Masarykova univerzita
Fakulta informatiky

Detekce anomálií a vizualizace síťového provozu

Diplomová práce

Petr Bartel

Brno, podzim 2011

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Petr Bartel

Vedoucí práce: RNDr. Jan Vykopal

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu práce RNDr. Janu Vykopalovi za vstřícný přístup, odborný dohled a cenné rady. Stejně tak bych rád poděkoval mé rodině a přátelům za trpělivost, ohleduplnost a neustálou podporu.

Shrnutí

Detekce anomálií v síťových tocích je při zvětšujícím se objemu dat i počítačových útoků klíčová pro provoz sítě a bezpečnost dat v ní přenášených. Argumentem pro použití síťových toků jsou rychlosti přenosu a objem dat na páteřních linkách. Kompletní záznamy pomocí paketů jsou při těchto podmínkách velmi náročné na výkon zařízení. Dostupné nástroje pro analýzu síťových toků často neposkytují automatickou detekci anomálií.

NfSen je jednou z open-source aplikací, která slouží primárně pro vizuální analýzu síťových toků. Úkolem práce je vytvoření rozšiřujícího pluginu této aplikace umožňujícího automatickou detekci a vizualizaci síťových anomálií.

Práce se zabývá porovnáním dostupných technologií využitelných při návrhu a implementaci pluginu. Kromě vymezení oblasti působnosti pluginu představuje možné alternativní aplikace s podobnou funkcionalitou. S potřebou ukládání velkých objemů dat a rychlého přístupu k nim souvisí přehled databázových řešení a jejich porovnání. Práce přináší návrháři výběr vizualizačních knihoven, přehled jejich funkcionality a praktické ukázky výstupů i konfigurací. Část věnující se představení a srovnání kompresních algoritmů je v práci obsažena z důvodu využití komprese při odhadu entropie IP adres a portů obsažených v síťových tocích.

Výstupem práce je praktická implementace pluginu, která staví na dříve představených technologiích, schopná vizualizace a automatické detekce anomálií. Objevené anomálie jsou hlášeny pomocí e-mailu. Funkčnost a výkon pluginu byly ověřeny při testovaní na studentském kolektoru, jehož data pochází z reálného provozu Masarykovy univerzity.

Práce tedy potvrzuje možnost automatické detekce a reakce na síťové anomálie pomocí pluginu NfSen a přináší konceptuální podklady pro tvorbu vlastního pluginu.

Klíčová slova

nfdump, NfSen, NetFlow, síťová bezpečnost, vizualizace, anomálie, entropie, komprese

Obsah

Kapitola 1

Úvod

Sběr a zpracování informací o provozu na síti je jedním ze základních kamenů mnoha oborů informačních technologií. S potřebou zajištění těchto dat se můžeme setkat při účtování poskytnutých služeb, sledování zátěže a výkonu či zabezpečení firemního prostředí a bezproblémového provozu.

S narůstajícím objemem dat předávaných prostřednictvím sítě Internet a evolucí datových spojů se stává nemožným sledovat každý paket. Při rychlostech mnohdy přesahujících desítky gigabajtů by to bylo extrémně výpočetně náročné. Z tohoto důvodu bylo vymyšleno jisté zjednodušení, které si však zachovalo svoji výpovědní hodnotu – **síťový tok**.

Vzhledem k tomu, že síťový tok je poměrně mladý standard definovaný původně firmou Cisco, není ani mnoho aplikací zabývajících se jeho zpracováním a analýzou dat v něm obsažených. Situaci neulehčuje ani fakt, že každý větší výrobce má svou vlastní specifikaci síťového toku. Pomineme-li existenci nákladných a komplexních komerčních nástrojů, o jejichž existenci se zmiňuje kapitola 2.2 na straně 16, zůstane nám několik světlých výjimek. Do této kategorie patří i open-source aplikace NfSen, která umožňuje práci se síťovými toky NetFlow a následnou vizualizaci dat v nich obsažených. Co ovšem NfSen neovládá, je automatická detekce anomálií.

Právě zde nacházíme oblast, která zasluhuje více pozornosti. Vizuální analýza časových řad prováděná lidskou silou je sice přesná, ale pro obsluhu znamená neustálé sledování vývoje síťového provozu a návazné aktivity v případě podezření na anomálii. Nahrazení stálého dohledu aplikací poskytující automatickou detekci anomálií a upozornění o jejich výskytu je úkolem této práce.

NfSen umožňuje rozšíření svých schopností díky pluginům. Cílem této práce je navrhnout a implementovat plugin pro tuto aplikaci, který by splňoval následující požadavky:

- Sledování objemu přenesených dat (toky, pakety a bajty).
- Sledování entropie IP adres a portů.
- Sledování provozu na významných aplikačních portech.
- Schopnost nastavení citlivosti sledování odchylek od průměru.
- Schopnost přizpůsobení detekce anomálií vývoji provozu na síti.
- Schopnost dokončení své činnosti v horizontu maximálně pěti minut.
- Vizualizace síťového provozu včetně vyznačených anomálií v grafech.
- E-mailové upozornění správce sítě na možnou anomálii.
- Funkční a výkonnostní testování modulu v síti Masarykovy univerzity.

Jak jsme již zmínili, síťový provoz zahrnuje enormní množství dat, které je nutné uložit. Samotný NfSen sice poskytuje úložiště síťových toků, nicméně dle výše zmíněných požadavků lze usoudit, že navrhovaný plugin využije pouze určitou část uložených dat. Z tohoto důvodu je vhodné, aby si plugin řešil ukládání dat ve vlastní režii. Vznikne tak prostor i pro uložení vypočtených hodnot závislých na původních datech. Neboť má práce sloužit i jako opora návrháře, nabízí zástupce různých typů databází potencionálně vhodných pro ukládání dat časových řad a jejich anomálií a porovnání základních vlastností.

Uchování dat nám otevírá prostor pro jejich další zpracování. Vhodným doplňkem je vizualizace časových řad i detekovaných anomálií. Práce umožňuje návrháři výběr z několika vizualizačních knihoven, kde srovnáním jejich vlastností poskytuje vodítka k výběru knihovny odpovídající požadavkům specifikace.

Zamyslíme-li se nad detekcí anomálií, jistě nalezneme mnoho způsobů, jak k problému přistupovat. Jedním z nich může být pro tuto práci zvolený způsob detekce anomálií v závislosti na změně entropie. Rozmanitost či podobnost IP adres nebo portů obsažených v síťových tocích nás může upozornit na potencionální anomálii, která představuje např. DoS útok či skenování. Entropie uvedených statistik je v práci odhadována pomocí komprese. Právě proto je zmíněno několik kompresních algoritmů a jejich porovnání, které nám pomůže vybrat adekvátní algoritmus vhodný pro naše požadavky.

V okamžiku kdy vybereme všechny komponenty v návrhu pluginu, můžeme přejít k vlastní implementaci. Práce obhajuje zvolené komponenty a představuje jednotlivé funkce částí pluginu – tedy backendu a frontendu. Správnost implementace a schopnost reagovat na skutečný provoz jsou následně ověřeny funkčním a výkonnostním testováním na kolektoru síťovým toků Masarykovy univerzity.

1.1 Síťové toky

Síťový tok (NetFlow) je definován sedmicí údajů, a to konkrétně zdrojovou a cílovou IP adresou a portem, obsaženým protokolem, rozhraním a TOS [1]. Ačkoli by tyto informace mohly dostačovat, návrháři připojili ještě další, jako například počet přenesených bajtů a paketů, dobu trvání a jiné. NetFlow obsahuje data o komunikaci pouze v jednom směru. Jinými slovy, NetFlow nám poskytuje mnoho užitečných informací o tom kde, kdo, co, kdy a jak dlouho provozoval na síti.

Nyní jsme si ve stručnosti nastínili, co to vlastně síťový tok je, a proto zbývá ještě zmínit několik faktů o tom, jak je s ním nakládáno. Informace o vzniku toku je zaznamenána do cache paměti. Máme tedy povědomí o aktivních tocích. Každý záznam v cache paměti časem expiruje, a dojde k jeho exportu. Mezi důvody k expiraci patří dlouho neaktivní tok (inactive timeout, obvykle 30 vteřin), toky, které překročí stanovenou hranici (active timeout, např. 30 minut, v reálném nasazení se obvykle nastavuje 5 minut), regulérní ukončení toku (příznaky FIN, případně RST protokolu TCP) či přeplnění cache.

1. type of service; pole nacházející se v IPv4 hlavičce paketu, původně označující prioritu, se dnes používá pro protokol DiffServ a ochranu proti zahlcení ECN

Více se lze dozvědět pročtením relevantních RFC dokumentů[2]:

- RFC 2720 – Traffic Flow Measurement: Meter MIB,
- RFC 3334 – Policy-Based Accounting,
- RFC 3917 – Requirements for IP Flow Information Export (IPFIX),
- RFC 3954 – Cisco Systems NetFlow Services Export Version 9,
- RFC 3955 – Candidate Protocols for IP Flow Information Export (IPFIX).

1.1.1 Vývoj NetFlow protokolu

Technologii NetFlow (resp. standard) vymyslela společnost Cisco, jako součást svého operačního systému pro síťová zařízení (IOS). Má za úkol ulehčit směrovačům při jejich hlavní činnosti – směrování, namísto zpracovávání informací o tocích a jejich exportu. Protokol NetFlow se od svého vzniku dočkal několika verzí. Ačkoli současné označení verze je NetFlow v9, stále se používají i verze NetFlow v5 a NetFlow v8.

NetFlow v5 rozšiřuje původní specifikaci verze 1 o informaci o zdrojovém a cílovém autonomním systému (AS), a také o sekvenční číslování exportovaných záznamů o tocích (pro případ výpadku, neboť samotný protokol UDP toto nezajišťuje). **NetFlow v8** definuje sumarizaci toků před jejich exportem, a tím umožňuje šetřit přenosové pásmo. Zatím poslední verzí společnosti Cisco je **NetFlow v9**, který přidává podporu pro šablony, a tak dovoluje budoucí přidávání polí v exportním paketu bez změny jeho formátu.

1.1.2 IPFIX

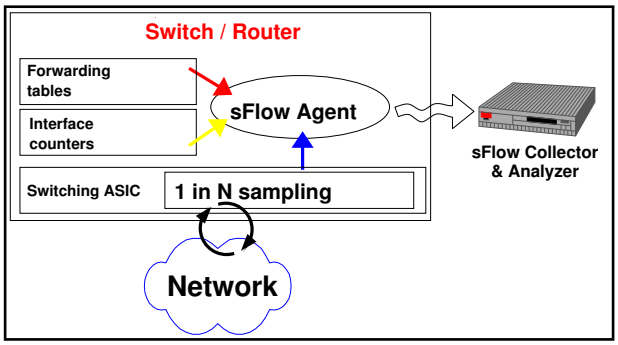
IPFIX je standardem skupiny IETF pro export informací o IP tocích a jejich formátování. Tento standard vychází z výše uvedeného NetFlow v9 a jeho formátu exportního paketu, pouze byla upravena hlavička, která je nyní následovaná libovolnou kombinací tzv. *FlowSet* (Template FlowSet, Options Template FlowSet a Data FlowSet). Hlavička je rozšířena o možnost specifikovat vlastní informace. IPFIX se spojuje s protokolem PSAMP, který zajišťuje sbírání vzorků paketů. IPFIX se také chová jinak z pohledu využití časových známek (timestamps), kde v hlavičce vynechává pole *sysUpTime*, které se u NetFlow v9 používá pro přepočítávání časových značek (může zvyšovat zatížení kolektoru).

1.1.3 sFlow

SFlow [75] je jednou z dalších alternativ k NetFlow. Jedná se o technologii vzorkování toků podporovanou mnoha výrobci síťových prvků. Prvky sFlow lze rozdělit na dvě skupiny a to na sFlow agenty a sFlow kolektor (plní také roli analyzátoru). SFlow agent můžeme nalézt jako součást ovladačů síťových rozhraní v přepínačích a směrovačích. Role kolektoru bude povětšinou přiřazena dedikovanému počítači či serveru. Samotné zpracování probíhá na všech rozhraních zároveň a není příliš výpočetně náročné, tedy nesnižuje významně propustnost sítě.

2. Libovolný Request For Comments dokument lze nalézt na adrese https://tools.ietf.org/html/.

SFlow agent komunikuje s ASIC (Application-Service Integrated Circuit neboli obvod naprogramovaný pro specifickou činnost. Jeho struktura se obvykle popisuje jazykem VHDL). Tento obvod poskytuje sFlow agentu počítadla paketů, vzorků, informaci o vstupních a výstupních rozhraních a také každý n-tý vzorek hlavičky paketu. Dodejme, že se zaznamenávají i informace o přeposílání/směrování paketů. Jak ukazuje obrázek 1.1, sFlow agent tyto informace přetvoří do podoby sFlow datagramu a odešle je přes síť sFlow kolektoru, který se postará o jejich zpracování.



Obrázek 1.1: sFlow[75]

Jednou z hlavních výhod této technologie je schopnost zpracovávat data na síťových rozhraních při rychlostech teoreticky se blížících jejich fyzickým možnostem (neboli sFlow je prakticky využitelné i na sítích, kde rychlosti přenosu přesahují desítky gigabajtů za vteřinu). Je třeba podotknout, že sFlow je určeno pro sítě s přepínáním a směrováním.

Mezi další vlastnosti sFlow, které můžeme zmínit, patří *přesnost* (díky vzorkování se nemusí zaobírat obrovským množstvím informací), *detailnost* (poskytuje informace o provozu na vrstvách L2–L7), *rozšiřitelnost* (možnost monitorovat tisíce síťových prvků s sFlow), *nízká cena* (jednoduchá implementace do síťových prvků, kterou podporuje většina výrobců až např. na firmu Cisco) a *rychlost* (rychlá odezva a získání informací o celé síti).

SFlow tedy poskytuje dodatečné informace tam, kde některé z ostatních dostupných technologií (např. NetFlow, RMON) nemají tyto vlastnosti implementované, případně to není ani principiálně možné. Připomeňme ku příkladu další protokoly třetí vrstvy ISO OSI modelu, jako jsou Appletalk či IPX anebo informace o VLAN a prioritách.

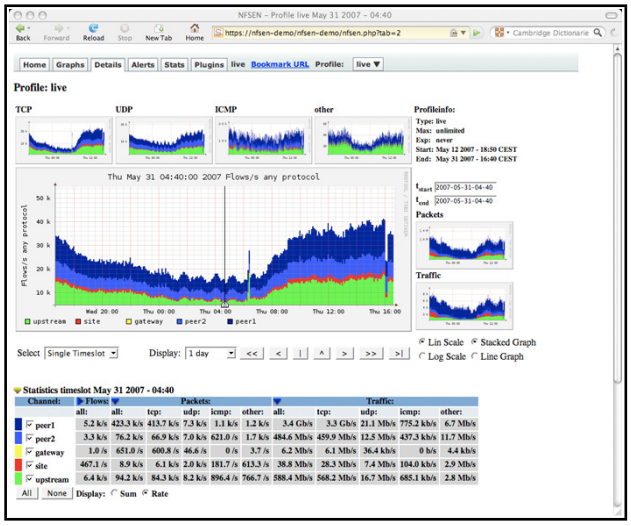
1.2 NfSen, nfdump

1.2.1 Nfdump

Nfdump je balíkem několika programů [43], které jsou schopny pracovat s NetFlow toky (v5, v7 a v9). Umožňují sběr toků a jejich zpracovaní z prostředí příkazové řádky. Jedním z nich je *nfcapd*, který je zodpovědný za ukládání toků a jejich rotaci po zvoleném časovém intervalu. Pro dotazování nad uloženými daty slouží nástroj jménem *nfdump*, jehož možnosti filtrování toků jsou velmi podobné nástroji *tcpdump* postaveného nad knihovnou *libpcap*. S jeho pomocí lze vytvářet i různorodé statistiky nebo seskupovat toky dle shodných parametrů. Nástroj je také schopný spojovat data z různých zdrojů (tedy několik sond zásobujících NetFlow kolektor) a pro svůj běh nepoužívá oprávnění privilegovaného uživatele. Typický požadavek na nfdump by mohl vypadat např. takto: *„Najdi všechny toky TCP s cílovým portem 443 a vypiš deset strojů s největším objemem přenesených bajtů.“*

1.2.2 NfSen

Je jistě vhodné použít nfdump pro zjištění informací o konkrétní události. Jeho výstup však není příliš přátelský, a také neposkytuje dlouhodobé statistiky, trendy a grafy (nicméně to co dělá, dělá správně, viz unixová filozofie malých nástrojů pro specifické funkce). Z tohoto důvodu vznikla grafická webová nadstavba. Obrázek mnohdy řekne tisíc slov v jednom okamžiku, zde popíše vývoj na sledované síti.



Obrázek 1.2: NfSen[44]

NfSen [44], jehož prostředí ukazuje obrázek 1.2, je napsaný v jazycích Perl (backend) a PHP (frontend). Data, která používá pro zobrazení grafů, ukládá do cyklické databáze RRD (viz kapitola 3.1.1 na straně 20), jejíž funkci `RRD:Graph` (viz kapitola 4.1.1 na straně 29) využívá pro vykreslování grafů. Poskytuje denní, týdenní a měsíční pohled. Různé typy profilů jsou navzájem odlišeny filtrem profilu nebo časovým obdobím. NfSen taktéž umí upozorňovat na překročení předem definovaných mezních hodnot.

Za zmínku rovněž stojí skutečnost, že celý systém předpokládá oddělení backendu a frontendu a komunikace mezi nimi je zajištěna pomocí funkcí a soketu. Stejně jako přibývá toků získaných ze sond, i NfSen zpracovává periodicky nová data a stará se o jejich zobrazení. Nakonec se hodí podotknout, že NfSen podporuje rozšíření svojí funkcionality pomocí pluginů (zásuvných modulů), a právě tvorba pluginu je hlavním cílem této práce.

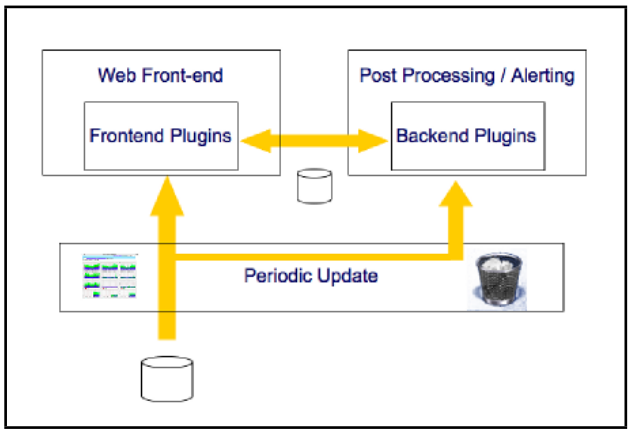
NfSen plugin a jeho rozhraní

Plugin umožňuje rozšířit funkcionalitu NfSen. Stejně jako NfSen má dvě části (znázorněné na obrázku 1.3), backend starající se o periodické zpracovaní dat dostupných v datovém úložišti a frontend poskytující prostor pro výstup do webového prohlížeče. Jak již bylo zmíněno, používá se kombinace skriptovacích jazyků Perl (pro backend) a PHP, XHTML, CSS a dalších jazyků běžně užívaných při vývoji webových aplikací (pro frontend).

Backend se spouští společně se samotným NfSen. Backend plugin je odpovědný za periodické zpracování dat poskytovaných sondami, a také za reakci na podmínky definovaných událostí (alerts), např. *„V případě, že počet spojení na port TCP/22 překročí hodnotu tisíc, zašli administrátorovi systému informační email.“*.

Backend pluginu musí obsahovat některé parametry. Proceduru *Init*, která slouží k zavedení globálních proměnných, jejíž návratová hodnota určuje, zda je plugin aktivní či nikoliv (viz výše), název pluginu (package název), určení verze pluginu. Mezi další, obvykle vyskytující se procedury můžeme zařadit *run* (odpovídá za periodické zpracování dat), *RunProc* (slouží pro komunikaci s frontendem), *Cleanup* (pro úklid po běhu pluginu) a *alert_condition*, *alert_action*, jejichž funkce je zjevná z názvu. Podotkněme, že parametry lze předat pluginu i konfigurací v souboru nfsen.conf – v hashi *PluginConf*.

Při změně backendu pluginu je třeba donutit NfSen k jeho znovunačtení, což je možné pomocí volání init skriptu *nfsen* s parametrem restart nebo reload. Pokud potřebujeme reagovat na události objevující se při běhu, máme tři možnosti. Zapsat danou informaci do syslogu (démon zajišťující správu logování v systému typu Unix), do souboru a v neposlední řadě můžeme informaci poslat přes standardní komunikační soket *nfsend-comm* frontendu.



Obrázek 1.3: Struktura pluginů v NfSen[44]

Pokud se zaměříme na frontend pluginu, zjistíme, že pro jeho umístění je použit jiný adresář (definovaný parametrem $FRONTEND_PLUGINDIR), nicméně jméno musí být shodné s backendem a přípona *php*. Frontend pluginu musí obsahovat minimálně dvě funkce, a to *<plugin_name>_ParseInput* a *<plugin_name>_Run*.

První zmiňovaná slouží pro vytvoření hlavičky HTTP protokolu (případně rozšířené o XML), zobrazení navigačního panelu (umožnujícího definici profilů, jejich přepínání, zobrazení základních statistik pro toky, bajty a pakety a správu alertů) a zpracování a validace dat získaných z formulářů, zatímco druhá se stará o zpracování dat získaných z backendu, případně o reakci na vstup uživatele.

1.3 Definice pojmů

Tato podkapitola slouží k získání vědomostí, jejichž znalost výrazně usnadní čtení dalšího textu. Jedná se o základní pojmy, se kterými se v této oblasti často setkáváme. Síťový provoz se odehrává v reálném čase, proto i data síťových toků tvoří soubor po sobě jdoucích pozorování – časové řady. Obecné informace o kompresi poslouží k pochopení uváděných algoritmů a díky kompresi získaná entropie přispěje k detekci anomálií.

1.3.1 Časová řada

Časová řada je soubor věcně a prostorově srovnatelných pozorování (dat), která jsou jednoznačně uspořádána z hlediska času ve směru minulost → přítomnost. Intervaly mezi jednotlivými datovými body jsou pevné. Analýzou dat v časových řadách se snažíme objevit

jejich statistické a charakteristické vlastnosti. Dle časových řad lze také s jistou úspěšností předpovědět vývoj situace v budoucnosti, neboť známe několik předchozích bodů.

Nejčastěji můžeme vidět časovou řadu znázorněnou pomocí bodového či spojnicového grafu, kde jsou hodnoty časové řady vynášeny na osu y a čas na osu x. Čas označujeme jako nezávislou veličinu, neboť jeho tok neumíme ovlivnit.

Časové řady můžeme rozdělit na diskrétní (jednotlivá pozorování) a spojité (např. srdeční tep). Časové řady se používají v mnoha oblastech, namátkou vyberme několik zástupců – ekonomika, meteorologie či informatika. V informatice nalezneme využití časových řad např. právě při periodickém odečítání hodnot síťových toků [10, 28].

1.3.2 Entropie

Nejdříve si zavedeme některé pojmy pravděpodobnosti. Definujme *jev jistý* s pravděpodobností $p = 1$ a *jev nemožný* s pravděpodobností $p = 0$. Všechny ostatní jevy, které mohou nastat, se tedy pohybují mezi jevem jistým a nemožným a jejich pravděpodobnost je tedy $0 < p < 1$. Nazvěme je *jevy možnými*. Zároveň platí, že pokud nelze jev vyjádřit jako sjednocení dvou jiných možných jevů, jedná se o *jev elementární*.

Jestliže následně vytvoříme úplný soubor S_N složený z N jevů s pravděpodobností p_i pro $i = \{1, \ldots, N\}$, pro který bude platit, že vždy nastane pouze jeden z nich a suma všech pravděpodobností p_i bude rovna jedné, budeme schopni definovat *entropii H(p)*, kde vztah $\mathbf{p} \equiv \{p_i\}_1^N$ vyjadřuje naši nejistotu o konkrétním zvoleném jevu z výše uvedeného úplného souboru. Entropie je funkce symetrická, monotónní, spojitá a aditivní [19].

Pojem entropie vznikl v termodynamice, a přestože existuje spojení mezi termodynamikou a informační entropií, my se budeme věnovat pouze druhé jmenované. Entropii lze popsat jako nejistotu o výsledku aktivity, která dovoluje více možných výsledků, jako např. hod mincí či kostkou. Informační entropie se také někdy nazývá *Shannonovou entropií*, neboť její vlastnosti formuloval C. E. Shannon. Shannonova entropie (viz vzorec 1.4) pracuje s *náhodnou veličinou*, což je proměnná, jejíž hodnota je jednoznačně určena až výsledkem náhodného pokusu.

$$H = -\sum_{i=1}^{n} P_i \, log_b \, P_i$$

Vzorec 1.4: Shannonova entropie pro diskrétní hodnoty

Shannonova entropie [18, 31] určuje střední hodnotu informace na jeden symbol zprávy. A mimo jiné, určuje i absolutní limit pro nejlepší dosažitelnou bezztrátovou kompresi libovolné komunikace. Jednotkou entropie je 1 bit, který např. u hodu mincí označuje samotný hod a jeho neurčitost výsledku, nikoliv však hodnotu „hlava“ nebo „orel“. V případě že entropii vyjadřujeme v bitech, je ve výše uvedené rovnici použita proměnná b s hodnotou 2. Další typické hodnoty proměnné b jsou Eulerovo číslo e či 10.

1.3.3 Komprese dat

Komprese je technika umožnující zmenšení velikosti zprávy, v důsledku čehož dojde k úspoře zdrojů, jako jsou sekundární a terciální paměti či šířka přenosového pásma. Tohoto zmen-

šení se docílí pomocí vynechání redundantních informací obsažených ve zprávě. Opačný postup se nazývá dekomprese. Po dekompresi nemusí být získaná zpráva shodná se zprávou originální v důsledku použitého typu komprese.

Existují dva typy komprese. Prvním je *komprese bezztrátová*, která po dekompresi poskytne zprávu shodnou s originálem. Je tedy vhodná v oblastech kryptografie či komprese textů, kde nám záleží na kompletní a neporušené zprávě. Druhou možností je využití *komprese ztrátové*, kde jsou některé části zprávy ztraceny a není je možné zpětně obnovit.

Ztrátová komprese se používá v oblastech, jako jsou obraz, zvuk či video. Informace, o které jsme ochuzeni, nám neschází z důvodu nedokonalosti lidských smyslů, jako jsou zrak či sluch anebo nám záleží hlavně na velikosti výsledné zprávy. Kompresní poměr, což je podíl velikostí zkomprimované a původní zprávy, je výhodnější při použití ztrátové komprese. Ztrátová komprese je často založena na netriviálních matematických algoritmech, kterými se zde nebudeme zabývat. Zástupci této kategorie jsou např. *diskrétní kosinová transformace*, *vektorová kvantifikace* či *A-law*.

Bezztrátová komprese dále nabízí jemnější dělení. Mezi používané metody patří především RLE, slovníkové metody, Burrows-Wheelerova transformace a metody založené na samotné entropii. **RLE** bývá často používána jako výukový příklad pro svou jednoduchost. Jde totiž o metodu, při které jsou stejné hodnoty v datovém proudu reprezentovány číselnou hodnotou počtu opakování a samotným znakem (*např. aaaakkkww* → *4a3k2w*). Je zjevné, že v případě neopakujících se znaků komprese až zdvojnásobí původní velikost zprávy.

Bezztrátové kompresní algoritmy neposkytují příliš dobrý kompresní poměr, pokud jsou zdrojem náhodná data. Příkladem kompresních algoritmů tohoto typu jsou dále uvedené deflate, LZMA, LZO a Burrows-Wheelerova transformace.

Pro účely kryptografie se často při šifrování využívá skutečnosti, že celková entropie je *téměř* stejná jako u originální zprávy, nicméně díky kompresi má zpráva méně bitů. Z toho plyne, že zpráva je více nepředvídatelná. Je tedy výhodné zprávu před samotným šifrováním zkomprimovat.

Pokud bychom se zajímali o vztah komprese a entropie, vycházejme z faktu, že čím více náhodná je vstupní sekvence, tím je vyšší entropie. Pokud obsah konečné vstupní zprávy převedeme do binárního formátu a zprávu následně zkomprimujeme, velikost zkomprimované zprávy odpovídá (v případě ideální komprese) entropii původní zprávy, protože bychom se zbavili veškeré nadbytečné informace. Ideálně tedy bude mít komprimovaná zpráva entropii 1 bit entropie na 1 bit komprimované zprávy. Entropii komprimované zprávy tedy dostaneme jako podíl velikosti původní zprávy ku velikosti komprimované zprávy (viz vzorec 1.5). Nicméně neboť současné kompresní algoritmy neposkytují perfektní kompresi dostaneme drobný rozdíl mezi entropiemi, který však pro naše účely můžeme zanedbat.

$$H_{komprese}(x) = \frac{velikost(x)}{velikost(komprese(x))}\,[bit/bit]$$

Vzorec 1.5: Entropie komprimované zprávy konečné délky

Pro účely porovnání zjistíme originální entropii zprávy dosazením hodnot do výše uvedené rovnice, kde jednotlivé pravděpodobnosti pro symbol vzniknou jako podíl frekvence

výskytu symbolu *f(x)* ku jeho velikosti *l(x)* (viz vzorec 1.6). V našem případě jsou IP adresy reprezentovány 32 bity (IPv4) či 128 bity (IPv6) a porty 16 bity. Hodnoty entropie komprimované zprávy a entropie původní zprávy by měly být srovnatelné.

$$p(i) = \frac{f(x)}{l(x)} \qquad H(x) = -\sum_{i=1}^{n} p_i \, log_2 \, p_i \, [bit/symbol]$$

Vzorec 1.6: Entropie původní zprávy konečné délky

1.4 Obsah práce

Nastiňme si obsah jednotlivých kapitol pro získání lepší představy o tom, jak je tato práce koncipována. První kapitola, kterou právě pročítáte, nás seznamuje se síťovým tokem a formáty pro práci s ním. Dále je představena grafická aplikace NfSen a možnosti jejího rozšíření a také klient pro textovou konzoli nfdump. Ke konci první kapitoly jsou definovány některé základní pojmy potřebné dále v textu.

Druhá kapitola, která nese název *Detekce anomálií*, vysvětluje některé obvyklé techniky útoků na počítačové sítě a jejich infrastrukturu. Dále je představeno několik komerčních i open-source nástrojů, které se detekcí anomálií a vůbec dohledem nad sítí zabývají.

Další, totiž třetí kapitola představuje aplikace pro ukládání dat. Nosným tématem této kapitoly jsou databáze, které poskytují binární formát a pokročilé funkce pro správu a práci s daty. Kapitola nás seznamuje se zástupci aplikačních, relačních, cyklických a dokumentových databází. U každého produktu jsou popsány jeho vlastnosti a typické situace z praxe, ve kterých bývá nasazen.

Čtvrtá kapitola definuje pojem vizualizace a představuje možnosti grafické reprezentace včetně zmínění vyšších dimenzí. Následně se čtenář dozvídá o vizualizačních pluginech a o tom, co mohou nabídnout. Po spíše technickém popisu jednotlivých pluginů, kde jsou k dispozici ukázkové konfigurace a jim odpovídající grafy, přichází na řadu srovnání, které se snaží porovnat produkty z mnoha pohledů. V závěru kapitoly je vybrán konkretní produkt a tato volba je diskutována.

Pátá kapitola se věnuje entropii a kompresi, neboť tento způsob byl vybrán pro detekci anomálií portů a IP adres v grafech. V této kapitole je představeno několik algoritmů a jejich principy. Ke každému z nich je vybrána jedna implementace. Následuje několik srovnávacích testů, které prověří implementace algoritmů při účinnosti komprese, době strávené kompresí či dekompresí, a také při využití výpočetních zdrojů. V závěru kapitoly se objeví doporučení pro volbu konkretního algoritmu a jeho implementace v závislosti na požadavcích.

Vytvořením pluginu pro aplikaci NfSen se zaobíráme v kapitole šesté, kde se postupně budeme věnovat jeho jednotlivým částem nejen z pozice návrháře, ale i v roli programátora. Dále se dozvíme o testování pluginu v rámci Masarykovy univerzity a jeho výsledcích.

Závěr hodnotí přínos práce a ukazuje čeho jsme dosáhli.

Kapitola 2

Detekce anomálií

Při automatickém sběru dat o síťových tocích pomocí NetFlow sond, směrovačů či přepínačů, a také dalších dat dostupných ze systémů prevence a detekce, získáváme obrovské množství informací. Tyto informace nám mají sloužit k tomu, abychom správně vyhodnotili potřeby a výkon sítě a doručili služby jejím legitimním uživatelům.

Pro zpřehlednění můžeme data vizualizovat (viz kapitola 4 na straně 27) pomocí grafů či organizovat do tabulek, nicméně není v silách člověka ručně procházet všechna dostupná data a hledat odchylky od normálního provozu. Pro tento účel existují statistické nástroje i empirické postupy s jejichž pomocí lze tyto anomálie objevit.

Jejich přítomnost obvykle naznačuje problémy. Ať už se jedná o již nedostačující výkon a kapacitu sítě nebo problém způsobený špatným nastavením či porouchaným zařízením anebo dokonce o počítačový útok. Vždy je třeba adekvátně reagovat, aby se daná situace co nejrychleji vyřešila, ideálně bez finančních ztrát a povšimnutí uživatelů.

V této práci je detekce anomálií prováděna analýzou dat časové řady tak, že se hledají hodnoty převyšující určitý průměr v omezené lokální oblasti. Tyto hodnoty jsou pak považovány za anomálie, které jsou při vizualizaci dat časové řady zaznamenaných síťových toků explicitně označeny jako místa potenciálních útoků či problémů v síti. O technickém řešení a typech útoků na které dokáže plugin reagovat, se dozvíte v kapitole 6 na straně 57.

2.1 Útoky na počítačovou síť a snaha o její zneužití

Počítačové sítě – jejich infrastruktura, zdroje i aplikace jsou častým cílem útočníků za účelem získání důvěrné informace obchodního významu, využití drahých výpočetních zdrojů či dokonce vyřazení služby nebo zařízení z provozu. Za tímto účelem vzniká mnoho škodlivého software a různých technik útoků, které si přiblížíme v následujících odstavcích. Jedině znalost principů a myšlení útočníků může pomoci při zabezpečení sítě a jejích zdrojů.

2.1.1 Rozdělení anomálií

Provozní anomálie

Provoz na síti má obvykle nějaké standardní chování. Poruchou zařízení, jeho špatným nastavením či v důsledku počítačového útoku může dojít v obvyklém chování ke změně, která ovlivňuje základní síťové parametry, jako jsou rozptyl, zpoždění a odezva. Na vině může

být rozsáhlá softwarová aktualizace (objem dat), chyba ve směrování, špatné či chybějící záznamy v DNS.

Bezpečnostní anomálie

Jedná se o činnosti nebo události, které zapříčiňují snížení či prolomení bezpečnosti aplikace, komunikace nebo zařízení. Do této kategorie lze zařadit nakaženou stanici šířící spam nebo viry. Software téměř vždy obsahuje chyby a proces jejich odhalování a záplatování sledují nejen administrátoři a vývojáři, ale také samotní útočníci. Je tedy zásadní udržovat systém aktuální.

2.1.2 Metody detekce anomálií

Porovnávání signatur je metoda využívaná v nástrojích pro detekci průniků, které označujeme jako NIDS (network intrusion detection system). Tyto signatury jsou vlastně popisem specifického chování pozorovatelného během konkrétního známého útoku. V případě shody může NIDS upravit pravidla firewallu či změnit QoS. Metoda se používá při zkoumání síťového provozu na úrovni paketů i při sledovaní toků. Její hlavní nevýhodou je neschopnost reakce na neznámé podněty a potřeba aktuálnosti databáze signatur.

Stavová analýza spoléhá na přesné definice protokolů provozovaných na síti, definovaných pomocí stavů v dokumentech RFC či jiných standardech. Činnost protokolu je tedy daná a přechody mezi jednotlivými stavy řídí stavový automat. Jestliže se tedy provoz na síti nechová dle stavu definovaného v protokolu, lze tento stav považovat za anomálii.

Behaviorální analýza je závislá na vytvoření vzoru provozu na sítí (tzv. baseline) a následném porovnávání provozu s takto vytvořeným vzorem v pravidelných intervalech. Hodnoty, které statisticky vybočují z hranic definovaných vzorem, jsou považovány za anomálie. Bohužel tato metoda může vyvolat falešný poplach. Jestliže upravujeme vlastnosti vzoru učením, můžeme se naučit i samotnou anomálii. Právě tuto metodou plugin využívá k detekci anomálií.

2.1.3 Typy anomálií

Následující odstavce představují několik obvyklých hrozeb, kterým běžná větší síť čelí velmi často. Tento výběr uvažuje pouze techniky útoků, jež lze detekovat za pomoci síťových toků [26]. Jedná se o metody útoků, při kterých je útočník mimo naši síť a nemá tedy přístup ke stroji ani softwarovou cestou ani fyzicky. Záleží tedy na službách, které stroj poskytuje, a na jejich konfiguraci i zabezpečení. Některé metody slouží k průzkumu a získání informací o oběti, jiné k ovládání a další si kladou za cíl přerušit poskytování služeb. Tyto druhy anomálií by měl uvažovaný plugin pro NfSen odhalit.

Botnet

Skupina napadených a ovládaných počítačů bývá označována pojmem botnet. K ovládání je obvykle použita komunikace s IRC nebo webovým serverem. Uživatelé napadených strojů

nemají často ani zdání o podezřelých aktivitách probíhajících uvnitř počítače, dokud se neobjeví v seznamu šiřitelů nevyžádané pošty či nejsou upozorněni svým poskytovatelem připojení. Botnety jsou tedy často využívány pro šíření nevyžádané pošty, pro DDoS útoky či pro svou výpočetní sílu pro luštění hesel a sběr citlivých informací. Takovéto sítě mnohdy dosahují až desítky miliónů napadených strojů. Dostupná výpočetní síla, úložiště a množství IP adres se také pronajímá zájemcům pro jejich vlastní škodlivé aktivity. Botnet nemusí mít centrálního správce, nýbrž celou hierarchii lidí, kteří ho ovládají. Topologie takové sítě je hvězda, větší množství ovládacích serverů, hierarchická či dokonce náhodná.

Škodlivý software

Tato kategorie anglicky též nazývaná *malware* zužuje počítače velmi často. I zde je několik kategorií, kam můžeme zařadit viry, červy, trojské koně a roboty. Z jeho pomocí může útočník poškodit, ale pravděpodobněji získat data z infiltrovaného stroje. V poslední době útočníci využívají malware k získání dalších výpočetních zdrojů pro podporu své činnosti, ať už se jedná o vykonání DDoS útoků, hádání hesel či rozesílání spamu.

Skenování portů

Existují dva typy skenování portů v závislosti na požadovaných informacích. První se nazývá *horizontální skenování* a je zpravidla zaměřen na síť. Stroje v ní umístěné jsou testovány, zda-li jsou náchylné pro určitou zranitelnost, případně se zjišťují IP adresy běžících strojů. Obvykle se jedná o malý počet portů testovaný na skupině strojů či síti. *Vertikální skenování* je naopak zaměřeno na konkrétní stroj. Testuje dostupné porty, aby zmapoval služby běžící na daném stroji. Většinou probíhá za pomoci protokolů TCP či UDP umístěných v transportní vrstvě ISO OSI.

DoS a DDoS

DoS, což je zkratka pro Denial of Service (česky odepření služby), je druh síťového útoku, při kterém útočník svojí aktivitou vytíží případně obsadí kapacitu poskytované služby či zdroje tak, že legitimní uživatelé nemohou k službě přistoupit nebo ji použít. DDoS je pak tentýž druh útoku, nicméně útočníků je daleko větší počet a jsou rozmístěni po síti. Většinou útočníci používají kompromitované stroje zformované do sítě, tzv. botnetu. Útoky DoS i DDoS jsou velkou hrozbou současného internetu. Jejich činnost lze v podstatě rozdělit do dvou kategorií – na objemové útoky a na útoky na aplikační vrstvu.

Mezi prvně jmenované patří např. záplava paketů neboli snaha o zahlcení přenosového pásma a síťové infrastruktury. Typicky se jedná o záplavy ICMP, UDP či TCP SYN paketů. V případě UDP útočník navazuje spojení s náhodnými porty na cílovém stroji, ten je nucen zjistit, zda na daném portu poslouchá nějaká aplikace a v momentě kdy zjistí, že tomu tak není, odesílá zprávu protokolu ICMP *Destination Unreachable*. Stroj je tak vytížen rozesíláním ICMP zpráv, že přestává reagovat na legitimní dotazy. Obvykle je též zdrojová adresa falešná, aby pakety nezahlcovaly útočníka. V rámci grafu by jistě bylo zajímavé sledovat UDP provoz ve vztahu k ICMP zprávám o nedosažitelnosti.

Pokud se jedná o TCP útok, využívá se polootevřených spojení se serverem k vyčerpání zdrojů. Standardně je spojení ustanoveno pomocí třícestné výměny zpráv neboli TCP paketů s nastavenými příznaky v tomto pořadí SYN, SYN+ACK a ACK. Útočník nezašle potvrzující TCP ACK a vznikne polootevřené spojení. Server je ovšem schopen obsluhovat pouze určitý počet spojení, a tedy po určité době začne odmítat legitimní požadavky.

DoS zaměřený na protokol ICMP má hned dvě varianty – **Smurf** a ping záplava. Smurf útok je založený na zaslání ICMP echo požadavku na všesměrovou adresu sítě se zdrojovou IP adresou oběti. Všichni hosté na sítí použijí ICMP echo odpověď, a tak zahltí oběť. **Flood ping**, což je název dalšího útoku založeného na ICMP, zahlcuje oběť ICMP echo požadavky. Podmínkou je ovšem mít větší šířku pásma než oběť.

Dalším typem DoS útoků, jsou ty, které útočí na aplikační vrstvu. Využívají menší část síťové infrastruktury, je tedy obtížnější je detekovat. Postupně vyčerpají obslužnou kapacitu dané služby, např. HTTP či DNS, a tím ji vyřadí z provozu. Do této kategorie můžeme zařadit útok nazývaný **ping smrti** (anglicky ping-of-death), což je obvykle fragmentovaný IP paket s ICMP protokolem, jehož velikost je větší než 65 535 B. Při opětovném sestavení v cíli se obvykle stroj při snaze o potvrzení zhroutí a tím způsobí DoS.

Speciálním případem jsou pomalé DoS útoky, které udržují velké množství spojení po maximální možnou dobu. Tento druh je špatně detekovatelný a příkladem může být např. Slowloris [24] útok.

2.2 Nástroje pro detekci anomálií a práci ze síťovými toky

Jak již bylo zmíněno výše, primárním cílem osob odpovědných za síťovou infrastrukturu je plynulý chod a vysoká kvalita služeb dodávaných uživatelům. Detekce anomálií postavená na analýze síťových toků je poměrně nové odvětví počítačové bezpečnosti, a proto ani nástrojů v této oblasti není mnoho. Bylo tedy vybráno několik relevantních IDS nástrojů z komerční i open-source sféry, které umí zpracovat síťové toky z NetFlow sond, analyzovat je a vizualizovat podobně jako systém NfSen a jeho pluginy. Získat detaily o komerčních produktech je takřka nemožné [26], proto uvádíme pouze základní informace o dvou zástupcích této kategorie. Zbývající produkty jsou postaveny na open-source. Podkapitola se zaobírá vlastnostmi vybraných nástrojů, jako jsou typy rozpoznatelných anomálií, modulárnost, možnosti vizualizace a hlášení incidentů a v neposlední řadě nároky na systém.

Některé z nich pracují kromě síťových toků i s pakety či signaturami, a tak spojují oblast detekčních systémů postavených na síťových tocích a NIDS. Data jsou tedy kumulována z různých zdrojů, což umožňuje identifikovat vazby v rámci jednotlivých útoků. Tyto aplikace umožňují kontinuálně sledovat dění na síti, od běžných provozních charakteristik, přes účtování služeb až po detekci a reakci na počítačové útoky.

2.2.1 Arbor Networks PeakFlow

Arbor PeakFlow SP [1] je komerční soubor nástrojů postavený na proprietárním ArbOS/ArbUX s podporou NetFlow, sFlow, cflowd, jFlow, IPFIX a Netstream, který slouží k detekci, analýze a minimalizaci hrozeb síťových útoků. Za tímto účelem používá in-

formace získané z IP toků, směrování, a také z aplikační vrstvy pomocí NIDS (network intrusion detection system). Jeho architektura je modulární, což umožňuje lépe rozložit zátěž, pokrýt větší oblast sítě senzory a reagovat na DoS/DDoS útoky. Kromě senzorů obsahuje také kolektor a webové rozhraní správy. Webové rozhraní umožňuje pohled do statistik reálného i historického provozu, stejně jako dokáže informovat o anomáliích způsobených DDoS útoky, změnách ve směrování či aplikacích zabírajícími enormní kapacitu přenosového pásma. Ve stejném rozhraní jsou vedeny i informace o tranzitních partnerech, zákaznících, směrovačích a aplikacích.

Arbor PeakFlow SP poskytuje nástroj pro potírání DDoS útoků a dalších hrozeb, který je bezstavový a sleduje jednotlivé síťové vrstvy. Umožňuje tak odstranit pouze provoz samotného DoS/DDoS, nikoliv provoz celé postižené sítě. Dále sleduje výkonnost sítě pomocí síťových charakteristik, jako jsou zpoždění, rozptyl, ztrátovost paketů či odezvu (round trip time). V neposlední řadě monitoruje důležité služby a dovoluje BGP blackhole routing či BGP flow-spec. Systém je postaven na sdílení otisků útoku pomocí služby **ATF** (Active Threat Feed), jejíž obsah vytváří členové asociace **FSA** (Fingerprint Sharing Aliance).

2.2.2 Caligare Flow Inspector

Caligare Flow Inspector [3] je nejen komerční softwarový nástroj podporující zařízení schopné exportovat síťové toky od firem Cisco Systems, 3COM, Juniper, Extreme Networks, případně softwarové sondy. Mezi je funkce patří inteligentní analýza síťových toků, sledování toků v reálném čase, inteligentní multikriteriální filtrování, agregace a statistické informace. K dispozici je i zkušební verze a volně dostupná verze pro nekomerční využití, která je limitována na maximálně dvě zařízení, objem do 250 toků/s, jednodenní historii a absencí detekce síťových anomálií. Komerční verze umí detekovat skenování portů, DoS útoky postavené na záplavě paketů, problémy ve směrování, případně provoz počítačových her nebo peer-to-peer aplikace. Pro svůj běh požaduje linuxové prostředí s webovým serverem podporujícím dynamické stránky a databází.

Flow Inspector komunikuje s obsluhou pomocí webového rozhraní, ve kterém lze kromě nastavení kolektoru sledovat dění ze současnosti i minulosti pomocí vizualizace trendů grafy. Systém je schopen poskytnout detailnější informace na úrovni autonomního systému, IP adresy či rozhraní. Umožňuje definování skupin a uživatelských práv, sítí a profilů. Většinu informací lze z prostředí exportovat, a to buď ve formě `CSV` souboru, nebo jako NetFlow tok pro zpracovaní na jiném kolektoru.

2.2.3 IBM AURORA, IBM Tivoli Netcool Performance Manager

Tivoli Netcool Performance Manager [14] je komerční aplikace založená na výzkumném projektu **AURORA** [13]. Jedná se o aplikaci zabývající se analýzou a vizualizací síťových toků pro velké drátové, bezdrátové a konvergované sítě, detekcí anomálií a virů, distribuovaným zpracováním a ukládáním síťových toků či sledováním dynamického směrování. Podporuje formáty NetFlow a IPFIX, ze kterých umí vyextrahovat detailní statistiky o komunikaci a využití protokolů.

Aplikace je vytvořena v jazycích Java a Jython (Python v Javě) a obsahuje dvě komponenty – pro drátové a pro bezdrátové sítě. Pro uložení dat je použita databáze Oracle, pro adresářové služby LDAP. Aplikaci je možné provozovat na operačním systému Solaris, Redhat Enterprise Linux či AIX. Komponenta pro drátové sítě je modulární a zajišťuje sběr dat mj. ze SNMP, systémového logu či TCP soketu, jejich agregaci a detekci hraničních hodnot. Pro správu a reakci na anomálie je k dispozici GUI aplikace a webové rozhraní, které poskytuje tabulky a grafy. Bezdrátová komponenta poskytuje několik částí, mezi které patří databázový modul a moduly pro sběr a konverzi dat z bezdrátových sítí. Správa uživatelů využívá Tivoli Directory Server (LDAP). Volitelně mohou být informace vyprodukované oběma komponentami prezentovány také díky **Tivoli Common Reporting**.

2.2.4 OSU flow-tools

Flow-tools [40] je open-source knihovnou s podporou komprese **zlib** a souborem unixových nástrojů pro záznam, zpracování a analýzu síťových toků NetFlow verzí 1, 5, 6 a 8 získaných exportem ze zařízení firem Cisco, Juniper či sond. Flow-tools mohou být použity pro sledování výkonnosti sítě, účtování služeb, plánování síťových struktur a v neposlední řadě i detekci a analýzu bezpečnostních incidentů.

Data jsou ukládána komprimovaně do logů, případně posílána na standardní výstup. Je možné je dále replikovat či exportovat do formátů, jako jsou ASCII, **CSV**, cflowd, pcap, či databází MySQL a PgSQL. Data jsou v logu uložena v binárním formátu z důvodu spotřeby úložného prostoru, nicméně pro potřeby analýzy je možné je převést do čitelné formy. Analýza dat probíhá podobně jako s nástrojem tcpdump pomocí filtrů, stejně jako produkce statistik provozu. Ty mohou být vstupem pro Gnuplot či převedeny do RRD. Z bezpečnostních anomálií jsou tyto nástroje schopny odhalit např. DoS či skenování portů, neboť umí ohlásit enormní množství paketů v toku.

K původní pravděpodobně již nevyvíjené verzi vznikl fork stejného jména hostovaný na *code.google.com* (portál pro vývojáře), který se snaží pokračovat v práci původního autora. Obě verze jsou schopny spolupracovat s programem FlowScan [22], který může přidat vizualizaci grafy pomocí knihovny RRD.

Kapitola 3

Uchovávání získaných dat

V závislosti na velikosti a rychlosti sítě, a také na množství zařízení schopných sběru informací o síťových tocích, roste i objem dat k dispozici. Sebraná data je třeba zpracovat případně vizualizovat. Takové ojedinělé měření je však často neprůkazné a ničím nepřínosné. Potřebujeme získaná data zkoumat v závislosti na ostatních datech, abychom dokázali říct, co se změnilo a jakým směrem se provoz na síti vyvíjí, anebo porovnat data s jinými získanými včera, před měsícem či rokem. Za tímto účelem potřebujeme data uložit, což nám umožní pracovat s větším souborem dat, a tak poskytnout adekvátní výsledky.

Základním požadavkem na datové úložiště pluginu je efektivní organizace. Ta poskytuje metody jak data ukládat s minimální režií, vyhledávat, měnit a přistupovat k nim pokud možno co nejrychleji. Pro úložiště dat je většinou použita energeticky nezávislá paměť umožňující přístup na libovolné místo, např. pevný disk. Výkon pluginu přímo závisí na rychlosti operací s úložištěm dat. Zajímá nás tedy i otázka formátu uložených dat. Jistě by se nabízely čistě textové soubory, ovšem práce s nimi je sice snadná, ale neefektivní a pomalá kvůli potřebě sekvenčního čtení. Z tohoto důvodu nám více vyhovuje binární formát nabízený databázovými aplikacemi. Ty nám poskytnou nejen rychlost a téměř okamžitý přístup ke konkrétním datům, ale i další funkce. Ať už se jedná o chování ve víceuživatelském režimu, souběžný přístup, zálohování či přístupová práva pro uživatele a skupiny.

Právě z tohoto důvodu si nyní představíme několik databází a jejich vlastností, abychom si mohli vybrat tu, která bude nejvíce vyhovovat potřebám pluginu. V přehledu (viz tabulky 3.1 a 3.2) nalezneme známou cyklickou databázi RRDtool, dále se zmíníme o aplikačních databázích, neopomeneme představit komplikované, ale zároveň velmi propracované a výkonné relační databáze a nakonec zmíníme databáze dokumentové. Databáze lze klasifikovat i podle modelu. Kromě uvedeného relačního, se můžeme setkat i s hierarchickým, síťovým či objektově-orientovaným.

Název	Typ databáze	Verze	Četnost	Licence
RRDtool	cyklická	1.4.5	střední	GPL
BerkeleyDB	aplikační	5.1.19	střední	GPL kompatibilní
SQLite	aplikační	3.7.6.2	velká	Public domain
MySQL	relační	5.5.11	malá	GPL 2
PosgreSQL	relační	9.1.2	malá	PosgreSQL/BSD
MongoDB	dokumentová	2.0.1	velká	AGPL 3.0

Tabulka 3.1: Uchovávání dat – databáze (typ DB, verze, četnost vydávání a licence

Četnost: malá – 1-2x ročně, střední – několikrát ročně, velká – několikrát měsíčně

Název	API
RRDtool	C/C++, Lua, Erlang, Java, Perl, PHP, Python, Ruby, Tcl
BerkeleyDB	C/C++, Java, Perl, PHP, Python, Ruby, Smalltalk, Python, Ruby, Tcl, Smalltalk
SQLite	BASIC, C, C++, Clipper/Harbour, Common Lisp, Delphi, Free Pascal, Haskell, Java, Lua, Javascript, Objective-C, OCaml, Perl, PHP, Python, REBOL, R, REALbasic, Ruby, Scheme, Smalltalk, Tcl, ...
MySQL	C/C++, Eiffel, Ocaml, Perl, PHP, Python, Ruby, Tcl
PosgreSQL	C/C++, Java, Ocaml, Perl, PHP, Python, Ruby, Smalltalk
MongoDB	C/C++, D, Delphi, Erlang, Haskell, Lua, MatLab, Java, Javascript, .NET, Ocaml, Perl, PHP, Python, Ruby, Scala, Tcl, ...

Tabulka 3.2: Uchovávání dat – databáze (API)

3.1 Cyklické databáze

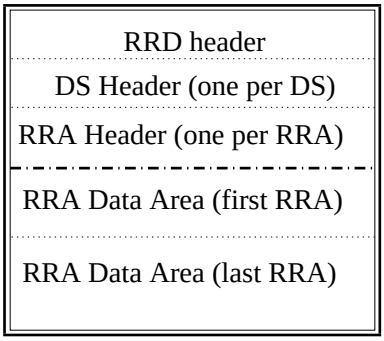
Jedná se o druh databáze, která pro uchování dat používá cyklický buffer. Tento buffer je datová struktura pevné velikosti, která používá dva ukazatele. První z ukazatelů označuje začátek dat a druhý konec. Pokud při ukládání hodnot dosáhneme hodnoty rovné velikosti databáze, oba ukazatelé se posunou o jedno místo (koncový se přesune na začátek bufferu). Tímto způsobem databáze zapomene nejstarší uloženou hodnotu. Asi nejznámějším představitelem je zde uvedená Round Robin Database.

3.1.1 Round Robin Database

RRD [63], jak se často Round Robin Database zkracuje, je průmyslový standard, zahrnující několik nástrojů, pro zpracovávání a uchovávání dat v časových řadách (viz kapitola 1.3.1 na straně 8) a jejich případnému vizuálnímu znázornění v podobě grafu. RRD je napsána v jazyce C, což je jistým vodítkem k efektivnosti a rychlosti. Celé řešení je založeno na cyklickém bufferu, což jinými slovy znamená, že soubor s příponou *rrd*, je schopen uchovat jen určité, předem specifikované, množství dat, než začne přepisovat nejstarší hodnoty.

RRD očekává data v pravidelných intervalech, jejichž vzdálenost je při vytváření databáze definována parametrem **step** a počátek zaznamenávání určíme parametrem **start**.

Nicméně to by zřejmě nestačilo, neboť potřebujeme definovat, jaká data budeme ukládat. To můžeme zařídit pomocí parametru **DS** (data source) pro každou sledovanou veličinu. DS přiřadíme typ dle předpokládaných dat. Konkrétně jeden z *gauge*, *counter*, *derive*, *compute* a *absolute*. Takto můžeme získat opravdu velké množství dat, které by mohlo být velmi náročné zpracovat.



Obrázek 3.1: Struktura RRD souboru[62]

Proto definujeme **CF** (consolidation function) jako jednu z MAX, MIN, AVG, last či total a uchováváme právě její výsledky. Data konsolidační funkce stejného typu se sdružují do archivů **RRA** (round robin archive). Pro tyto archivy nastavujeme za jaké období nás sbírané hodnoty zajímají, v jakém rozsahu se měřená veličina může nacházet, ale také co dělat, pokud data nepřicházejí v očekávaných časových intervalech (tzv. *heartbeat*) a kdy se má hodnota prohlásit za *neznámou (unknown)*.

Existují mnohá rozšíření pro různé programovací jazyky, ku příkladu Perl, PHP, Tcl, Python, Ruby, Lua a další, které tak usnadňují prací s databází RRD od vytvoření, přes

ukládání získaných hodnot až k jejich zpětnému dolování. Mimo jiné, je k dispozici i rozšíření umožnující práci v paralelním prostředí s vlákny. Pro zpracovaní dalšími nástroji poskytuje RRD také export do XML.

Při velkém množství RRD databází a přístupu k nim, lze pro zvýšení efektivity využít démona **rrdcached**, který ulehčí hard disku z pohledu přístupů pro zápis. Hlavička RRD zabírá v paměti 4 kB, stejně jako alespoň jedno aktivní RRA (viz schéma 3.1). Objem dat při aktualizaci RRD je osm bajtů na jedno DS. Při 4 kB velikosti bloku na disku, to může být poněkud neefektivní. Proto má smysl požadavky na zápis schraňovat a zapsat najednou (základní časový interval pro zapsaní do RRD je pět minut).

Ačkoli může tento démon velmi pomoci, je třeba mít na vědomí jistá úskalí. RRDcached komunikuje pomocí textového ASCII protokolu a nepodporuje ani šifrování ani autorizaci. Jedinou, i když poněkud chabou ochranou je kompilace s libwrap a následné použití *hosts_allow* a *hosts_deny* pro přístup jednotlivých klientů. Rozšíření tohoto druhu jsou plánována do dalších verzí.

RRD se často využívá pro uložení hodnot fyzikálních jevů, jako teplota, tlak, vlhkost, ale nejen v tomto oboru. Často můžeme potkat RRD skryto za počítačovými sítěmi, kde může fungovat jako počítadlo paketů či rámců, sledovat odezvy služeb nebo dostupnost a mnohé další. Cokoli, co produkuje měřitelné veličiny v pravidelných intervalech, může být zdrojem pro DS.

3.2 Embedded Database (aplikační databáze)

Embedded databáze jsou další možností, jak uchovávat data organizovaně a přistupovat k nim za pomoci dostupných funkcí či metod, které zachovávají integritu a zjednodušují programátorovi práci. Aplikace zpracovávají uživatelské vstupy a jejich výsledky, neboli výstup, je často potřeba nejen prezentovat, ale také uchovat. Ať už se jedná o vlastní nastavení aplikace, naměřené hodnoty či výsledky operací, jsou to obvykle data sloužící pouze dané aplikaci. Z tohoto důvodu je vhodné, aby se databáze nacházela na stejném stroji, není tedy potřeba stroj vyhrazený ani řízení přístupu a práv.

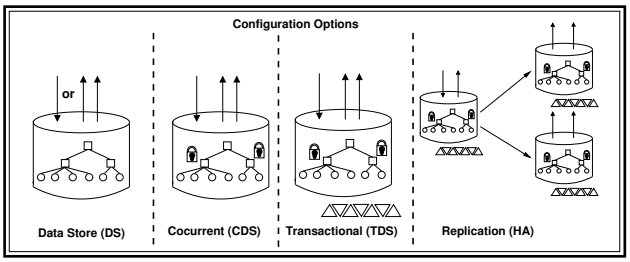
Databáze je spojena s aplikací pomocí (obvykle dynamické) knihovny, která následně poskytuje funkční API umožnující pracovat s daty v nativním formátu databáze. Oproti velkým DBMS (database management system) využívajícím obvykle SQL, embedded databáze vynikají jednoduchostí API, malou velikostí, velkou rychlostí (nemusejí se zabývat parsováním dotazu, sestavením plánu, jeho prováděním, transakcemi a mnoha dalšími činnostmi) a v neposlední řadě také snadnou správou (je třeba pouze malá či žádná údržba).

Představme si několik významnějších zástupců.

3.2.1 Berkeley DB

Embedded databáze pracující se schématem klíč/hodnota (v současné verzi poskytuje i SQL) poskytující vysoký výkon. Databáze je napsaná v jazyce C a existují také API rozšíření pro jazyky C++, Java, Python, Perl, Ruby a další. Berkeley DB [73] vychází z DBM a původně vznikla na univerzitě v Berkeley. BDB ukládá klíče i hodnoty do „*byte arrays*“

a umožňuje uložit více hodnot k jednomu klíči. BDB podporuje databáze ve velikostech v rozmezí kilobajtů až petabajtů.



Obrázek 3.2: Funkcionalita BerkeleyDB[73]

Databáze neposkytuje síťový přístup a celá komunikace s aplikací se děje v tzv. *in-process* módu (aplikace využívá poskytované API). Současná verze poskytuje funkce jako jsou ACID transakce, zamykání, neblokující zápis, čtení s nízkou odezvou či replikace. Jistě je vhodné podotknout, že BDB poskytuje zpětnou kompatibilitu s unixovými DBM a NDBM (in-memory hash tables). Mezi základní datové indexy databáze patří *btree*, *hash*, *queue*, *recno* (v abecedním pořadí). Hash je pravděpodobně nejpoužívanějším. Mimo formátu klíč/hodnota je možné využít i funkcí relačních databází a SQL díky implementované podpoře SQLite3 API, ODBC, JDBC a ADO.NET. BDB podporuje kompresi, paralelní přístup pomocí vláken/procesů a dokonce i šifrování a HW akceleraci na platformě Intel.

Jak můžeme vidět na obrázku 3.2, původní přístup **DS** (Data store) umožňuje pouze zápis nebo čtení. S dalšími verzemi BDB přišel **CDS** (Concurrent data store), který dovoluje provádět zápis i čtení zároveň díky použití zámků. Vzhledem k tomu, že BDB podporuje i SQL, najdeme i transakční přístup **TDS**. Pro ochranu dat a jejich integritu a spolehlivost nabízí BDB replikaci **HA** (high availability, vysoká dostupnost) a automatický failover na jednu ze záložních kopií.

V současnosti vlastní BDB společnost Oracle, která poskytuje tři produkty, vlastní BDB, BDB Java Edition a BDB XML, kde každý z nich má své specifické výhody (vazbu na Javu nebo ukládání XML dokumentů). Tento typ embedded databáze je často nasazovaný do mnoha aplikací i díky své licenci (BSD, Sleepycat licence) umožnující šíření v binární formě bez zdrojového kódu (podmínkou je pouze uvedení copyrightu). Připomeňme si několik důležitějších, jako např. Asterisk PBX (softwarová telefonní ústředna), IMAP server Cyrus IMAP, databázový server MySQL, adresářová služba OpenLDAP, poštovní server Postfix či balíčkovací systém RPM.

3.2.2 SQLite

SQLite [47] je další z embedded databází, ovšem díky velkému množství implementovaných vlastností se pomalu stírá rozdíl mezi embedded a relačními databázemi. Jde opět o databázi uloženou v jednom souboru, která nepotřebuje server ani konfiguraci, a nabízí SQL i transakce. Na domovské stránce se tvrdí, že jde o nejvíce rozšířenou databázi tohoto druhu (k čemuž jistě přispívá i zveřejnění zdrojového kódu v public domain, a tedy možnosti využití i v komerčních produktech).

Přesuňme se tedy k architektonickým vlastnostem. Patří mezi *in-process* databáze podobně jako BDB. Mimo již zmíněných, vyhovuje SQLite standardu SQL'92 a vyniká velmi malou velikostí kódu (325 kB v plné konfiguraci, méně než 190 kB při vynechání volitelných funkcí). Samotný kód je napsaný v jazyce ANSI C, a poskytuje přímo propojení

s jazykem Tcl. Mimo jiné nabízí i rozšíření API pro mnohé další jazyky jako jsou C, Perl či PHP. Rovněž můžeme považovat za výhodu multiplatformnost (Unix, Windows, OS/2) a přítomnosti klienta pro příkazovou řádku.

Jak již bylo zmíněno, SQLite podporuje transakce ACID a téměř kompletní implementaci standard SQL'92. Můžeme tedy používat tabulky, indexy, spouště i pohledy. Podpora little-endian i big-endian, stejně jako 32bitové i 64bitové architektury je samozřejmostí.

SQLite je obvykle využívána jako cílový formát aplikačních dat (odpadá parsování), díky své velikosti a minimální potřebě údržby je vhodná pro implementaci do různých mobilních zařízení a spotřební elektroniky. Často je použita v roli backendu pro webové aplikace. Pokud by nás zajímaly konkrétní příklady, můžeme se zaměřit např. na Mozilla Firefox, Skype, Google Chrome či Adobe Photoshop Lightroom.

SQLite nabízí i precizně zpracovanou dokumentaci zahrnující Wiki server a tutoriály. Pokud dáváte přednost tištěné dokumentaci, SQLite je tématem několika knih (např. *Using SQLite* či *The Definite Guide to SQLite*).

3.3 Relační databáze

Jsou databáze postavené na **relačním modelu** a **relační algebře**. Relační model pro databázové systémy formuloval E. F. Codd již v roce 1969. Je založen na predikátové logice prvního řádu. Základní myšlenkou je snaha popsat databázi jako množinu predikátů (např. $\forall X$ *platí, že* $(Y \wedge Z) \Rightarrow W$) nad konečnou množinou predikátových proměnných tak, abychom vyjádřili omezení (constraints). Databáze obsahují n-ární relace (tabulky), které jsou kartézským součinem n domén (atributů). Každý výrok lze vyhodnotit dvěma hodnotami (true, false, existuje i tříhodnotový systém obsahující NULL).

Relační algebra je matematický aparát sloužící k popisu relačního modelu. Obsahuje základní operace jako jsou projekce π, selekce σ a přejmenování ρ a dále jsou definovány operátory spojení (např. přirozené $\bowtie$, všechny dvojice spojovaných relací) a nakonec také agregační funkce (sum, count, average, minimum a maximum).

Často potřebujeme, aby tabulka splňovala určité požadavky, zejména neobsahovala zbytečnou redundanci a nezávislé hodnoty. K dosažení těchto cílů se používá technika nazývaná **normalizace**, což je proces převodu do jedné z normálních forem. Ač těchto norem existuje více, v praxi se používají převážně první tři (tj. 1NF, 2NF, 3NF, kde ku příkladu 1NF požaduje, aby každý atribut obsahoval pouze atomické hodnoty)

RDBMS (relation database management system), což je DBMS kde se data i vztahy mezi nimi ukládají do tabulek. RDBMS povětšinou pracuje v režimu klient/server. RDBMS můžeme dělit dle různých kritérií. Může nás například zajímat závislost na konkrétním operačním systému, schopnost použití transakcí a ACID, referenční integrity, práci s kódováním pomocí Unicode, ale také jistá omezení jako maximální velikost databáze, tabulky, řádku či přímo jednotlivých datových typů. Jistě nás bude zajímat práce s indexy, materializované pohledy, dočasné tabulky a mnohé další atributy.

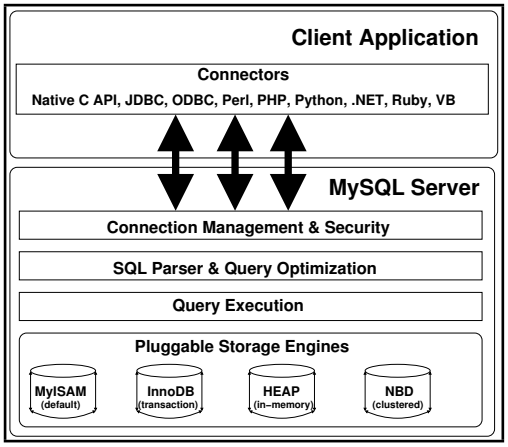
Nyní si můžeme představit několik klasických zástupců v open-source světě.

3.3.1 MySQL

MySQL [58] je multiplatformní klient/server relační databáze napsaná v jazycích C a C++, obsahující parser SQL napsaný v *yacc* a vlastní lexikální analyzátor. Pro spojení s MySQL z programovacích jazyků byla vyvinuta různá API (viz schéma 3.3). Můžeme tedy používat MySQL z C, C++, Perlu (přes DBI), PHP, Pythonu, C#, Javy a mnoha dalších.

MySQL je oficiálně dodávána pouze s klientem pro příkazovou řádku, nicméně existuje několik GUI nadstaveb (např. phpMyAdmin). Ačkoli je dostupná open-source verze databáze, firma MySQL AB poskytuje i verze komerční (tzv. Enterprise Server s podporou).

Databáze MySQL má většinu vlastností, které jsou u relační databáze s podporou SQL očekávány. Mimo již zmíněné multiplatformnosti, splňuje MySQL standard ANSI SQL'99, dále poskytuje možnost využití uložených procedur, spouští a kurzorů, stejně tak jako podporu SSL, hesel a replikace (umožnující zvýšit dostupnost, HA) a částečnou podporu Unicode a ACID v transakcích.



Obrázek 3.3: Architektura MySQL[59]

Daleko zajímavějším údajem však může být informace o tom, kde se MySQL liší od ostatních RDBMS. V této oblasti je to především velký výběr nativních formátů uložení relaci (MyISAM, Berkeley DB, InnoDB, stejně jako dovede pracovat s konkurenčními SolidDB či IBM DB2).

Nicméně každá technologie má své limity. MySQL dovoluje vytvořit až 64 indexů na jedné tabulce (od verze 4.1.2), kde každý může zahrnovat 16 sloupců (atributů). V tabulkách můžeme narazit na omezeni maximální velikosti záznamu nastavené na 65 kB, stejně tak maximální počet sloupců/atributů je nastaven v kódu na 4096. Samotná velikost databáze je omezena většinou systémem souborů, na kterém se vyskytuje (např. Linux Ext3 – 4 TB, Windows NTFS – 2 TB).

MySQL je používána pro mnohé webové aplikace, jako jsou např. sociální síť Facebook, Flickr, YouTube, redakční systémy Joomla, WordPress, phpBB anebo také např. Wikipedia. Často bývá spojována s dalšími nástroji (Linux jako operační systém, webový server Apache, MySQL a některý ze skriptovacích jazyků Perl/PHP/Python) do balíku LAMP jako základ pro provoz či vývoj webových aplikací.

3.3.2 PostgreSQL

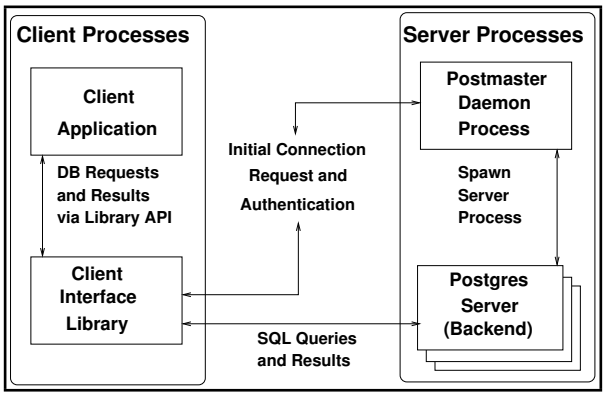
PostgreSQL [68] je ORBMS (objektová relační databáze). Její název se často zkracuje na pouhé „*Postgres*“. Postgres vychází z myšlenek DBMS Ingres, která vznikla původně na univerzitě v Berkeley pod MIT licencí.

Postgres má následující rysy. Umožňuje provádět **uložené procedury** (tzv. PL, procedural language, s vazbou na PL/SQL od Oracle, Tcl, Perl a Python), tedy uživatelsky definované rutiny, spouště a kurzory.

Jako téměř každá databáze, i Postgres má zabudované **indexy**, konkrétně založené na B+ stromech, hash, GIST a GIN a uživatelsky definované indexy. Ty dovolují specifikovat index jako výraz („*spočítej něco nad sloupcem a výsledek považuj za index*“) anebo omezit řádky zahrnuté do indexu klauzulí *where*. Navíc plánovač provedení dotazu bere v úvahu všechny dostupné indexy a umí je k dosažení výsledku zkombinovat.

Spoušť (**trigger**) je uložená procedura, jejíž kód se vykoná, pokud se stane určitá událost (např. Update, Select, Delete). Postgres umožňuje přiřadit spouště pouze tabulkám, pro pohledy (views) existují jiná pravidla.

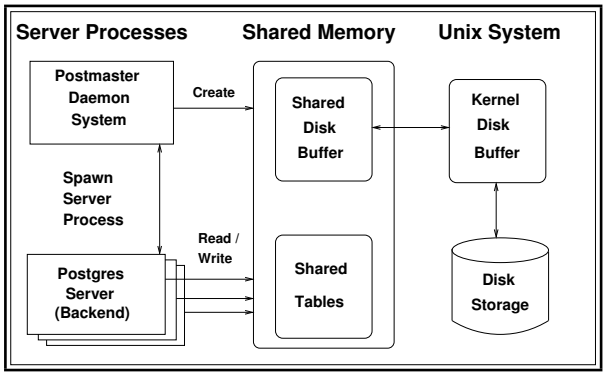
Opravdu zajímavým způsobem je v Postgresu řešena souběžnost. Je totiž použita technologie nazvaná **MVCC** (multi-version concurrency control), která poskytne každému uživateli jeho vlastní snímek/kopii databáze, čímž eliminuje možnost ovlivňování ostatních uživatelů a jejich výpočtů. Po skončení transakce se změny uloží.



Obrázek 3.4: PostgreSQL – klient/server komunikace[69]

Jak již bylo zmíněno, pro pohledy se v Postgresu spouště nepoužívají. Místo nich mohou být definována tzv. **pravidla** (jinak taky query re-write rules). Tyto pravidla říkají, jak změnit příchozí DML (data manipulation language) dotaz.

Mimo těchto klíčových vlastností zmiňme ještě SSL, dědičnost vlastností od nadřazených tabulek, kompatibilitu se standardem SQL'2008, omezení na referenční integritu, TOAST (pro ukládání velkých atributů, jakožto MIME příloh či XML dokumentů s automatickou kompresí), podporu regulárních výrazů a replikaci.



Obrázek 3.5: PostgreSQL – inter-server komunikace[69]

Výkonnost Postgresu je závislá na množství paměti, kterou mu přidělíme. Tato paměť je následně použita pro cache na bloky databáze či na třídění. Bohužel, standardní nastavení používá pouze malé množství paměti, protože současný kernel nepovoluje přidělení velkých bloků paměti. Ke zvýšení výkonnosti si lze dopomoci úpravou parametrů jádra.

I Postgres má svá omezení. Server nemůže být rozprostřen přes více strojů, neboť jeho instance spolu komunikují přes sdílenou paměť (viz schéma 3.5). Ačkoli velikost databáze ani počet indexů není omezen, existují limity na velikost tabulky (32 TB), velikost řádku (1.6 TB) či počet sloupců na tabulku (250-1600 dle typu).

Pro přístup k databázi potřebujeme nějakého klienta (viz schéma 3.4). K výběru se nám nabízí *psql* (textový klient pro příkazovou řádku) či některý z klientů grafických (pgAdmin nebo phpPgAdmin, open-source, GPL). Postgres je široce nasazován a byl volbou i pro velké projekty Internetu jako Yahoo, MySpace, či OpenStreetMap.

3.4 Dokumentové databáze

Tento druh databází patří do širšího kruhu tzv. NoSQL (nejen SQL, not only SQL) databází společně s databázemi grafovými, klíč/hodnota či tabulkovými. Koncept, který tento druh databází utváří, je nerelační řešení umožňující horizontální rozšiřitelnost, distribuovatelnost, často open-source řešení, bez pevného schématu a připravenost pro velké objemy dat a replikaci. Jejich nasazení je obvykle spojováno s webovými aplikacemi.

Základní datovou jednotkou dokumentových databází je dokument, který je obvykle reprezentován jedním z formátů XML, YAML, JSON či BSON. Jednotlivé dokumenty jsou v databázi identifikovány jednoznačnými klíči. Dokumenty připomínají řádky tabulek v SQL světě. Společně tvoří kolekce, což evokuje zmíněné tabulky. Dokumentové databáze často implicitně podporují verzování dokumentů v nich obsažených.

3.4.1 MongoDB

MongoDB [32] je zástupce NonSQL dokumentových databází. Jedná se o open-source projekt, který je naprogramován v jazyce C++ a je multiplatformní. Pro využití v dalších jazycích poskytuje knihovny, jejichž seznam je opravdu rozsáhlý. Namátkou vyberme např. C++, Java, Python, Ruby, PHP, Perl a mnohé další. MongoDB nabízí ukládání JSON dokumentů, resp. jejich binární serializaci tzv. BSON. Databáze využívá tradičního přístup klient/server a poskytuje i základní webové rozhraní.

MongoDB je zajímavá především způsobem, kterým uchovává data. MongoDB poskytuje databáze a kolekce, což je jistá obdoba databází a tabulek v pojetí relačních databází. Tam ovšem podobnost struktury dat končí. Databáze i kolekce není třeba vytvářet, neboť prostě začnou existovat s prvním zápisem. Dalším podstatným rozdílem, oproti relačním databázím, je absence tradičního schématu – kolekce neobsahují nic jako sloupce označené typem dat. Celé schéma kolekcí je dynamické, a tak dovoluje tedy uchovávat dokumenty s odlišnými poli, ačkoli je zvykem, že jedna kolekce obsahuje dokumenty stejného typu.

Jednotlivé dokumenty jsou označeny jednoznačným identifikátorem UUID, který je následně použit při práci s nimi. MongoDB poskytuje dotazovací jazyk, jehož možnosti jsou rozsáhlé. Je nám umožněno používat funkci *find*, která je obdobou selectu z SQL databází. Data dokumentů lze samozřejmě i upravovat a mazat. Většina operací s MongoDB vrací kursor o velikosti dvaceti záznamů namísto celého dokumentu. Poněkud zvláštní je skutečnost, že operace prováděné z klienta nejsou standardně ověřovány, a tak se klient spoléhá na bezproblémový průběh. Tomuto chování lze zamezit použitím bezpečného módu.

Z pokročilých funkcí MongoDB podporuje indexy, agregační funkce, replikaci či odkazy mezi jednotlivými kolekcemi. Databáze je připravena k uchovávání velkých souborů, a také pro běh v prostředí vyžadující vysokou dostupnost. Na stránkách projektu nalezne kromě tutoriálů i dokumentaci k API.

Mezi velké projekty využívající MongoDB patří např. repozitář zdrojových kódů SourceForge [41] či nekomerční distribuovaná sociální síť Diaspora [78].

Kapitola 4

Vizualizace

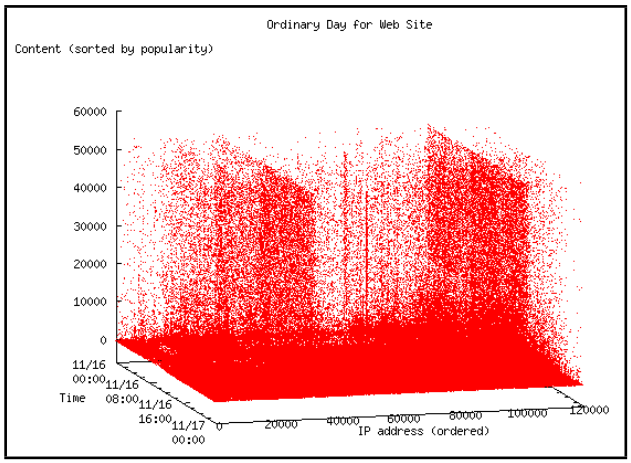
Co rozumíme pod pojmem vizualizace? Jedná se o jasné a efektivní prezentování dat za pomoci grafických prvků. V první řadě nás zajímá srozumitelnost prezentovaných dat a až na místě druhém je grafická stránka vizualizace, která dokáže zaujmout pozornost a data zpřehlednit. Data se dají formou vizualizace prezentovat anebo zkoumat.

Vizualizací se snažíme zmírnit množství dostupných informací a provádíme tzv. dolování dat (data mining). Pomocí grafické reprezentace se dokážeme vyrovnat i s daty, která pocházejí z nestejnorodých zdrojů či obsahují chyby. Dolování lze provádět i strojově, ovšem jedná se o složité statistické algoritmy založené na dlouhodobějším pozorovaní dat.

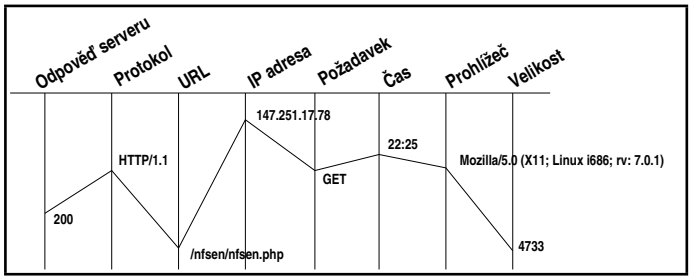
Samotná prezentace se často zabývá otázkami co prezentujeme, komu, jak a proč. Pokud se budeme zajímat o zdroj dat, neboli „co“, můžeme se zaměřit na jedinou veličinu či na data obsahující více veličin a jejich případné vztahy. Data mohou být souvislá nebo rozdělená dle jednotlivých kategorií. Mimo to mohou být samotná data změněna či agregována pro účely prezentace.

Grafická reprezentace může mít mnoho forem. Mezi často používané zástupce při dvoudimenzionálním zobrazení patří např. koláčový graf, sloupcový graf, histogram nebo mapy (zeměpisná šířka a délka). Pro vyšší dimenze existují techniky hledající geometrické transformace zobrazující vícerozměrná data ve 2D/3D prostoru. Za zvážení jistě stojí fakt, že některé formy nejsou vhodné pro určitá data (např. je nemožné zobrazit data jedné veličiny souvislá v čase pomocí koláčového grafu).

Pro vícedimenzionální vizualizace můžeme použít například rozptylový diagram anebo souběžné souřadnice. **Rozptylový diagram** (scatter plot) poskytuje všechny možné kombinace dvou veličin (matice rozptylových diagramů, scatter plot matrix) pro 2D či 3D prostor. Využívá kartézskou soustavu souřadnic a další dimenze lze přidat pomocí vlastností bodů (barva, tvar, velikost). Nicméně metoda rozptylových diagramů je zamýšlena pouze pro malý počet veličin. Příklad rozptylového grafu uvádí obrázek 4.1.



Obrázek 4.1: Ukázka rozptylového grafu – požadavky na webový server během jednoho dne [27]



Obrázek 4.2: Ukázka grafu pomocí systému souběžných souřadnic – požadavek na webový server

Pokud uvažujeme více veličin, přichází v úvahu např. **systém souběžných souřadnic** (viz obrázek 4.2, paralel coordinates) [30], což je metoda, využívající souběžné osy pro jednotlivé dimenze. Osy jsou stejné, co se týče orientace a délky, měřítko se obvykle liší. Bod v tomto typu zobrazení tvoří lomená čára, jejíž části jsou úsečky spojující jednotlivé osy. U tohoto typu zobrazení může docházet k určité ztrátě informace, neboť některá data se mohou překrývat, obzvláště při zobrazení v prostoru 2D. Mezi nejdůležitější parametry tohoto zobrazení můžeme zařadit pořadí, rotaci a měřítko os.

Informace o dalších technikách používaných pro vizualizaci vícedimenzionálních dat, jako jsou např. tabulky relačních databází, vztahy v textu, hierarchické vztahy na webu, vizuální znázornění algoritmů a další, rozebírá ve svém článku Daniel A. Kean. [16]

Dalším krokem po výběru formy je nastavení určitých parametrů tak, aby data mohla být prezentována správně. Jedním z nich je volba měřítka. Je třeba rozhodnout o četnosti a rozmístění bodů na osách, o koncových bodech, případně jak se má algoritmus pro rozmístění značek zachovat, jestliže data vystoupí z očekávaných hranic (např. hodnota 105, pro interval <0-100>). Důležité je také samotné umístění osy, kde bývá typickou hodnotou bod 0, nicméně nemusí to být pravidlem.

Pořadí zobrazených hodnot je obvykle určováno dle abecedy a kategorií, případně dle hodnot dalších proměnných, velikosti či geografického rozmístění dat. Důležitou složkou vizuální reprezentace je titulek a legenda. Titulek má jasně popisovat o jaká data se jedná, případně také uvést jejich zdroj. Legenda přináší podrobnější informace o jednotlivých zobrazených datových řadách [6].

4.1 Možnosti vizualizačních knihoven

Pro potřeby vizualizace dat získaných aplikací NfSen a NetFlow sondami bylo vyzkoušeno několik grafických knihoven, které se věnují oblasti 2D grafů, pokud některá dovoluje jiné typy grafů, je to explicitně zmíněno. V následujících podkapitolách poskytuji přehled konfiguračních možností jednotlivých knihoven týkajících se reprezentace NetFlow toků a zjištěných anomálií. Přehled uvádí nejen popis vybraných konfiguračních voleb, ale i informace o požadavcích na systém, spotřebované paměti, výkonu a v neposlední řadě i upozornění na možné problémy či nedostatky. Může tedy dobře posloužit jak při návrhu, tak při implementaci. Velikost vybraných knihoven je většinou kolem 1 MB.

Pro srovnání je v každé podkapitole uveden výpis konfigurace a výsledný graf. Jako testovací data byla použita anonymizovaná data ze skutečného provozu na síti Masarykovy univerzity (ze dne 22. září 2011). V podkapitole 4.1.6 na straně 39 nalezneme porovnání grafických knihoven týkající se požadavků na systém, dokumentace a konfigurace. Také jsou zde zmíněny metody testování a obhajoba výsledné volby vizualizační knihovny.

4.1.1 RRD

Round robin database (RRD) [63], je nástroj k uchovávání a zpracovávání dat (detailnější informace a vysvětlení některých pojmů týkajících se datové struktury naleznete v kapitole 3.1.1 na straně 20) v časových řadách, který mimo jiné disponuje také možností uložená data vizualizovat. Za tímto účelem používá funkci **graph**. Představme si její možnosti, ať už konfigurační či grafické.

Zdrojový kód 4.1: RRDtool – definice dat pro graf

```
DEF:<vname>=<rrdfile>:
            <ds-name>:
            <CF>
             [:step=<step>]
             [:start=<time>]
             [:end=<time>]
             [:reduce=<CF>]

VDEF:vname=RPN expression

CDEF:vname=RPN expression
```

Abychom byli schopni vyprodukovat graf, potřebujeme nejprve příhodná data ve správném tvaru. Data získáme z RRD databáze, konkrétně z RRA pomocí definice **DEF**, **VDEF** či **CDEF**. Každá z těchto instrukcí musí mít identifikátor *vname*, který se může skládat pouze z alfanumerických znaků, pomlčky a podtržítka, vše do maximální délky 255 znaků.

DEF musí kromě *vname* ještě obsahovat jméno RRD databáze, jméno zdroje dat (DS) a konsolidační funkci (CF, jednu z maximum, minimum, průměr, poslední hodnota). Konsolidační funkce zpracuje data získaná v průběhu jednoho kroku a uloží pouze vypočtenou hodnotu. Pro DEF lze nastavit velikost kroku (step) a čas startu a ukončení. Navíc lze nadefinovat ještě jednu konsolidační funkci pro případ, že RRA obsahuje více dat, než je požadováno. Mezi archivy RRA se automaticky vybere ten s odpovídajícím rozlišením.

VDEF opět používá unikátní *vname* a uchovává pouze jedinou hodnotu. Pro tuto hodnotu lze využít pouze agregačních funkcí. Poněkud jiný způsob využití nabízí CDEF. Tento příkaz vytvoří novou sadu bodů, ovšem pouze v paměti. Pro definici funkcionality se používá RPN (reverzní polská nebo také postfixová notace) jazyk pro definici CDEF nebo VDEF) a obvykle jde o jednoduchou matematickou funkci (např. převod bitů na bajty, tedy násobení osmi). Uvedené funkce pro definici dat shrnuje ukázka 4.1.

RPN pracuje se zásobníkem a načítá z něj potřebný počet hodnot dle použitého operátoru. Mezi dostupné operátory patří booleovské (*LT, GT, LE, GE, EQ, NE*), operátory pro porovnání s neznámou hodnotou či nekonečnem a také podmínka *IF*. Dále lze využít agregační či matematické operátory a několik speciálních hodnot (*UNKN, INF, NOW*). Pro podrobnější informace je k dispozici stránka manuálu *rrdgraph_rpn*.

Pokud jsou definované zdroje dat, je možné přistoupit k tvorbě samotného grafu příkazem `graph`. Příkaz má např. volby pro velikost výsledného grafu, osy, legendu, barvy atd. Zaměřme se nyní na některé volby podrobněji. Kompletní seznam konfiguračních voleb naleznete v manuálu *rrdgraph* a *rrdgraph_**.

rrdtool graph|graphv filename [option ...] [data definition ...][data calculation ...] [variable definition ...] [graph element ...] [print element ...]

Třebaže je název souboru jediným povinným parametrem funkce `graph`, k vytvoření grafu nestačí. Pokud si přejeme jiný než standardní výstup, můžeme volit mezi **PNG**, **SVG**, **PDF** či **EPS**. Dále je třeba specifikovat časové období, pro které se má graf vykreslit a velikost kroku (parametry `-s`, `-e` a `-S`). Kromě unixového času, který je vyjádřen počtem

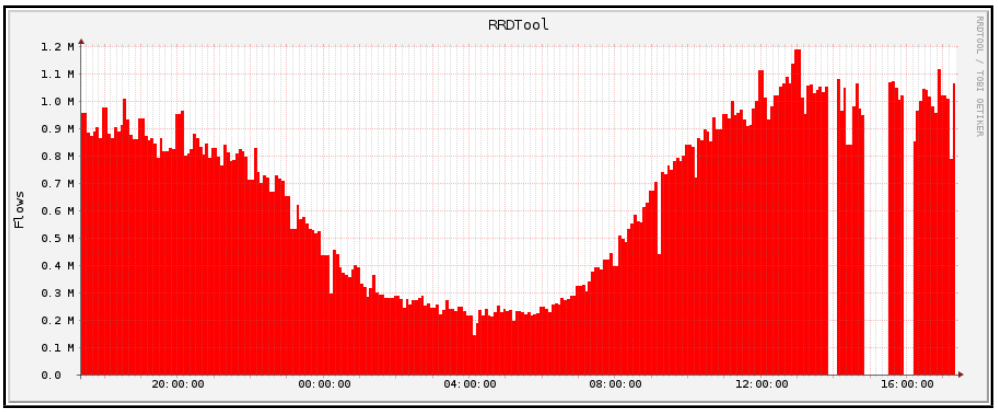
sekund od 1. 1. 1970, akceptuje i časové údaje příkazu *at* určeného pro spouštění úloh v určitém čase.

Zdrojový kód 4.2: Ukázka konfigurace RRDtool

```
rrdtool graph flows.png,
  "--start", "1316618400",
  "--end", "1316704800",
  "--title", "RRDtool",
  "--width", "800",
  "--height", "300",
  "--vertical-label", "Flows",
  "--x-grid", "MINUTE:10:HOUR:1:HOUR:4:0:%X",
  "DEF:flow=$rrdFile:flows:AVERAGE",
  "AREA:flow#FF0000"
```

Neméně důležité je nastavení rozměrů grafu (`-w`, `-h`). Mimo jiné je možné vytvořit náhledový obrázek s použitím parametru `--only-graph` s výškou do 32 pixelů. Specifikujeme popisek (`--title`), škálování a očekávané špičkové hodnoty. Důležitým bodem při nastavování grafu je popsání souřadnicových os, včetně definice kroku.

Můžeme přidat ještě legendu/vysvětlivky, rámeček, font či adresu démona rrdcached. Pro doplňující informace lze použít ještě funkce `COMMENT` a `GPRINT`. Zajímavostí by mohl být parametr `--lazy`, který vytváří graf jen tehdy pokud se data změnila. Poté do definice grafu doplníme hodnoty DEF, CDEF či VDEF a definici grafické části.



Obrázek 4.3: Ukázka grafu v RRDtool

Zde je na výběr spojnicový graf (`LINE`) a graf oblasti (`AREA`). Oblasti je na sebe možné navršit (`STACK`). Pro tyto volby je třeba zahrnout DS či DEF jako zdroj dat, volitelně navíc barvu a popisek. Dále je možné zobrazit data z různých časových úseků pomocí volby `SHIFT`. Uvedené volby jsou použity v konfiguraci 4.2, ze které po zpracování nástrojem RRDtool vznikne graf 4.3. V neposlední řadě je vhodné vyzdvihnout integraci statistické metody Holt-Winters (informace o této metodě jsou k nalezení v článcích C. Chatfielda [4, 5]) pro odhalování anomálií v průběhu časové řady přímo do RRDtool.

RRDtool grafy jsou často využívány jako součást dalších produktů pro např. vizualizaci odezvy sítě, objemu dat či jiná periodická data. Mezi nejvýznamnější zástupce open-source produktů patří MRTG [61], SmokePing [64], Cacti [36], Mailgraph [72]. Z komerčních pak můžeme uvést AirWave Management Platform [35] či tacLOG/tacMON [76]. Autor i uživatelé považují RRDtool za hotový nástroj, proto se vývoj omezil jen na případné opravy chyb či změny z důvodu kompatibility (např. kompilace s GCC 4.5 -std=c99). I tak je poslední změna v kódu stabilní větve stará sedmnáct měsíců.

4.1.2 jqPlot

JqPlot [53] je knihovna pro jQuery, *což je javascriptová knihovna umožňující skriptování na klientské straně ve spolupráci s HTML, podporující výběr DOM objektů, práci s událostmi, CSS a AJAXem* [48], který teprve nedávno dospěl do verze 1.0 (beta). Umožňuje vytvářet čárové, sloupcové a koláčové grafy. Kromě grafů statických lze vytvářet i grafy dynamické, jejichž data jsou známa později než při generování webové stránky. Podporuje uživatelské nastavení os, ať jde o popisky (datum, rotace), jejich počet (vlastní osa pro každou množinu bodů v jednom grafu) a další. Ukažme si, jaké má požadavky, konfiguraci a co je nám schopen nabídnout.

Předně je třeba uvědomit si, že se jedná tzv. „client-side“ aplikaci, což znamená, že webový server poskytne prohlížeči na koncovém zařízení pouze definici grafu a knihovnu pro její zpracování. Poté, co prohlížeč přesune data do paměti, použije vlastní zdroje k tomu, aby graf vygeneroval. Ulehčí se tak serveru, ovšem některé prohlížeče mohou mít zakázaný skriptovací jazyk JavaScript (jqPlot → jQuery → (JavaScript) → HTML), a tak mohou nastat problémy s generováním grafu.

Prvním krokem ke zprovoznění knihovny jqPlot je jeho začlenění do kódu webové stránky. V hlavičce HTML stránky přidáme záznamy pro načtení jQuery, jqPlot a jeho případných rozšiřujících pluginů. Pro prohlížeč Internet Explorer je k dispozici knihovna *excanvas.js*, která zpřístupňuje HTML5 tag `canvas` pro ohraničení 2D kresby, který Internet Explorer do verze 9 nepodporuje.

Použití pluginů poskytuje projektu jqPlot rozšíření funkcionality. Kromě několika pluginů přiložených v distribučním balíčku, které podporují například použití časových či logaritmických os, umožňuje jqPlot i definici pluginů vlastních. Používání pluginů je třeba povolit pomocí `$.jqplot.config.enablePlugins = true`, případně `{ show: true }`.

Dalším krokem je vyznačení místa v těle HTML dokumentu označujícím umístění grafu vzhledem k okolnímu obsahu [54].

Zdrojový kód 4.3: Ukázka konfigurace jqPlot

```
plot = $.jqplot('plot', [points], {
  title:'jqPlot',
  legend:{
    show:true,
    location: "n"
  },
  series:[{
    fill: true,
    label:'flows',
    color:"#ff0000",
    shadowAngle:0,
    shadowOffset:1.5,
    shadowAlpha:.08,
    shadowDepth:6
  }],
  axesDefaults: {
    tickRenderer: \\
      $.jqplot.CanvasAxisTickRenderer,
    tickOptions: {
      angle: -30,
      fontSize: '8pt'
    }
  },
  axes: {
    xaxis: {
      renderer: $.jqplot.DateAxisRenderer,
      tickOptions: \\
        {formatString:'%#d/%m/%y\t%a␣%H:%M'},
      autoscale: true,
      label:'Time',
      labelRenderer: \\
        $.jqplot.CanvasAxisLabelRenderer
    },
    yaxis: {
      label: "Flows",
      autoscale: true,
      labelRenderer: \\
        $.jqplot.CanvasAxisLabelRenderer,
      min: 0,
    }
  },
  cursor: {
    showVerticalLine:true,
    showHorizontalLine:false,
    showCursorLegend:true,
    showTooltip:false,
    zoom:true
  }
});
```

Definici grafu začneme klíčovou funkcí `$.jqplot()` a do jejího těla postupně přidáme všechny potřebné parametry. Mezi parametry povinné zařadíme pouze dva, a to odkaz na dříve definovaný prostor v těle HTML dokumentu a alespoň jednu datovou řadu. Takto nám vznikne pouze samotný obrázek spojnicového grafu. Ostatní volby jsou nepovinné a jestliže neřekneme explicitně jinak, zvolí se automaticky. Můžeme tak ze základního spojnicového grafu učinit graf plošný vyplněním prostoru pod křivkou volbou `fill`.

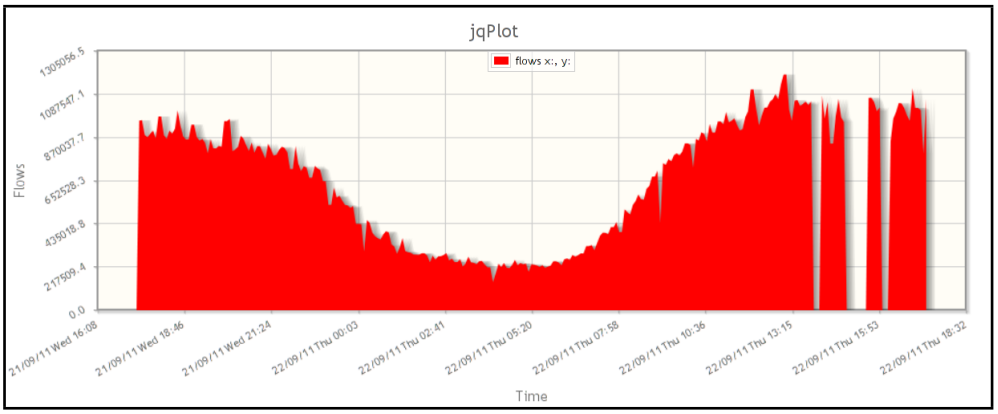
Graf má mít řádně označené osy (volby `axes`, `xaxis` a `yaxis`) a být ve správném měřítku. Jestliže bychom v grafu rádi použili logaritmické či exponenciální souřadnicové osy, musíme načíst patřičný plugin a použít volbu `renderer: $.jqplot.LogAxisRenderer`. Označení os lze ještě natočit pod zadaným úhlem či zvolit velikost písma.

Pro koláčové či sloupcové grafy je třeba použít speciální vykreslovač (`PieRenderer` pro koláčové a `BarRenderer` pro sloupcové). Pokud bychom chtěli specifikovat další nastavení, použijeme volbu `rendererOptions`, díky níž můžeme nastavit průměr, šířku, mezery mezi sloupci anebo horizontální či vertikální orientaci sloupců.

Pro zvýšení přehlednosti grafu nabízí jqPlot několik funkcí, které si nyní představíme. Chceme-li graf pojmenovat, použijeme volbu `title`. Pokud je nežádoucí mít název datové řady přímo u křivky, můžeme vytvořit legendu (`legend`). Jejího zobrazení docílíme až zadáním příkazu `show: true` a její umístění v grafu je ovlivnitelné příkazem `location` s udáním světové strany, např. „n“ (North, sever). Pro rozlišení datových řad obsažených v grafu můžeme použít nastavení barev či stín.

Pro zvýraznění bodu v grafu poskytuje jqPlot funkci `cursor` a k ní volbu `tooltip`, která zobrazí informace o hodnotě nacházející se pod kurzorem myši. Pokud chceme určit vlastnosti pro více sérií nebo os, můžeme použít volby `axesDefaults` nebo `seriesDefaults`. Ze statistických nástrojů podporuje jqPlot tzv. *trendline*, což je přímka spojující některé body v grafu (maxima nebo minima), která určuje směr a sílu vývoje v datech grafu. JqPlot je schopen tuto křivku generovat automaticky pomocí funkce `trendline`. V neposlední řadě dovoluje jqPlot přesun bodu grafu pomocí funkce *táhni a pusť* či zobrazení mřížky.

Ukázka grafu na obrázku 4.4 byla vytvořena parametry uvedenými v konfiguraci 4.3. JqPlot byl také použit pro vizualizaci při projektu CNDet [15] (Chuck Norris botnet detection plugin) pro NfSen, který vznikl na Masarykově univerzitě.



Obrázek 4.4: Ukázka grafu v jqPlot

4.1.3 Flot

Flot [50] je vizualizační knihovna pro kreslení grafů napsaná v jazyce JavaScript využívající jQuery. Její přednosti spočívají v jednoduchosti použití, atraktivním vzhledu a interaktivnosti s uživatelem. Můžeme tak zvětšovat vybrané oblasti grafu nebo reagovat na události kurzoru myši. I zde se jedná o „client-side“ aplikaci, a tak je vykreslení grafu úkolem pro prohlížeč na koncové platformě. Pro použití v tagu `canvas` je pro prohlížeč Internet Explorer ve verzí nižší než 9 třeba plugin `excanvas.js`. Flot podporuje vykreslování bodového, spojnicového, sloupcového a s dodatečným pluginem i koláčového grafu.

Prvotním krokem je propojení HTML dokumentu, jQuery a knihovny flot. Následně v těle téhož dokumentu označíme místo, kam si přejeme výsledný graf umístit. Na toto místo se budeme odkazovat při konfiguraci grafu. Flot očekává, že data budou pouze čísla a jejich přísun ideálně ve formátu JSON, což je datový formát pro přenos informací např. mezi PHP a JavaScriptem ve formě řetězce.

Definici započneme užitím funkce `$.plot()`, jejíž parametry budou ostatní konfigurační volby. Uvedením `$("#id")` zajistíme propojení s rezervovaným místem a volbou `data:` grafu podsuneme data k zobrazení, která lze definovat výčtem prvků nebo předaným polem. Graf může být spojnicový, sloupcový či bodový, což zajistíme výběrem jedné z možností `lines`, `bars` či `points`.

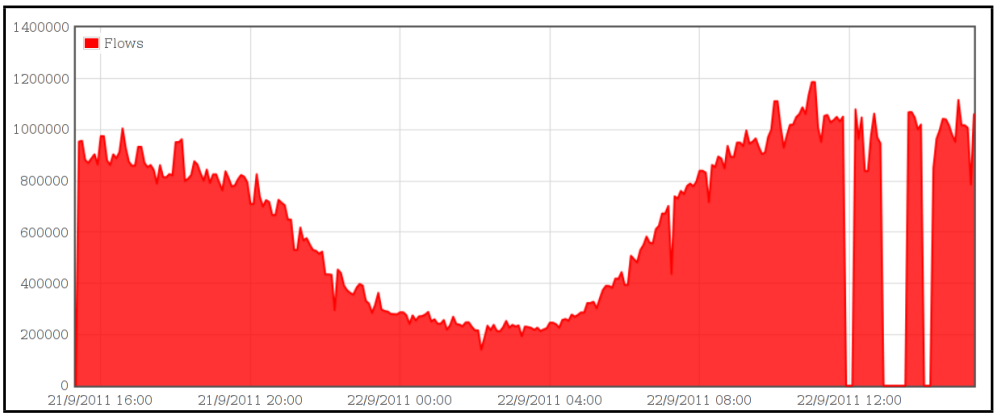
Zdrojový kód 4.4: Ukázka konfigurace flot

```
$.plot($("#flot"), [{
  label: "Flows",
  color: "rgb(255, 0, 0)",
  lines: {
    show: true,
    fill: true,
    fillColor: "rgba(255, 0, 0, 0.8)"
  },
  shadowSize: 1,
  data: points
}],
{
  legend: {
    show: true,
    position: "nw"
  },
  xaxis: {
    mode: "time",
    ticks: 5,
    timeformat: "%d/%m/%y\t %H:%M"
  }
});
```

Další volby jsou volitelné. Souřadnicové osy mohou být nastaveny volbami `xaxis` a `yaxis`. U definice souřadnicových os grafu lze zvolit, zda vynášené hodnoty budou pouze čísla nebo čas. Toto nastavení je ovlivnitelné volbou `mode`. Pokud budou na ose časové hodnoty, je vhodné použít volbu `timeformat`, jejíž parametrem je řetězec definující čas (např. `'%h:%m'` pro hodiny a minuty). Pro jednotlivé osy můžeme nastavit jejich umístění (`position`), barvu a parametry jednotlivých bodů na ose. Tyto body se nazývají *ticks* a kromě jejich velikosti a barvy lze pro jejich formát ještě zvolit vlastní funkci předanou volbě `tickFormatter`. Flot umožňuje vytvářet logaritmické či exponenciální souřadnicové osy nebo poskytuje možnost vytvořit osy pro každou ze zobrazených datových řad.

Legenda grafu může být definována volbou `legend` s parametrem `show = true`. Nicméně lze definovat také její barvu a průhlednost pozadí, pozici a dokonce i umístění mimo graf (volbou `container` spojenou s libovolným objektem jQuery či elementem DOM). Pokud chceme upravit popisky datových řad, aby např. fungovaly jako webové odkazy, můžeme si nadefinovat vlastní funkci a flot o této skutečnosti informovat volbou `labelFormatter`.

Nakonec graf ještě opatříme titulkem pomocí volby `label` a případně můžeme zvolit některá z *vizuálních* vylepšení jako jsou barva grafu (`color`), stín (`shadowSize`) či zda-li a jakým způsobem má graf reagovat na přejetí kurzorem anebo kliknutí. Ukázka grafu na obrázku 4.5 byla vytvořena parametry uvedenými v konfiguraci 4.4.



Obrázek 4.5: Ukázka grafu v flot

V současně době jsou k dispozici i rozšiřující pluginy pro koláčové grafy a zvětšování vybrané oblasti grafu. Tyto a některé další lze najít na wiki projektu v tématu Plugins [51]. Plugin flot je vysoce konfigurovatelný, ovšem mnohé funkce si musíte napsat sami. To však zjevně nebraní mnohým, aby jej zapojili do svých projektů, o čemž se můžete přesvědčit na wiki projektu v tématu FlotUsage [52].

4.1.4 Highcharts

Highcharts [45] je knihovna napsaná v JavaScriptu pro vytváření grafů, spolupracující s jedním z frameworků jQuery [48], MooTools [70] nebo Prototype [71]. Highcharts umožňuje volné použití, pokud se nejedná o komerční účely.

Poskytuje podporu pro všechny moderní webové prohlížeče prostřednictvím formátu SVG. Pro Internet Explorer do verze 9, který nepodporuje HTML značku `canvas` ani SVG, je spolupráce s Higcharts vyřešena s pomocí VML. Jedná se o „client-side“ aplikaci, kde je generování grafu ponecháno na klientském prohlížeči. Highcharts umožňuje modifikovat grafy i po jejich vytvoření (asynchronními zpětnými dotazy serveru na pozadí, tzv. AJAX).

Highcharts disponuje velice rozsáhlou a dobře zpracovanou dokumentací [46] ke svému API, která je protkána demonstračními příklady grafů s danou funkcionalitou (vytvořených v prostředí pro testovaní JavaScriptu a frameworků na něm postavených – jsFiddle [37]). Stejně tak dokumentace obsahuje návod pro samotné zprovoznění knihovny či představení klíčových funkcí. Nyní si ukážeme některé volby, které jsou využitelné při vizualizaci NetFlow toků a zjištěných anomálií.

Vložením knihoven jQuery, Highcharts a případných rozšiřující pluginů do hlavičky HTML dokumentu zajistíme dostupnost jimi poskytovaných funkcí v celém dokumentu. Následně označíme v těle HTML dokumentu místo pro vykreslení grafu.

Před konfigurací samotného grafu si ještě představíme několik globálních konfiguračních voleb, na které bychom neměli zapomenout. Globální parametry prostředí Highcharts můžeme upravovat prostřednictvím funkce `Highcharts.setOptions()`. Uvedením sekce `global: {}` a volby `useUTC: false` lze vypnout podporu UTC, abychom mohli používat lokálního čas na časových osách. Sekce `lang: {}` poskytne možnost upravit nastavení lokalizačních voleb, jako jsou názvy dnů a měsíců či oddělovače číslic.

Posledním krokem před konfigurací grafu je získání dat a jejich upravení pro Highcharts. Data je možné předat výčtem hodnot, ve formátu JSON, v XML či **CSV**. Pro zpracování externích dat se využívá jQuery funkce `.get` a vlastní funkce pro analýzu jejího výstupu. S formátem JSON však mohou nastat menší komplikace, neboť funkce `json_encode` a `json_decode` používané pro práci s ním jsou definovány až pro jazyk PHP verze 5.2.0. Tato verze bohužel ještě není zcela běžná pro některé stabilní serverové distribuce (např. CentOS 5.6) a je tak potřeba využít např. `jsonwrapper` postavený kolem `PEAR JSON knihovny`. Jestliže máme připravena data, jsme již schopni nadefinovat samotný graf.

Zdrojový kód 4.5: Ukázka konfigurace Highcharts – první část

```
chart = new Highcharts.Chart({
  chart: {
    renderTo: 'plot',
    defaultSeriesType: 'line',
    zoomType: 'x'
  },
  title: {
    text: 'Highcharts'
  },
  subtitle: {
    text: 'Flows'
  },
  legend: {
    align: 'center',
    verticalAlign: 'top',
    y: 50,
    floating: true
  },
  tooltip: {
    crosshairs: {
      width: 2,
      color: 'white',
      dashStyle: 'shortdot'
    },
    formatter: function() {
      return '<b>Date:</b> '
      + Highcharts.dateFormat \
        ('%d.%m.%Y', this.x)
      + '<br>' + '<b>Time:</b> '
      + Highcharts.dateFormat \
        ('%H:%M', this.x)
      + '<br>'
      + '<b>' + this.series.name
      + ':</b>' + this.y;
    }
  },
  plotOptions: {
    series: {
      marker: {
        enabled: false,
        states: {
          hover: {
            enabled: true
          }
        }
      }
    }
  },
```

Jaké typy grafů Highcharts umožňuje vytvořit? K dispozici máme bodový, spojnicový, plošný, sloupcový a koláčový graf. Pokud si přejeme, aby čára mezi jednotlivými body procházela plynule, můžeme použít místo úseček standardně použitých ve spojnicovém grafu křivku popsanou polynomem s aproximací, tzv. spline. Implicitní volbou je spojnicový graf.

Pro definici samotného grafu existuje několik parametrů, jejichž použití je povinné. Do této kategorie zařadíme volbu `renderTo` v sekci `chart: {}` spojující místo určující pozici grafu v rámci webové stránky a definici grafu. Dalším povinným parametrem pak bude `data` v sekci `series: {}` definující data k zobrazení. Použití ostatních parametrů je volitelné.

V sekci `chart {}` můžeme dále definovat typ grafu. Na výběr máme jednu z následujících možností: `scatter`, `line`, `area`, `bar` a `column`, `pie` a také `spline` a `areaspline`, které označují výše zmíněné grafy ve stejném pořadí. V rámci jednoho obrázku lze použít více různých grafů.

To, co graf mění ze spleti čar a bodů na vizualizaci dat s určitou informační hodnotou, jsou souřadnicové osy. Highcharts poskytuje možnost separátních os pro jednotlivé datové řady, a také inverzní souřadnicové osy. Pro jejich nastavení poskytuje Highcharts dvě sekce, `xAxis` a `yAxis`. Společnými parametry pro obě osy jsou typ, který může nabývat hodnot (`date`, `linear`), název, popisky bodů a jejich barvy i rotaci. Pokud hodláme vynášet na osu časové hodnoty, je vhodné nastavit, v jakém formátu budou zobrazovány. Jeho specifikaci zajistíme funkcí `Highcharts.dateFormat`. Mimo jiné také nabízí volby pro zvýraznění určité hranice nebo oblasti (`plotLines`, `plotBands`) využitelné např. pro rozlišení noci a dne či označení víkendů a volbu `startOfWeek` definující zda týden začíná v neděli dle zemí Severní Ameriky nebo v pondělí dle evropských zemí.

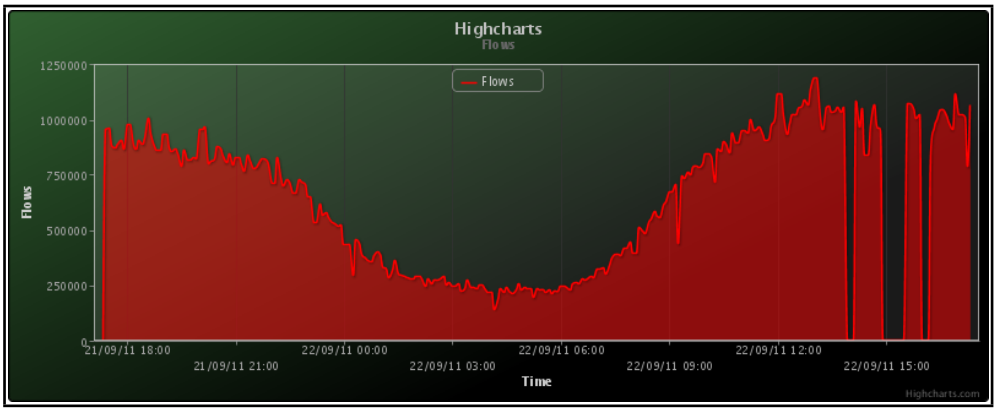
Zdrojový kód 4.6: Ukázka konfigurace Highcharts – druhá část

```
  xAxis: {
    title: {
      text: 'Time'
    },
    type: 'datetime',
    labels: {
      formatter: function() {
        return Highcharts.dateFormat \
        ('%d/%m/%y %H:%M', this.value);
      },
      staggerLines: 2
    }
  },
  yAxis: [{
    title: {
      text: 'Flows'
    },
    labels: {
      formatter: function() {
        return this.value;
      }
    }
  }],
  series: [{
    name: 'Flows',
    color: '#ff0000',
    data: points
  }]
});
```

Jednotlivé datové řady a jejich parametry nastavíme v sekci `series: {}`. Mimo již zmiňovaného přiřazení dat jde o název (`name`), který se následně zobrazí v legendě, typ grafu a barvu, případně přiřazení jedné z dříve definovaných os. V případě spojnicového grafu či polynomiální křivky lze zvolit typ čáry a její tloušťku.

Společné parametry po více grafů stejného typu lze nastavit v sekci `plotOptions: {}`. Do této kategorie patří vrstvení spojnicových, sloupcových či plošných grafů tzv. `stacking`, který může být normální či procentní.

Pro podrobné informace o grafu a datech poskytuje Highcharts hned několik možností. V první řadě se jedná o titulek a podtitulek (sekce `title`, `subtitle`). Dále pak legendu (`legend: {}`, u které můžeme upřesnit parametry jako umístění (pevné/plovoucí), barvu, rámeček, stín a vybrat zda jednotlivé série dat umístíme pod sebe či vedle sebe (`layout`)). V neposlední řadě jsme schopni přidat nápovědu k jednotlivým bodům (`tooltip`) a rozhodnout o jejím stylu i obsahu. Dále lze definovat kurzor (`crosshairs`) usnadňující výběr jednotlivých bodů. Více se o datech v grafu dozvíme i díky možnosti zvětšovat vybrané oblasti. Highcharts umožňuje povolit zvětšování jen v rámci osy x či y anebo pro obě osy. Ukázka grafu na obrázku 4.6 byla vytvořena parametry uvedenými v konfiguracích 4.5 a 4.6.



Obrázek 4.6: Ukázka grafu v Highcharts

Highcharts dovoluje také vytvářet dynamické grafy, kde s pomocí technologie AJAX a metod `update`, `remove`, `addSeries` a `addPoint` umožňuje upravovat data obsažená v grafu a jeho vlastnosti. V neposlední řadě umí Highcharts měnit i svůj vzhled za pomoci zásuvných modulů (ve formě .js), kdy poskytuje tzv. témata.

Mezi pokročilé vlastnosti zahrneme i modul pro export. Po vložení modulu `export.js` do kódu webové stránky, je Highcharts schopen využít PHP skript na vzdáleném či lokálním serveru pro export grafu. Highcharts poskytuje tuto službu na svém serveru zdarma. Použit je produkt Batik [34] (PHP/Java), který obrázek ve formátu **SVG** získaný metodou POST převede na jeden z podporovaných formátů, kterými jsou **PDF**, **PNG** a **JPG**. V nastavení Highcharts je možno definovat název souboru a standardní typ generovaného obrázku.

4.1.5 pChart

`Autor: Jean-Damien POGOLOTTI`

pChart [67] je knihovna v jazyce PHP pro vytváření grafů, postavená na knihovnách GD (podpora dynamického vytváření obrázků ve formátech **GIF**, **PNG** a **JPG**) a FreeType (font engine), která podporuje vyhlazování (tzv. anti-aliasing). Jedná se o „server-side“ aplikaci, z čehož plyne, že o tvorbu grafu se stará pouze server a výsledný obrázek může být uložen lokálně, poslán do prohlížeče na klientské straně či vložen do souboru **PDF**. Navíc nám generování grafů na straně serveru přináší možnost neinteraktivního spouštění např. z `cronu` [77] v Unixu či pomocí `at` [56] z Windows.

V této kapitole jsou zmíněny dvě vývojové větve této knihovny, neboť v době testování byla k dispozici pouze verze 1.x [66], mezitím však vyšla kompletně přepsaná verze 2.x [67]. Nová verze přináší velká vylepšení a rozšíření funkcionality, a proto se pokusím zaměřit pouze na verzi 2.x a informace o verzi 1.x ponechávám čtenáři jako přílohu této práce, kterou můžete nalézt v dodatku F na straně 89.

pChart se skládá z několika tříd, konkrétně se jedná o **pData** zajišťující očekávaný formát vstupních dat, **pDraw** vykreslující rozličné typy grafů a **pImage**, která zpřístupňuje samotný objekt grafu následně modifikovaný metodami dříve jmenovaných tříd. Použití těchto tří tříd je povinné pro každý graf. pChart však ještě obsahuje třídu **pCache**, která umožňuje znovupoužití již vygenerovaných grafů a přispívá tak k šetření zdrojů. Další třídy se týkají speciálních rozšiřujících modulů, jako např. **pStock** pro grafy akcií.

Pro svoji správnou funkci požaduje pChart interpret jazyka PHP alespoň ve verzi 4.x, ovšem verze 5.x dokáže významně urychlit práci knihovny. Před započetím práce s knihovnou je třeba zahrnout její zdrojový kód do webové stránky.

Data je třeba připravit pro použití v grafu, o což se postará třída pData. Data je možné získat z SQL, souboru **CSV**, a samozřejmě také ručně nadefinovat. Nejdříve je potřeba vytvořit objekt třídy pData, s nimž budeme dále pracovat. Nejdůležitější funkcí této třídy je bezpochyby `addPoints` umožňující přidání jednoho nebo více bodů do datové řady. Další funkce této třídy umožňují vybrat barvu pro datovou řadu, přiřadit data k jednotlivým osám, zvolit jejich umístění, typ a popisky.

pChart umí poskytnout základní statistické údaje o datech, mezi které patří funkce pro výpočet průměrů (medián, geometrický a harmonický) a směrodatné odchylky. Mimo jiné umožňuje vytvořit novou datovou řadu aplikací matematické funkce/vzorce pomocí funkce `createFunctionSerie`. V neposlední řadě zvládne i datové řady s chybějícími body (např. nedefinovaná hodnota).

Jestliže jsme nadefinovali data, můžeme vytvořit objekt třídy pImage pro samotný graf. V tomto místě přiřadíme k objektu grafu dříve připravená data a určíme celkový rozměr obrázku. Následně vymezíme část obrázku pro graf pomocí funkce `setGraphArea`. Další z povinných funkcí je `drawScale` zajišťující vykreslení souřadnicových os do obrázku. U této funkce se na chvíli zastavíme a podíváme se detailněji na její parametry.

Standardně pChart prohlédne definovaná data a vyhodnotí jejich maximum a minimum, z čehož následně vytvoří popisky pro osy. Nicméně zřejmě pro přehlednost připojí na ose y prázdnou oblast pod a nad tyto zjištěné hodnoty. Osa y tak začíná např. hodnotou -20 místo očekávané nuly. Toto chování lze odstranit uvedením parametru `SCALE_MODE_START0`.

Zdrojový kód 4.7: Ukázka konfigurace pChart 2.x

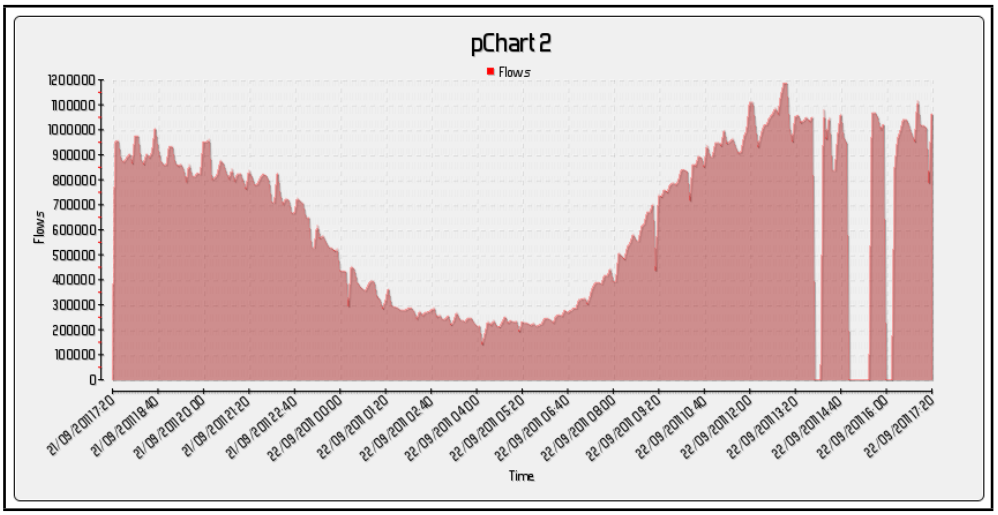
```
for ($i=0; $i<count($data); $i+=2 ) {
        $flows_osa_x[] = $data[$i];
        $flows_osa_y[] = $data[$i+1];
}
$FlowDataSet->AddPoints($flows_osa_y, "Flows");
$FlowDataSet->SetAxisName(0, "Flows");
$ssSet = array("R"=>255,"G"=>0,"B"=>0,"Alpha"=>100);
$FlowDataSet->setPalette("Flows",$ssSet);
$FlowDataSet->AddPoints($flows_osa_x, "Time");
$FlowDataSet->setSerieDescription("Time", "Time");
$FlowDataSet->setAbscissa("Time");
$FlowDataSet->setAbscissaName("Time");
$FlowDataSet-> \
setXAxisDisplay(AXIS_FORMAT_DATE, "d/m/Y␣H:i");

$Flows = new pImage(850,430,$FlowDataSet);
$Flows->setFontProperties(array( \
"FontName"=>"Forgotte.ttf", "FontSize"=>11));
$Flows->setGraphArea(80,50,810,330);
$FillRecSet = array( \
"R"=>240,"G"=>240,"B"=>240,"Alpha"=>100);
$Flows->drawFilledRectangle(7,7,843,423, $FillRecSet);
$RecSet = array("R"=>0,"G"=>0,"B"=>0,"Alpha"=>100);
$Flows->drawRoundedRectangle(5,5,845,425,5,$RecSet);
$Flows->setShadow(TRUE, \
array("X"=>1,"Y"=>1,"R"=>0,"G"=>0,"B"=>0,"Alpha"=>20));
$scaleSettings = array( \
"Mode"=>SCALE_MODE_START0, "XMargin"=>10, \
"YMargin"=>10,"Floating"=>TRUE,"DrawSubTicks"=>TRUE, \
"LabelRotation"=>40,"LabelSkip"=>15);
$Flows->drawScale($scaleSettings);
$Flows->drawAreaChart();
$TextSettings = array( \
"R"=>0,"G"=>0,"B"=>0,"Angle"=>0,"FontSize"=>20);
$Flows->drawText(400,40,"pChart␣2",$TextSettings);
$Flows->drawLegend(415,50,array( \
"Style"=>LEGEND_NOBORDER,"Mode"=>LEGEND_HORIZONTAL));
$Flows->Render("peakock_test/flows2.png");
```

Pokud je hodnot vynášených na osu x příliš mnoho a jejich popisky by pozbyly čitelnosti, pChart dovoluje uvést pouze každý x-tý bod na ose x, pomocí parametru `LabelSkip`. Pokud jsou data vynášená na osu x např. časem, pChart umožňuje nastavit uvedení popisku jen tehdy, jestliže se liší od předchozí hodnoty. Jestliže jsou i po těchto úpravách popisky stálé nečitelné, lze použit ještě funkci pro jejich rotaci `LabelRotation`. Mezi další parametry pro nastavení souřadnicových os patří jejich pozicování, obarvení či označení os šipkami.

Doposud jsme si připravili data a prostředí pro jejich zobrazení, nyní můžeme vytvořit samotný graf. Mimo typických zástupců grafů ve 2D prostoru, jako jsou bodový a spojnicový graf (scatter, line), plošný graf (area), graf ohraničený kubickou křivkou (neboli polynomem třetího stupně, cubic curve), sloupcový a koláčový graf (bar, pie), nabízí pChart i mnohem neobvyklejší varianty grafů a vizualizačních prvků. Do této kategorie patří např. grafy akcií (stock), diagramy průběhu (progress), obrázkové mapy nebo čárové kódy (barcode).

Dvourozměrným prostorem ovšem možnosti knihovny pChart nekončí. Jsme schopni vytvořit 3D koláčový či prstencový graf. Autor plánuje rozšířit možnosti 3D grafů o sloupcové, spojnicové a křivkové grafy. Toto téma se řeší na diskuzním fóru projektu [23]

Vylepšení přehlednosti a srozumitelnosti grafu můžeme dosáhnout přidáním rámečku, legendy s jednotlivými datovými řadami, popisků významných bodů nebo titulkem grafu. Grafickou stránku grafu lze ovlivnit přidáním stínu, průhlednosti či zrcadlového efektu. pChart navíc umí zvýraznit určitou část grafu dle definované podmínky. Ukázka grafu na obrázku 4.7 byla vytvořena parametry uvedenými v konfiguraci 4.7.



Obrázek 4.7: Ukázka grafu v pChart 2.x

pChart využívá mnoho velkých společností, mezi které patří např. HP, Sony, British Telecom, Intel, Airbus. Kompletní seznam naleznete na webové stránce projektu pChart [67].

4.1.6 Srovnání

V následujících několika odstavcích se pokusím o srovnání vlastností jednotlivých projektů. Úvodem porovnáme projekty z hlediska softwarových licencí pod nimiž jsou vydány. Shrneme si údaje o četnosti vydávání nových verzí i vyzrálosti projektů a dále porovnáme knihovny dle programovacího jazyka, ve kterém byly vytvořeny.

Postupně si ukážeme, jak jsou jednotlivé projekty dokumentovány a zda je možno požadovanou informaci najít rychle a pohodlně. Zastavíme se také u formy poskytované dokumentace a její aktuálnosti. Díky dokumentaci můžeme knihovnu používat a nakonfigurovat tak, abychom mohli vyprodukovat graf. Porovnáme tedy knihovny z hlediska rozsáhlosti a přehlednosti konfigurace.

V dalším kroku budu hodnotit vzhled a vizualizační možnosti obecně. Zhodnotím také přehlednost výstupu jednotlivých knihoven. A nakonec se zaměřím na srovnání knihoven z pohledu náročnosti na výpočetní zdroje. Poté vyberu na základě tohoto srovnání nejvhodnější projekt pro vizualizaci síťových toků a anomálií.

Funkcionalita a licence

Nástroj	Licence	Verze	Datum vydání	Četnost vydávání	Dostupnost
flot	MIT	0.7	03/2011	malá	web, google
highcharts	CC-BY-NC	2.1.6	07/2011	velká	web, GitHub
jqPlot	GPL, MIT	1.0.0b2_r947	10/2011	velká	web, Mercurial
pChart	GPLv3	2.1.3	10/2011	střední	web, SF
RRDtool	GPL	1.4.5	10/2010	střední	web, SVN

Tabulka 4.1: Vizualizační knihovny – licenční podmínky, velikost kódu, verze a periodicita vydávání

Četnost: malá – 1-2x ročně, střední – několikrát ročně, velká – několikrát měsíčně

Jak můžeme vidět v tabulce 4.1, všechny z testovaných knihoven jsou poskytovány pro volné použití, modifikace a opětovné šíření. Až na projekt Highcharts, který je uveden pod licencí Creative Commons (CC-BY-NC 3.0 [7]), jsou zbývající projekty uvedeny pod licencemi GNU GPL nebo MIT, které jsou si velmi podobné. Licence CC-BY-NC vyžaduje uvedení autora a zakazuje bezplatné komerční využití. Pokud chceme Highcharts používat v komerčních projektem, nezbývá než zakoupit licenci. Kromě těchto omezení dovoluje zdrojový kód upravovat i šířit. Samozřejmostí je opětovné uvedení podmínek licence.

V rámci nekomerčního projektu vizualizace síťových toků a anomálií v prostředí univerzity lze vybrat libovolný projekt. Jediným problémem je neslučitelnost licence CC-BY-NC a GPL a následná potřeba duální licence v případě volby Highcharts.

Tatáž tabulka také uvádí vyzrálost jednotlivých projektů a periodicitu vydávání nových verzí. Projekt RRDtool většinou přináší jen opravy chyb. Ostatní projekty se zdají být aktivní, jak lze ověřit dle data vydání poslední verze a vychází několikrát měsíčně či ročně. Každý z projektů používá kromě přístupu přes web i některý ze systému správy verzí. Zdrojové kódy všech projektů se svojí velikostí v komprimované formě pohybují maximálně na hranici jednoho megabytu.

Tabulka 4.2 shrnuje vizualizační knihovny z pohledu programovacích jazyků a podle provádění svého kódu na serveru či klientu. RRDtool je ze zvolených nástrojů nejstarší, pracující na straně serveru, napsaný v jazyce C a není primárně určen pro vizualizaci dat, nýbrž pro jejich uchovávání. Ovšem jako jediný nabízí API pro další programovací jazyky (viz tabulka 3.2).

Nástroj	Programovací jazyk	Vykreslování
flot	Javascript	klient
highcharts	Javascript	klient
jqPlot	Javascript	klient
pChart 1.x	PHP	server
pChart 2.x	PHP	server
RRDtool	C	server

Tabulka 4.2: Vizualizační knihovny – programovací jazyk, generování grafu

Dalším projektem, provozovaným na straně serveru, je pChart napsaný v jazyce PHP. Vyžaduje tedy funkční web server s podporou PHP. Zbývající tři projekty, totiž Highcharts, flot a jqPlot, jsou napsány v jazyce JavaScript a jejich kód se spouští na klientské straně v prohlížeči. Všechny uvedené knihovny v JavaScriptu jsou závislé na knihovně jQuery. Highcharts případně dovoluje využití alternativních knihoven Mootools či Prototype.

Dokumentace

Srovnáním dostupné dokumentace jednotlivých projektů podstoupíme první krok k volbě vizualizačního nástroje. Podívejme se tedy, co jednotlivé projekty nabízejí.

Plugin **jqPlot** nabízí HTML dokument *Usage*, který nás provede základními kroky vedoucími ke zprovoznění knihovny. Dalším dokumentem je *jqPlot Options*, který poskytuje informace o některých konfiguračních volbách, avšak hned na jeho začátku autor návštěvníka informuje o neaktuálnosti a nekompletnosti tohoto dokumentu. jqPlot může také nabídnout CSS strukturovanou dokumentaci API. Ta se zdá být kompletní a je rozčleněna do sekcí. Jednotlivé rozšiřující pluginy pro jqPlot obsahují svoji vlastní dokumentaci. V neposlední řade jqPlot také nabízí několik příkladů demonstrujících dostupné funkce.

Knihovna **flot** nabízí krátké *Readme*, kde je ukázáno, jak ji zprovoznit. Příklady s grafy jsou také přiloženy, jen jejich zdrojový kód není zobrazen přímo do webového dokumentu, ale je nutné nahlédnout do zdrojového kódu stránky. Mimo dokumentace k API můžeme ještě nalézt dokument často kladených dotazů (FAQ). Dokumentace je bohužel skryta v jediném dokumentu, a to pouze jako textová, bez jakéhokoli zvýraznění.

Další JavaScriptová knihovna **Highcharts** nabízí dokument *How to use*, který nás provede od instalace, přes základní konfiguraci, až k možnostem využití knihovny a nezapomene zmínit odkaz na kompletní dokumentaci API. Referenční dokumentace je vytvořená jako kontextové menu za pomoci kaskádových stylů a JavaScriptu. Valná většina konfiguračních voleb poskytuje éci *Try it*, která odkazuje do testovacího prostředí jsFiddle, kde je kromě samotného grafu vypsán i HTML kód, CSS styl a konfigurační volby. Tím ovšem možnosti tohoto zobrazení nekončí, neboť jednotlivé parametry lze editovat. V neposlední řadě obsahuje Highcharts galerii příkladů (Demo gallery), která nabízí ukázky všech základních typů grafů, včetně možnosti vybrat si téma (theme).

PHP knihovna **pChart** je zastoupena ve dvou vývojových větvích, jak jsem již zmínil v kapitole 4.1.5 na straně 37. Verze 1.x má dokumentaci rozdělenu dle jednotlivých tříd, dále obsahuje dokument často kladených dotazů a příklady (včetně konfigurací) pro základní i pokročilé případy grafů. Verze 2.x pak přidává prohlídku funkcí (Features) a samotná dokumentace k API je rozdělená dle jednotlivých funkcionalit a pluginů. Dokumentace je umístěna ve wiki projektu, která kromě prohlížení, nabízí i stažení ve formátu PDF a sledování změn. Balíček pChart 2.x také obsahuje příklady grafů, nicméně než si je budete moci prohlédnout, je potřeba nechat vygenerovat obrázky. Jednou z možností je nasměrovat prohlížeč do adresáře *examples*.

Nástroj **RRDtool** bere vizualizaci pouze jako funkci přidané hodnoty, a tak ani dokumentace není příliš rozsáhlá. Její forma připomíná manuálovou stránku systému Unix, a je tedy celkem přehledná. Dokumentace je rozdělena na jednotlivá témata, mezi která patří např. příkaz `graph`, formát dat, jazyk pro definici DEF prvků či popis rozhraní DBI.

Vzhled

Hodnocení vzhledu může být dosti subjektivní. Může na něj být nazíráno pouze jako na vizuální prezentaci či jako na posouzení praktické využitelnosti a přehlednosti rozhraní. Ideální je oba pohledy zkombinovat. Pro utvoření vlastní představy si prohlédněte obrázky grafů v podání knihoven na konci jednotlivých kapitol spolu s jejich konfigurací.

Knihovny postavené na spolupráci JavaScriptu a jQuery, totiž flot, Highcharts a jqPlot, mají srovnatelný vzhled. **Highcharts** navíc poskytuje možnost volby tématu (themes),

který umožňuje kompletní změnu vzhledu grafu. Silnou stránkou těchto knihoven je možnost interakce s již vytvořenými grafy. Ať už jde o kurzor, okno nápovědy, zvětšení části grafů či dokonce přesun jednotlivých bodů.

Grafy produkované projektem **pChart** ve verzi 1.x umožňují použít vyhlazování (anti-aliasing), průhlednost většiny prvků, obrázek jako pozadí. Na druhou stranu, pChart 1.x podporuje pouze osm slotů pro jednotlivé barvy, což by mohla být překážka při větším počtu datových řad v jednom grafu. pChart 2.x využívá možnosti grafické knihovny GD mnohem lépe, což potvrzuje možností vykreslit více druhů specializovaných grafů, šipek na souřadnicových osách anebo použití zrcadlového efektu. Jistěže jde pouze o detaily, ovšem právě tyto detaily dotvářejí lepší celkový dojem z grafu, a tak zvyšují jeho přehlednost.

Co se grafické části týče, poněkud opomenutý **RRDtool** nabízí pouze základní zobrazení čar a oblastí, případně jejich vrstvení. Svou práci dělá dobře a rychle. Ovšem ve srovnání z výše uváděnými je bohužel poněkud pozadu. Postrádá bodový graf, popisky jednotlivých bodů přímo v grafu, komplexnější nastavení souřadnicových os včetně rotace popisků a typu vynášených dat.

Konfigurace

Konfigurace **RRDtool** je poměrně přímočará, nicméně může být poněkud nepřehledná, neboť jde o jeden příkaz a jeho parametry, a tak postrádá nějakou formu strukturalizace. Bez přečtení dokumentace, případně struktury samotné RRD databáze, nemusíme získat jasnou představu o budoucím grafu.

Projekty napsané v JavaScriptu, tedy **jqPlot**, **Highcharts** i **flot**, mají podobně strukturovanou konfiguraci. Jedná se o definici objektů a jejich vlastností. Díky velice přehledné dokumentaci je konfigurace Highcharts lehce pochopitelná, a výsledek je pak sám o sobě dokumentující. Případný čtenář nemá problém přizpůsobit si ji pro své potřeby. Na druhou stranu, oproti ostatním projektům je poněkud rozsáhlá. Flot obsahuje vskutku minimalistickou konfiguraci, se kterou není problém dosáhnout vykreslení základních grafů. Dojde-li však na některou z pokročilých vlastností, nabízí flot vývojáři volnou ruku při vytváření vlastních funkcí obsluhujících chování některých objektů. Tento způsob se mi jeví bohužel dost nepřehledný a vyžadující více než uživatelské znalosti.

pChart také využívá objekty, které postupně modifikujeme. Volání probíhá způsobem *objekt → funkce (parametry)*, kde sice vhodným formátováním konfiguraci zpřehledníme, ale v porovnání s konfiguracemi knihoven v JavaScriptu *objekt: {vlastnost, vlastnost}* neposkytuje čtenáři kódu tolik komfortu. Pokud používáte editor nebo IDE se zalamováním řádků na 80. sloupci, může se stát občas konfigurace nepřehledná právě kvůli zalomení. Informace o rozsáhlosti konfigurace při definici demonstrovaného grafu časové řady shrnuje tabulka 4.3.

Rozsáhlost konfigurace pro vybraný průběh síťových toků						
nástroj	**flot**	**highcharts**	**jqPlot**	**pChart 1.x**	**pChart 2.x**	**RRDtool**
počet řádků	18	84	46	24	28	10
počet bytů	337	1406	880	1269	1463	277

Tabulka 4.3: Vizualizační knihovny – Rozsáhlost konfigurace

Výpočetní zdroje

Jistě podstatným kritériem pro výběr je rychlost zpracování a generování grafu, stejně tak jako náročnost na výpočetní zdroje, kde se snažíme minimalizovat využití CPU a paměti, případně prostoru na sekundární paměti.

Název souboru	MIME	Velikost [KB]	Doba přenosu [ms]
flot			
style.css	text/css	0,341	196
jquery.min.js	application/javascript	89,78	1280
jquery.flot.min.js	application/javascript	32,52	992
4 požadavky (125,40 KB), obsazená dynamická paměť: 2,81 MB, čas na procesoru: 4,83 s			
pChart 1.x			
test_pchart.php	text/html	0,532	4239,36
style.css	text/css	0,341	232
flows.png	image/png	30,11	809
3 požadavky (30,98 KB), obsazená dynamická paměť: 1,84 MB, čas na procesoru: 2,48 s			
pChart 2.x			
test_pchart2.php	text/html	0,537	10024,96
style.css	text/css	0,341	265
flows.png	image/png	66,21	1075,20
3 požadavky (67,09 KB), obsazená dynamická paměť: - MB, čas na procesoru: - s			
Highcharts			
test_highcharts.php	text/html	3,17	435
style.css	text/css	0,341	243
jquery.min.js	application/javascript	89,78	2252
highcharts.js	application/javascript	77,45	2048
dark-green.js	application/javascript	3,24	417
exporting.js	application/javascript	6,69	694
6 požadavků (180,67 KB), obsazená dynamická paměť: 3,52 MB, čas na procesoru: 5,35 s			
jqPlot			
test_jqplot.php	text/html	3,06	428
jquery.jqplot.min.css	text/css	3,27	261
style.css	text/css	0,342	374
jquery.min.js	application/javascript	89,78	2396,16
jquery.jqplot.min.js	application/javascript	107,70	2611,20
jqplot.canvasTextRenderer.min.js	application/javascript	17,18	1443,84
jqplot.canvasAxisLabelRenderer.min.js	application/javascript	4,35	996
jqplot.canvasAxisTickRenderer.min.js	application/javascript	4,80	705
jqplot.dateAxisRenderer.min.js	application/javascript	5,61	796
jqplot.cursor.min.js	application/javascript	18,38	2037,76
jqplot.logAxisRenderer.min.js	application/javascript	8,02	1167,36
11 požadavků (262,49 KB), obsazená dynamická paměť: 3,53 MB, čas na procesoru: 18,11 s			
RRDtool			
test_rrdtool.php	text/html	0,529	578
style.css	text/css	0,341	253
flows.png	image/png	18,76	581
3 požadavky (19,63 KB), obsazená dynamická paměť: 1,84 MB, čas na procesoru: 4,06s			

Tabulka 4.4: Vizualizační knihovny – využití zdrojů na straně klienta (webový prohlížeč)

Pro účel měření bylo využito rozšíření **firebug** [57] pro prohlížeč Firefox a **devtools** [42] pro prohlížeč Google Chrome. Tyto nástroje jsou schopny poskytnout informace o velikosti

přenášených souborů, době jejich přenosu (včetně DNS dotazu) a v neposlední řadě i obraz paměti zabrané stránkou s grafem a profil využití procesoru.

Tabulka 4.4 obsahuje data vzniklá při měření objemu dat, doby přenosu pro jednotlivé grafy, času využití CPU a velikosti obsazené dynamické paměti. Všechny příklady používaly stejná data. Jednalo se o jeden den průběhu síťových toků ze sond na síti Masarykovy univerzity, který obsahuje 288 hodnot odpovídající pětiminutovým intervalům. Testovací soubory jsou přiloženy na připojeném disku CD. Jejich obsahem je vždy pole výše zmíněných dat, inicializace skriptu či knihovny, externí styl CSS a kontejner `<div>` určující místo grafu na stránce.

Pro testování bylo využito procesoru Intel T2300 s frekvencí 1,6 GHz a 1GB RAM. Software na klientské straně představoval OS Debian GNU/Linux wheezy/sid s jádrem 2.6.39-4 a prohlížeči Firefox v8.0 (včetně pluginu Firebug v1.9.0b3) a Google Chrome v17.0.963.0-dev (včetně vestavěného pluginu devtools). Na straně serveru se jednalo webový server Apache 2.2.21, PHP 5.3.8-1+b1 a operační systém Debian GNU/Linux wheezy. Testování proběhlo za pomoci asynchronního připojení do sítě Internet o rychlosti 256 kB/2 MB a modemu CDMA Anydata ADU-300H.

Závěr

Dle předcházejících informací a měření byl pro vizualizaci dat ze síťového provozu Masarykovy univerzity získaného za pomoci NetFlow sond vybrán projekt **Highcharts**. Díky své výborné dokumentaci, přehledné konfiguraci a vzhledu. Oproti projektům jako **RRDtool** či **flot** je sice náročnější na přenosovou kapacitu, nicméně to lze při kapacitách dnešních internetových přípojek zanedbat. Nevýhodou zůstává větší náročnost na klienta při generování grafu.

Kapitola 5

Entropie za pomoci komprese

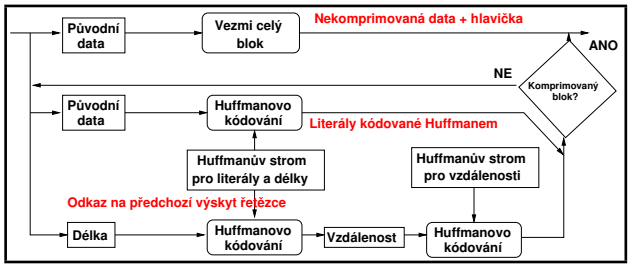
V této kapitole se zaměříme na entropii v síťových tocích, konkrétněji na entropii v nich se vyskytujících IP adres a portů. Definujeme pojem entropie. Dále bude popsána spojitost mezi entropií a kompresí. Po představení jednotlivých uvažovaných kompresních algoritmů a jejich vlastností provedeme jejich srovnání z pohledu úspěšnosti komprese, času potřebného k jejímu dokončení a také náročnosti na výpočetní zdroje. Toto srovnání nám bude sloužit jako vodítko při výběru optimálního kompresního algoritmu.

Obecně se symboly ve zprávě, v našem případě IP adresy a porty, mohou vyskytovat s různou četnosti. Na rozdíl od přirozeného jazyka jako je angličtina nebo čeština, kde je tato četnost daná, ovlivňuje entropii v síťových tocích právě způsob využití sítě. Na jednom konci bude stát běžná aktivita uživatelů při práci, kdesi uprostřed nalezneme sítě pro P2P sdílení či počítačový útok a na konci opačném např. šíření internetového červa nebo DoS případně DDoS útok.

Jak již bylo uvedeno výše, pomocí komprese můžeme získat odhad entropie. Tento fakt můžeme využít v pluginu pro zjišťování anomálií v grafech entropie síťových toků. Bylo dokázáno [29], že tato metoda, obzvláště je-li použita pro IP adresy a porty, napomáhá odhalit odchylky od normálního stavu provozu. Pokud se jedná o útok, často se rapidně sníží entropie, neboť provoz míří na konkrétní stroje, komunikuje se s určitými porty či se využívá pouze určitého protokolu. Rozhodli jsme se tuto metodu implementovat v praxi do pluginu pro systém NfSen, a ověřit tak její použitelnost v síti Masarykovy univerzity.

5.1 Deflate

Deflate je bezztrátový kompresní algoritmus (viz schéma 5.1), který vymyslel Phil Katz pro svůj archivační nástroj PKZIP. Později byl specifikován v RFC 1951 [8] (request for comments, pod záštitou organizace IETF). Tento algoritmus není zatížen patenty (ačkoli původní varianta pro nástroj PKZIP byla patentována v roce 1991), a proto byl úspěšně použit v mnoha implementacích. Nejčastěji je spojován implementaci gzip, dle RFC 1952 [9] (GNU zip, autoři Jean-Loup Gailly a Mark Adler). Je



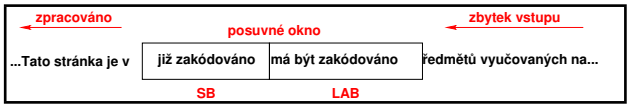
Obrázek 5.1: Schématický diagram algoritmu Deflate

postaven na kombinaci algoritmu LZ77 a Huffmanova kódování. Abychom si mohli přiblížit princip tohoto algoritmu, musíme se alespoň letmo seznámit s jeho částmi.

5.1.1 LZ77

Je také bezztrátový kompresní algoritmus, který navrhli Abraham Lempel a Jacob Ziv již v roce 1977. Kompresní metoda LZ77 [25] je slovníková, jednoprůchodová, adaptivní, asymetrická a komprese je mnohem náročnější než dekomprese.

Algoritmus je postaven na posuvném okně (viz obrázek 5.2, sliding window). Toto okno je rozděleno na dvě části, kde první z nich je vyhledávací buffer (search buffer, SB) a druhá je buffer dopředný (look-ahead buffer, LAB). Velikost těchto bufferů je konstantní a velikost dopředného bufferu musí být menší nebo rovna velikosti bufferu vyhledávacího.



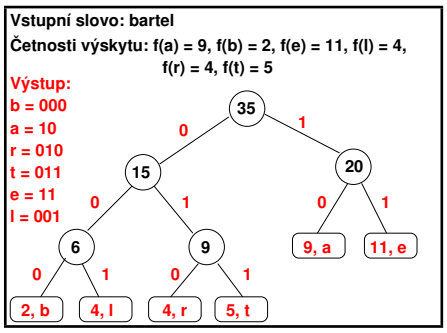
Obrázek 5.2: LZ77 posuvné okno

Po inicializaci je SB prázdný a LAB obsahuje část vstupu. Následně se v SB hledá nejdelší (poslední nalezený, při několika stejně dlouhých řetězcích) prefix z LAB (řetězec S) a to zprava doleva! Řetězec S začíná v SB a v některých případech může dokonce pokračovat až do LAB. Takovýto řetězec je následně uložen jako trojice (x, y, z), kde x označuje vzdálenost začátku S v SB, y je délkou řetězce S a z je symbol v LAB následující po S. V originální terminologii jde o trojici *offset, length* a *next symbol*. Jakmile se uloží tato trojice do výstupního proudu, posuvné okno se posune o $y + 1$ znaků vpravo. Jestliže není nalezeno nic, uloží se $(0, 0, symbol)$, nazývaný také literál.

Velikost posuvného okna se pohybuje v rozsahu 2^{12} - 2^{16} B (gzip 2^{15} B), zatímco velikost LAB je pouze v řádech 10^1 až 10^2 B (gzip 256 B). Pokud se zaměříme na jednotlivé hodnoty zmíněné trojice, můžeme zjistit. že offset je $log_2|SB|$, délka dosahuje $log_2(|LAB| - 1)$ a symbol je $log_2 A$, kde A je počet znaků abecedy. Pokud není nalezeno nic $(0, 0, znak)$, uloží mnohem více než 8 bitů pro hodnotu symbolu (např. 24 b).

5.1.2 Huffmanovo kódování

Druhou složkou algoritmu `deflate` je Huffmanovo kódování postavené na prefixech. Každý kód je několik bitů označující prvek abecedy (např. ASCII). Váha, která je stěžejním parametrem pro tento druh kódování, je relativní četností výskytu prvků, která byla zjištěna průchody či předem daná. Prvky označené váhou pak tvoří binární strom s větvemi označenými 0 či 1. Listy uzlů mají stejnou váhu. Pro Huffmanův strom (viz obrázek 5.3) platí, že kratší kódy jsou uloženy napravo a pokud se vyskytne více kódů stejné délky, řadí se lexikograficky. Cesta v takto zkonstruovaném Huffmanově stromu tvoří Huffmanův kód.



Obrázek 5.3: Huffmanův strom

5.1.3 Funkcionalita deflate

Nyní si již můžeme vysvětlit princip algoritmu `deflate` [11]. Algoritmus pracuje s vyhledávacím bufferem (SB) o velikosti 32 kB a dopředným bufferem (LAB) o velikosti 258 B. Pro vyhledávání v řetězci S je použita hašovací tabulka (což je pouze doporučená metoda, která však výrazně urychluje proces hledaní oproti vyhledávání lineárnímu), ve které jsou uloženy objevené trojice. Deflate však nevyužívá dva buffery, nýbrž buffer jediný o velikosti 32 kB, jehož rozdělení na SB a LAB je řízeno ukazatelem.

Deflate na rozdíl od původního LZ77 ukládá pouze dvojice (offset, length) jako token a pokud nenalezne žádnou shodu, zapíše pouze symbol (literál). Offset se v terminologii algoritmu deflate nazývá **vzdáleností** a pro zápis komprimovaného proudu se používá Huffmanovo kódování (konkrétně dvě kódové tabulky, jedna pro literály a délku, druhá pro offsety, resp. vzdálenosti).

Kódy pro symboly/literály [25]

- 0-255: pro symboly/literály
- 256: konec bloku
- 257-285: spojeno s extra bity, označuje délku 3-258 bytů.
- 286-287: nepoužívané, rezervované

Kódy pro délku a vzdálenost [25]

- 0-3: vzdálenost 1-4
- 4-5: vzdálenost 5-8, 1 extra bit
- 28-29: vzdálenost 16,385-32,768, 13 extra bitů
- 30-31: nepoužívané, rezervované

Literál je maximálně 1 B, tedy <0, 255>. Kodér hledá v první tabulce kód pro délku, ten nahradí Huffmanovým kódem a následně připojí 5bitový prefix a Huffmanův kód pro vzdálenost z druhé tabulky. Deflate používá upravenou variantu LZ77, která hledá shodu dvěma průchody (z nalezené shody vezme pouze první symbol/literál, a hledá delší shodu, najde-li takovou pošle symbol z prvního průchodu na výstup a za ním delší shodu). To jak se zachází s druhým průchodem, závisí na míře zvolené komprese. Komprese pomocí algoritmu Deflate probíhá v blocích různé délky, z nichž každý může být jednoho ze tří typů. Tyto typy jsou odvozeny od módů Deflate algoritmu.

První mód je tzv. „*bez komprese*“, který je používán hlavně pro náhodná či již zkomprimovaná data a přináší pouhé rozdělení na části. Tento mód nepoužívá kódové tabulky. Druhý mód „*komprese s pevně danými kódovacími tabulkami*“ využívá dvě kódové tabulky, viz výše uvedené a poslední mód „*komprese s proměnlivými kódovacími tabulkami*“, kde se komprese řeší adaptivně dle vstupu a vlastní kód se následně ještě zkomprimuje pomocí algoritmu RLE (run-length encoding, princip naleznete na straně 10).

Na začátek každého bloku je připojen Huffmanův strom pro literály/délku a pro vzdálenosti. Duplikované řetězce jsou nalezeny pomocí hašovací tabulky. Dekódování probíhá pomocí funkce inflate(), která vytváří několikaúrovňové vyhledávací tabulky pro Huffmanovy kódy. V praxi bývá první úroveň obvykle 9bitová, díky čemuž stačí pro kratší a tedy častější kódy pouze jeden dotaz.

Teď když už víme, jak deflate funguje, vraťme se ke schématu 5.1. To nám ukazuje tři možné cesty průchodu algoritmem, které odrážejí chování jednotlivých módů. Mód bez komprese označuje horní cestu. V případě samotného literálu na výstupu uvažujeme

střední cestu a dolní cesta řeší delší sekvence znaků. Nakonec zmiňme, že existuje i varianta Deflate64, která využívá 64 kB slovník. Deflate má mnoho implementací, mezi ty významnější patří PKZIP, ZIP, zlib/gzip [12] či 7-zip [65]. Mimo jiné existují i hardwarové implementace dekodéru i enkodéru.

5.2 LZMA a 7-zip

Algoritmus LZMA vymyslel Igor Pavlov společně se svým nástrojem pro kompresi a dekompresi – 7-zip. Nástroj samotný je koncipován jako open-source, podporující běh ve vláknech a hyperthreading. Jeho vlastnostmi se zde však nebudu zabývat, neboť nás zajímá hlavně algoritmus. Ten je podobný předchozímu deflate z výjimkou toho, že namísto Huffmanova kódování využívá kódování rozsahové, které si v následujícím odstavci popíšeme.

5.2.1 Rozsahové kódování

Ideu rozsahového kódování [17] (range encoding) prezentoval G. N. N. Martin v roce 1979. Tento druh kódování, který není již od svého vzniku zatížen žádnými patenty, vychází z kódování aritmetického a umožňuje odstranit dva typy nadbytečných informací. První z nich je založen na kontextu a faktu, jak často se jednotlivé symboly vyskytují v okolí jiných symbolů. Druhý je postaven na četnosti výskytu jednotlivých symbolů. Celé kódování začíná určením frekvencí výskytu, kumulativních frekvencí a velikosti počátečního rozsahu (počtu bitů a báze). Kumulativní frekvence jsou spočteny ze všech předcházejících frekvencí výskytu a určují, jaký rozsah daný symbol obsadí. Symboly jsou seřazeny dle svých frekvencí výskytu od nejmenšího po největší. Z definovaného rozsahu si každý ze symbolů nekomprimované vstupní zprávy vezme svůj díl, následující symbol již operuje pouze s rozsahem zvoleným předchozím symbolem. Na rozdíl od Huffmanova kódování, které vytváří bitové vzorky a ty následně spojuje, rozsahové kódování vrací jako svůj výstup jediné číslo. Žádané číslo označující kód pro zadanou zprávu určíme jako střed výsledného intervalu. Dokonce můžeme omezit i počet významných číslic, tak že vynecháme ty místa číslic jejichž libovolnou volbou nelze opustit interval. Pro dekomprimaci musí mít protistrana k dispozici frekvence výskytů a velikost počátečního rozsahu. Rozsahové kódování je implementováno binárně, čímž se vyhneme pomalému dělení, a jeho úspěšnost se blíží entropii dat.

5.2.2 funkcionalita LZMA

Jak již bylo zmíněno, LZMA stejně jako *deflate* využívá algoritmu LZ77 (detailněji na straně 46), který vyhledává nejdelší shodný řetězec v dopředném bufferu a následně zapisuje na výstup trojici (vzdálenost, délka, další symbol). Pokud je nalezena shoda, výstup je vždy kódován pomocí rozsahového kódování, v opačném případě jde na výstup literál. Algoritmus si udržuje historii pro čtyři poslední zjištěné vzdálenosti, a pokud by se hodnota měla opakovat, pošle se na výstup místo vzdálenosti index na dotyčnou hodnotu v historii. Lokalizace shody ve vyhledávacím bufferu je docíleno pomocí dvou, tří nebo čtyřbytové hašovací funkce. Výstupem této funkce je index do hašovaného pole, které je zhruba polo-

viční oproti slovníku. Tento index se použije pro vyhledávání v hašovaném poli jednou ze dvou metod. K dispozici jsou rychlá *hash-chain* nebo více efektivní *binární strom*.

Implementací LZMA je již zmíněný produkt **7-zip** [65], produkující soubory s příponou `7z`. S pomocí nástroje lze upravit parametry algoritmu. Tímto způsobem můžeme zvolit kompresní mód, velikost slovníku či funkci pro vyhledávání v hašovaném poli. K dispozici je i vývojový balík, který umožňuje využití algoritmu v jazycích C, C++, C# a Java. Velikost slovníku může dosáhnout teoreticky až 4 GB. Dalším projektem využívajícím LZMA je **xz** [38], který používá algoritmus LZMA2 [39].

5.3 LZO

LZO [20, 21] (Lempel-Ziv-Oberhumer) je dalším zástupcem algoritmů bezztrátové komprese napsaný v jazyce C, konkrétně podle normy ANSI. Některé rutiny algoritmu jsou optimalizované pro assembler, nicméně celý kód je snadno přenositelný, což dokazuje velký počet platforem, pro které byl přeložen a testován. Algoritmus je navržen pro kompresi a dekompresi v reálném čase. Dává tedy přednost rychlosti zpracování vůči samotnému kompresnímu poměru.

Rychlost komprese i kompresní poměr se blíží algoritmu *deflate*. Přidanou hodnotou oproti zmíněnému deflate je velmi rychlá dekomprese. LZO je blokový kompresní algoritmus, který potřebuje minimální množství paměti ke kompresi a dekomprimovat dokáže bez využití další paměti tzv. *in-place* při užití techniky překrývajících se bloků. LZO také umí pracovat ve vláknech a garantuje jejich bezpečné využití.

LZO vlastně není jediným algoritmem, nýbrž rodinou několika menších algoritmů, jejichž chování je odvozeno od použití rozdílných kompilačních parametrů. Velké množství algoritmů rodiny LZO je udržováno z důvodu zpětné kompatibility. Označení se liší dle požadavků na kompresní poměr, rychlost a využití operační paměti. Výchozím kompresním algoritmem je **LZ01X**, jehož paměťové nároky jsou pro 32bitovou architekturu pouhých 64 kB (ve speciálních případech dokonce jen 8 kB). Zde se jedná o variantu LZO1X-1, která je optimalizována spíše pro rychlost než pro úroveň komprese. Pokud bychom chtěli lepší kompresi, můžeme využít variantu LZO1X-999, zaplatíme však rychlostí a větším množstvím využité paměti – 448 kB pro 32bitovou platformu.

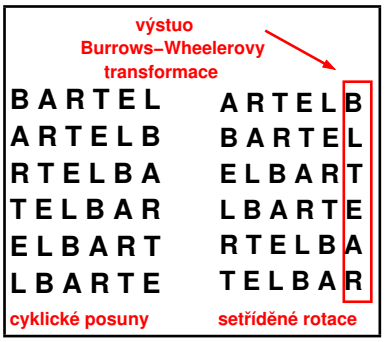
Algoritmus LZO využívá opět slovníku a posuvného okna, neboli vychází z algoritmu LZ77. Konkrétní detaily o způsobu modifikace LZ77 pro potřeby LZO bohužel nenajdeme v dokumentaci a museli bychom nahlédnout přímo do zdrojového kódu. Sám autor nás však od tohoto kroku odrazuje, neboť tvrdí, že kód je optimalizován hlavně pro rychlost a je tedy velice špatně čitelný.

V implementační rovině je algoritmus LZO zastoupen open-source projektem **lzop** [60], který je velmi podobný gzipu, jen díky knihovně LZO poskytuje mnohem vyšší rychlosti komprese i dekomprese. Existují i implementace knihovny LZO v Javě či Visual Basicu.

5.4 Burrows-Wheelerova transformace

Tato metoda není ve své podstatě samotnou kompresí, pouze poskytuje jeden z článků pro dále zmiňovanou implementaci **bzip2**. Techniku objevili Michael Burrows a David Wheeler v roce 1993 [2]. Metoda je založená na permutaci znaků či podřetězců v řetězci a jejich přerovnání takovým způsobem, aby výsledný řetězec byl snáze komprimovatelný.

Zvolený řetězec doplněný o ukončovací znak o délce n rotujeme vždy o jedno místo doleva. Tímto způsobem získáme n rotací původního řetězce (jak ukazuje obrázek 5.4). Výsledky lexikograficky srovnáme a na výstup zašleme poslední sloupec znaků. Řetězcem samozřejmě může být i celý soubor a znak EOF. Technika poskytuje uspokojivé výsledky, pokud je zadán dostatečně dlouhý vstup (alespoň v jednotkách kilobajtů) a jeho obsah má smysl – jedná se např. o text v češtině. Metoda využívá ukazatelů a pro uchovávání rotací se používá suffixový strom, což je struktura obsahující podslova řetězce.



Obrázek 5.4: Burrows-Wheelerova transformace

Další výhodou této transformace je její obousměrnost. Jsme tedy schopni vygenerovat původní řetězec. Toho dosáhneme tak, že výstupní řetězec lexikograficky setřídíme a následně přidáme znovu výstupní řetězec před právě setříděný. Po n-tém opakování získáme originální řetězec.

5.4.1 Move-to-front transformace

Vstup	Výstup	Vstupní abeceda
BLTEAR	1	(a**b**cdefghijklmnopqrstuvwxyz)
B**L**TEAR	1,11	(**b**acdefghijk**l**mnopqrstuvwxyz)
BL**T**EAR	1, 11, 21	(**l**bacdefghijkmnopqrs**t**uvwxyz)
BLT**E**AR	1, 11, 21, 6	(**t**lbacd**e**fghijkmnopqrsuvwxyz)
BLTE**A**R	1, 11, 21, 6, 4	(**e**tlb**a**cdfghijkmnopqrsuvwxyz)
BLTEA**R**	1, 11, 21, 6, 4, 18	(**a**etlbcdfghijkmnopq**r**suvwxyz)
BLTEAR	1, 11, 21, 6, 4, 18	(**r**aetlbcdfghijkmnopqsuvwxyz)

Tabulka 5.1: Move-to-front transformace

Dalším krokem pro implementaci *bzip2* je tzv. *Move-to-front* transformace. Cílem této metody je zlepšení vlastností entropie vstupního řetězce. Vstupem je výstupní řetězec Burrows-Wheelerovy transformace a lexikograficky seřazený seznam vstupní abecedy.

Podstatou této metody je nahrazení znaků celočíselnými indexy odkazujícími se do vstupní abecedy. Ta je průběžně upravována tak, že na její počátek přesunujeme znaky přečtené ze vstupu. Často využívané symboly abecedy se vyskytují na jejím počátku a jejich indexy jsou tedy následně malá čísla.

Používá-li transformace anglickou abecedu tedy znaky {a-z}, bude pro zakódování použito čísel 0 až 25. Stejné znaky ve vstupním řetězci nebudou kódovány stejným indexem, pokud nebudou následovat ihned za sebou. Tato transformace je taktéž obousměrná. Příklad přináší tabulka 5.1

5.4.2 Bzip2

Bzip2 [74] je bloková kompresní metoda s variabilní velikostí bloků, která se pohybuje od 100 kB po 900 kB. Velikost bloku je nastavitelná pomocí přepínačů. Od velikosti bloku se odvíjí i paměťová náročnost. Ta se liší pro kompresi i dekompresi (8 * velikost bloku + 400 kB pro kompresi a zhruba poloviční pro dekompresi).

Bzip2 stejně jako *gzip* zvládá komprimovat pouze jediný soubor. Pokud chceme komprimovat více souborů, musíme využit např. archivačního nástroje **tar**. Soubory zkomprimované pomocí bzip2 jsou označeny koncovkou `.bz2`

Po obdržení výstupu *Move-to-front* transformace zbývá pouze poslední krok. Tímto krokem je využití kódování založeného na entropii. Nabízí se RLE či Huffmanovo kódování (více v podkapitole 5.1.2 na straně 46). Bzip2 využívá právě Huffmanovo kódovaní.

5.5 Srovnání

Uvedli jsme si základní principy vybraných bezztrátových kompresních algoritmů, a také jejich nejobvyklejší implementace. V této podkapitole vybereme nejvhodnější algoritmus pro potřeby pluginu za účelem získávání informací o entropii obsažené v síťových tocích.

Nejprve v tabulce 5.2 srovnáme jednotlivé implementace z pohledů softwarových licencí, což může pomoci při výběru implementace kompatibilní s ostatním software použitým v rámci pluginu. Záměrně zahrnujeme open-source implementace, neboť i plugin má být volně šiřitelný. Výjimku tvoří produkt PKZIP, který je původní implementací algoritmu deflate a je zde uveden pouze pro úplnost. Moduly Perlu pro kompresi uvádíme z důvodu jejich využití pravě s pluginem, jehož backend využívá tohoto jazyka.

Algoritmus / knihovna	Implementace	Licence implementace
deflate	PKZIP, zlib/gzip, 7-zip, IO::Compress::Gzip	komeční, GPL, LGPL, GPL
LZMA	7-zip, xz, IO::Compress::Lzma	LGPL, LGPL a GPL, GPL
LZO	Lzop, IO::Compress::Lzop	GPL, GPL
Burrow-Wheelerova transformace	bzip2, 7-zip, IO::Compress::Bzip2	BSD, LGPL, GPL

Tabulka 5.2: Implementace algoritmů a jejich licence

Zajímavým faktorem pro srovnání může být kromě licence i zralost projektu, a tedy četnost vydávání nových verzí. Vzhledem k open-source povaze jednotlivých projektů nás může zajímat i programovací jazyk, ve kterém byly napsány, pro případ že bychom chtěli knihovny připojit k jinému projektu. Tyto informace spolu s cestami jak daný nástroj získat uvádí následující přehledná tabulka 5.3. Bohužel ne všechny z vybraných projektů umožňují ověření staženého nástroje pomocí kontrolního součtu či podpisu. Lzop a 7-zip tímto způsobem integritu svého nástroje nezaručují.

Výkon a výpočetní zdroje

Provedli jsme několik srovnávacích testů, jejichž výsledky nyní předkládáme. Testovali jsme vlastností kompresních algoritmů pomocí řádkových klientů jejich implementací, abychom

Implementace	Aktuální verze	Datum	Četnost	Programovací jazyk	Dostupnost a ověření
7-zip	9.22	04/2011	střední	C/C++	web, sourceforge
bzip2	1.0.6	10/2010	malá	ANSI C	web + md5
lzop	1.0.3	11/2010	malá	ANSI C	web
xz	5.0.3	05/2011	střední	ANSI C	web, GIT + OpenPGP
zlib	1.2.5	04/2010	střední	C	web, sourceforge + md5

Tabulka 5.3: Implementace algoritmů – verze, dostupnost a programovací jazyk

malá – 1x za x let, střední – několikrát ročně

zjistili, jaká je náročnost na výpočetní zdroje, tedy procesor a paměti, jakou rychlostí jsou jednotlivé implementace schopny komprimovat či dekomprimovat soubor na disku a v neposlední řadě jak dlouho jim daná operace bude trvat. Měření probíhalo s pomocí nástroje `GNU time`, který poskytl čas strávený příkazem, informace o obsazené paměti a procentuální vytížení procesoru. Testovali jsme několik souborů.

Testovací prostředí zahrnovalo:

- notebook Toshiba Satellite M100-164,
- procesor Intel Core Duo processor T2300 s frekvencí 1,66 GHz,
- 1 GB RAM,
- pevný disk TOSHIBA MK6034GSX,
- operační systém GNU/Linux Debian wheezy/sid,
- jádro Linux 2.6.39.4 (preempt),
- gzip 1.4, bzip2 1.0.5, lzop 1.03 a p7zip 9.20,
- nástroj GNU time 1.7.

Test velikosti režijních nákladů komprimovaného souboru

V prvním testu jsme zkoušeli zjistit, jak velká je režie zkomprimovaného souboru. Vytvořili jsme tedy soubor obsahující číslici 0 o velikosti 1 B. Tabulka 5.4 ukazuje velikosti komprimovaných souborů, jaké jsme obdrželi. V případě komprimace velmi malých souborů je třeba zahrnout při porovnávání kompresních poměrů dosažených jednotlivými algoritmy i velikost režijních nákladů. V opačném případě by pro malé vstupní soubory byla zkreslena entropie, neboť jak bylo uvedeno výše, při odhadu entropie kompresí se jedná o poměr původní a komprimované zprávy. *Pokud by tedy zdroj dat obsahoval během jednoho časového slotu např. pouze 150 bajtů a použili bychom algoritmy z tabulky, obdržíme diametrálně rozdílné kompresní poměry neodpovídající entropii dat.*

Algoritmus	Velikost archivu [B]	Velikost režie [B]
gzip	23	19
bzip2	37	33
lzop	56	52
P7zip	104	100

Tabulka 5.4: Kompresní algoritmy – velikosti režijních nákladů

Test entropie IP adres a portů za pomoci komprese

Další test se přímo týkal principu plánovaného pro implementaci pluginu. V celkově čtyřech souborech byla čísla portů. V prvních dvou se opakoval port 80. V souboru `dp_repeating` vždy na novém řádku a v souboru `dp_repeating_string` za sebou. Další dva soubory (`dp_201111131120 a dp_201111131120_string`) kopírovaly stejný formát, nicméně data pocházela z pěti minut anonymizovaného provozu na síti Masarykovy univerzity v období 13. 11. 2011 11.20 až 11.25 středoevropského času a odrážela tudíž provoz na skutečné síti.

Název souboru	dp_repeating	dp_repeating_string	dp_201111131120	dp_201111131120_string
Originální velikost [B]	439 161	292 774	1 024 709	544 227
Gzip [B]	479	341	260	212
Gzip [%]	0,11	0,12	0,03	0,04
Bzip [B]	51	44	256	256
Bzip [%]	0,01	0,02	0,02	0,05
Lzop [B]	2 453	1 670	576	320
Lzop [%]	0,56	0,57	0,06	0,06
P7zip [B]	259	250	320	320
P7zip [%]	0,06	0,09	0,03	0,06

Tabulka 5.5: Kompresní algoritmy – test entropie portů

Porovnání souborů s porty v řetězci a na jednotlivých řádcích mělo za úkol potvrdit, že s nimi lze nakládat rovnocenně a obsahují stejnou entropii. Mezi uměle vytvořeným provozem a daty získanými z reálné sítě měl být pozorovatelný rozdíl v entropii, totiž opakující se porty měly být snáze zkomprimovatelné a vykazovat nižší entropii. Z přiložené tabulky 5.5 daný jev není zcela zjevný pravděpodobně proto, že jsme použili příliš malý soubor hodnot (cca 150 000). Srovnatelnost souborů s daty oddělenými novými řádky a souborů obsahující řetězec se prokázala.

Test vlastností kompresních algoritmů

Nejrozsáhlejší test proběhl s archivem tar současného stabilního linuxového jádra verze 3.0.4. Soubor `linux-3.0.4.tar` s velikostí úctyhodných 451 031 040 B obsahující zdrojové kódy převážně v jazyce C a komentáři v angličtině a několika dalších jazycích byl dostačující pro test vlastností jednotlivých komprimačních nástrojů. Všechny nástroje byly testovány pro různé úrovně komprese. Těchto úrovní bylo devět, nicméně nástroj lzop rozlišuje jen pět úrovní, jelikož úrovně 2-6 jsou shodné a v tabulce jsou vyznačeny kurzívou. Nástroj p7zip sice v manuálu takovou informaci neposkytuje, ovšem výsledky v tabulce vypovídají o tom, že mezi každými dvěma následujícími úrovněmi není rozdíl (tedy např. 1-2).

První charakteristikou při komprimaci byla velikost komprimovaného souboru. Těsnými výsledky si mohly konkurovat nástroje bzip2 a p7zip, které dosáhly pod hranici 20 % původní velikosti. O něco horší byl nástroj gzip a poslední skončil lzop, jehož výsledek byl téměř dvojnásobný oproti p7zip. Pokud jde o volbu optimálního přepínače, půjde pravděpodobně o hodnotu 6, neboť jak uvedeno dále, vyšší hodnoty byly příliš časově nebo výpočetně náročné a zlepšení komprese nebylo nikterak významné.

Doba komprese spolu s vypočtenou rychlostí přenosu dat ukázala poněkud jiné pořadí. Nejlépe se umístil gzip, jehož čas rostl pouze lineárně dle zvyšující se úrovně komprese. Následoval jej lzop, jehož časová náročnost významně vzrostla až pro poslední tři úrovně komprese. Bzip2 byl zhruba třikrát horší než gzip, nicméně také vykazoval pouze lineární nárůst. A nejhůře si vedl p7zip, který s každou úrovní zhoršil svůj výkon v řádu minut, kde nejvyšší úroveň komprese dokončil až po 7 minutách oproti jedné minutě gzipu.

Využití výpočetních zdrojů ovládl nástroj lzop, který při nižších úrovních komprese vytížil procesor na méně než 20 %. Využití paměti u gzip a lzop bylo v podstatě konstantní a bzip2 spotřeboval paměť lineárně k úrovni komprese. P7zip byl jednoznačně nejnáročnější na výpočetní zdroje, protože mu nestačil 1 GB RAM a potřeboval využít i odkládací prostor. U využití procesoru se právě díky využití odkládacího prostoru dostal přes 100 %, což je dle manuálové stránky nástroje time způsobeno odložením do odkládacích prostoru během měření, případně je možné, že nástroj GNU time nezvládá práci s více jádry při paralelizaci výpočtu.

Při dekompresi jasně zvítězil p7zip následovaný nástroji lzop a gzip. Jediným pomalejším účastníkem se stal bzip2, který dosáhl téměř na trojnásobek hodnot nástroje p7zip. P7zip a bzip2 vytěžovaly nadměrně procesor i během dekomprese a p7zip si vzal i významnou část operační paměti. Všechny údaje jsou shrnuty v přehledné tabulce 5.6.

Ačkoli tento test sice přímo neodrážel oblast vstupních dat se kterou se bude plugin zabývat, zvolený vstupní soubor sloužil díky svému obsahu a velikosti jako ideální materiál pro testovaní obecných vlastností kompresních algoritmů a pomohl nám demonstrovat chování každého z nich.

Závěr

Implementace P7zip jako jediná byla implicitně optimalizovaná pro paralelní běh na více jádrech, a tak nebylo měření zcela přesné. Existují sice i verze, které umí výpočet paralelizovat pro gzip a bzip2, nicméně těmi jsme zde nezabývali. Pro naše potřeby využijeme implementace algoritmů v Perlu z důvodů zmíněných výše. Tyto implementace jsme sice netestovali, ale předpokládejme, že jsou funkčně stejné jako testované. V případě entropie bude jistě zajímavé porovnat výstup více komprimačních algoritmů, abychom zjistili zda dokáží správně reagovat na vzniklé anomálie.

Shrňme si tedy, co jsme zjistili. Nejlepší komprese dosáhl sice p7zip, nicméně ten je velmi náročný na zdroje. V případě účinnosti komprese tedy vyberme bzip2. Z pohledu využití výpočetních zdrojů můžeme zvažovat gzip, bzip2 a lzop, který přes nepříliš kvalitní kompresi alespoň vyniká rychlostí a nízkou náročností.

Dekompresí se v našem pluginu zabývat nebudeme, neboť pouze potřebujeme zjistit entropii obsaženou v datech, a tak rychlé provedení této operace u nástrojů lzop a p7zip můžeme ignorovat.

Ve výsledku tedy můžeme počítat s nástrojem gzip, který sice v ničem neexceluje, poskytuje však kvalitní výsledky ve všech zmíněných testech. Pro porovnání můžeme přidat lzop, díky jeho rychlosti. Bzip2 bychom vybrali pravděpodobně pouze na silném hardware a v případě, že by plugin nemusel soupeřit s ostatními.

Zdrojový soubor – Linux kernel 3.0.4 tar archiv (linux-3.0.4.tar)

Velikost zkomprimovaného archivu a kompresní poměr

Komprese	Gzip [B]	Poměr [%]	Bzip2 [B]	Poměr [%]	Lzop [B]	Poměr [%]	P7zip [B]	Poměr [%]
1	120 981 447	26	91 068 959	20	162 803 275	36	89 205 195	19
2	115 340 935	25	85 048 796	18	*161 813 056*	*35*	89 205 195	19
3	111 368 218	24	82 298 718	18	*161 813 056*	*35*	81 494 386	18
4	103 365 755	22	80 612 795	17	*161 813 056*	*35*	81 494 386	18
5	99 191 591	21	79 415 536	17	*161 813 056*	*35*	66 559 073	14
6	97 528 006	21	78 570 545	17	*161 813 056*	*35*	66 559 073	14
7	97 066 963	21	77 758 199	17	112 591 553	24	64 440 660	14
8	96 753 439	21	77 354 576	17	111 305 208	24	64 440 660	14
9	96 695 794	21	76 759 291	17	111 259 101	24	63 787 963	14

Doba a rychlost komprese

Komprese	Gzip [s]	Rychlost [MB/s]	Bzip2 [s]	Rychlost [MB/s]	Lzop [s]	Rychlost [MB/s]	P7zip [s]	Rychlost [MB/s]
1	26,70	16,10	109,79	3,91	31,22	13,77	90,33	4,76
2	24,29	17,70	120,09	3,58	*31,07*	*13,84*	65,54	6,56
3	25,36	16,96	125,17	3,43	*31,07*	*13,84*	83,71	5,13
4	28,30	15,19	126,78	3,39	*31,07*	*13,84*	83,83	5,13
5	36,21	11,87	133,91	3,21	*31,07*	*13,84*	321,49	1,33
6	44,35	9,69	136,21	3,15	*31,07*	*13,84*	292,54	1,47
7	47,44	9,06	139,34	3,08	102,02	4,21	387,83	1,10
8	58,20	7,39	141,39	3,04	163,51	2,63	386,10	1,11
9	68,58	6,27	143,29	3,00	171,33	2,51	430,67	0,99

Využití operační paměti a procesoru při kompresi

Komprese	Gzip [kB]	Gzip [%]	Bzip2 [kB]	Bzip2 [%]	Lzop [kB]	Lzop [%]	P7zip [kB]	P7zip [%]
1	3 552	65	6 368	96	4 672	19	14 688	72
2	3 552	75	9 504	89	*4 400*	*19*	14 672	94
3	3 536	78	12 624	87	*4 400*	*19*	42 960	99
4	3 552	77	15 792	89	*4 400*	*19*	42 976	99
5	3 504	77	18 928	87	*4 400*	*19*	790 720	**149**
6	3 520	79	22 032	88	*4 400*	*19*	790 736	**162**
7	3 488	85	25 200	88	5 984	72	1 544 416	**159**
8	3 504	91	28 336	89	5 968	77	1 544 384	**159**
9	3 488	90	31 456	90	5 968	80	2 789 376	**154**

Doba a rychlost dekomprese

Komprese	Gzip [s]	Rychlost [MB/s]	Bzip2 [s]	Rychlost [MB/s]	Lzop [s]	Rychlost [MB/s]	P7zip [s]	Rychlost [MB/s]
1	27,81	15,46	43,24	9,94	22,92	18,76	17,00	25,30
2	26,99	15,93	43,77	9,82	*23,97*	*17,94*	17,46	24,63
3	26,13	16,46	43,83	9,81	*23,97*	*17,94*	15,72	27,36
4	25,30	17,00	44,87	9,58	*23,97*	*17,94*	16,35	26,30
5	26,45	16,26	44,56	9,65	*23,97*	*17,94*	17,57	24,48
6	24,73	17,39	45,75	9,40	*23,97*	*17,94*	15,80	27,22
7	26,86	16,01	48,38	8,89	20,97	20,51	16,17	26,60
8	24,89	17,28	48,72	8,82	19,73	21,80	15,65	27,48
9	25,12	17,12	48,17	8,92	21,93	19,61	24,17	17,79

Využití operační paměti a procesoru při dekompresi

Komprese	Gzip [kB]	Gzip [%]	Bzip2 [kB]	Bzip2 [%]	Lzop [kB]	Lzop [%]	P7zip [kB]	P7zip [%]
1	5 168	29	3 760	65	3 072	17	12768	98
2	5 184	28	5 344	63	*3 056*	*16*	12 768	95
3	5 168	28	6 912	64	*3 056*	*16*	16 624	93
4	5 184	28	8 496	67	*3 056*	*16*	16 608	90
5	5 168	26	10 048	68	*3 056*	*16*	78 048	74
6	5 168	27	11 632	70	*3 056*	*16*	78 064	81
7	5 168	25	13 216	68	3 056	15	143 616	78
8	5 168	27	14 800	70	3 072	15	143 616	81
9	5 168	26	16 352	72	3 056	14	274 672	52

Tabulka 5.6: Srovnání kompresních algoritmů a jejich implementací

Kapitola 6

Plugin pro detekci anomálií a vizualizaci

Jedním z hlavních cílů této diplomové práce bylo vytvoření pluginu pro detekci a vizualizaci anomálií v síťovém provozu. Požadavky v úvodní kapitole 1 na straně 3 jsou v této kapitole rozvinuty ve specifikaci a následně implementaci a testování. V následujících podkapitolách si představíme jednotlivé části pluginu, jejich účel a provázanost.

6.1 Funkční specifikace

Plugin aplikace NfSen

Architektura požadovaného pluginu musí respektovat typickou strukturu pluginů pro NfSen, jak bylo definováno v kapitole 1.2.2 na straně 7. Očekává se tedy, že bude obsahovat dvě navzájem oddělené části – backend a frontend – komunikující síťovým soketem. Plugin bude ukládat a zpracovávat data periodicky ukládaná aplikací NfSen pomocí povinné funkce `run`. Plugin by měl být schopen detekce anomálií a poskytne vizualizaci dat včetně těchto anomálií. Sekundárním výstupem pluginu bude e-mailové hlášení o detekovaných anomáliích. Plugin bude o své činnosti zapisovat hlášení do systémového logu. Pro nastavení provozních parametrů může plugin použít konfigurační soubor `nfsen.conf` případně vlastní konfiguraci.

Měřené charakteristiky

Plugin bude schopen sledovat vybrané charakteristiky síťového provozu v rámci zvoleného profilu. Data těchto charakteristik získá pomocí dotazování se nad datovým úložištěm aplikace NfSen nástrojem nfdump. Plugin uvažuje objem vybraných hodnot v aktuálním timeslotu případně další závislé hodnoty jako např. entropii. Mezi charakteristiky vhodné ke sledování byly vybrány:

- síťové toky, pakety a bajty,
- zdrojové porty a jejich entropie,
- cílové porty a jejich entropie,
- zdrojové IP adresy a jejich entropie,
- cílové IP adresy a jejich entropie,
- účastníci provozu a komunikační páry,
- aplikační porty (DNS, HTTP, HTTPS, SMTP, SSH).

Požadavky na detekci

Plugin bude schopen detekovat odchylky od standardního provozu na síti. Plugin poskytne algoritmus pro detekci anomálií, jehož činnost bude ovlivnitelná uživatelem. Ke každé detekované anomálii budou dohledány relevantní informace z datového úložiště NfSen, které napomohou k získání kontextu o potencionálním útoku. Tyto informace poskytnou identifikaci účastníků provozu, kteří se největší měrou podíleli na vzniku anomálie. Nalezené anomálie budou zaznamenány, vizualizovány a také ohlášeny pomocí e-mailu.

Požadavky na vizualizaci

Nalezené anomálie budou vizualizovány společně s průběhem časové řady v podobě 2D grafů. Průběh časové řady bude zobrazen pomocí spojnicového grafu nebo grafu oblasti. Detekované anomálie budou vyznačeny bodovým grafem. Každý bod výše uvedeného seznamu měřených charakteristik bude vizualizován jedním obrázkem obsahujícím grafy časových řad a jejich anomálií pro období 24 hodin. Vizualizace umožní vypnout/zapnout zobrazení konkrétního grafu v obrázku. Libovolný bod obsažený v grafu poskytne informaci o typu charakteristiky, ke které náleží a časový údaj svého vzniku. Vybranou oblast grafu bude možné přiblížit a získat tak detailnější pohled pro kratší období.

E-mailová hlášení

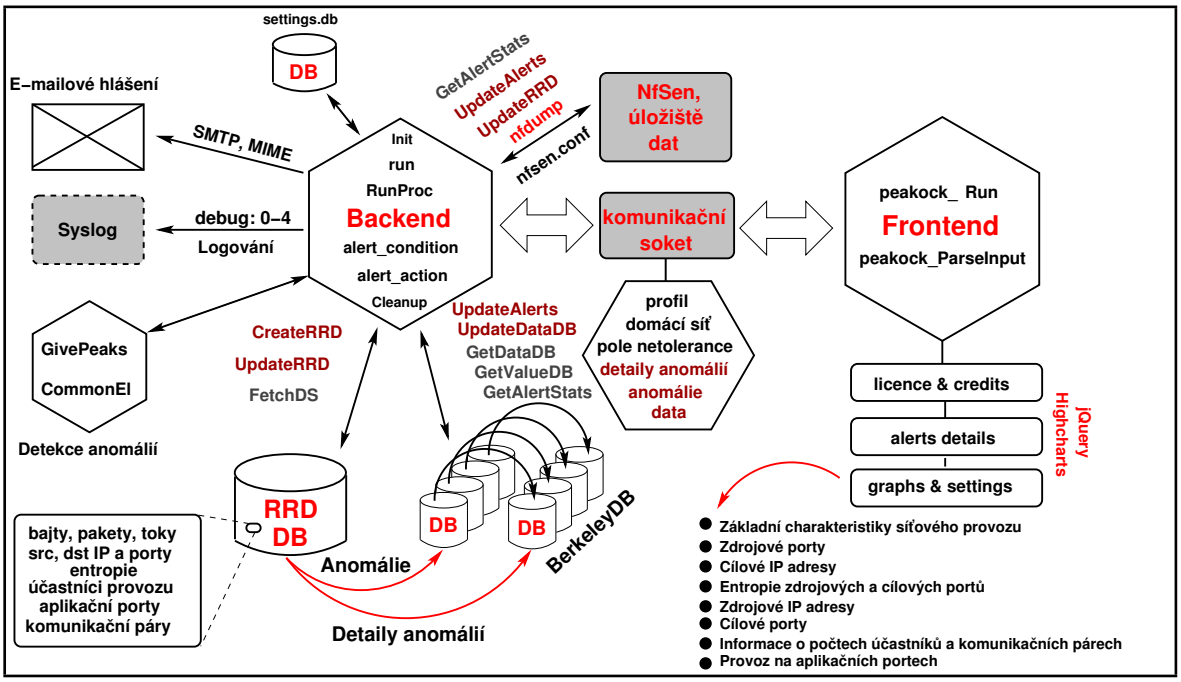
Výstupem pluginu bude i e-mailové hlášení shrnující detekované anomálie včetně jejich detailů za stanovené časové období. Kromě informací o anomáliích poskytne hlášení také informace o zvoleném profilu a jeho typu, skupině, filtru, zdroji dat a časovém období ze kterého anomálie pocházejí. V e-mailu bude dále uveden odkaz na vizualizaci těchto informací. Příjemce zpráv i počet objevených anomálií potřebných k odeslání e-mailu bude možné nastavit pomocí konfiguračního souboru `nfsen.conf`.

Logování

Plugin bude schopen informovat o svojí činnosti do systémového logu. Plugin poskytne několik úrovní logování, které budou rozlišeny dle detailnosti. Úroveň logování bude ovlivnitelná uživatelem z frontendu pluginu či konfiguračního souboru `nfsen.conf`. Plugin umožní vypnout logování do systémového logu.

6.2 Návrh a implementace

Dle požadavků v kapitole 1 na straně 3 a specifikace v kapitole 6.1 na straně 57 vznikl plugin pro aplikaci NfSen, který byl pojmenován **peaKock** (PEAK observation connections kit). Jeho části i funkcionalitu dokumentuje obrázek 6.1. Jednotlivé komponenty tohoto pluginu budou představeny v následujících odstavcích.



Obrázek 6.1: Plugin peaKock – architektura

6.2.1 Frontend/Backend

Frontend pluginu peaKock

Frontend pluginu peaKock (viz schéma 6.1 a snímek obrazovky v dodatku C na straně 81) slouží k prezentaci dat získaných a zpracovaných backendem a interakci s uživatelem. Úkolem frontendu je poskytnout tyto data uživateli v přehledné formě a umožnit nastavit parametry ovlivňující funkcionalitu backendu. Frontend pluginu peaKock poskytuje nejen vizualizaci těchto dat pomocí 2D grafů s vyznačenými anomáliemi, ale i doplňující detailní informace o detekovaných anomáliích v přehledné tabulce.

Frontend pluginu peaKock je tedy webové rozhraní zakomponované do struktury poskytované aplikací NfSen. Pro vytvoření této části pluginu bylo použito značkovacího jazyka XHTML v kombinaci s jazykem PHP, který umožňuje vytvářet obsah webového rozhraní dynamicky na základě předaných hodnot. V XHTML byla vytvořena základní kostra webového rozhraní a PHP se stará o obsluhu komunikačního soketu a získávání dat z backendu. Po přijetí dat ze síťového soketu je jejich část transformována do formátu JSON a přenesena na vstup vizualizační knihovny. Frontend obsahuje povinné funkce uvedené v podkapitole 1.2.2 na straně 7.

Na první pohled viditelná prezentační vrstva frontendu využívá javascriptové knihovny jQuery [48] pro organizaci prvků rozhraní, jejich vzhled a funkcionalitu. Vykreslení grafů následně zajišťuje javascriptová knihovna Highcharts (viz podkapitola 4.1.4 na straně 34), která služeb knihovny jQuery využívá.

Po vizuální stránce poskytuje frontend obrazovku rozčleněnou pomocí horizontálních záložek podobně jako samotný NfSen. Toto členění vytváří menu o třech záložkách. První

z nich se jmenuje *graphs & settings* a kromě informací o profilu, filtru a domácí síti nabízí grafy sledovaných síťových charakteristik. Každý graf zobrazuje údaje o 24 hodinách síťového provozu. K dispozici jsou grafy těchto síťových charakteristik:

- **Základní charakteristiky síťového provozu** – graf zobrazující počty paketů, bajtů a síťových toků.
- **Cílové porty** – graf zobrazuje objem cílových portů a kompresní poměry této veličiny získané pomocí gzip a lzop.
- **Zdrojové porty** – graf vystihuje zdrojové porty a jejich entropii získanou pomocí komprese gzip a lzop.
- **Entropie cílových a zdrojových portů** – graf, který spojuje v jednom obrázku entropii zdrojových a cílových portů.
- **Zdrojové IP adresy** – graf zdrojových IP adres a jejich entropie.
- **Cílové IP adresy** – graf cílových IP adres a jejich entropie.
- **Informace o počtech účastníků a komunikačních párech**
- **Provoz na aplikačních portech** – graf zobrazuje počty spojení na několika aplikačních portech, konkrétně se jedná o 22/SSH, 25/SMTP, 53/DNS, 80/HTTP a 443/HTTPS.

Záložka *graphs & settings* dále umožňuje úpravu některých parametrů detekce anomálií uživatelem. Tato funkcionalita je řešena pomocí formulářových polí. Uživateli je umožněno zvolit profil, domácí síť, úroveň logování a pole netolerance. Změnou prvních dvou jmenovaných dojde k odstranění dat a vytvoření nových grafů.

Každá z měřených charakteristik dovoluje nastavit šířku uživatelského pásma netolerance v procentech (podrobněji v podkapitole 6.2.4 na straně 63). Tyto hodnoty jsou platné od chvíle jejich nastavení a používají se v při detekci anomálií.

Další záložka, totiž *alerts details*, dovoluje uživateli procházet detaily detekovaných anomálií (ukázku naleznete v dodatku C na straně 81). Navigace po obsahu stránky je řešena s použitím křížových odkazů usnadňující výběr požadovaného druhu anomálie. Jednotlivé druhy anomálií jsou označeny nadpisem a poskytují přehlednou tabulku hodnot anomálií a časů ve kterých byly objeveny. Detailní informaci o anomálii získáme kliknutím na řádek tabulky, které způsobí vysunutí/zasunutí panelu obsahujícím požadované informace. Panel zobrazuje pět nejaktivnějších účastníků provozu, kteří k vzniku anomálie přispěli. Kromě jejich IP adres jsou zobrazeny počty paketů, bajtů i síťových toků, a také doba trvání s časovou značkou prvního výskytu.

Záložka s názvem *licence & credits* obsahuje informace o autorovi a licenci pluginu.

Backend pluginu peaKock

Backend pluginu peaKock (viz schéma 6.1) zajišťuje sběr a zpracování dat měřených charakteristik dle specifikace na straně 57. Backend obsahuje povinné funkce uváděné v kapitole 1.2.2 na straně 7. V rámci svého periodického spouštění poskytuje několik funkcí.

Data získaná z úložiště dat aplikace NfSen ukládá do databází, které slouží k jejich uchování po dobu 24 hodin. Data pro tento časový úsek jsou následně zobrazena ve frontendu v podobě grafů a tabulek.

Hlavním úkolem backendu pluginu peaKock je detekce anomálií a jejich detailů, které se věnuje podkapitola 6.2.4 na straně 63. Detekované anomálie a data časových řad jsou odesílána frontendu, aby posloužila k vytvoření prezentace pro uživatele. Zde autor pravděpodobně narazil na omezení komunikačního soketu, neboť se mu nepodařilo poslat z backendu více než cca 1 kB dat pro prvek pole, což představuje asi devět řádek textu. Proto kompletní data detailů anomálií obsahuje pouze e-mailové hlášení.

Informace o objevených anomáliích jsou také zasílány správci systému pomocí e-mailu. Účelem toho druhu hlášení je přitáhnout pozornost správce v případě detekovaných anomálií. E-mailová hlášení jsou založena na sledování podmínky backendem plugin peaKock. Je-li podmínka splněna, zpráva je odeslána. Podrobně se s principem této funkcionality seznámíme v podkapitole 6.2.6 na straně 65

Komunikační soket

Aby měl frontend pluginu peaKock vůbec co prezentovat, musí o data požádat backend. Komunikace je zařízena pomocí síťového soketu. NfSen počítá s případnou modularitou a je tedy možné provozovat backend i frontend na různých strojích. Na straně backendu existuje funkce `socket_send_ok` pro Perl a frontend zase disponuje funkcí `nfsend_query` pro PHP. Pro přenos se používají asociativní pole – tzv. hashe.

V případě pluginu peaKock probíhá skrze soket oboustranná komunikace. Frontend informuje backend o změnách nastavení provedených uživatelem pomocí formulářů a zároveň čeká na data síťových charakteristik a jejich anomálií. Pokud bychom chtěli shrnout typy dat procházejících soketem, můžeme se pro začátek inspirovat schématem 6.1 na straně 59. Ze strany backendu přicházejí data časových řad sledovaných charakteristik, informace o detekovaných anomáliích a jejich detailech a v neposlední řadě informace o profilu a jeho vlastnostech. Fronted zasílá tímto komunikačním kanálem požadavky na změnu konfigurace. Mezi tyto požadavky patří změna aktivního profilu, definice domácí sítě či korekce pole netolerance u jednotlivých grafů.

Profily versus kanály

NfSen používá jako organizační jednotky dat z NetFlow sond profily a kanály. Profily se liší svými vlastnostmi. Každý profil je jistého typu. Kromě implicitního profilu *live* se můžeme setkat s profilem kontinuálním, historickým či stínovým. Typ rozhoduje zda je profil vymezen jistým časovým obdobím nebo sleduje provoz na síti průběžně. Všechny profily mimo *life* jsou omezeny nějakým filtrem. Dříve zmíněné kanály mohou sloužit jako zjemnění uvnitř profilu. Toto rozdělení však není ustálené, a tak záleží zcela na správci NetFlow sond a NfSen, zda záznamy rozdělí na profily či kanály.

Plugin peaKock je připraven na práci s profily. Backend pluginu přistupuje k datovému úložišti, aby vydoloval data týkající se měřených charakteristik. NfSen předává backendu jméno a skupinu profilu zvoleného ve webovém rozhraní společně s timeslotem. S pomocí

asociovaného pole `%profileinfo` je umožněno získat podrobné informace o profilu. Jediný údaj, který není dostupný, je filtr profilu. Jenže právě ten potřebujeme, pokud chceme pracovat s jiným profilem než *live*.

Z tohoto důvodu je práce s profily a jejich přepínání řešeno přímo v pluginu peaKock nezávisle na NfSenu. Backend tedy zajistí načtení filtru ze souboru v úložišti dat profilu a všechny následující dotazy pomocí nástroje nfdump obohacuje právě o definici filtru. Plugin peaKock je tedy schopen pracovat se všemi výše uvedenými typy profilů, a tak může skvěle posloužit jako nástroj zpětné analýzy při použití historického profilu. Pokud není možné vytvořit plugin dle našich potřeb, je doporučeno vytvořit stínový profil sledující profil *live* omezený patřičným filtrem případně volbou kanálů.

V případě historického profilu je důležitá volba správného filtru, neboť plugin dopočítává veškeré hodnoty pro posledních 24 hodin (288 hodnot) v profilu. V případě ostatních typů profilu připravuje plugin data pouze z poslední hodiny (12 hodnot). Je však třeba upozornit, že pro profily s velkým objemem dat to může trvat velmi dlouho.

Data získaná z profilu může uživatel dále ovlivnit volbou tzv. **domácí sítě** neboli sítě ve které se nachází sledované stroje. Tímto nastavením se doplní filtr pluginu o další hodnotu a díky ní nejsou do dat získaných backendem z úložiště NfSenu započítávána data vzniklá uvnitř této sítě.

6.2.2 Vizualizace

Vizualizaci v pluginu peaKock představují grafy. Jedná se o grafy spojnicové případně grafy oblastí. Anomálie jsou v grafech vyznačeny jako bodový graf a jsou označeny kolečkem. Grafy jsou vytvořeny pomocí javascriptové knihovny **Highcharts**, která byla vybrána na základě srovnání v podkapitole 4.1.6 na straně 39. Popis knihovny je k dispozici v podkapitole 4.1.4 na straně 34. Nyní si ukážeme, jaké možnosti grafy síťových charakteristik nabízejí (ukázku grafu naleznete v dodatku C na straně 81).

Jednotlivé veličiny či jejich anomálie v grafu lze vypínat a zapínat podle potřeby pouhým kliknutím na název veličiny v legendě grafu. Tím se graf zpřehlední. Jestliže je v grafu obsaženo více veličin, každá z nich má svou vlastní osu y. Osa x je pak vždy popsána časem.

Pokud nás zajímá vývoj časové řady pro kratší období než je 24 hodin, můžeme použít funkci přiblížení. Ta umožňuje přiblížení pro obě osy zároveň jestliže myší vybereme obdélníkový výřez grafu. Pro návrat k původnímu zobrazení je v pravém horním rohu k dispozici odkaz `Reset zoom`. Dále je možné grafy vytisknout nebo exportovat do formátů **PDF**, **JPEG**, **PNG** a **SVG**.

6.2.3 Uložení dat

Data, která sesbírá NfSen pomocí sond, jsou filtrována po potřeby pluginu a je potřeba uchovat je po 24 hodin. Zde přichází na řadu databáze (více v kapitole 3 na straně 19).

Plugin pracuje s několika druhy dat. V první řadě se jedná o data získaná z úložiště NfSenu, která jsou vždy celými čísly a označují množství dané veličiny. K těmto datům

jsou dopočítávány další údaje, tam kde není potřebný údaj dostupný přímo. Jedná se např. o entropii za pomoci komprese či počty komunikujících párů a účastníků provozu.

Dalším typem dat jsou samotné anomálie, které jsou objevovány na základě dat získaných z NfSenu a algoritmu, který bude popsán v následující podkapitole. Posledním druhem dat jsou detailní informace o anomáliích, které získáme dalšími dotazy nad úložištěm NfSenu, v okamžiku kdy anomálii objevíme.

V pluginu peaKock jsou uchovávány informace za posledních 24 hodin. Vývoj síťového provozu se však mění neustále. Z tohoto důvodu se potřebujeme zbavovat starých hodnot. To se v pluginu peaKock řeší pomocí cyklické databáze RRDtool (více v kapitole 3.1.1 na straně 20), která udržuje pouze potřebné hodnoty sama o sobě. Data získaná z NfSenu jsou uložena hromadně v jedné RRD databázi. K jejímu vytvoření dochází při prvním periodickém spuštění pluginu peaKock nebo v okamžiku změny profilu či definice domácí sítě vyvolané z frontendu.

Ukládání informací o anomáliích a jejich detailech je realizováno zcela jiným způsobem. Autor vychází z myšlenky, že anomálie se vůbec nemusí vyskytnout a je tedy zbytečné od začátku vytvářet prostor pro jejich uložení. Zde je použito aplikační databáze, neboť stále jde pouze o data, která bude využívat pouze plugin. Důvodem pro volbu jiné databáze než RRD, byl fakt, že tento druh databáze očekává pravidelný přísun dat. To je ovšem přesný opak toho, co můžeme čekat od výskytu anomálií. Díky BSD licenci a použití v mnoha projektech byla zvolena databáze BerkeleyDB (viz kapitola 3.2.1 na straně 21). Pro každou veličinu jejíž anomálie nás zajímají plugin peaKock vytváří dvě databáze. První pro samotné anomálie a druhou pro detailní informace o nich. I databáze anomálií a jejich detailů jsou odstraněny při změně profilu nebo domácí sítě.

Poslední místo kde se plugin zabývá ukládáním dat je databáze nastavení. Tato databáze udržuje konfigurační parametry zvolené uživatelem ve frontendu. Opět byla použita aplikační databáze BerkeleyDB.

Tabulka D.1 na straně 83 představuje funkce, které jsou v pluginu peaKock použity při manipulaci s daty. Konkrétnější představu o ukládání dat může čtenář získat ze schématu architektury 6.1 na straně 59.

6.2.4 Detekce a komprese

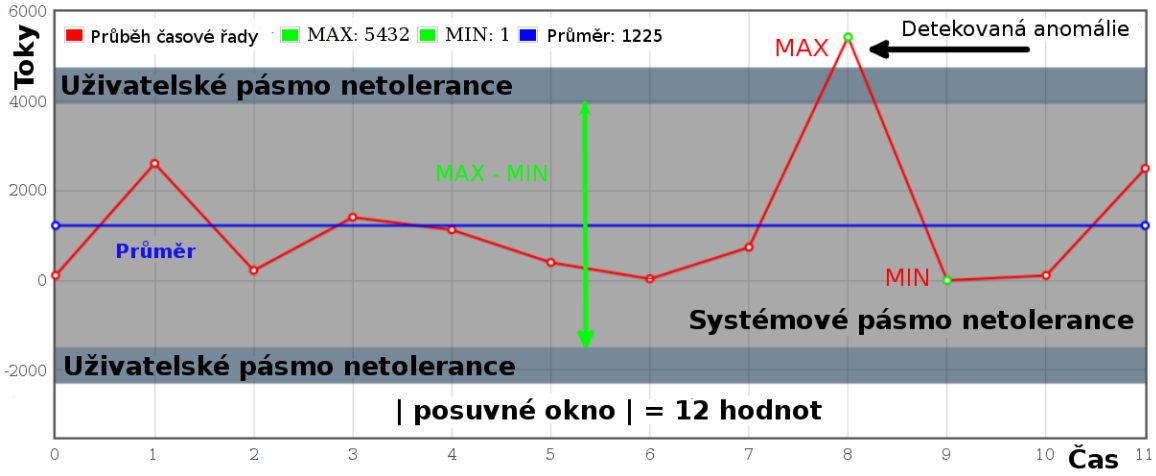
Primárním cílem pluginu peaKock je detekce anomálií v síťovém provozu. Data, se kterými pracuje, získá v databázi RRD, kterou plugin používá pro uchování dat získaných z úložiště NfSen a výpočtů. O tom jak jsou data uložena jsme se dozvěděli v kapitole 6.2.3. Zbývá už jen představit způsob, kterým je získána entropie. Ta se počítá pro porty a IP adresy. Data vrácená nfdumpem jsou po setřídění spojena v jeden řetězec, který se následně zkomprimuje. Pro porty se používá dvou kompresních algoritmů – gzip a lzop, zatímco pro IP adresy je použito pouze gzip z výkonnostních důvodů (volba algoritmů vychází ze závěru kapitoly 5.5 na straně 54). Podle vzorce 1.5 na straně 10 se vypočte entropie z kompresních poměrů. Kromě entropie, portů a IP adres jsou zpracovávány další charakteristiky provozu na síti – bajty, pakety, toky, počty účastníků a komunikačních párů a v neposlední řadě objem provozu na vybraných aplikačních portech.

Nyní si vysvětlíme, jakým způsobem plugin anomálie hledá. Algoritmus je postaven na použití posuvného okna, jehož velikost lze zvolit v konfiguračním souboru *nfsen.conf*. Implicitní hodnota tohoto okna je 12 hodnot, tedy jedna hodina. V oblasti okna hledáme události, jejichž hodnota výrazně vybočuje z průměru. Za tímto účelem algoritmus nejdříve hledá lokální extrémy.

Poté co je vypočten průměr, maximum a minimum předaných hodnot, rozdělí pomyslný graf průběhu časové řady hodnotou průměru. Vznikne tak minimálně jedna oblast nad i pod průměrem v případě, že vstupní data obsahovala alespoň tři různé hodnoty. Rozdělení na oblasti probíhá pravě při průchodu přes hodnotu průměru. Následně se prochází všechny hodnoty a přechází se mezi jednotlivými oblastmi. Pokud se jedná o oblast nad průměrem, algoritmus hledá maximální hodnotu. V případě dolní oblasti se hledá hodnota minimální. Výsledkem jsou tedy lokální extrémy z jednotlivých oblastí.

Předpokládejme však, že zdaleka ne každý lokální extrém znamená detekovanou anomálii, neboť se může jednat pouze o odchylku v rámci jednotek. Z tohoto důvodu zavadí algoritmus *pásmo netolerance*, které slouží k nastavení citlivosti algoritmu. Toto pásma definované dvěma hranicemi – spodní a horní – zahrnuje lokální extrémy, které nás svým významem nezajímají. Šířka tohoto pásma je daná jako násobek poloviny rozdílu maxima a minima a čísla 1,*xx*, kde *xx* představuje hodnotu v procentech zadanou uživatelem ve frontendu. Horní hranici tedy dostaneme přičtením tohoto pásma k průměru a spodní pak odečtením od průměru.

Ze zjištěných lokálních extrémů se hledají takové, které přesahují jednu z hranic. Pokud jsme požadovali minima, zajímají nás všechny extrémy z dolních oblastí, které jsou pod spodní hranicí. Jestliže jsme chtěli maxima, funkce hledá extrémy z horních oblasti přesahující horní hranici. Výstupem základního algoritmu jsou tedy anomálie – lokální minima či maxima (dle nastavených parametrů), která se vymanila z pásma netolerance. Základní algoritmus ukazuje obrázek 6.2



Obrázek 6.2: Plugin peaKock – detekce anomálií

Často nám však vyhodnocení jedné časové řady nestačí a je třeba vyhledat lokální extrémy na více souvisejících charakteristikách provozu. Pro tuto situaci můžeme využít další funkce, která využívá výsledky základního algoritmu. Pro závislé charakteristiky si vyžádá

maxima či minima a následně hledá společné body mezi výsledky – ty jsou prohlášeny za anomálie. *Příkladem může být zvyšující se objem cílových portů, tedy maxima, a snižující se entropie, tedy minima, což by mohlo být důkazem aplikačního DoS útoku.*

Nicméně i objevené anomálie mají po určitou dobu dočasný charakter. Tou dobou je jedna hodina resp. velikost posuvného okna. Pokud hodnota procestuje skrz celé posuvné okno a nebude nahrazena významnějším lokálním extrémem, můžeme ji pokládat za skutečnou anomálii a informovat o ní správce systému. *Na obrázku 6.2 je detekovaná anomálie devátou hodnotou. Pokud se tedy v následujících devíti timeslotech neobjeví lokální extrém s vyšší hodnotou, opustí anomálie posuvné okno a může být ohlášena.*

Pro dokončení detekce anomálií je ještě třeba dohledat detailní informace o potencionálních anomáliích. Pro tuto operaci je opět využito nfdumpu k vygenerování statistiky deseti nejvýznamnějších účastníků provozu, který vedl k zjištěné anomálii. Tento údaj se následně uloží do databáze detailů konkrétního typu anomálie. Tabulka D.2 na straně 83 představuje funkce, které jsou v pluginu peaKock při detekci anomálií použity. Plugin peaKock tedy napomáhá k odhalení útoků uvedených v kapitole 2 na straně 13 či slovníkovému útoku na aplikační porty.

6.2.5 Logování

Plugin peaKock může informovat o svojí činnosti do systémového logu. Tato činnost je závislá na nastavení úrovně logování. Plugin poskytuje pět úrovní upovídanosti, kde nula logování vypíná a standardně zvolenou úrovní je úroveň jedna. Volba úrovně logování je na uživateli a požadovanou úroveň lze zvolit ve frontendu pluginu. Jednotlivé úrovně se doplňují. S vyšší úrovní logování jsou v systémovém logu i všechny zprávy nižších úrovní. Tabulka 6.1 představuje jednotlivé úrovně a zprávy, které se k nim vztahují.

N	Zpráva
0	inicializace a ukončení pluginu peakock
0	aktivace podmínky upozornění správce
1	doba provádění základních procedur
1	objevení či zneplatnění anomálie
1	počet objevených anomálií a informace o zaslání e-mailu
2	časový rozsah pluginu a předaná časová značka
2	timeslot zasahující mimo profil či chybějící data
2	doba provádění komprese gzip, lzop2 a kompresní poměry
2	scházející data u zdrojových a cílových portů i adres
2	počty pozorovaných aplikačních portů
2	umístění frontendu v rámci webového serveru
2	změna profilu či domácí sítě a mazání databází
2	použitá domácí síť a také profil, jeho typ a skupina
2	počty anomálií v jednotlivých souborech databáze
3	detaily objevených anomálií
3	hodnoty vrácené funkcí `GetValueDB` a název databáze
3	prováděcí doby dalších procedur
3	aktuální timeslot, časová značka a datum
3	hodnoty ukládané pomocí funkce `UpdateRRD`
3	počet hodnot uložených v profilu
4	filtry profilu
4	nfdump dotazy pro systém nfsen
4	hodnoty lokálních extrému funkce `GivePeaks`
4	hodnoty získané z databáze RRD pomocí funkce `FetchDS`

Tabulka 6.1: Plugin peaKock – Úrovně logování

6.2.6 E-mailové hlášení

Jak již bylo zmíněno, NfSen nabízí pro reakci na jistou mezní událost dvě procedury, a to `alert_condition` a `alert_action`, jichž plugin peaKock využívá. Ve frontendu NfSenu je nutné informovat, že plugin je schopen reagovat na určité události pomocí těchto procedur.

První funkce má za úkol sledovat stav podmínky definované v pluginu. Plugin peaKock prochází všechny soubory obsahující nalezené anomálie a počítá jejich celkový počet v zadaném časovém rozpětí. Při prvotní inicializaci pluginu je počátek tohoto rozpětí nastaven na unixový čas 0, tedy půlnoc 1. 1. 1970 a konec označuje poslední objevená anomálie či

současný čas. Jestliže tedy počet detekovaných anomálií dosáhne nebo překročí určitou hranici, je podmínka aktivována a na řadu přichází funkce `alert_action`. V opačném případě zapíše plugin peaKock do systémového logu pouze informaci o časovém horizontu a počtu v něm objevených anomálií. Logování musí být samozřejmě povoleno. Hranice pro aktivaci podmínky je implicitně stanovena na hodnotu deset a lze ji upravit v konfiguračním souboru aplikace NfSen. Touto hodnotou lze regulovat četnost e-mailových zpráv s hlášením o anomáliích zaslaných správci systému. Po aktivaci podmínky je počátek časového rozmezí nastaven na poslední ohlášenou anomálii.

O samotné vygenerování zprávy se stará funkce `alert_action`. Plugin peaKock v této funkci vytváří tělo a hlavičku e-mailové zprávy, která je následně obalena do MIME a odeslána. Předmět zprávy informuje o skutečnosti, že se jedná o plugin systému NfSen, a také o časovém rozmezí, ze kterého detekované anomálie pochází. Tělo zprávy obsahuje základní informace o profilu, jehož data byla použita při vyhodnocování anomálií a dále zde nalezneme specifikaci domácí sítě a odkaz na grafy zobrazující průběh časové řady včetně vyznačených anomálií. Následuje seznam jednotlivých anomálií spolu s detailními informacemi o nejvýznamnějších účastnících provozu v době výskytu anomálie. Ukázku e-mailového upozornění naleznete v příloze D.1 na straně 84.

6.3 Testování v rámci MU

Tato podkapitola slouží k prezentování výsledků testování pluginu peaKock v prostředí Masarykovy univerzity. Testování pluginu má za úkol odhalit jeho funkčnost, výkonnost a splnění požadavků.

Testovaní probíhalo na studentském kolektoru a anonymizovaných datech, neboť původně zamýšlený vývojový kolektor s neanonymizovanými daty bohužel s pluginem nespolupracoval. Za touto skutečností stojí zřejmě jeho 64bitová architektura. Veškeré komponenty softwarového vybavení obou serverů byly porovnány, nicméně další významný rozdíl nebyl nalezen. Autor se pokoušel o zprovoznění pluginu za pomoci vedoucího práce více než týden. Aplikace NfSen na vývojovém stroji plugin ignoruje mimo první spuštění po restartu. Kvůli absenci přímého přístupu k tomuto serveru nebylo možné problém dále zkoumat.

Plugin peaKock vyhodnocoval provoz na studentském kolektoru, který zaznamenává provoz mezi ICS MUNI a Internetem. Kromě simulovaných útoků bylo provedeno i sedmidenní testování. Tento rozsah byl vybrán, aby se podařilo sledovat reakci pluginu na provoz v různých stavech vytíženosti sítě. Víkend se zcela jistě bude lišit od pracovního týdne, stejně jako provoz v nočních hodinách od denního. Pokud testovaní vyžaduje samotný kanál, je třeba vytvořit stínový profil s tímto kanálem, neboť to je způsob jakým plugin s daty pracuje. Testování bylo rozděleno na dvě části.

6.3.1 Funkční testování

Funkční testování vyhodnocuje reakci na reálné či simulované anomálie a jejich správnou detekci a následnou vizualizaci. Pro tento cíl je třeba upravit pásmo netolerance jednot-

livých statistik, abychom nebyli zahlceni falešnými poplachy. Po pozorování provozu na studentském kolektoru navrhl autor hodnoty, které uvádí tabulka 6.2.

Bajty	Pakety	Toky	CP	ZP	CA	ZA	U	KP	SSH	DNS	HTTP	HTTPS	SMTP
45	55	55	35	35	40	40	45	50	50	45	50	55	65

Tabulka 6.2: Testování – nastavení netolerance anomálií

Zkratky: CP, CA – cílové porty a adresy, ZP, ZA – zdrojové porty a adresy, U – účastníci, KP – komunikační páry

První funkční testování proběhlo dne 3. 12. 2011 na studentském kolektoru a kanálu p3000 s anonymizovanými daty. Jednalo se o následující útoky:

- **Slovníkový útok na službu SSH pomocí nástroje sshtrix** [49] – jednalo se o 92 pokusů uhádnout uživatele a heslo. Následně došlo k zablokování přístupu aplikací DenyHosts [1]. Pluginu peaKock se anomálii podařilo zachytit a označit. Tato událost se udála ve slotu 10.20–10.25 a trvala 134,14 sekund.
- **Aplikační DoS pro webový server** – pomocí nástroje **ab** [33] pro testování výkonnosti webového serveru Apache, kterým byly vytvořeny požadavky na cílové porty 80/TCP a 443/TCP. V případě portu 80 bylo vysláno 1 000 požadavků o velikosti zhruba 9 kB, kdežto pro port 443 bylo odesláno 10 000 požadavků stejné velikosti. Program pracoval se čtyřmi vlákny a odesílal tedy požadavky souběžně. Cílový stroj zaznamenal v časovém slotu 11.10–11.15 celkem 880 požadavků na webovou službu v době trvání 15,46 sekundy. Obdobný útok na port 443 byl objeven s hodnotou 4 419 v časovém slotu 11.25–11.30. Obě události byly správně označeny v průběhu časové řady.
- **Blokové skenování** – s použitím nástroje **nmap** [55] byla síť 147.251.16.0/22 podrobena několika skenováním. První skenování testovalo zařízení v provozu pomocí zaslání zprávy ICMP Echo Request. Další zkoušelo síť spojeními s nastaveným příznakem TCP SYN na port 443 a spojeními s příznakem TCP ACK na port 80. Poslední ze skenování posílalo zprávu ICMP Timestamp. Skenování se významně projevilo v grafech cílových portů a cílových adres, kdy došlo k poklesu entropie oproti rostoucímu objemu provozu. Jednalo se o časové období 12.10-12.15, kdy se skenování provádělo na síti sledovaného kanálu a byla detekována anomálie. Celková doba provádění přesáhla jednu hodinu.
- **Objemový DoS ICMP záplavou** – nástrojem **ping** s parametrem *-f* proběhl pokus o tento typ útoku, nicméně v grafech se nikterak neprojevil, a tudíž nebyla ani detekována, neboť svým objemem nemohl konkurovat ostatnímu provozu ve stejném okamžiku. Útok probíhal cca 30 min z jediného stroje s internetovým připojením zdaleka nedosahujícím připojení univerzity. Pro úplnost uvádím, že doba běhu zahrnovala časové sloty 16.00–16.35.

Následující sedmidenní testování na anonymizovaných datech bylo zaměřeno na počty objevených anomálií získaných z e-mailových oznámení pluginu a databází pluginu.

1. DenyHost je systém aktivní obrany proti útokům na službu SSH instalovaný na studentském kolektoru.

Díky testování na autorovi přístupném kolektoru nebyl přístup k datům nikterak omezen. Oprávněnost vybraných anomálií byla posouzena v porovnání se zmíněnými daty.

6.3.2 Výkonnostní testování

Výkonnostní testovaní poskytuje informace o schopnosti pluginu peaKock vyrovnat se s určitým objemem dat. Zajímá nás tedy kolik síťových toků, paketů či bajtů je schopen zpracovat v předem daných časových intervalech, jako např. 1 sekunda či 1 minuta. Případně může přibližovat pohled z jiného úhlu, kde se budeme zajímat o prováděcí dobu zpracování určitého objemu dat z provozu sítě, např. 1 000, 10 000 či 100 000. Rovněž zajímavými údaji je doba běhu nejdůležitějších procedur backendu, stejně jako celková doba běhu.

Tyto údaje byly získány z pluginem zaznamenaných událostí v systémovém logu, který byl dále filtrován. Požadovaná úroveň logování pluginu je dva. Pro účely tohoto testu je pole netolerance nastaveno stejně jako při funkčním testování (viz tabulka 6.2) a byla dočasně vypnuta komprese pomocí bzip2, neboť její provádění je velmi časově i výpočetně náročné.

6.3.3 Výsledky testování

Testování probíhalo v období mezi 19. a 25. prosincem 2011 na studentském kolektoru Masarykovy univerzity. Pro dolování dat potřebných pro zmíněné testy byl vytvořen analytický modul pluginu peaKock, který umožňuje zobrazit informace o datech starších než jeden den. Zmiňovaná data získává jednoduchým skriptem spouštěným z cronu, který uloží databáze síťového provozu, anomálií i systémový log do adresáře. Analytický modul použije databáze, aby znovu vytvořil grafy s vyznačenými anomáliemi, tabulky detailních informací o anomáliích, a také tabulky funkčních a výkonnostních testů.

Hodnoty získané během sedmidenního testování na studentském kolektoru jsou uvedeny v tabulkách a grafech umístěných v příloze E na straně 85. Tabulka reprezentující funkční testování zobrazuje počty detekovaných anomálií. Tabulky výkonnostního testování uvádějí časy běhu jednotlivých procedur pluginu či základních operací, jako jsou zpracování jednoho záznamu nebo počet záznamů prověřených během jedné sekundy.

Celkem plugin nalezl 218 anomálií. Průměrně se počty anomálií pohybovaly v rozmezí 10–30 na den a nejvyšší počet v jedné charakteristice byl sedm. Velmi často se anomálie objevovaly v charakteristikách bajtů, paketů, síťových toků a aplikačních portů. Naopak zřídka bylo možné pozorovat anomálie v charakteristikách cílových a zdrojových portů. Denní provoz předčil dle očekávání provoz noční, ovšem víkendový provoz také vykazoval mnoho anomálií.

Zpracování tisíce síťových toků obvykle nepřekročilo hodnotu několika setin sekundy. Pro pakety a bajty se jednalo o hodnoty ještě menší. Jedna sekunda tedy poskytla dostatek času pro zpracování průměrně dvaceti tisíc síťových toků. Celková doba běhu pluginu se pohybovala mezi jednou a dvěma sekundami, z čehož zhruba polovinu vždy spotřebovalo dolování dat z úložiště NfSenu.

Ověření detekovaných anomálií

Deset vybraných anomálií bylo předáno vedoucímu práce k ověření na neanonymizovaných datech. Vyhodnocení je shrnuto v následující tabulce 6.3 jsou shrnuty detekované anomálie, které byly vybrány pro ověření vedoucím práce na neanonymizovaných datech. Zvolené anomálie se projevily buď výskytem ve více měřených charakteristikách, nebo byly nápadné svojí hodnotou výrazně převyšující své okolí o jeden či dva řády.

Pořadí	Čas výskytu	Detekováno v	Typ anomálie
1.	19.12.2011 2.40	pakety, bajty	provozní
2.	19.12.2011 19.30 19.12.2011 19.50	SSH	bezpečnostní
3.	21.12.2011 11.55	síťové toky	neurčeno
4.	21.12.2011 21.40	bajty	provozní
5.	22.12.2011 22.55	SSH	bezpečnostní
6.	23.12.2011 18.25 23.12.2011 18.30	HTTPS, SMTP cílové i zdrojové IP a porty, SSH	bezpečnostní
7.	23.12.2011 19.00 23.12.2011 19.10	bajty, pakety síťové toky, bajty, pakety	provozní
8.	24.12.2011 12.05	cílové IP, komunikační páry, síťové toky	provozní
9.	25.12.2011 10.25	cílové i zdrojové porty, zdrojové IP	neurčeno
10.	25.12.2011 11.10	DNS	neurčeno

Tabulka 6.3: Vybrané anomálie detekované pluginem peaKock

Anomálie jsou v tabulce seřazeny chronologicky, a tedy i jejich vyhodnocení dodržuje stejné pořadí. První anomálie nebyla nalezena v neanonymizovaných datech a dle vedoucího se jednalo o chybu při anonymizaci. Z tohoto důvodu byla prohlášena za provozní anomálii. Druhá anomálie byla bezpečnostní, protože se jednalo o SSH útok, který byl nezávisle na peaKocku nahlášen jinými nástroji. Třetí anomálie byla zaznamenána v charakteristice síťových toků. Vedoucí následně zjistil, že šlo o náhlý nárust dotazů na hlavní DNS server Masarykovy univerzity. Zda se jednalo o bezpečnostní či provozní anomálii, nebylo bez podrobné analýzy možné zjistit. Čtvrtá anomálie nebyla sice přímo označena pluginem peaKock, ale ve vizualizaci byla natolik výrazná, že ji autor vybral k ověření. Vedoucí ji vyhodnotil jako provozní anomálii, která vznikla v důsledku tuzemského VoIP provozu.

Patá anomálie se ukázala jako bezpečnostní, neboť se jednalo o skenování SSH portu na vybrané skupině sledovaných strojů pocházející z ruské IP adresy. Šestá anomálie označovala simulované skenování několika vybraných portů na sledované podsíti. PeaKock ji správně označil a vedoucí bezchybně identifikoval IP adresu, ze které útok pocházel. Sedmá anomálie zaznamenala přenos pomocí služby SSH mezi stroji Masarykovy univerzity, a proto byla označena jako provozní. I další, tedy osmá, anomálie byla označena jako provozní, neboť se jednalo o komunikaci více strojů s jedním v distribuovaném prostředí. Poslední dvě anomálie byly velmi podobné anomálii třetí a bez podrobnější analýzy nebylo možné určit, zda se jednalo o provozní či bezpečnostní anomálie. Vedoucí práce nicméně potvrdil, že plugin peaKock je označil zcela oprávněně.

Z vyhodnocení je zřejmé, že pluginu se podařilo zachytit několik provozních anomálií i bezpečnostních anomálií a je schopen správně identifikovat a označit anomálie v síťovém provozu. Nastavení citlivosti pluginu ovlivní úroveň nahlášených *false positive*, které se mezi vybranými anomáliemi již nevyskytovaly, protože šlo o cíleně zvolený soubor.

Kapitola 7

Závěr

Ačkoli principy monitorování rozsáhlých a rychlých sítí pomocí síťových toků jsou známé, nástroje pro jejich zpracování se liší dle funkcionality i výrobce. Tato roztříštěnost trhu zpomaluje ustálení technologie a vznik kvalitních open-source nástrojů a dokumentace. Jednou ze slibných aplikací je NfSen, který bohužel neumožňuje automatickou detekci anomálií vyskytujících se v síťovém provozu. Úkolem této práce bylo prozkoumat možnosti zpracování a vizualizace síťových toků NetFlow a následně navrhnout a implementovat plugin pro aplikaci NfSen poskytující požadovanou detekci anomálií.

V úvodní kapitole jsme se seznámili s technologií síťových toků, aplikací NfSen a základními pojmy. V následujících kapitolách se autor snažil usnadnit návrháři práci při volbě nástrojů pro tvorbu pluginu. Prvotním problémem bylo uchování dat. Uložení potřebných dat ve vlastní režii pomohlo ulehčit úložišti dat aplikace NfSen, neboť neustálé prosévání kompletních dat zatěžovalo systém a prodlužovalo výpočetní čas. Pestrá nabídka databázových aplikací umožnila autorovi získat patřičný rozhled při návrhu pluginu a vybrat databázi odpovídající požadavkům pluginu.

Prezentace dat byla neméně důležitá. Další kapitola přinesla podrobné srovnání vizualizačních knihoven, které se soustředí na dokumentaci a její dostupnost, licenční politiku, vzhled, rozsáhlost konfigurace a v neposlední řadě i na využití výpočetních zdrojů. Tato kapitola posloužila autorovi v roli programátora, neboť představila relevantní konfigurační volby využitelné při vizualizaci síťových toků. Celá kapitola navíc obsahuje vzorové konfigurace včetně grafů pro každou z vizualizačních knihoven. Autorovi si nepodařilo najít na internetu takto rozsáhlé srovnání, a proto věří, že tato práce ušetří návrháři i vývojáři podobné aplikace mnoho času stráveného testováním dostupných možností.

Ovšem pokud bychom nechtěli vlastní implementaci, NfSen data ukládá i vizualizuje sám o sobě. Co bylo tedy nejdůležitější? Detekce anomálií. Data síťových toků jsou pouhým průběhem časové řady, jak nám byla představena v úvodní kapitole. V této práci bylo navázáno na výzkum, ve kterém byla pro detekci anomálií použita entropie získaná za pomoci komprese. Přesně proto jedna z kapitol představila principy některých běžně používaných kompresních algoritmů, a hlavně přinesla komplexní srovnání jejich vlastností.

V tomto okamžiku získal autor dostatek informací pro návrh a implementaci pluginu. Tomuto tématu byla věnována jedna z dalších kapitol. Poté co byly jasně formulovány požadavky na plugin, využil autor nabyté vědomosti pro jejich uspokojení. Výstupem této práce je tedy funkční implementace pluginu pro aplikaci NfSen, které je schopná zpracovat a vizualizovat síťové toky, a umožňuje automatickou detekci anomálií. Pro tuto detekci byl

navržen algoritmus, který umožňuje detekovat maxima či minima v průběhu časové řady a uvažovat pouze ta z nich, která jsou významná z pohledu uživatele pluginu. Vizualizace byla doplněna o e-mailové hlášení shrnující detekované anomálie, které odstraňuje nutnost sledovat grafy vývoje průběhu síťových toků v pravidelných intervalech.

V neposlední řadě byly chování i vlastnosti pluginu ověřeny za pomoci funkčního a výkonnostního testování na provozu Masarykovy univerzity. Plugin tak pravděpodobně poslouží jako další díl skládačky pro zajištění bezpečnosti a analýzu síťového provozu na Masarykově univerzitě. Samotný text práce obsahuje mnoho užitečných informací jak pro návrháře, tak pro vývojáře dalších nástrojů pro analýzu síťových toků.

V budoucnu by bylo možné plugin obohatit o zásuvné moduly jednotlivých kompresních algoritmů či vizualizačních knihoven, a tak umožnit jeho přizpůsobení konkrétním požadavkům a prostředí.

Literatura

[1] ARBOR NETWORKS. *Peakflow SP: Traffic Anomaly Detection* [online]. 2011. [cit. 11. 11. 2011]. Dostupné z: <http://www.arbornetworks.com/peakflow-sp-traffic-anomaly-detection.html>.

[2] BURROWS, M. – WHEELER, D. J. *A block-sorting lossless compression algorithm* [online]. 1993. [cit. 11. 11. 2011]. Dostupné z: <http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-124.pdf>.

[3] CALIGARE. *NetFlow Analyzer Protocol – Caligare Flow Inspector* [online]. 2011. [cit. 11. 11. 2011]. Dostupné z: <http://www.caligare.com/netflow/caligare_flow_inspector.php>.

[4] CHATFIELD, C. The Holt-Winters Forecasting Procedure. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*. 1978, 27, 3, s. 264–279.

[5] CHATFIELD, C. – YAR, M. Holt-Winters Forecasting: Some Practical Issues. *Journal of the Royal Statistical Society. Series D (The Statistician)*. 1988, 37, 2, s. 129–139.

[6] CHEN, C.-h. et al. *Handbook of Data Visualization (Springer Handbooks of Computational Statistics)*. Santa Clara, CA, USA : Springer-Verlag TELOS, 1 edition, 2008. ISBN 3540330364, 9783540330363.

[7] CREATIVE COMMONS. *Creative Commons Attribution-NonCommercial 3.0* [online]. 2007. [cit. 19. 10. 2011]. Dostupné z: <https://creativecommons.org/licenses/by-nc/3.0/cz/>.

[8] DEUTSCH, P. *DEFLATE Compressed Data Format Specification version 1.3* [online]. IETF, May 1996. Dostupné z: <http://www.ietf.org/rfc/rfc1951.txt>.

[9] DEUTSCH, P. *GZIP file format specification version 4.3* [online]. IETF, May 1996. Dostupné z: <http://www.ietf.org/rfc/rfc1952.txt>.

[10] EASTON, V. J. – MCCOLL, J. H. *Statistics glossary – Time series data* [online]. 1997. [cit. 19. 10. 2011]. Dostupné z: <http://www.stats.gla.ac.uk/steps/glossary/time_series.html#timeseries>.

[11] FELDSPAR, A. *An Explanation of the DEFLATE Algorithm* [online]. 1997. [cit. 25. 9. 2011]. Dostupné z: <http://www.gzip.org/deflate.html>.

[12] GAILLY, J.-L. – ADLER, M. [online]. 1997. [cit. 25. 9. 2011]. Dostupné z: <http://www.gzip.org/algorithm.txt>.

[13] IBM. *AURORA: Traffic analysis and visualisation* [online]. 2011. [cit. 11. 11. 2011]. Dostupné z: <http://www.zurich.ibm.com/aurora/>.

[14] IBM. *IBM Performance Management – Tivoli Netcool Performance Flow Analyzer* [online]. 2011. [cit. 11. 11. 2011]. Dostupné z: <http://www-01.ibm.com/software/tivoli/products/netcool-performance-mgr/>.

[15] ICS CYBER. *Botnet Chuck Norris* [online]. [cit. 25. 9. 2011]. Dostupné z: <http://www.muni.cz/ics/research/projects/4622/web/chuck_norris_botnet>.

[16] KEIM, D. A. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*. 2002, 8, 1, s. 1–8.

[17] MARTIN, G. N. N. *Range encoding: an algorithm for removing redundancy from a digitised message* [online]. 1979. [cit. 11. 11. 2011]]. Dostupné z: <http://web.archive.org/web/20041014083730/http://www.compressconsult.com/rangecoder/rngcod.pdf.gz>.

[18] MAŘÍK, V. *Kybernetika a umělá inteligence, 1. přednáška* [online]. 2000. [cit. 11. 1. 2011]. Dostupné z: <http://cyber.felk.cvut.cz/gerstner/teaching/kui/prednasky/1/>.

[19] OBDRŽÁLEK, J. *Entropie a informace* [online]. 2007. [cit. 11. 1. 2011]. Dostupné z: <http://utf.mff.cuni.cz/~jobdr/download/ENTR-INF4.pdf>.

[20] OBERHUMER, M. *LZO – a real-time data compression library* [online]. 2011. [cit. 11. 11. 2011]]. Dostupné z: <http://www.oberhumer.com/opensource/lzo/lzodoc.php>.

[21] OBERHUMER, M. *LZO Frequently Asked Questions* [online]. 2011. [cit. 11. 11. 2011]]. Dostupné z: <http://www.oberhumer.com/opensource/lzo/lzofaq.php>.

[22] PLONKA, D. *FlowScan: A Network Traffic Flow Reporting and Visualization Tool* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://net.doit.wisc.edu/~plonka/lisa/FlowScan/>.

[23] POGOLOTTI, J.-D. *Introduction of 3D charts (support forum)* [online]. 2010-2011. [cit. 18. 10. 2011]. Dostupné z: <http://wiki.pchart.net/forum/viewtopic.php?f=3&t=2125>.

[24] RSNAKE. *Slowloris HTTP DoS* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://ha.ckers.org/slowloris/>.

[25] SALOMON, D. *Data compression: the complete reference*. London, UK : Springer-Verlag, 2007. With contributions by Giovanni Motta and David Bryant. ISBN 1-84628-602-6.

[26] SPEROTTO, A. et al. An Overview of IP Flow-Based Intrusion Detection. *IEEE Communications Surveys & Tutorials*. 2010, 12, 3.

[27] USHAKANTH, V. *A New Visualization for Web Server Logs* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://opensourcedevelopmentsgdhub.blogspot.com/2007/03/new-visualization-for-web-server-logs.html>.

[28] VESELÝ, V. *Úvod do časových řad* [online]. 2003. [cit. 19. 10. 2011]. Dostupné z: <http://www.econ.muni.cz/~vesely/papers/ad03cr.pdf>.

[29] WAGNER, A. *Entropy-Based Worm Detection for Fast IP Networks*. PhD thesis, Eidgenossische Technische Hochschule Zurich, 2008.

[30] WEGMAN, E. J. Hyperdimensional Data Analysis Using Parallel Coordinates. *Journal of the American Statistical Association*. 1990, 85, s. 664–675. Dostupné z: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.1902>.

[31] WIKIPEDIA. *Entropy (information theory) — Wikipedia, The Free Encyclopedia* [online]. 2011. [cit. 11. 11. 2011]]. Dostupné z: <http://en.wikipedia.org/w/index.php?title=Entropy_(information_theory)>.

Seznam programů

[32] 10GEN. *MongoDB* [online]. [cit. 29. 11. 2011]. Dostupné z: <http://www.mongodb.org/>.

[33] APACHE SOFTWARE FOUNDATION. *ab – Apache HTTP server benchmarking tool* [online]. [cit. 2. 12. 2011]. Dostupné z: <http://httpd.apache.org/docs/2.2/programs/ab.html>.

[34] APACHE SOFTWARE FOUNDATION. *Batik SVG Toolkit* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://xmlgraphics.apache.org/batik/>.

[35] ARUBA NETWORKS, INC. *AirWave* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.arubanetworks.com/products/management-security-software-2/airwave/>.

[36] CACTI GROUP, THE. *Cacti - The Complete RRDTool-based Graphing Solution* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://cacti.net/>.

[37] CALDWELL, O. *jsFiddle – Online editor for the Web (JavaScript, MooTools, jQuery, Prototype, YUI, Glow and Dojo, HTML, CSS)* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://jsfiddle.net/>.

[38] COLIN, L. *XZ Utils* [online]. [cit. 11. 11. 2011]. Dostupné z: <http://tukaani.org/xz/>.

[39] COLIN, L. *The .xz file format* [online]. [cit. 6. 12. 2011]. Dostupné z: <http://tukaani.org/xz/xz-file-format.txt>.

[40] FULLMER, M. – OSU. *Flow-tools* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.splintered.net/sw/flow-tools/>.

[41] GEEKNET, INC. *SourceForge.net: Find, Create and Publish Open Source software for free* [online]. [cit. 20. 11. 2011]. Dostupné z: <http://sourceforge.net/>.

[42] GOOGLE. *Google Chrome Developer Tools* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://code.google.com/intl/cs/chrome/devtools/>.

[43] HAAG, P. *Nfdump* [online]. [cit. 29. 7. 2011]. Dostupné z: <http://nfdump.sourceforge.net/>.

[44] HAAG, P. *NfSen – Netflow Sensor* [online]. [cit. 29. 7. 2011]. Dostupné z: <http://nfsen.sourceforge.net/>.

[45] HIGHSOFT SOLUTIONS AS. *Highcharts - Interactive JavaScript charts for your webpage* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.highcharts.com/>.

[46] HIGHSOFT SOLUTIONS AS. *Highcharts - Options Reference* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.highcharts.com/ref/>.

[47] HIPP, D. R. *SQLite* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.sqlite.org/>.

[48] JQUERY PROJECT, THE. *jQuery: The Write Less, Do more, JavaScript Library* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://jquery.org/>.

[49] KAYAN, L. *sshtrix* [online]. [cit. 2. 12. 2011]. Dostupné z: <http://www.nullsecurity.net/tools.html>.

[50] LAURSEN, O. *flot - Attractive Javascript plotting for jQuery - Google Project* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://code.google.com/p/flot/>.

[51] LAURSEN, O. *Plugins – flot – Links to flot plugins* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://code.google.com/p/flot/wiki/Plugins>.

[52] LAURSEN, O. *FlotUsage – flot – Sites and projects using Flot* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://code.google.com/p/flot/wiki/Plugins>.

[53] LEONELL, C. *jqPlot Charts and Graphs for jQuery* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.jqplot.com>.

[54] LEONELL, C. *jqPlot options* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.jqplot.com/docs/files/jqPlotOptions-txt.html#jqPlot_Options>.

[55] LYON, G. *Nmap – Free Security Scanner For Network Exploration & Security Audits* [online]. [cit. 2. 12. 2011]. Dostupné z: <http://nmap.org/>.

[56] MICROSOFT, CORP. *AT command* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/at.mspx?mfr=true>.

[57] MOZILLA FOUNDATION. *Firebug* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://getfirebug.com/>.

[58] MYSQL AB. *MySQL – The world's most popular open source database* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.mysql.org/>.

[59] MYSQL AB. *MySQL 5.1 Reference Manual – MySQL Storage Engine Architecture* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://dev.mysql.com/doc/refman/5.1/en/pluggable-storage-overview.html>.

[60] OBERHUMER, M. *Lzop file compressor* [online]. [cit. 11. 11. 2011]. Dostupné z: <http://www.lzop.org/>.

[61] OETIKER, T. *MRTG - Tobi Oetiker's MRTG - The Multi Router Traffic Grapher* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://oss.oetiker.ch/mrtg/>.

[62] OETIKER, T. *Tuning RRDtool for performance* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://oss.oetiker.ch/rrdtool-trac/wiki/TuningRRD>.

[63] OETIKER, T. *RRDTool* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://oss.oetiker.ch/rrdtool/>.

[64] OETIKER, T. *SmokePing* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://oss.oetiker.ch/smokeping/>.

[65] PAVLOV, I. *7-zip* [online]. [cit. 11. 11. 2011]. Dostupné z: <http://www.7-zip.org/>.

[66] POGOLOTTI, J.-D. *pChart | a PHP Charting library* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://pchart.sourceforge.net/>.

[67] POGOLOTTI, J.-D. *pChart 2.0 – a PHP charting library* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.pchart.net/>.

[68] POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL – The world's most advanced open source database* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.postgresql.org/>.

[69] POSTGRESQL GLOBAL DEVELOPMENT GROUP. *A Tour of PostgreSQL Internals* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.postgresql.org/files/developer/tour.pdf>.

[70] PROIETTI, V. et al. *MooTools - a compact javascript framework* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://mootools.net/>.

[71] PROTOTYPE CORE TEAM. *Prototype JavaScript framework: Easy Ajax and DOM manipulation for dynamic web applications* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.prototypejs.org/>.

[72] SCHWEIKERT, D. *Mailgraph - a RRDtool frontend for Mail statistics* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://mailgraph.schweikert.ch/>.

[73] SELTZER, M. – BOSTIC, K. *Berkeley DB* [online]. [cit. 17. 11. 2011]. Dostupné z: <http://www.oracle.com/technology/products/berkeley-db>.

[74] SEWARD, J. R. *Bzip2 and libzip2* [online]. [cit. 11. 11. 2011]. Dostupné z: <http://bzip.org/>.

[75] SFLOW.ORG. *sFlow – Making the Network Visible* [online]. [cit. 29. 7. 2011]. Dostupné z: <http://www.sflow.org/>.

[76] TERREACTIVE AG. *Die Spezialisten für umfassende IT-Security. – terreActive* [online]. [cit. 20. 9. 2011]. Dostupné z: <http://www.terreactive.ch/>.

[77] VIXIE, P. *ISC (Internet Systems Consortium) CRON* [online]. [cit. 20. 9. 2011]. Dostupné z: <https://secure.wikimedia.org/wikipedia/en/wiki/Vixie_cron#Modern_versions>.

[78] ZHITOMIRSKIY, I. et al. *The Diaspora Project* [online]. [cit. 20. 11. 2011]. Dostupné z: <http://diasporafoundation.org/>.

Příloha A

Seznamy obrázků, tabulek, konfigurací a zdrojových kódů

Seznam obrázků

Seznam tabulek

Seznam konfigurací a zdrojových kódů

Příloha B

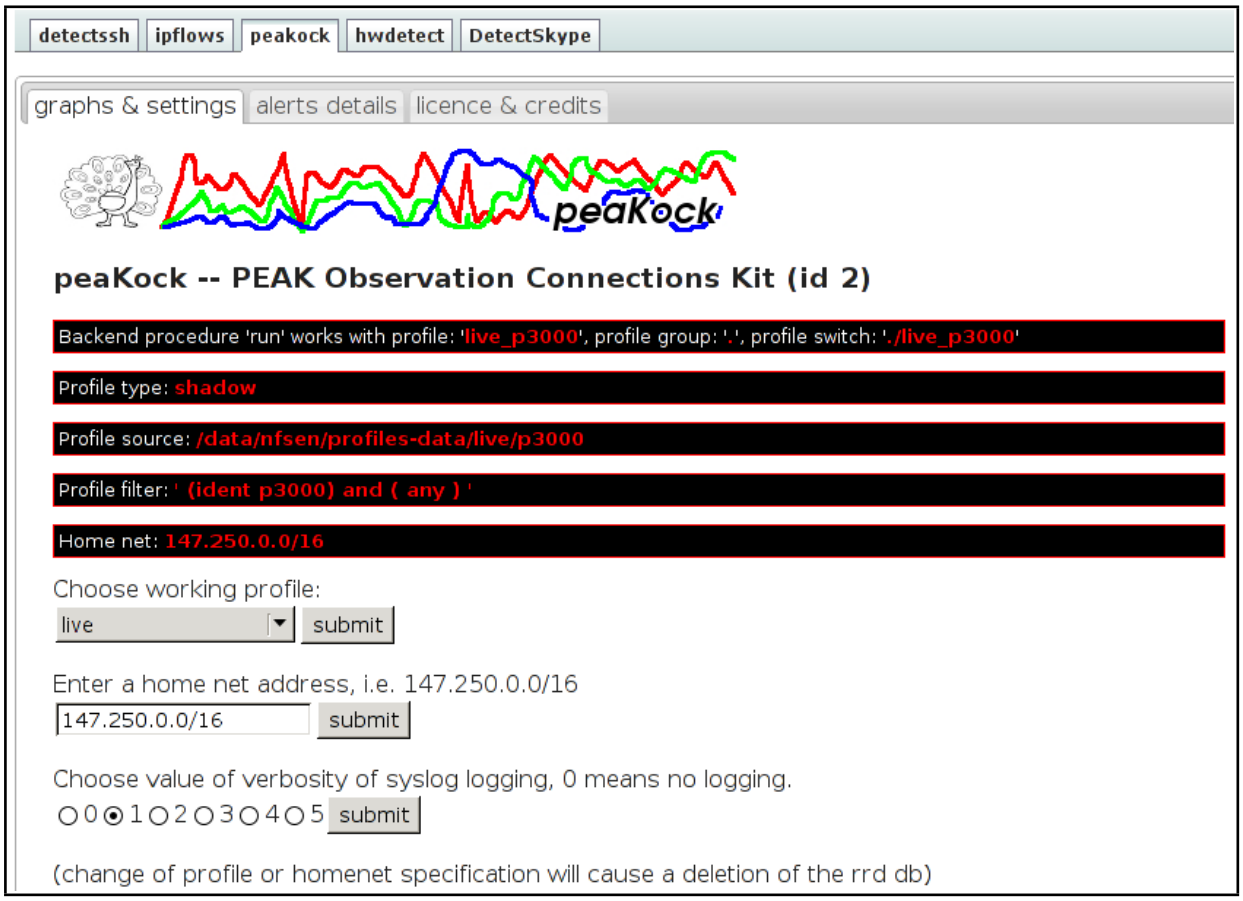
Obsah přiloženého CD

Kompaktní disk obsahuje následující strukturu:

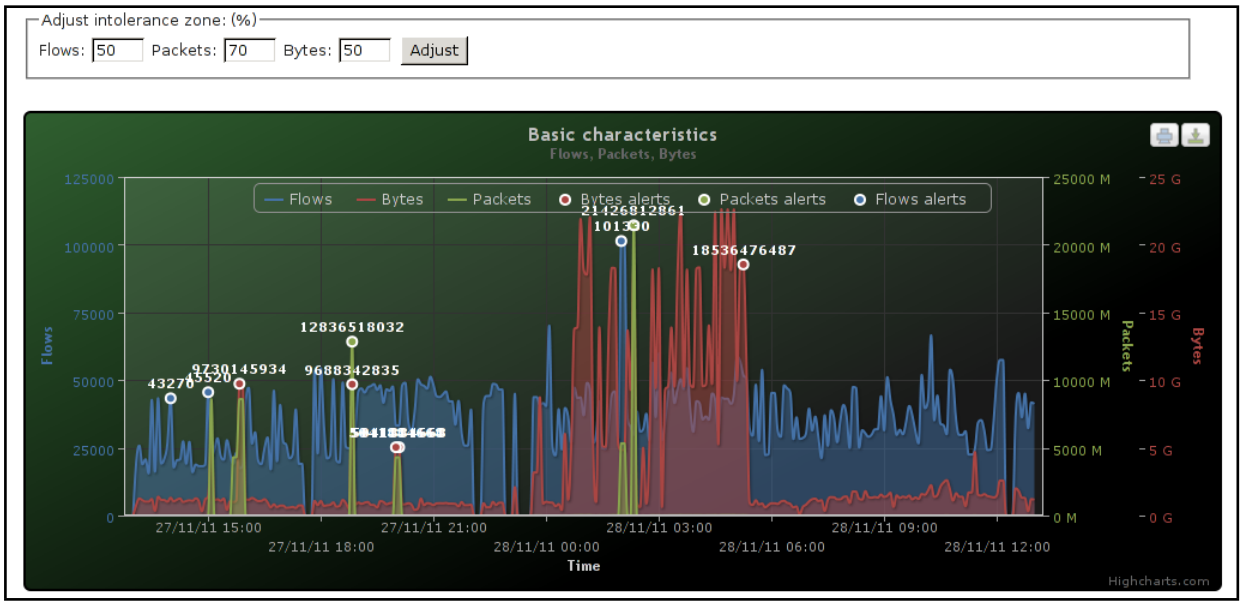
- Zdrojové kódy zásuvného modulu.
- Výstupy a nastavení testování.
- Text této práce v PDF.

Příloha C

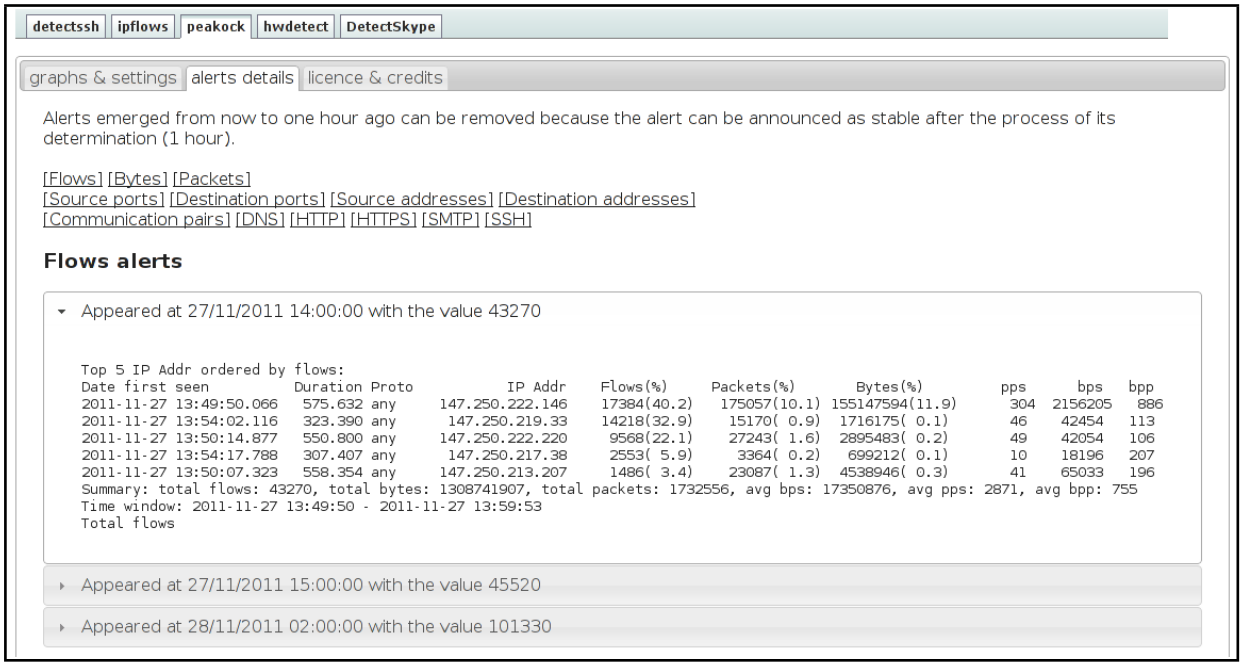
Frontend pluginu peaKock



Obrázek C.1: Ukázka rozhraní pluginu peaKock – Informace o profilu



Obrázek C.2: Ukázka rozhraní pluginu peaKock – Graf a nastavení pásma netolerance anomálií



Obrázek C.3: Ukázka rozhraní pluginu peaKock – Detailní informace o anomáliích

Příloha D

Backend pluginu peaKock

Data	DB	Funkce	R/W	Popis
Data	RRD	**CreateRRD**	W	Slouží k vytvoření databáze RRD, která uchovává data z úložiště NfSen po 24 hodin. Pro svoje prvotní naplnění interně volá funkci UpdateRRD.
Data	RRD	**UpdateRRD**	W	Funkce k získání požadovaných dat z úložiště NfSen dle profilu, filtru a domácí sítě. Dále vypočte hodnoty entropie za pomoci komprese a počty komunikačních párů a účastníků provozu. Výsledek uloží do RRD databáze. Je volána periodicky každých pět minut.
Data	RRD	**FetchDS**	R	Funkce získá z databáze RRD data jedné konkrétní veličiny pro určitý časový úsek.
Anomálie	BerkeleyDB	**UpdateDataDB**	W	Funkce ukládá data o anomáliích či jejich detailech do patřičné databáze a současně dohledává detailní informace v úložišti NfSen. Následně odstraní hodnoty starší než 24 hodin.
Anomálie	BerkeleyDB	**GetDataDB**	R	Funkce vrací kompletní obsah databáze anomálií či jejich detailů.
Anomálie	BerkeleyDB	**GetValueDB**	R	Funkce vrací jednu konkrétní hodnotu z dabáze anomálií či jejich detailů v zadaném čase.

Tabulka D.1: Plugin peaKock – Funkce pro ukládání dat

Funkce	Popis
UpdateAlerts	Funkce pro detekci anomálií. Načítá data z RRD, které dále prochází funkcemi **GivePeaks** a **CommonEl**. Výsledek ukládá do databáze anomálií. Při uložení se dohledávají detaily anomálií a ukládají se do vlastní databáze.
GivePeaks	Funkce pro výpočet lokálních extrémů v předaných datech – maxima či minima.
CommonEl	Funkce vyhledávající společné body mezi lokálními extrémy více síťových charakteristik.
GetAlertName	Funkce vracející název statistiky na základě názvu souboru s anomáliemi.
GetAlertStats	Funkce definující parametry pro statistiky vygenerovanou nfdumpem, dle typu anomálie.
GetAlertFilter	Funkce poskytující filtr pro dotazování se při získávání detailů anomálií.

Tabulka D.2: Plugin peaKock – Funkce pro detekci anomálií

Zdrojový kód D.1: Ukázka části e-mailového upozornění o anomáliích

```
Subject: peaKock (nfsen plugin) alerts! (27/11/11 15:45 - 27/11/11 20:50)

Alerts for time period 27/11/11 15:45 - 27/11/11 20:50
-------------------------------------------------------------------------------

Profile: live_p3000
Profile group: .
Profile type: shadow
Profile source: /data/nfsen/profiles-data/live/p3000
Profile filter: ' (ident p3000) and ( any ) '
Homenet: 147.250.0.0/16
-------------------------------------------------------------------------------
You can find graphs at: https://nftest.ics.muni.cz/nfsen/nfsen.php?sub_tab=2
-------------------------------------------------------------------------------

HTTP alerts
===============================================
Appeared at 27/11/11 17:20 with the value 70.
-----------------------------------------------

Top 5 Src IP Addr ordered by flows:
Date first seen          Duration Proto       Src IP Addr    Flows(%)      Packets(%)       Bytes(%)          pps      bps   bpp
2011-11-27 17:14:25.907    24.098 any     dffe:f8..84:7b7a      49(70.0)       381( 9.0)     50747(17.3)       15    16846   133
2011-11-27 17:14:30.013   271.359 any         66.249.92.84       7(10.0)        63( 1.5)      5724( 2.0)        0      168    90
2011-11-27 17:15:57.955   196.002 any     df01:18..f5:1c89       6( 8.6)      3724(88.3)    231560(79.1)       18     9451    62
2011-11-27 17:15:14.856    94.159 any         66.249.83.60       2( 2.9)        18( 0.4)      1600( 0.5)        0      135    88
2011-11-27 17:19:16.018     0.794 any       170.16.42.170       1( 1.4)         6( 0.1)       621( 0.2)        7     6256   103

Top 5 Dst IP Addr ordered by flows:
Date first seen          Duration Proto       Dst IP Addr    Flows(%)      Packets(%)       Bytes(%)          pps      bps   bpp
2011-11-27 17:14:25.907    24.098 any     20fe:e6..ff:0:d3      49(70.0)       381( 9.0)     50747(17.3)       15    16846   133
2011-11-27 17:14:30.013   271.359 any       147.250.205.90      10(14.3)        88( 2.1)      7938( 2.7)        0      234    90
2011-11-27 17:19:09.722     4.235 any     d500:eb..:0:ff06       3( 4.3)        15( 0.4)      4958( 1.7)        3     9365   330
2011-11-27 17:18:24.140    46.227 any     15ff:eb..ff00:10       2( 2.9)      3478(82.5)    211407(72.2)       75    36585    60
2011-11-27 17:18:18.796    58.016 any      147.250.214.141       2( 2.9)        12( 0.3)      1242( 0.4)        0      171   103

Summary: total flows: 70, total bytes: 292698, total packets: 4218, avg bps: 8049, avg pps: 14, avg bpp: 69
Time window: 2011-11-27 17:14:25 - 2011-11-27 17:19:16
Total flows processed: 39156, Blocks skipped: 0, Bytes read: 2041896
Sys: 0.022s flows/second: 1702805.0  Wall: 0.014s flows/second: 2613013.0

Appeared at 27/11/11 20:45 with the value 1196.
-----------------------------------------------
Top 5 Src IP Addr ordered by flows:
Date first seen          Duration Proto       Src IP Addr    Flows(%)      Packets(%)       Bytes(%)          pps      bps   bpp
2011-11-27 20:42:35.987    66.603 any       59.50.118.241    1177(98.4)     1464(46.1)     80363(42.0)       21     9652    54
2011-11-27 20:42:05.313    20.331 any     20fe:18..5b:7b2a       6( 0.5)        75( 2.4)      7465( 3.9)        3     2937    99
2011-11-27 20:40:39.618    14.497 any     20fe:18..a1:7a4d       3( 0.3)        32( 1.0)      3268( 1.7)        2     1803   102
2011-11-27 20:39:15.947   204.590 any     df01:18..f5:1c89       2( 0.2)      1554(49.0)     94621(49.4)        7     3699    60
2011-11-27 20:42:45.570     0.522 any     dffe:f8..36:20f8       2( 0.2)         8( 0.3)      1575( 0.8)       15    24137   196

Top 5 Dst IP Addr ordered by flows:
Date first seen          Duration Proto       Dst IP Addr    Flows(%)      Packets(%)       Bytes(%)          pps      bps   bpp
2011-11-27 20:42:36.359    57.412 any       147.250.205.90       7( 0.6)        47( 1.5)      3601( 1.9)        0      501    76
2011-11-27 20:39:57.644   205.838 any      147.250.214.141       6( 0.5)        32( 1.0)      2573( 1.3)        0      100    80
2011-11-27 20:42:36.442    60.128 any      147.250.205.194       6( 0.5)        30( 0.9)      2449( 1.3)        0      325    81
2011-11-27 20:40:36.619   177.394 any       147.250.205.79       5( 0.4)        26( 0.8)      2210( 1.2)        0       99    85
2011-11-27 20:42:36.429    59.869 any       147.250.205.91       5( 0.4)        25( 0.8)      2042( 1.1)        0      272    81

Summary: total flows: 1196, total bytes: 191383, total packets: 3173, avg bps: 5741, avg pps: 11, avg bpp: 60
Time window: 2011-11-27 20:39:15 - 2011-11-27 20:43:42
Total flows processed: 47863, Blocks skipped: 0, Bytes read: 2497128
Sys: 0.025s flows/second: 1841238.7  Wall: 0.018s flows/second: 2641300.1
```

Příloha E

Testování pluginu peaKock

Anomálie objevena v	Pondělí		Úterý		Středa		Čtvrtek		Pátek		Sobota	Neděle
	den	noc	den	noc	den	noc	den	noc	den	noc	víkend	
Síťové toky	0	1	4	1	4	4	1	0	0	0	1	2	3
Bajty	0	1	1	3	1	0	0	2	1	2	2	0	1
Pakety	3	2	5	3	1	0	0	2	1	2	3	1	1
Zdrojové porty	0	0	2	0	0	0	0	0	0	0	1	0	1
Cílové porty	0	0	0	0	0	0	0	0	0	0	1	0	1
Zdrojové adresy	0	2	4	1	0	1	0	0	0	0	2	0	2
Cílové adresy	0	1	1	1	0	0	0	0	0	0	2	1	0
Komunikační páry	1	0	2	1	0	0	2	1	0	0	3	7	4
Účastníci provozu	2	0	1	1	0	0	0	1	0	1	1	0	2
DNS	0	1	2	0	1	1	0	0	1	0	0	2	2
HTTP	5	2	1	5	3	5	3	1	1	4	0	4	5
HTTPS	3	1	2	0	0	1	1	0	1	2	3	6	3
SMTP	0	0	0	0	0	0	2	0	0	0	1	2	0
SSH	3	4	2	2	1	1	0	4	0	1	2	4	4
Celkem anomálií	17	15	27	18	11	13	9	11	5	12	22	29	29
Studentský kolektor, 19. prosince 2011 – 25. prosince 2011, profil live_p3000													

Tabulka E.1: Plugin peaKock – Funkční testování

Časové období: den (6.00–17.00), noc (17.00–6.00), víkend (pá 17.00–po 6.00)

Měřená veličina		Pondělí		Úterý		Středa	
		den	noc	den	noc	den	noc
Toky	1 000 [s]	0,038 9	0,043 6	0,042 0	0,073 4	0,030 2	0,059 4
	10 000 [s]	0,388 8	0,436 2	0,420 1	0,733 6	0,301 7	0,594 2
	100 000 [s]	3,887 2	4,362 1	4,200 5	7,335 6	3,017 1	5,941 5
	1 s [počet]	21 772	20 086	17 555	6 949	19 457	8 199
	60 s [počet]	1 306 293	1 205 169	1 053 319	416 949	1 167 421	491 947
Bajty	1 000 [s]	$9{,}0\times10^{-7}$	$1{,}0\times10^{-6}$	$1{,}0\times10^{-6}$	$2{,}6\times10^{-6}$	$7{,}0\times10^{-7}$	$8{,}0\times10^{-7}$
	10 000 [s]	$9{,}0\times10^{-6}$	$1{,}0\times10^{-5}$	$1{,}0\times10^{-5}$	$2{,}6\times10^{-5}$	$7{,}0\times10^{-6}$	$8{,}0\times10^{-6}$
	100 000 [s]	$9{,}0\times10^{-5}$	0,000 1	0,000 1	0,000 3	7.0×10^{-5}	8.0×10^{-5}
	1 s [počet]	964 699 605	893 295 896	733 837 097	194 539 826	849 895 880	617 914 989
	60 s [počet]	57 881 976 319	53 597 753 764	44 030 225 827	11 672 389 541	50 993 752 771	37 074 899 349
Pakety	1 000 [s]	$7{,}9\times10^{-6}$	$9{,}3\times10^{-6}$	$3{,}5\times10^{-6}$	$7{,}7\times10^{-6}$	$2{,}46\times10^{-5}$	0,000 8
	10 000 [s]	$7{,}9\times10^{-5}$	$9{,}3\times10^{-5}$	$3{,}5\times10^{-5}$	$7{,}7\times10^{-5}$	0,000 2	0,007 6
	100 000 [s]	0,000 8	0,000 9	0,000 4	0,000 8	0 002 5	0,076 3
	1 s [počet]	107 757 970	93 934 625	209 911 764	65 892 680	23 825 864	638 147
	60 s [počet]	6 465 478 194	5 636 077 497	12 594 705 844	3 953 560 805	1 429 551 866	38 288 797
Funkce	**UpdateRRD** [s]	0,521 3	0,495 2	0,574 6	0,991 0	0,806 2	0,909 3
	UpdateAlerts [s]	0,147 1	0,178 6	0,210 8	0,204 4	0,170 0	0,268 1
	run [s]	1,181 6	1,141 3	1,356 1	1,961 7	1,703 5	2,052 7
Studentský kolektor, 19. prosince 2011 – 21. prosince 2011, profil live_p3000							

Tabulka E.2: Plugin peaKock – Výkonnostní testování (průměry) (a)

Měřená veličina		Čtvrtek		Pátek		
		den	noc	den	noc	víkend
Toky	1 000 [s]	0,034 3	0,038 4	0,037 1	0,040 0	0,037 8
	10 000 [s]	0,343 1	0,383 8	0,370 5	0,399 6	0,377 9
	100 000 [s]	3,431 3	3,838 4	3,705 4	3,996 2	3,778 8
	1 s [počet]	20 097	17 219	19 094	16 304	20 216
	60 s [počet]	1 205 832	1 033 142	1 145 637	978 248	1 212 978
Bajty	1 000 [s]	$1{,}0\times10^{-6}$	$1{,}4\times10^{-6}$	$8{,}0\times10^{-7}$	$1{,}5\times10^{-6}$	$5{,}0\times10^{-7}$
	10 000 [s]	$1{,}0\times10^{-5}$	$1{,}4\times10^{-5}$	$8{,}0\times10^{-6}$	$1{,}5\times10^{-5}$	$5{,}0\times10^{-6}$
	100 000 [s]	0,000 1	0,000 1	$8{,}0\times10^{-5}$	0,000 2	$5{,}0\times10^{-5}$
	1 s [počet]	714 780 303	468 254 457	878 857 233	426 649 566	1 557 250 524
	60 s [počet]	42 886 818 152	28 095 267 407	52 731 434 003	25 598 973 952	93 435 031 417
Pakety	1 000 [s]	0,000 8	$4{,}5\times10^{-6}$	$9{,}4\times10^{-6}$	$8{,}4\times10^{-6}$	$7{,}3\times10^{-6}$
	10 000 [s]	0,007 9	$4{,}5\times10^{-5}$	$9{,}4\times10^{-5}$	$8{,}4\times10^{-5}$	$7{,}3\times10^{-5}$
	100 000 [s]	0,078 6	0,000 5	0,000 9	0,000 8	0,000 7
	1 s [počet]	877 535	145 631 203	75 456 313	77 471 064	104 605 775
	60 s [počet]	52 652 124	8 737 872 170	4 527 378 761	4 648 263 837	6 276 346 525
Funkce	**UpdateRRD** [s]	0,630 3	0,635 3	0,592 1	0,625 5	0,538 1
	UpdateAlerts [s]	0,132 7	0,145 6	0,132 9	0,193 2	0,223 6
	run [s]	1,450 1	1,513 0	1,413 4	1,534 8	1,309 0

Tabulka E.3: Plugin peaKock – Výkonnostní testování (průměry) (b)

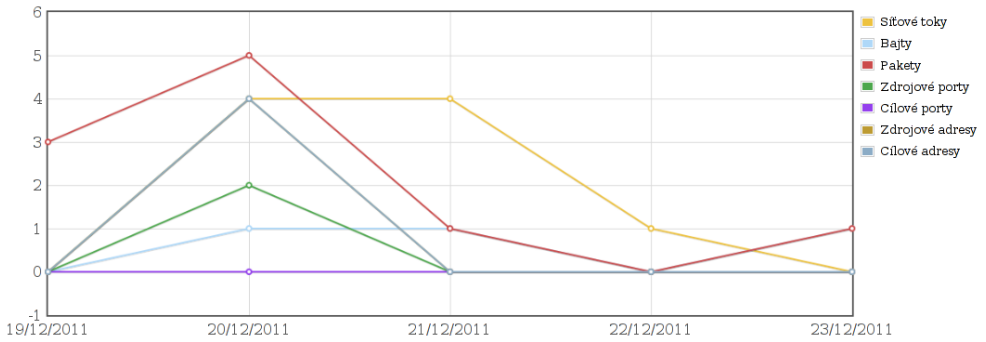
Časové období: den (6.00–17.00), noc (17.00–6.00), víkend (pá 17.00–po 6.00)

Měřená veličina		Sobota	Neděle
		víkend	
Toky	1 000 [s]	0,022 8	0,017 9
	10 000 [s]	0,227 9	0,179 0
	100 000 [s]	2,278 7	1,789 7
	1 s [počet]	61 875	92 070
	60 s [počet]	3 712 511	5 524 223
Bajty	1 000 [s]	$2{,}0\times10^{-7}$	$3{,}0\times10^{-7}$
	10 000 [s]	$2{,}0\times10^{-6}$	$3{,}0\times10^{-6}$
	100 000 [s]	$2{,}0\times10^{-5}$	$3{,}0\times10^{-5}$
	1 s [počet]	7 032 805 219	5 215 096 114
	60 s [počet]	421 968 313 143	312 905 766 851
Pakety	1 000 [s]	$2{,}11\times10^{-5}$	$1{,}3\times10^{-5}$
	10 000 [s]	0,000 2	0,000 1
	100 000 [s]	0,002 1	0,001 3
	1 s [počet]	66 845 170	127 115 825
	60 s [počet]	4 010 710 210	7 626 949 518
Funkce	**UpdateRRD** [s]	0,507 4	0,541 5
	UpdateAlerts [s]	0,146 5	0,137 6
	run [s]	0,709 2	0,606 9

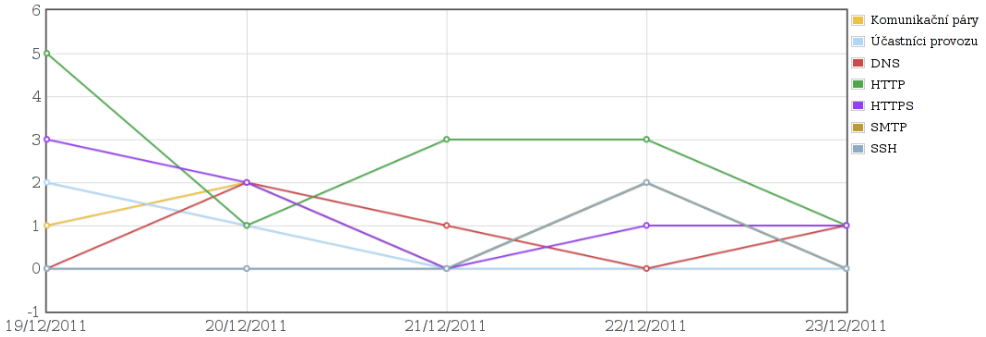
Tabulka E.4: Plugin peaKock – Výkonnostní testování (průměry) (c)

Časové období: den (6.00–17.00), noc (17.00–6.00), víkend (pá 17.00–po 6.00)

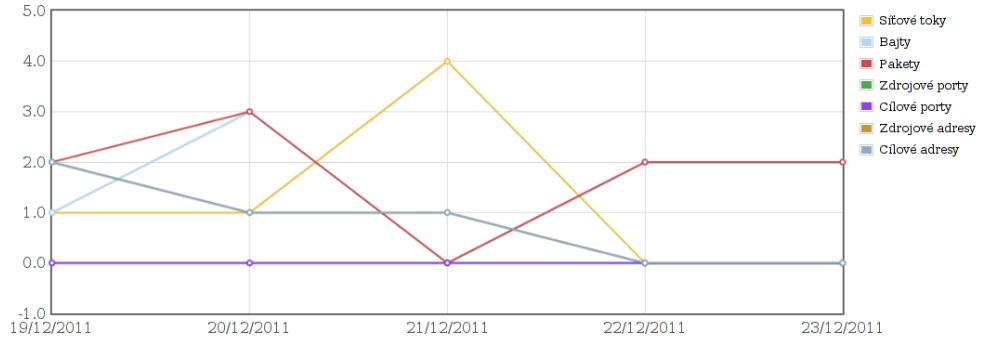
Následně jsou uvedeny grafy, které vizualizují data z uvedených tabulek pro lepší představu vzájemných souvislostí. Stejně jako data v tabulkách i grafy jsou rozděleny na období dne, noci a víkendu. Pro funkční testování zobrazují grafy počty anomálií na ose y a jednotlivé dny na ose x. Grafy výkonnostního testování zobrazují dobu běhu vybrané funkce na ose y a čas na ose x.



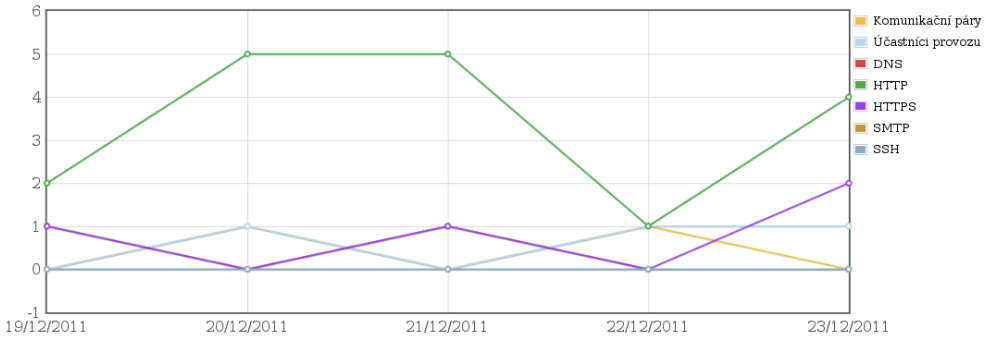
Obrázek E.1: Plugin peaKock – Funkční testování – den (a)



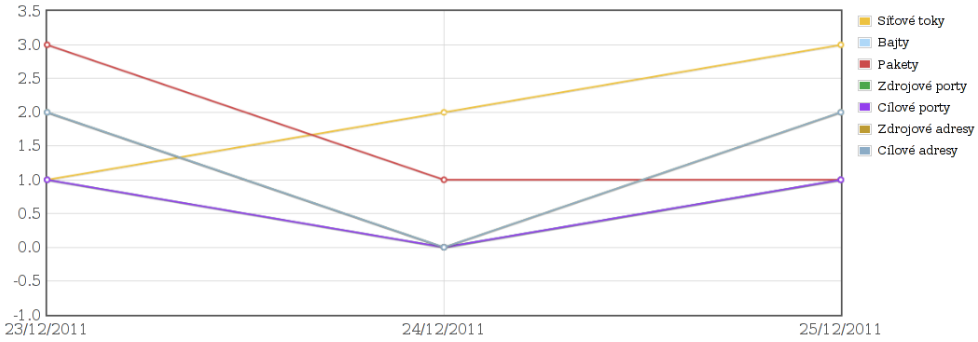
Obrázek E.2: Plugin peaKock – Funkční testování – den (b)



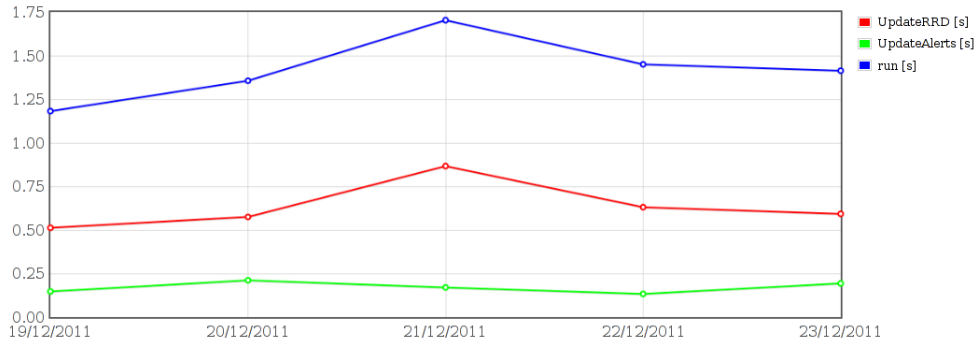
Obrázek E.3: Plugin peaKock – Funkční testování – noc (a)



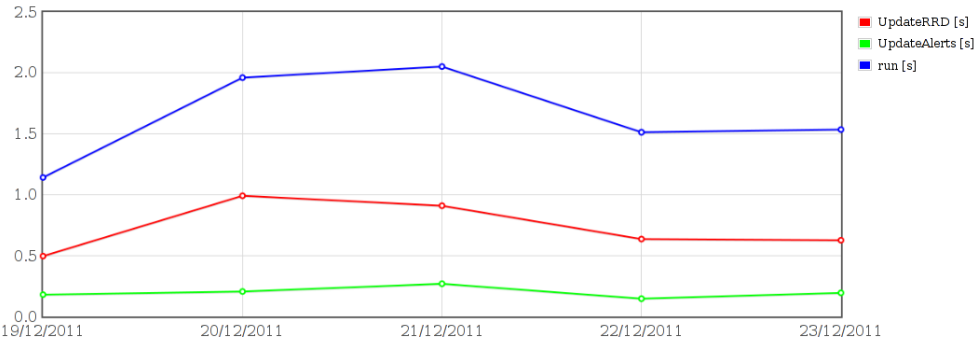
Obrázek E.4: Plugin peaKock – Funkční testování – noc (b)



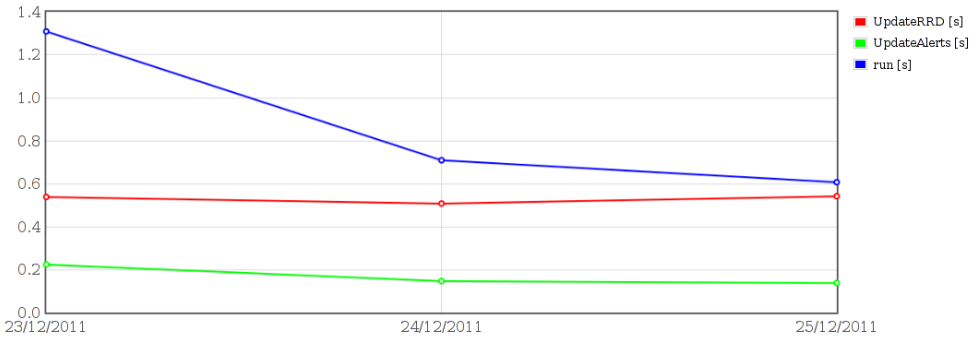
Obrázek E.5: Plugin peaKock – Funkční testování – víkend



Obrázek E.6: Plugin peaKock – Výkonnostní testování, funkce – den



Obrázek E.7: Plugin peaKock – Výkonnostní testování, funkce – noc



Obrázek E.8: Plugin peaKock – Výkonnostní testování, funkce – víkend

Příloha F

pChart 1.x

Autor: Jean-Damien POGOLOTTI

Knihovna pChart 1.x je v současnosti již zastaralou a je doporučováno přejít na verzi 2.x (informace můžete nalézt v kapitole 4.1.5 na straně 37). Z důvodu uvedených ve zmíněné kapitole uvádím pouze základní informace o této verzi. Konfigurace a graf mohou posloužit k získání přehledu o tom, jakým směrem knihovna pChart postoupila.

Stejně jako novější verze i pChart 1.x [66] je „server-side“ knihovnou v jazyce PHP pro vytváření grafů postavená na knihovnách GD a FreeType podporující vyhlazování. pChart 1.x se skládá z několika tříd, konkrétně se jedná o **pChart** zajišťující tvorbu grafů, třídu **pData** pro organizaci vstupních dat a třídu **pCache** umožňující znovupoužití již vygenerovaných grafů a šetření zdrojů. Použití tříd pChart a pData je povinné, a je třeba zahrnout jejich kód do kódu webové stránky.

Data je třeba připravit pro použití v grafu. O to se postará třída pData. Jejich zdrojem může být SQL, soubor **CSV** či ruční nadefinování. Asi nejpodstatnějšími funkcemi této třídy jsou `addPoint` umožnující přidání jednoho nebo více bodů do datové řady a `addSerie` zajišťující přiřazení datové řady ke grafu. Dále třída pData obsahuje funkce pro zvýšení přehlednosti grafu, jako např. nastavení barvy, umístění, jména a pozice souřadnicových os.

Zdrojový kód F.1: Ukázka konfigurace pChart 1.x

```
$FlowDataSet = new pData;
for ($i=0; $i<count($flow); $i+=2 ) {
        $flows_osa_x[] = $flow[$i];
        $flows_osa_y[] = $flow[$i+1];
}
$FlowDataSet->AddPoint($flows_osa_y,"Serie1");
$FlowDataSet->AddPoint($flows_osa_x,"Serie2");
$FlowDataSet->AddSerie("Serie1");
$FlowDataSet->SetSerieName("Flows","Serie1");
$FlowDataSet->SetXAxisName("Time");
$FlowDataSet->SetYAxisName("Flows");
$FlowDataSet->SetAbsciseLabelSerie("Serie2");
$FlowDataSet->SetXAxisFormat("date");
$Flows = new pChart(850,430);
$Flows->setDateFormat("d/m/Y␣H:i");
$Flows->setColorPalette(0,255,0,0);
$Flows->setFontProperties \
  ("plugins/peakock/Fonts/pf_arma_five.ttf",9);
$Flows->setGraphArea(130,50,810,330);
$Flows->drawFilledRoundedRectangle \
  (7,7,843,423,5,240,240,240);
$Flows->drawRoundedRectangle \
  (5,5,845,425,5,0,0,0);
$Flows->drawGraphArea(252,252,252);
$Flows->drawScale($FlowDataSet->GetData(),
  $FlowDataSet->GetDataDescription(),
  SCALE_NORMAL,130,130,130,TRUE,35,2,TRUE,25);
$Flows->drawGrid(4,FALSE,230,230,230,255);
$Flows->drawFilledLineGraph( \
  $FlowDataSet->GetData(), \
  $FlowDataSet->GetDataDescription());
$Flows->setFontProperties \
  ("plugins/peakock/Fonts/tahoma.ttf",8);
$Flows->drawLegend \
  (135,55, \
  $FlowDataSet->GetDataDescription(), \
  255,255,255);
$Flows->setFontProperties \
  ("plugins/peakock/Fonts/tahoma.ttf",15);
  $Flows->drawTitle(280,35, "pChart", \
  50,50,50,585);
$Flows->Render("plugins/peakock/flows.png");
```

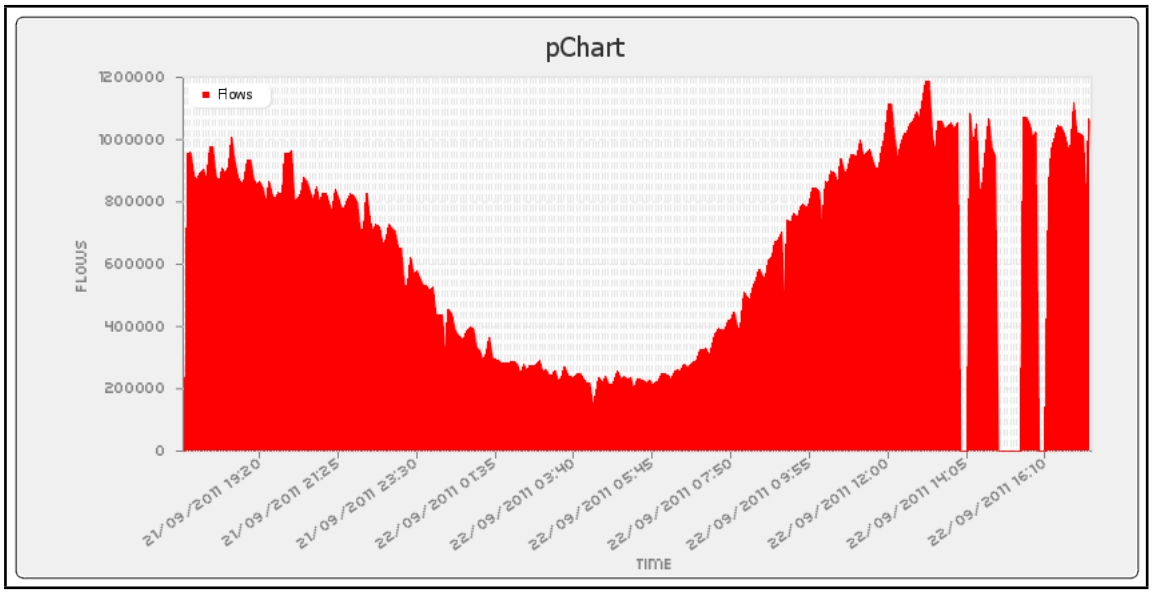
Než budeme schopni kreslit graf, je třeba ještě nadefinovat oblast pro jeho vykreslení pomocí funkce `setGraphArea` a nastavit měřítko funkcí `drawScale`. Další možnosti nastavení jsou nepo-

vinné, ale dokáží zlepšit čitelnost grafu. Do této skupiny zařadíme funkce pro vykreslení legendy, titulku grafu, stínu či rámečku.

V oblasti vykreslování poskytuje pChart 1.x kromě grafů i některé jednoduché grafické prvky. Lze tedy vytvořit čáru, Beziérovu křivku, kružnici, kruh, čtyřúhelník, vyplněný čtyřúhelník a mnohoúhelník. Jestliže se zaměříme na možnosti vizualizace grafů, zjistíme, že pChart 1.x umí vytvořit bodový a spojnicový graf (scatter, line), graf ohraničený křivkou (cubic curve), dále pak plošný graf (area) a v neposlední řadě také sloupcový, radar a koláčový graf (bar, radar, pie). Posledně jmenovaný zvládá i ve variantě pro 3D prostor.

Samotné vytvoření grafu zajišťuje jedna z funkcí `render()` či `stroke()`. Prvně jmenovaná vytvoří soubor s obrázkem v definovaném formátu a uloží jej do složky na serveru. Druhá pošle obrázek ve formě MIME přímo prohlížeči.

Pro ladění chyb poskytuje pChart 1.x funkci `ReportWarnings()`, jejíž parametrem je buďto `GD` anebo `CLI` a která umožňuje vypisovat chyby na konzoli či přímo do grafu. Výpis na konzoli může být příhodný pokud pChart spouštíme neinteraktivně např. z `cronu` [77] v Unixu či pomocí `at` [56] z Windows.



Obrázek F.1: Ukázka grafu v pChart 1.x