím si, že tato data jsou malého rozsahu a proto se pro statistické metody moc nehodí.

Jenom během dvou hodinového rozpětí provozu jsem pomocí funkcí pluginu našel hned několik typů útoku. Jednalo se nejčastěji o pokusy o připojení ke službě ssh na pár desítkách IP adres vždy s podobnými charakteristikami. Tedy krátký čas připojení, počet bajtů několik desítek (často stejné hodnoty), odpovědi většinou také podobné, připojení obvykle ze zahraničí (Čína, Hong Kong). Našel jsem také obdobné pokusy o připojení na službu telnet s postupně rostoucí cílovou adresou, tentokrát šlo o několik adres z Kanady.

Další objevený útok z Číny vyhledával Microsoft SQL servery (port 1433) na desítce různých adres. Plugin také několikrát objevil různé podivné komunikace, které jsem nedokázal zhodnotit, zda může jít o útok či nikoliv. Například z jedné adresy bylo vytvořeno několik spojení pomocí HTTPS protokolu, každé ovšem mělo stovky bajtů, přestože průměrně mají toky tohoto protokolu tisíce bajtů. Pokud by se jednalo o nějaký mírně sofistikovanější útok, předpokládám, že bych ho bez větších zkušeností stejně nerozeznal.

Zajímavé je, že pár z těchto útoků mělo hodnotu LOF mírně nižší než jedna, což znamená, že hustota bodů v okolí je přibližně stejná jako okolí tohoto bodu. Problém byl v nastavení parametru $k$ na moc nízkou hodnotu. Po zvýšení již byly hodnoty LOFu větší než jedna.

Poměrně často plugin vracel jako podezřelé IP adresy zahraničních DNS serverů, které se nejspíše snažily o překlad nějaké adresy. Šlo obvykle o několik toků a dále již nic. Přeloženou adresu tedy nejspíše vrátili svému klientovi, který se poté připojil na získanou adresu. Myslím si, že v takových případech

nejde o žádnou anomálii, ale o planý poplach. Nicméně toto také nedokážu moc dobře posoudit. Řešením by mohlo být separovat DNS provoz do zvláštního profilu NfSenu. Takto by se DNS provoz srovnával pouze s ostatním DNS provozem a množství anomálií by bylo pravděpodobně stabilnější a věrohodnější. Toto jsem ovšem nevyzkoušel. Samotná analýza jednotlivých IP adres označených jako anomálie byla velmi pracná a zdlouhavá.

Jako anomálie byl také označen jeden z webových serverů fakulty, kde ovšem převažoval provoz služby ssh. Ověřil jsem si, že na tomto serveru služba ssh běží, připojení se mi však nepodařilo, pro připojení nejspíše potřebuji certifikát serveru. Na tento server se nicméně opakovaně pokoušelo připojit několik adres z Číny, Hong Kongu i USA. Jiná adresa z Číny se pokoušela připojit na Microsoft SQL server. Anomálie to tedy byla, ale daná IP adresa útok neprováděla, ale byla spíše cílem útoků.

# Závěr

Cílem této práce bylo seznámit se s metodami detekce anomálií v oblasti síťové bezpečnosti a navrhnout plugin pro softwarovou sondu síťových toků NfSen, který takovou detekci provádí. Toto zadání bylo splněno a na datech z univerzitní sítě byly pomocí pluginu detekovány pravděpodobné útoky. V následujících odstavcích krátce shrnuji hlavní body práce a na závěr uvádím možné směry dalšího budoucího rozvoje pluginu.

Seznámil jsem se se způsoby získávání a zpracování dat ze síťového provozu, tedy s paketovou analýzou a hlavně se síťovými toky. Popsal jsem pojem anomálie a rozebral jejich možné druhy podle nalezené literatury. Dále jsem uvedl různé způsoby jejich detekce. Zvláště jsem se zabýval detekcí pomocí behaviorální analýzy a nástroji které používá. Jsou to algoritmy shlukové analýzy a způsoby detekce a hodnocení odlehlých bodů.

Pro implementaci jsem vybral programovací jazyk python (druhou možnou variantou byl jazyk R) s balíčky pro práci s tabulkovými daty pandas a pro data mining scikit-learn. Pro vývoj jsem zvolil prostředí IPython notebooků. Vybral jsem množinu atributů. Některé z nich jsem navrhl sám, použil jsem např. možnosti geolokace IP adres. Navrhnul jsem způsob zpracování atributů, aby bylo možné porovnat nejnovější síťový provoz s historickým. Vybral jsem také algoritmus pro detekci shluků DBSCAN, který sám vrací body, jež nejsou součástí shluku. Tyto body pak hodnotím mírou odlehlosti LOF. Navrhl jsem také sledovat body, které změnily shluk, a hodnotit jejich míru změny pomocí vzdálenosti.

Nastudoval jsem si program nfdump a softwarovou sondu NfSen, pro kterou jsem naprogramoval plugin *ad_plugin*. Chvíli mi trvalo, než jsem se zorientoval a naučil se všemi použitými programy a balíčky pythonu. Například dokumentace NfSenu je poměrně stručná a několikrát jsem tedy musel procházet jeho zdrojový kód v perlu (, který moc neovládám), abych zjistil, jak nějaká část funguje. Několik problémů jsem měl i s nfdumpem, podrobněji jsem se rozepsal v sekci 3.1.1.1. Python samotný i jeho balíčky jsou poměrně dobře dokumentované a nejspíše i poměrně používané. Případné řešení růz-

ných zádrhelů jsem totiž obvykle poměrně rychle dohledal na internetu.

Během testování jsem ověřil, že vytvořený plugin bez problému zvládne zpracovat síťové toky z části provozu fakulty a to i při startu bez pomocných dočasných souborů, které podstatně zkracují čas běhu pluginu. Ověřil jsem i samotnou funkčnost pluginu. Ten odhalil různé podezřelé chování, které odpovídá například vyhledávání zařízení se spuštěnou službou ssh či telnet, které je možná doprovázeno i zkoušením výchozích hesel. Tento provoz byl odhalen i přesto, že šlo někdy např. pouze o několik toků. Mezi podezřelé byly poměrně často zařazeny i IP adresy, které komunikovaly s fakultním dns serverem. Podle mého názoru nejspíše nešlo i nic podezřelého. Řešení problému častého výskytu dns serverů (či jiných výrazných serverů) je pomocí vytvoření vlastního profilu pouze s dns provozem. Tak lze zjistit, jestli jsou dané dns provozy anomální vzhledem k ostatním.

## Možnosti dalšího rozvoje

Vytvořený plugin lze dále vylepšit mnoha způsoby. Lze například přidat výpočet dalších atributů (například ty nepoužité z návrhu v sekci 2.2). Atributy lze také počítat nejenom pro časové úseky, ale i pro posledních $n$ toků pro dané IP adresy. Má implementace je pro tato rozšíření dobře připravena. Je však třeba brát zřetel na zvyšující se časovou náročnost při zvýšení počtu atributů. Geolokace IP adres jde například zjemnit na úroveň měst a částí zemí. Záleží však na přesnosti databáze, zda by tato změna měla nějaký přínos.

Zajímavé by bylo rozšířit plugin rozhraním pro další analýzu podezřelých IP adres. Je škoda, že do pluginu nejde jednoduše zakomponovat některé funkce NfSenu, které by se na tuto analýzu hodily.

Nejzajímavější by bylo vymyslet způsob, jak analyzovat, co konkrétně má vliv na odlehlost nalezených bodů. Jestli je to hlavně jeden atribut či spíše jejich kombinace a tuto informaci nějak vhodně předat v uživatelském rozhraní. Uživatel pluginu by pak mohl lépe rozeznat relevanci nalezených anomálií a případně poupravit nastavení algoritmů. Je také možné přidat váhy k jednotlivým atributům pro posílení nebo zeslabení jejich vlivu.

# Literatura


[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper. [cit. 29. 5. 2015]. Dostupné z: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[2] Internet users. [cit. 29. 5. 2015]. Dostupné z: http://www.internetlivestats.com/internet-users/

[3] IT threat evolution in Q1 2015. [cit. 29. 5. 2015]. Dostupné z: https://securelist.com/analysis/quarterly-malware-reports/69872/it-threat-evolution-in-q1-2015/

[4] Cisco 2014 midyear security report. [cit. 29. 5. 2015]. Dostupné z: http://www.cisco.com/web/offer/grs/190720/SecurityReport_Cisco_v4.pdf

[5] nfsen. [cit. 29. 5. 2015]. Dostupné z: http://nfsen.sourceforge.net/

[6] Listina základních práv a svobod. 1992, [cit. 29. 5. 2015]. Dostupné z: http://www.psp.cz/docs/laws/listina.html

[7] Trestní zákoník, § 182: Porušení tajemství dopravovaných zpráv. 2010, [cit. 29. 5. 2015]. Dostupné z: http://portal.gov.cz/zakon/40/2009/182

[8] ČT24: Monitorování zaměstnanců versus právo na soukromí. [cit. 29. 5. 2015]. Dostupné z: http://www.ceskatelevize.cz/ct24/svet/312803-monitorovani-zamestnancu-versus-pravo-na-soukromi/

[9] TAP vs. SPAN. [cit. 29. 5. 2015]. Dostupné z: http://www.networkinstruments.com/includes/popups/taps/tap-vs-span.php

[10] NetManiak: Příklad NetFlow architektury. [cit. 29. 5. 2015]. Dostupné z: https://cs.wikipedia.org/wiki/NetFlow#/media/File:NetFlowModerniArchitektura.png

[11] Berk, V.: The NetFlow/sFlow®/CFlow/JFlow Flow Dilemma. [cit. 29. 5. 2015]. Dostupné z: http://www.flowtraq.com/corporate/resources/whitepapers/the-netflowsflowcflowjflow-flow-dilemma/

[12] Cisco Systems NetFlow Services Export Version 9. [cit. 29. 5. 2015]. Dostupné z: https://www.tools.ietf.org/html/rfc3954

[13] Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. [cit. 29. 5. 2015]. Dostupné z: https://tools.ietf.org/html/rfc7011

[14] FlowMon plugin: Flowmon Traffic Recorder. [cit. 29. 5. 2015]. Dostupné z: https://www.invea.com/cs/produkty-sluzby/flowmon/flowmon-pluginy#FlowMonTrafficRecorder

[15] ABZ.cz: slovník cizích slov – anomálie. [cit. 29. 5. 2015]. Dostupné z: http://slovnik-cizich-slov.abz.cz/web.php/slovo/anomalie

[16] Blažek, R. B.; Kim, H.; Rozovskii, B.; aj.: A novel approach to detection of 'Denial-of-Service' attacks via adaptive sequential and batch-sequential change-point detection methods. In *Proc. 2nd IEEE Systems, Man, and Cybernetics Information Assurance Workshop*, West Point, NY, Červen 2001.

[17] Tartakovsky, A.; Rozovskii, B.; Blazek, R. B.; aj.: A Novel Approach to Detection of Intrusions in Computer Networks via Adaptive Sequential and Batch-sequential Change-point Detection Methods. *IEEE Transactions on Signal Processing*, ročník 54, č. 9, Září 2006: s. 3372–3382.

[18] Tartakovsky, A.; Rozovskii, B.; Blažek, R. B.; aj.: Detection of Intrusions in Information Systems by Sequential Change-point Methodss. *Statistical Methodology*, ročník 3, č. 3, Červenec 2006: s. 252–293.

[19] 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, doi:10.1145/1541880.1541882, [cit. 29. 5. 2015]. Dostupné z: http://doi.acm.org/10.1145/1541880.1541882

[20] DNS Tunneling. [cit. 29. 5. 2015]. Dostupné z: http://resources.infosecinstitute.com/dns-tunnelling/

[21] Ping smrti. [cit. 29. 5. 2015]. Dostupné z: https://cs.wikipedia.org/wiki/Ping_smrti

[22] Distributed Denial of Service. [cit. 29. 5. 2015]. Dostupné z: https://cs.wikipedia.org/wiki/Denial_of_service#Distributed_DoS

[23] Neumann, M.: Služba a NfSen plugin pro analýzu síťových toků pomocí sekvenčních statistických metod. Praha: ČVUT, 2014. Diplomová práce. ČVUT, Fakulta informačních technologií.

[24] Examples of metric spaces. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/Metric_space#Examples_of_metric_spaces

[25] Comparing different clustering algorithms on toy datasets. [cit. 29. 5. 2015]. Dostupné z: http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

[26] Dempster, A. P.; Laird, N. M.; Rubin, D. B.: Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, ročník 39, č. 1, 1977: s. 1–38, [cit. 29. 5. 2015]. Dostupné z: http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.4884

[27] Hierarchical clustering. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/Hierarchical_clustering

[28] Ester, M.; peter Kriegel, H.; S, J.; aj.: A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press, 1996, s. 226–231, [cit. 29. 5. 2015]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.71.1980

[29] DBSCAN Illustration example. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/DBSCAN

[30] Tao, H.; Yanna, T.: Research Outlier Detection Technique Based on Clustering Algorithm. In *2014 7th Conference on Control and Automation (CA)*, Prosinec 2014, s. 12–14, doi:10.1109/CA.2014.10, [cit. 29. 5. 2015]. Dostupné z: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&tp=&arnumber=7026251

[31] k-d tree. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/K-d_tree

[32] Ball tree. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/Ball_tree

[33] R* tree. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/R*_tree

[34] Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; aj.: LOF: Identifying Density-based Local Outliers. *SIGMOD Rec.*, ročník 29, č. 2, Květen 2000: s. 93–104, ISSN 0163-5808, doi:10.1145/335191.335388, [cit. 29. 5. 2015]. Dostupné z: http://doi.acm.org/10.1145/335191.335388

[35] Lazarevic, A.; Ertöz, L.; Kumar, V.; aj.: A comparative study of anomaly detection schemes in network intrusion detection. In *In Proceedings of SIAM Conference on Data Mining*, 2003, [cit. 29. 5. 2015]. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.215.3893

[36] Ertöz, L.; Eilertson, E.; Lazarevic, A.; aj.: MINDS – Minnesota Intrusion Detection System. 2004, [cit. 29. 5. 2015]. Dostupné z: http://minds.cs.umn.edu/

[37] Bhuyan, M.; Bhattacharyya, D.; Kalita, J.: Network Anomaly Detection: Methods, Systems and Tools. *Communications Surveys Tutorials, IEEE*, ročník 16, č. 1, First 2014: s. 303–336, ISSN 1553-877X, doi:10.1109/SURV.2013.052213.00046, [cit. 29. 5. 2015]. Dostupné z: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=6524462

[38] PYPL PopularitY of Programming Language. [cit. 29. 5. 2015]. Dostupné z: https://pypl.github.io/PYPL.html

[39] Piatetsky, G.: R leads RapidMiner, Python catches up, Big Data tools grow, Spark ignites. [cit. 29. 5. 2015]. Dostupné z: http://www.kdnuggets.com/2015/05/poll-r-rapidminer-python-big-data-spark.html

[40] The R Project for Statistical Computing. [cit. 29. 5. 2015]. Dostupné z: http://www.r-project.org/

[41] Python. [cit. 29. 5. 2015]. Dostupné z: https://www.python.org/

[42] R vs Python for Data Science: The Winner is .... [cit. 29. 5. 2015]. Dostupné z: http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html

[43] R vs Python for data analysis. [cit. 29. 5. 2015]. Dostupné z: https://programmers.stackexchange.com/questions/181342/r-vs-python-for-data-analysis

[44] SAS vs. R (vs. Python) – which tool should I learn? [cit. 29. 5. 2015]. Dostupné z: http://www.analyticsvidhya.com/blog/2014/03/sas-vs-vs-python-tool-learn/

[45] rpy2 – R in Python. [cit. 29. 5. 2015]. Dostupné z: http://rpy.sourceforge.net/

[46] NumPy. [cit. 29. 5. 2015]. Dostupné z: http://www.numpy.org/

[47] SciPy. [cit. 29. 5. 2015]. Dostupné z: http://www.scipy.org/scipylib/index.html

[48] LAPACK – Linear Algebra PACKage. [cit. 29. 5. 2015]. Dostupné z: http://www.netlib.org/lapack/

[49] BLAS – Basic Linear Algebra Subprograms. [cit. 29. 5. 2015]. Dostupné z: https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms

[50] Wolphram Mathematica. [cit. 29. 5. 2015]. Dostupné z: https://www.wolfram.com/mathematica/

[51] MATLAB. [cit. 29. 5. 2015]. Dostupné z: https://www.mathworks.com/products/matlab/

[52] Optimized R and Python: standard BLAS vs. ATLAS vs. OpenBLAS vs. MKL. [cit. 29. 5. 2015]. Dostupné z: http://blog.nguyenvq.com/blog/2014/11/10/optimized-r-and-python-standard-blas-vs-atlas-vs-openblas-vs-mkl/

[53] pandas – Python Data Analysis Library. [cit. 29. 5. 2015]. Dostupné z: http://pandas.pydata.org/

[54] scikit-learn – Machine Learning in Python. [cit. 29. 5. 2015]. Dostupné z: http://scikit-learn.org/

[55] matplotlib. [cit. 29. 5. 2015]. Dostupné z: http://matplotlib.org/

[56] Pérez, F.; Granger, B. E.: IPython: a System for Interactive Scientific Computing. *Computing in Science and Engineering*, ročník 9, č. 3, Květen 2007: s. 21–29, ISSN 1521-9615, doi:10.1109/MCSE.2007.53, [cit. 29. 5. 2015]. Dostupné z: http://ipython.org

[57] Jupyter. [cit. 29. 5. 2015]. Dostupné z: https://jupyter.org/

[58] Geolocation Providers. [cit. 29. 5. 2015]. Dostupné z: http://whatismyipaddress.com/geolocation-providers

[59] GeoLite2 Free Downloadable Databases. [cit. 29. 5. 2015]. Dostupné z: https://dev.maxmind.com/geoip/geoip2/geolite2/

[60] MaxMind. [cit. 29. 5. 2015]. Dostupné z: https://www.maxmind.com/

[61] Hofstede, R.; Fioreze, T.: SURFmap: A network monitoring tool based on the Google Maps API. In *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, s. 676–690, doi:10.1109/INM.2009.5188876, [cit. 29. 5. 2015]. Dostupné z: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5188876

[62] SURFmap. [cit. 29. 5. 2015]. Dostupné z: http://sourceforge.net/projects/surfmap/

[63] IP2Location. [cit. 29. 5. 2015]. Dostupné z: https://www.ip2location.com/

[64] nfdump. [cit. 29. 5. 2015]. Dostupné z: http://nfdump.sourceforge.net/

[65] tcpdump & libpcap. [cit. 29. 5. 2015]. Dostupné z: http://www.tcpdump.org/

[66] fprobe. [cit. 29. 5. 2015]. Dostupné z: http://fprobe.sourceforge.net/

[67] INVEA-TECH. [cit. 29. 5. 2015]. Dostupné z: https://www.invea.com/

[68] Install nfdump and nfsen netflow tools in Linux. [cit. 29. 5. 2015]. Dostupné z: http://www.linuxscrew.com/2011/02/23/install-nfdump-and-nfsen-netflow-tools-in-linux/

[69] Installation and configuration of NFDUMP and NfSen on Ubuntu. [cit. 29. 5. 2015]. Dostupné z: http://terraltech.com/installation-and-configuration-of-nfdump-and-nfsen-on-ubuntu/

[70] CPAN – Comprehensive Perl Archive Network. [cit. 29. 5. 2015]. Dostupné z: http://www.cpan.org/

[71] Apache HTTP server. [cit. 29. 5. 2015]. Dostupné z: https://httpd.apache.org/

[72] pylof – Python implementation of Local Outlier Factor algorithm. [cit. 29. 5. 2015]. Dostupné z: https://github.com/damjankuznar/pylof

[73] Improving performance of Local outlier factor with KD-Trees. [cit. 29. 5. 2015]. Dostupné z: http://www.bistaumanga.com.np/blog/lof/

[74] bokeh. [cit. 29. 5. 2015]. Dostupné z: http://bokeh.pydata.org/

[75] mpld3. [cit. 29. 5. 2015]. Dostupné z: https://mpld3.github.io/

[76] PyPy – python JIT compiler. [cit. 29. 5. 2015]. Dostupné z: http://pypy.org/

[77] geoip2 – MaxMind GeoIP2 Python API. [cit. 29. 5. 2015]. Dostupné z: https://geoip2.readthedocs.org/en/latest/

[78] pip – python package installation tool. [cit. 29. 5. 2015]. Dostupné z: https://pip.pypa.io/en/stable/

[79] pypi – Python Package Index. [cit. 29. 5. 2015]. Dostupné z: https://pypi.python.org/pypi

[80] Installing scikit-learn. [cit. 29. 5. 2015]. Dostupné z: http://scikit-learn.org/dev/install.html

Příloha A

# Seznam použitých zkratek

**IDS** Intrusion Detection System

**IPFIX** IP Flow Information eXport

**SLA** Service Level Agreement

**P2P** Peer-to-peer

**DDoS** Distributed Denial of Service

**DNS** Domain Name System

**FAR** False Alarm Rate

**DBSCAN** Density-based spatial clustering of applications with noise

**LOF** Local Outlier Factor

**MINDS** Minnesota Intrusion Detection System

**BLAS** Basic Linear Algebra Subprograms

**LAPACK** Linear Algebra PACkage

**HTTP** Hypertext Transfer Protocol

**BSD** Berkeley Software Distribution

**TCP** Transmission Control Protocol

**UDP** User Datagram Protocol

**ICMP** Internet Control Message Protocol

**IMAPS** Internet Message Access Protocol over TLS/SSL

**TLS** Transport Layer Security

**SSL** Secure Sockets Layer

**RFC** Request For Comments

**PHP** Hypertext Preprocessor

**CPAN** Comprehensive Perl Archive Network

**RRDtool** Round Robin Database Tool

**HTML** HyperText Markup Language

Příloha **B**

# Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `ad_plugin` ..... zdrojové soubory pluginu ad_plugin
  - `ad_plugin.py` ..... hlavní program pluginu
  - `ad_plugin.pm` ..... backendový plugin do NfSenu
  - `ad_plugin.php` ..... frontendový plugin do NfSenu
  - `example_notebook.ipynb` ..... IPython notebook s příklady
  - `example_notebook.html` ... html verze IPython notebooku s příklady
  - `GeoDB` ..... složka s geolokační databází
- `text` ..... text práce a jeho zdrojové soubory
  - `DP_Lessner_Petr_2015.tex` ... zdrojový text práce ve formátu LaTeX
  - `DP_Lessner_Petr_2015.pdf` ..... text práce ve formátu pdf
  - `img` ..... složka obrázků použitých v textu práce