VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

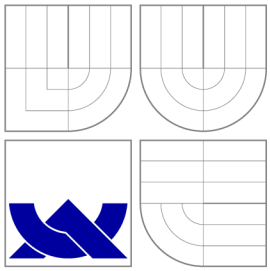
INFORMAČNÍ SYSTÉM PRO SPRÁVU FAIR USER POLICY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

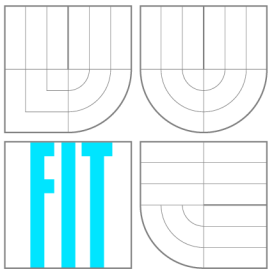
AUTOR PRÁCE
AUTHOR

JAKUB HORČIČKA

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INFORMAČNÍ SYSTÉM PRO SPRÁVU FAIR USER POLICY

INFORMATION SYSTEM FOR FAIR USER POLICY MANAGEMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAKUB HORČIČKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ TOBOLA
SUPERVISOR

BRNO 2010

Abstrakt

Tato práce uvádí metody pro monitorování síťové aktivity se zaměřením na technologii NetFlow firmy Cisco Systems. V následujících kapitolách je popsána implementace informačního systému, který tuto technologii využívá zejména pro tvorbu statistik o jednotlivých uživatelích a umožňuje tak kontrolovat dodržování pravidel a limitů stanovených v rámci nabízených tarifů.

Abstract

This bachelor thesis presents methods for monitoring network activity with an aim to Cisco Systems NetFlow technology. Following chapters describe implementation of an information system that uses this technology especially for making the stats about particular users and allows checking of rules and data limits determined in offered tariffs.

Klíčová slova

informační systém, NetFlow, Fair User Policy, FUP, Python, Django, nfdump, monitorování sítí

Keywords

information system, NetFlow, Fair User Policy, FUP, Python, Django, nfdump, network monitoring

Citace

Jakub Horčička: Informační systém pro správu Fair User Policy, bakalářská práce, Brno, FIT VUT v Brně, 2010

Informační systém pro správu Fair User Policy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jiřího Toboly. Uvedl jsem veškeré literární prameny a zdroje, ze kterých jsem čerpal.

.....
Jakub Horčička
14. května 2010

Poděkování

Zde bych chtěl poděkovat vedoucímu Ing. Jiřímu Tobolovi za užitečné připomínky, rady a odbornou pomoc, kterou mi během práce poskytoval.

Obsah

1 Úvod **3**

2 Monitorování sítí **4**
2.1 Co je monitorování sítí . . . 4
2.2 Princip . . . 4
2.3 Technologická relizace . . . 5
2.4 Protokoly pro monitorování sítí . . . 7
2.4.1 SNMP . . . 7
2.4.2 RMON . . . 7
2.4.3 NetFlow . . . 8
2.4.4 IPFIX . . . 8
2.4.5 sFlow . . . 9
2.5 Výstup sledovacího systému . . . 9
2.6 Odhalování bezpečnostních rizik . . . 10
2.6.1 Druhy útoků . . . 10
2.6.2 IDS . . . 11
2.6.3 IPS . . . 11
2.7 FUP . . . 12

3 NetFlow **14**
3.1 Co je NetFlow . . . 14
3.2 Využití . . . 14
3.3 Architektura . . . 15
3.3.1 Exportér . . . 15
3.3.2 Kolektor . . . 15
3.3.3 Prezentér . . . 15
3.4 IP tok . . . 16
3.5 NetFlow cache . . . 17
3.6 Časovače . . . 18
3.7 Vzorkování . . . 18
3.8 Verze protokolu NetFlow . . . 19
3.9 Dostupné nástroje . . . 20

4 Analýza a návrh systému **21**
4.1 Specifikace požadavků . . . 21
4.2 Návrh řešení . . . 21
4.3 Databáze . . . 24

5 Implementace **25**
5.1 Použité technologie . . . 25
5.1.1 Django framework . . . 25
5.1.2 ReportLab Toolkit . . . 27
5.1.3 jQuery . . . 27
5.2 Zdrojové soubory projektu . . . 27
5.3 Periodické kontroly . . . 29
5.4 Vzhled aplikace . . . 31

6 Závěr **32**
6.1 Dosažené výsledky . . . 32
6.2 Další rozšíření . . . 32

A Popis instalace **35**
A.1 Předpoklady . . . 35
A.2 Nfdump . . . 35
A.3 Framework Django . . . 36
A.4 ReportLab . . . 36
A.5 Zprovoznění APACHE serveru . . . 36
A.6 Instalace webové aplikace . . . 37
A.7 Přihlášení do systému . . . 37
A.7.1 Přístup superuživatele . . . 37
A.8 Spuštění kolektoru . . . 37
A.9 Spuštění periodických skriptů . . . 38

Kapitola 1

Úvod

Propojování počítačových stanic do sítí přidává další rozměr ve využití moderní techniky. Lze tak snadno sdílet hardwarové prostředky nebo komunikovat a přenášet data i na velké vzdálenosti. Tyto a mnohé další výhody se využívají již od druhé poloviny 20. století, kdy se ve Spojených státech amerických realizoval armádní projekt, jenž položil základy dnešnímu Internetu.

Od té doby se však mnoho změnilo. Počítačovou síť provozuje většina společností, firem, ústavů, ale i domácností. Vznikla potřeba kontrolovat a zaznamenávat kdo s kým a jak dlouho komunikuje nebo kolik dat se přenáší. Odpovědi na tyto a jiné otázky přináší technologie pro monitorování síťové aktivity.

Cílem této práce je analyzovat, navrhnout, implementovat a otestovat informační systém, který by sloužil poskytovateli internetového připojení pro správu *Fair User Policy (FUP)*. Jde o politiku, která zabraňuje jednotlivým uživatelům omezovat ostatní nadměrným využíváním komunikačních linek. Jelikož je však v případě připojení přes pevnou linku na ústupu, zaměřuje se tento systém spíše na sledování síťové aktivity uživatelů a prezentaci statistik o evidované komunikaci. K těmto účelům byla použita technologie *NetFlow*, která je založena na datových tocích.

Hlavní obsah práce je rozdělen do pěti částí. Ve druhé kapitole jsou představeny metody, nástroje a prostředky, jež sledování sítí umožňují. Třetí kapitola se snaží čtenáři detailněji přiblížit technologii NetFlow. Čtvrtá a pátá kapitola popisují rozbor, návrh řešení a implementaci informačního systému. V závěru jsou shrnuty dosažené cíle a navrhnuty možnosti vylepšení či rozšíření systému.

Kapitola 2

Monitorování sítí

2.1 Co je monitorování sítí

Český termín *monitorování* vznikl z anglického slova *monitor*, popř. *monitoring*, které lze přeložit jako *sledovat*, popř. *sledování*. *Sledování sítě* je pak souhrn procedur (zasílání dotazů na jednotlivá zařízení, ověřování správné funkčnosti, ukládání přenášených dat, analýza a provádění statistik) vedoucí ke zjištění stavu síťové aktivity. Tedy kdo s kým, kdy a jak často komunikuje, kolik přenáší paketů, zda data spíše stahuje nebo vysílá a jaké aplikace či služby využívá [21].

Tyto informace jsou klíčové nejen pro majitele či správce sítí, ale také pro poskytovale internetového připojení nebo vedení firem. Je možné získat představu o využití jednotlivých zdrojů, předvídat potřebu rozšíření nebo naopak vyřadit prvky, které se nepoužívají. Získaná data napomáhají k odhalování anomálií, rozpoznávání poruch a výpadků nebo k označování zastaralých či poškozených prvků určených k vyřazení. Pomocí různých algoritmů a metod je možné zachytávat viry a odhalovat škodlivý kód, který by jinak mohl vyřadit chod sítě jejím zahlcením. Nespolehlivá, nefunkční nebo zavirovaná síť může způsobit nejen v oranizaci či společnosti nemalé finanční ztráty. Díky technologiím určeným k monitorování je možné podobným problémům čelit, případně se jim zcela vyhnout.

V neposlední řadě je možné monitorování sítě využívat pro zajištění kvality poskytovaných služeb, správu uživatelů s jejich tarify nebo jednoduše hlídat, zda zaměstnanci v pracovní době netráví většinu času na Internetu prohlížením stránek nebo komunikací s přáteli. Je možné odhalit uživatele, kteří nedodržují smluvní předpisy. Jako běžný příklad bych uvedl překročení limitu pro velikost stažených nebo odeslaných dat. Viníkovi je pak snížena rychlost připojení, zvýšena cena za přenesený paket nebo dokonce odpojena přípojka.

2.2 Princip

Základní princip získávání informací o síťové komunikaci spočívá v neustálém ukládání vzorků z procházejících paketů. V ideálním případě by se ukládala veškerá probíhající komunikace na všech síťových zařízeních. To ovšem z kapacitních a výkonových omezení není možné. Následují dotazy směrované do databáze a prezentace výsledků pomocí uživatelského rozhraní. K tomu slouží nejrůznější softwarové i hardwarové aplikace realizující jednotlivé podúlohy. Podle konkrétní situace je možné větši nebo menší podmnožinu těchto podúloh plně nebo částečně zautomatizovat a administrátory pouze informovat o nečeka-

ných událostech pomocí e-mailových nebo SMS zpráv.

Získané vzorky dat se ukládají do centrální databáze, kde jsou spravovány. Ve většině praktických aplikací ve skutečnosti nejde o jednu jedinou databázi. Často existuje mnoho lokálních úložišť, které ale v konečném důsledku svá data různým způsobem s centrální databází synchronizují. V některých případech nemusí hlavní úložiště fyzicky existovat. Požadovaná data se seskupí až v případě potřeby prostřednictvím aplikací realizujících dotazy na jednotlivá úložiště. Tento způsob snižuje kapacitní požadavky na úložná média, častou nevýhodou je pak pomalejší odezva. Záleží na požadavcích zákazníka, zda bude upřednostněna rychlost zpracování dotazů a předání výsledků na úkor větší vyžadované kapacity nebo časově náročnější vyhodnocení bez potřeby centrální databáze. Pro oba případy je nezbytné pečlivě zvolit jednotlivé prvky architektury. Existuje celá řada speciálních hardwarových karet nebo aktivních síťových prvků, které procházející data vzorkují, ukládají a výsledky odesílají do hlavní databáze.

Z pohledu běžného uživatele jsou pro potřeby monitorvní důležitá pouze metadata. V drtivé většině případů se neukládá textový obsah přenášených zpráv, zdrojový kód webových stránek nebo binární data přenášených souborů. Toto všechno není pro sledování síťové aktivity důležité. Každý paket obsahuje tak zvanou hlavičku, ve které jsou uloženy údaje potřebné pro doručení obsahu na správné místo. Mimo jiné se zde můžeme dozvědět odkud byla data odeslána, kam směřují a jak velký je jejich obsah. To jsou údaje, ze kterých je možné ve spolupráci s časovými razítky vyčíst vše podstatné pro správu a sledování síťě.

2.3 Technologická relizace

Možností pro monitorování síťové infrastruktry je celá řada. K dispozici jsou softwarové i hardwarové technologie a aplikace. Programovou část je možné obstarat i zdarma díky open source nástrojům. Zde je však často omezujícím faktorem potřeba unixového operačního systému, nad kterým tyto programy pracují. Nezbytnou součástí každé monitorovací architektury je využití celé řady síťových protokolů, které standardizují způsob komunikace jednotlivých oblastí nebo prvků.

V [2] se uvádí, že pokud disponujeme dostatečnými znalostmi a dovednostmi, zejména v oblasti skriptování, je možné realizovat sledovací systém svépomocí. Využije se některý z dostupných nástrojů a propojí se pomocí vlastních skriptů. V opačném případě jsou k dispozici kvalitní komerční nástroje, které ušetří čas a námahu spojenou s nastavováním. Výhodou vlastního řešení je mimo finanční stránky často lepší přehled o fungování celého systému a možnost přizpůsobení pro konkrétní potřeby.

V operačních systémech typu unix nebo windows je možné pro získání základních informací o dostupnosti jednotlivých prvků využít některé z následujících nástrojů. Jejich bližší popis a způsob použití je možné nalézt v manuálových stránkách nebo v [1].

- *ping (Packet Internet Groper)* – program využívající protokolu *ICMP (Internet Control Message Protocol)*. Na zvolený cílový prvek odešle sadu paketů *ECHO_REQUEST* a očekává odpověďi *ECHO_RESPONSE*. Do každého paketu vloží jedinečné identifikační číslo, díky kterému dokáže rozlišit jednotlivé odpovědi. Pomocí tohoto mechanismu lze ověřit dostupnost, ztrátovost paketů a rychlost odezvy cílového prvku.

- *traceroute (unix), tracert (windows)* – tento program využívá obdobného principu jako program ping, ale jeho výsledkem je výpis všech síťových uzlů, přes které zaslané pakety prošly. Využívá položky *TTL (Time To Live)* v hlavičce protokolu. Jde o hodnotu, která se při každém zpracování paketu sníží. V případě že se dostane na nulovou

hodnotu, je paket zahozen a odesilateli je vrácen paket s upozorněním *TIME_EXCEEDED* [1]. Program tedy nejprve odešle paket s hodnotou TTL rovnou jedné. Hned na prvním směrovači (nebo jiném prvku) je tato položka vynulována a směrovač vrací upozornění. Program uloží jeho *IP (Internet Protocol)* adresu a pokračuje s paketem, kde je TTL o jedničku větší. Takto se pokračuje dále, až je vráceno upozornění z cílového prvku.

- *netstat* – nástroj pro zobrazení statistik protokolů a aktivních *TCP/IP (Transmission Control Protocol / Internet Protocol)* spojení.
- *ifconfig (unix), ipconfig (windows)* – nástroj pro zobrazení konfigurace jednotlivých rozhraní TCP/IP systému.
- *finger* – unixová utilita umožňující zobrazit seznam přihlášených uživatelů na daném počítači a základní informace, které o sobě sami uživatelé uvedou do speciálního souboru. Kvůli bezpečnosti bývá tato utilita na mnoha síťích zakázána.
- *host* – program provádějící dotazy na *DNS (Domain Name Server).* Převádí zadanou IP adresu na doménové jméno či naopak.
- *nslookup* – jde o rozšíření programu host. Pro zadanou IP adresu vypíše všechna odpovídající doménová jména. Využívá se pro ověření správné funkčnosti DNS serveru.
- *tcpdump* – utilita vypisující základní informace o *TCP (Transmission Control Protocol)* paketech zachycených na zvoleném síťovém rozhraní. Podporuje filtrování podle jednotlivých položek.
- *wireshark, ethereal* – nadstavba programu tcpdump. Neomezuje se pouze na TCP pakety a podporuje grafické uživatelské rozhraní.
- *nmap* – pomocí tohoto programu je možné provádět tzv. skenování portů. Pro zadanou IP adresu se vypíše seznam všech používaných portů, jejich stav (otevřený, zavřený) a služba, která na nich běží.

Chceme-li komplexní monitorovací systém, je k dispozici několik alternativ. Ať už komerčních či volně dostupných.

- *Nagios* [15] slouží k monitorování síťové infrastruktury. Primární cílovou architekturou je Linux, ale lze ho využít i v ostatních operačních systémech unixového typu. Lze jej využít pro plánování změn nebo získání celkového pohledu na průběh sledovaných vlastností.
- *Zabbix* [23] je open–source software monitorující síťové parametry. Umožňuje nastavit upozornění ve formě e-mailu na nejrůznější události, automaticky detekuje servery a síťové prvky, obsahuje webové rozhraní pro celkovou analýzu a poskytuje komplexní systém uživatelských práv.
- *Zenoss* [24] je komerční open–source řešení pro správu síťové infrastruktury. Základem je detailní analýza všech síťových prvků. Data jsou shromážděna v hlavní databázi a pomocí webové konzole je možné kdykoli spravovat jejich stav.

[1] Jde o příznak vypršení času na doručení paketu

- *HP OpenView* [12] je sada komerčních aplikací pro správu sítě, diagnostiku bezpečnostních rizik a řadu dalších služeb od firmy *Hewlett Packard*.

2.4 Protokoly pro monitorování sítí

Pro účely sledování síťových aktivit existuje podpora ve formě různých *protokolů*. V následujících podkapitolách jsou přiblíženy nejpoužívanější z nich.

2.4.1 SNMP

SNMP (Simple Network Management Protocol) [1, str. 860] je standardní protokol podporující širokou škálu síťových zařízení. Cílem jeho vzniku v roce 1988 bylo sjednotit všechny existující protokoly využívané k podobnému účelu a vytvořit univerzální prostředek pro sledování a správu sítě. Během své dlouholeté existence se stal velice rozšířeným, používaným a oblíbeným.

Hlavní součástí je tzv. *MIB (Management Infromation Base)*. Jde o databázi definující síťové objekty v deseti skupinách: systém, rozhraní, překlad adres, IP, ICMP, TCP, UDP, EGP (Exterior Gateway Protocol), přenos dat a protokol SNMP. Jednotliví výrobci však implementují četná rozšíření, která narušují význam standardizace. Pro popis objektů se využívá dvou specifikací. První z nich je *ASN 1 (Abstrack Syntax Notation One)*. Jde o textový popis zahrnující základní prvky, jako je celé číslo, identifikátor objektu, nula a síťová adresa. Jednotlivé objekty jsou identifikovatelné pomocí *OID (Object Identificator)* a dělí se na singulární (jedna hodnota) a sloupcové. Sloupcové objekty představují seskupení více singulárních objektů ve formě tabulky nebo řady. Druhou je pak *BER (Basic Encoding Rules)*, která popisuje pravidla šífrování při převodu hodnot MIB objektu do binárního formátu vhodného pro přenos po síti.

Architektura je charakteristická dvěma hlavními prvky. *Správce – stanice* a *agent*. Stanice se využívá k prohlížení a analýze dat. Jedná se často o vyhrazený počítač, ke kterému mají přístup správci sítě. Agentem je počítačový program běžící na spravovaném zařízení, který zpracovává procházející data a ukládá je do své MIB databáze. Analýza dat je založena na dotazech stanice směřujících ke konkrétnímu zařízení. Agent požadavek zpracuje, sestaví požadovaná data do paketů a odešle je zpět do stanice. Pro přenos se využívá protokolů IP a UDP (port 161).

2.4.2 RMON

RMON (Remote Network Monitoring) [1, str. 865] vznikl v roce 1991 jako rozšíření standardní specifikace SNMP. Nad rámec SNMP poskytuje sadu objektů souvisejících s analýzou a sledováním sítě. Mezi základní skupiny patří:

- *statistics* (statistiky)
- *history* (historie)
- *alarms* (upozornění)
- *hosts* (hostitelé – MAC adresy zařízení)
- *hostTopN* (statistiky týkající se hostitelů v síti)
- *matrix* (statistiky o přenosu dat)

- *filter* (nastavení filtrace paketů)
- *capture* (zachytávání paketů podle nastavených filtrů)
- *event* (generováí událostí)

Dalším rozdílem jsou typy poskytovaných dat, která jsou více zaměřena na provoz v síti. Jedná se zejména o různé statistiky, historii a seznam událostí. Umožňuje zachytávat a filtrovat provoz na konkrétních zařízeních. Největší odlišností je však obrácení architektury klient – server. U klasického SNMP je serverem jedna stanice a agenti jsou na každém zařízení. To může způsobovat problémy zatížení při častých dotazech. V případě RMON je serverem každý agent, který nezávisle provádí veškerá zpracování a pravidelně odesílá pakety se statistikami do stanice. Tou je tedy jeden počítač, klient, který pravidelně přijímá aktualizace ze všech zařízení v síti a nemusí tak při každém požadavku uživatele zahltit síť jednotlivými dotazy.

2.4.3 NetFlow

NetFlow je protokol firmy *Cisco* určený k přenosu dat o síťových tocích. Základem architektury jsou dva prvky: *sonda* a *kolektor*. Sonda představuje prvek, který zpracovává data procházející sítí a vytváří statistiky o tocích, které odesílá kolektoru. Kolektor tedy data přijímá, shromažďuje a připravuje pro následnou prezentaci nebo analýzu. Detailněji je tento protokol popsán v samostatné části 3.1 na straně 14.

2.4.4 IPFIX

IPFIX (Internet Protocol Flow Information Export) [8], [17] vznikl jako snaha o vytvoření univerzálního standardu pro přenos datových toků. Základem se stal protokol *NetFlow verze 9* firmy *Cisco*. Princip přenosu je rozdělen do tří logických částí:

1. *Metering Process* generuje záznamy o tocích. Je spuštěn na sledovaném zařízení a zpracovává hlavičky procházejících paketů.
2. *Exporting Process* odesílá vytvořené záznamy ve formě paketů jednomu nebo více shromažďujicím procesům.
3. *Collecting Process* je shromaždujicí proces, který přijímá odesílané pakety a dále je zpracovává nebo ukládá do databáze.

Každý paket obsahuje hlavičku, za kterou následují *data* (kolekce záznamů) nebo *šablony* (popis struktury, interpreteace a způsobu využití jednotlivých datových položek). Formát hlavičky je uveden v tabulce 2.4.4. Význam jednotlivých položek je následující:

- *version* – číslo verze přenášeného záznamu, identifikuje jeho formát.
- *length* – celková délka paketu v *oktetech* (oktet = 8 bitů).
- *export time* – počet sekund od 1. 1. 1970.
- *sequence number* – pořadové číslo paketu v rozmezí 0–2^{32}.
- *observation domain ID* – unikátní identifikátor sledované domény. Shromažďovací proces tak může rozlišit jednotlivé exportující procesy.

version	length
export time	
sequence number	
observation domain ID	

Tabulka 2.1: Formát hlavičky v IPFIX paketu

2.4.5 sFlow

Technologie *sFlow (Samples Flow)* [16] je určena pro monitorování vysokorychlostních sítí (Gbit/s), které obsahují *switche* nebo *routery*. Charakteristickým rysem je tzv. *sampling (vzorkování)*, kdy se neanalyzují všechny procházející pakety, ale pouze některé, určitým způsobem zvolené. Architektura je složena ze dvou hlavních prvků:

1. *sFlow agent* – jde o zařízení vestavěné v síťovém prvku (switch nebo router), které zpracovává procházející pakety a získaná data odesílá kolektoru.
2. *sFlow kolektor* – centrální databáze pro data odeslaná agenty.

Pro vzorkování se používají dva mechanismy:

1. *Paketově orientované* vzorkování využívá cyklického čítače, který se při každém nezahozeném příchozím paketu dekrementuje. Pokud dosáhne nulové hodnoty, zpracuje aktuální paket a nastaví se do počáteční hodnoty.
2. *Časově orientované* vzorkování se používá pro celkové statistiky jednotlivých rozhraní. Každý agent má nastavenu periodu, se kterou odesílá statistická data o svém rozhraní na kolektor.

2.5 Výstup sledovacího systému

Ačkoliv je možné monitorování plně automatizovat, vždy existuje cílový uživatel (administrátor – člověk), který bude výsledky sledovat či analyzovat. Je tedy nezbytné, aby byl výstup přehledně a uživatelsky přívětivě prezentován. I jinak bezchybný sledovací systém, který nezvládne úlohu prezentace dat, bude pravděpodobně neoblíbený a tudíž se neuplatní.

Z hlediska prezentace je možné dle [2] rozlišovat dva typy dat. Prvním jsou *události*. Jedná se o situace, kdy dojde k náhlé změně stavu sledované vlastnosti. Příkladem může být porucha síťového prvku či překročení limitu odeslaných nebo přijatých dat. V tomto případě je podstatné zaznamenat kromě popisu a stavu také čas vzniku události. Prezentace je možná formou tabulky. Je důležité, aby bylo možné vyhledání konkrétní položky (případně dohledání v budoucnu). V závažných případech (např. výpadky) je typické automatické upozorňování formou SMS nebo e-mailu. Druhým typem jsou *aktuální hodnoty* sledovaných vlastností. Zde není obvykle důležité uchovávat a prezentovat každou naměřenou hodnotu. Důležitý je však trend, jakým se hodnoty mění. Vhodným způsobem prezentace je tedy grafické zobrazení. Příkladem může být vytížení sítě, konkrétního procesoru (prvku), ale i procentuální rozložení zátěže mezi jednotlivé uživatele.

2.6 Odhalování bezpečnostních rizik

Velice užitečnou vlastností sledovacích systémů je možnost odhalení a prevence bezpečnostních incidentů na síti. K těm může dojít dříve nebo později v každé větší firemní síti. Existují algoritmy, které rozpoznávají škodlivý kód v obsahu přenášených paketů, ale z hlediska klasického sledovacího systému je možné identifikovat *skenování portů*, *útok typu DoS (Denial of Service)* a *šíření „červů“ (worms)*.

2.6.1 Druhy útoků

Skenování portů

Jde o techniku, kterou je možné identifikovat běžící služby na síti. Útočník odesílá pakety pro navázání spojení na různé porty. V případě, že k navázání spojení dojde, jedná se o aktivní službu. V rámci této techniky je možné rozlišovat několik způsobů provedení, které jsou popsány v [19, str. 32]. Pro ukázku uvedu následující čtyři.

1. TCP connection: V tomto případě dochází k úplnému (třícestnému – SYN, SYN/ACK, ACK) navázání spojení. Na cílovém počítači je toto spojení jednoduše identifikovatelné.

2. TCP SYN scan: Jedná se o poloviční navázání spojení. Útočník vyšle paket s příznakem SYN a čeká pouze na odpověď. Pokud příjde SYN/ACK je pravděpodobné, že je port otevřený a služba dostupná. Když příjde RST, tak port otevřený není. Rozdíl oproti TCP spojení je v tom, že po příjmutí tohoto paketu útočník spojení sám uzavírá a nedochází k odeslání ACK paketu, kterým by se spojení dokončilo. Výhodou je zde fakt, že se toto spojení na cílovém počítači nezaznamenává. Nejde totiž o úspěšně navázané spojení.

3. TCP FIN scan: Na cílový port se odešle paket s příznakem FIN. V RFC 793 je definováno, že systém na FIN pakety odpoví paketem RST pro všechny uzavřené porty. Prakticky je ovšem možné tuto techniku použít pouze pro unixové systémy.

4. UDP scan: Na rozdíl od všech výše uvedených je v tomto případě odeslán paket protokolu *UDP (User Datagram Protocol)*. Jestliže je cílový port uzavřen, odešle systém odpověď ve formě *ICMP PORT UNREACHABLE* zprávy. Pokud však tuto zprávu neobdržíme, nemůžeme předpokládat, že je port dostupný. Důvod vyplývá z principu UDP komunikace, kde není snaha o zajištění spolehlivého doručení paketů. Proto si nemůžeme být jisti, zda námi odeslaný paket dorazil k místu určení nebo došlo ke ztrátě *ICMP* odpovědi (*ICMP* protokol využívá *UDP*).

Technik pro skenování portů existuje mnohem více, ale výše uvedené patří mezi nejběžnější a v případě *TCP connection* a *TCP SYN scan* také o nejspolehlivější. Ve statistikách je skenování portů typicky rozeznatelné podle velkého množství jednopaketových toků na různe porty. Často (ne však vždy) mají všechny tyto toky stejnou zdrojovou IP adresu.

Útok typu DoS

Útoky typu *DoS (Denial of Service, Odepření služby)* [19, str. 443] jsou vážným problémem potenciálně všech počítačových sítí. Cílem je zablokování poskytovaných služeb pro běžné

uživatele. Princip spočívá v nadměrném počtu požadavků, čímž dojde k přetížení serveru a zablokování služby.

Četnost těchto útoků vyplývá z relativní snadnosti provedení a díky možnému podvržení zdrojové IP adresy je téměř nemožné vystopovat skutečnou identitu útočníka.

Ve statistikách je tento útok charakteristický nadměrným množstvím toků se stejným cílovým portem (požadavky na stejnou službu). Často se jedná o pakety využívající protokolu *TCP* s příznakem *SYN*.

Worms

Pojem *worm, červ* [10, 1, str. 604] představuje v oblastni softwaru samovolně se šířící kód, který provádí na cílových stanicích škodlivou činnost. Jeho objevení není obecně snadné. Přibližným rysem, který lze identifikovat ve statistikách, je nápadně velké množství krátkých toků ze stejné IP adresy. Nemusí jít konkrétně o šířící se červ, nicméně uvedená situace naznačuje neobvyklý síťový provoz značící potenciální problém.

2.6.2 IDS

IDS (Intrusion Detection System) [9] je množina nástrojů a metod, které pomáhají odhalit nepovolenou či nebezpečnou síťovou aktivitu na *síťové vrstvě referenčního modelu OSI*. Často se jedná o součást většího bezpečnostního systému. Navzdory názvu nedetekuje IDS pouze skutečná vniknutí či hrozby. Princip spočívá v analýze procházejících dat, kdy se v případě překročení nastavených limitů provede příslušná reakce (např. odeslání upozornění správci). Mnohdy tak dochází k planým poplachům nebo se naopak některý útok neodhalí. Nastavení limitů a parametrů je tedy pro správnou činnost klíčové. V rámci IDS lze rozlišovat *HIDS (Host–based IDS)* , *NIDS (Network–based IDS)* a jejich kombinace.

- HIDS je detekční systém pracující na konkrétní počítačové stanici (host). Nemá přístup k celkové síťové infrastruktuře a neanalyzuje hlavičky jednotlivých paketů. Místo toho sleduje lokální aktivity, prochází logovací výpisy (např. syslog) a porovnává nalezené události se svojí databází. V ní jsou uloženy typické rysy různých bezpečnostních rizik a pokud dojde ke shodě, odešle se upozornění o potenciálním nebezpečí. Hodí se tedy spíše na odhalování útoků zevnitř sítě a je silně závislý na použitém operačním systému.

- Naopak NIDS je detekční systém sledující celou síť jako celek. Prochází hlavičky paketů a pracuje v reálném čase. Využívá se spíše k odhalení útoků zvenčí a je nezávislý na použitém operačním systému. Z finančního hlediska jde o méně nákladnou záležitost než v případě HIDS. Příkladem NIDS systému může být *Snort*. Jde o volně dostupný software využívající knihovnu *libpcap* pro nízkoúrovňovou práci s pakety a síťovými rozhraními.

2.6.3 IPS

IPS (Instrusion Prevention System) [9] je rozšířením IDS ve smyslu prevence. Zatímco IDS v případě podezřelé síťové aktivity skončil po odeslání upozornění, IPS systém se aktivně snaží hrozbám předcházet nebo je zastavit. IPS systémy se vyvíjí samostatně, ale jsou zařazovány mezi IDS systémy. Opět existuje rozdělení na *HIPS (Host–based IPS)* a *NIPS (Network–based IPS)*. Princip je ale odlišný. IPS systém se snaží potenciální nebezpečí

odhalit dříve, než nastane. Využívá k tomu seznam ověřených kontrolních součtů a v případě špatné či neznámé hodnoty nedovolí spuštění (přeposlání) daného programu (paketu). Typicky je možné v každém IPS systému rozlišit 4 prvky.

1. *Trafic normalizer* provádí rozbor a znovusestavování paketů.
2. *Service scanner* klasifikuje zpracovávaná data a vytváří referenční tabulku.
3. *Detection engine* porovnává řetězce v paketech s referenční tabulkou a rozhoduje, zda se paket přepošle dál nebo se zahodí.
4. *Traffic shaper* spravuje výsledný datový tok.

2.7 FUP

FUP (Fair Use Policy nebo Fair User Policy) [22] je politika posktovatelů internetového připojení. Její aplikací se snaží zajistit stejnou dostupnost a kvalitu poskytovaných služeb pro všechny své uživatele. Princip spočívá v zavedení datových limitů na stahování a odesílání dat. Při překročení limitu je dotyčný uživatel penalizován snížením rychlosti nebo zaplacením za data, která stáhl či odeslal navíc. Tato politika se s výhodou uplatňuje především při nižších poskytovaných rychlostech. V dnešní době (jaro 2010) jde převážně o Internet v mobilních zařízeních a wi-fi připojení, kde se běžně dostupné rychlosti pohybují pod hranicí 1 Mbit/s [2]. Pro klasické připojení počítačů přes *ADSL ()* ztrácí FUP svůj původní smysl. Z marketingových důvodů však podle [22] někteří poskytovatelé připojení přes pevnou linku stále FUP v určité formě používají.

Nejčastějším odůvodněním pro použití FUP je zamezení přetížení komunikačních linek tzv. *nadměrnými stahovači*. Jde o skupinu uživatelů, kteří nejen z Internetu stahují filmy, hudbu a další objemné soubory v nadměrném množství, čímž mohou omezit rychlost připojení ostatních uživatelů. FUP tedy zákazníkům internetových poskytovatelů zajišťuje kvalitu předplacených služeb.

Konkrétní podoba FUP se může u různých poskytovatelů lišit. Obecně lze rozlišit *datové* a *protokolové* omezení. Datové omezení nerozlišuje využívané služby a v některých případech slučuje data přijatá i odeslaná. V konečném důsledku jde tedy o datové omezení veškeré uživatelovy komunikace během stanovené časové periody (den, týden, měsíc). Na druhou stranu protokolové omezení podle [22] rozděluje využívané služby do tří tzv. *zón*. Označují se jako *bílá, šedá a černá*.

1. *Bílá zóna* zahrnuje protokoly HTTP (HyperText Transfer Protocol), DNS (Domain Name System), SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol) nebo POP3 (Post Office Protocol version 3). Tyto protokoly jsou využívány pro běžné činnosti na Internetu jako prohlížení webových stránek nebo odesílání a přijímání elektronické pošty. Ve většině případů se neodesílá ani nepřijímá větší množství dat.
2. *Šedá zóna* představuje využívání služeb pro přenos souborů protokolem *FTP (File Transfer Protocol)* nebo sledování on-line videa (*streaming*). V těchto případech jde už o větší množství přenášených dat, ale v rozumné míře by nemělo k omezování ostatních uživatelů docházet.

[2] Vycházím z nabídky tarifů poskytovatelů T-mobile (http://www.t-mobile.cz) a O2 (http://www.cz.o2.com)

3. *Černá zóna* je charakteristická především pro přímou komunikaci mezi dvěma uživateli. Jde o tzv. *P2P (Peer to Peer) spojení*. Typický je přenos a sdílení velkých souborů (filmů, hudby), kde je omezení ostatních uživatelů nejpravděpodobnější.

Kapitola 3

NetFlow

3.1 Co je NetFlow

NetFlow [5, 6, 7] je otevřený protokol firmy *Cisco*, který je využíván k monitorování počítačových sítí na základě datových *IP toků* (viz. kapitola 3.4 na straně 16) umožňující práci téměř v reálném čase. Původně vznikl jako pomůcka pro administrátory, kterým umožňoval lepší pochopení toho, jak jejich síť pracuje. Jednalo se o rozšíření směrovačů, které byly umístěny do klíčových uzlů síťové infrastruktury, odkud zpracovávaly procházející pakety a vytvářely statistiky. Dnes je však možné využít řadu softwarových i hardwarových nástrojů, které umožňují sledovat dostupnost a výkon jednotlivých zařízení, odhalovat bezpečnostní rizika, účtovat zákazníky na základě přenesených dat nebo plánovat rozšíření podle zatížení jednotlivých částí sítě.

3.2 Využití

Technologie NetFlow je dnes velice rozšířena a její využití je možné nalézt v mnoha směrech.

- *Monitorováním celé sítě* je možné získat přehled jak o jednotlivých částech, tak o síti jako celku. Z ukládané historie lze rozpoznat kdy je síť vytížena a kdy je naopak téměř nevyužitá, což lze využít pro plánovaní rozšiřování infrastruktury nebo zálohování dat v době nejmenší aktivity. Díky odezvám v reálném čase je možné odstraňovat vzniklé problémy okamžitě po jejich detekci nebo jim dokonce předcházet.
- *Sledování používaných aplikací a služeb* je vhodné jako odezva pro poskytovalete. Lze tak zjistit, jestli je potřeba nějakou službu posílit nebo na druhou stranu některé nevyužité služby zrušit.
- *Sledováním uživatelských aktivit* je možné naplánovat rozdělení síťových zdrojů nebo určit vhodný tarif (v případě poskytovatele internetu). Dále lze detekovat uživatele, kteří porušují bezpečnostní nebo jiná pravidla chování v síti a zastavit tak potenciální hrozbu v jejím zárodku.
- *Dlouhodobé statistiky* jsou vhodné pro celkové přehledy za konkrétní (delší) období. Využívají se při návrhu rozšíření nebo inovace síťové infrastruktury. Dále se uplatní při dodržování kvality poskytovaných služeb.

- *Detekce a rozbor bezpečnostních incidentů* jsou možné díky práci v reálném čase. V databázi jsou uloženy vzory pro známé typy hrozeb, které se porovnávají s probíhající síťovou aktivitou. V případě shody je možné upozornit správce, který zajistí potřebná opatření.
- *Účtování zákazníků* je možné na základě přenesených dat. Jelikož je znám čas, zdroj, cíl a velikost přenášených dat, mohou zákaznící i poskytovatelé internetu kdykoli zjistit aktuální využití daného tarifu. Dále je možné automaticky generovat a odesílat faktury během splátkového období.
- *Ukládání dat* nebo statistik z nich vytvořených je kromě dodržování *vyhlášky o rozsahu provozních a lokalizačních údajů, době jejich uchovávání a formě a způsobu jejich předávání orgánům oprávněným k jejich využívání* (vyhláška ze dne 7. prosince 2005) důležité pro pozdější analýzy. Nejen manažeři mohou získat přehled o vývojových trendech a přizpůsobit tak poskytované služby.

3.3 Architektura

Typická NetFlow architektura se skládá ze tří prvků: *exportér*, *kolektor* a *prezentér*. Základem jsou ale pouze první dva. Ukázka je na obrázku 3.1 na straně 16.

3.3.1 Exportér

Pod pojmem exportér nebo *sonda* se zde myslí zařízení (hardware) nebo program (software), který pracuje v klíčovém uzlu sítě. Jeho činností je zpracovávání hlaviček procházejících paketů. Z těchto dat vytváří záznamy o jednotlivých tocích, které ve vhodné době [1] odesílá na kolektor. Původně byl exportér vždy součástí Cisco směrovačů. Jedním z problémů je finanční nákladnost těchto zařízení. Tento fakt zabránil širšímu využití technologie NetFlow v malých a středně velkých sítích. Postupem času, kdy se zvyšovala rychlost síťové komunikace, přestávaly směrovače plnit svoji primární funkci, protože byly vytíženy vytvářením NetFlow záznamů. Tento problém se částečně vyřešil pomocí vzorkování (kapitola 3.7 na straně 18), ale jako efektivnější řešení se ukázalo nasazení speciálních hardwarových sond na klíčová místa síťové infrastruktury. Směrovače tak nebyly nadále zatíženy a sondy mohly transparentně provádět svoji činnost.

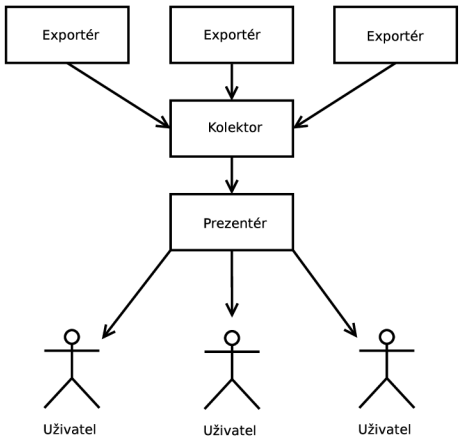
3.3.2 Kolektor

Kolektor bývá v dané síti jediný. Jde o zařízení (např. počítačová stanice), kde běží proces přijímající pakety se záznamy od exportérů. Veškerá data ukládá do centrální databáze, kde jsou připravena pro analýzu nebo prezentaci.

3.3.3 Prezentér

Prezentér není vždy považován jako základní součást NetFlow architektury, nicméně bez možnosti vizualizovat výsledky v přehledné a názorné formě by bylo využití této technologie silně omezeno. Často se jedná o webové grafické rozhraní poskytující filtrování dat uložených v databázi. Nezbytnou součástí jsou různorodé grafy a tabulky, které pomáhají uživateli v jednodušší orientaci.

[1] Viz. kapitola 3.6 na straně 18



Obrázek 3.1: Znázornění architektury NetFlow

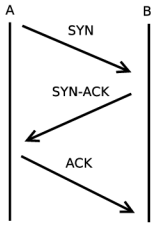
3.4 IP tok

Každý paket, který putuje sítí, obsahuje hlavičku, ve které jsou uvedeny údaje potřebné pro správné doručení. Díky těmto údajům je možné každý paket jednoznačně identifikovat. Pod pojmem IP tok *(IP flow)* se rozumí množina všech paketů sdílejících konkrétní vlastnost. V případě NetFlow je touto vlastností shoda v následujících sedmi položkách [4, str. 3].

1. *Zdrojová IP adresa (source address)* – umožňuje identifikovat uživatele, který paket odeslal.
2. *Cílová IP adresa (destination address)* – umožňuje identifikovat uživatele, který paket přijal.
3. *Číslo zdrojového portu (source port)* – charakterizuje aplikaci nebo službu, která paket odeslala.
4. *Číslo cílového portu (destination port)* – charakterizuje aplikaci nebo službu, která paket přijala.
5. *Protokol třetí (TCP/UDP) vrstvy modelu TCP/IP (layer 3 protocol type)* – rozlišuje druh služby (ICMP = 1, IGMP = 2, TCP = 6, UDP = 17, ...)
6. *Typ resp. kvalita služby (type of service)* – určuje prioritu dané služby.
7. *Název vstupního rozhraní (input interface)* – pomáhá určit využití konkrétních síťových prvků (switch, router).

	Zdrojová IP	**Cílová IP**	**Zdrojový port**	**Cílový port**	**Protokol**
Tok_{AB}	147.108.90.73	77.10.20.50	3333	80	6
Tok_{BA}	77.10.20.50	147.108.90.73	80	3333	6

Tabulka 3.1: Toky A a B



Obrázek 3.2: Znázornění komunikace dvou stran

Z obrázku 3.2 a příslušející tabulky 3.1 na straně 16 výplývá, že tok obsahuje pakety jednosměrné komunikace. V případě výměny dat mezi dvěma body existují naráz minimálně dva toky, pro každý směr jeden. Exportér k těmto údajům při tvorbě NetFlow záznamů přidává mimo jiné ještě časové razítko, velikost paketu a verzi NetFlow protokolu. V databázi je tedy uloženo více dat pro širší možnosti následné analýzy.

3.5 NetFlow cache

Každý exportér disponuje vlastní pamětí (*cache* [4, str. 4]), do které ukládá data o *aktivních tocích.* Aktivním se stává každý nově zaznamenaný tok a jeho aktivita trvá, dokud nedojde k jedné z následujích situací:

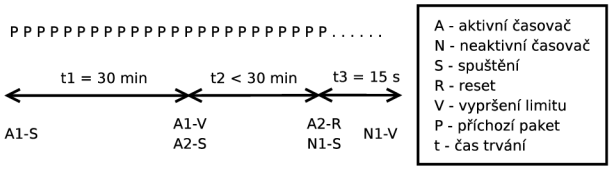
- Po určitou dobu není zaznamenán žádný nový paket daného toku. Dojde k vypršení tzv. *neaktivního časovače.* Bližší popis je v kapitole 3.6 na straně 18.
- Tok je příliš dlouhý. Neustále jsou zpracovávány další pakety stejného toku. Protože pracuje technologie NetFlow v reálném čase, je nutné o dané situaci informovat kolektor. Proto se tok uměle ukončí, odešle k dalšímu zpracování a v případě dalších paketů stejného toku je vytvořen tok nový. Dojde k vypršení tzv. *aktivního časovače.* Bližší popis je v kapitole 3.6 na straně 18.
- Naplnění celé paměti cache. V případě velkého množství toků probíhajících současně by velice rychle došlo k přeplnění paměti a další nové toky by nebyly zaznamenávány. Proto je v takovém případě lepší některé ještě nevypršené toky uměle ukončit a místo, které zabíraly v paměti, použít pro nové záznamy.
- Dojde k načtení TCP příznaků oznamujících ukončení toku. V případě protokolu TCP jsou obsahem hlavičky speciální hodnoty – příznaky, které informují, o jaký typ paketu se jedná. Mimo další jsou to příznaky pro vyjádření standardního ukončení toku (*FIN flag*) nebo předčasného ukončení (*RST flag*). Pokud je paket obsahující některou z těchto položek načten, je jasné, že tok nebude dále pokračovat. Není tedy potřeba čekat na další pakety a je možné vytvořený záznam odeslat a uvolnit tak paměť.

Všechny tyto události představují tzv. *vypršení toku.* Znamená to, že jejich aktivita skončila a je nutné vytvořit a odeslat záznam na kolektor. Exportér většinou neodesílá každý záznam o ukončeném toku samostatně. Často je v jednom paketu uvozeném společnou hlavičkou několik (až 30) takových záznamů. Snižuje se tím zatížení přenosové linky a urychluje se zpracování. Po odeslání záznamu dochází k jeho odstranění z paměti cache, aby bylo možné ukládat nová data.

3.6 Časovače

Při práci s jednotlivými toky využívá exportér pro každý z nich dvou časovačů (*Timers*). Prvním je *neaktivní časovač (inactive timer).* Jde o jednoduchý časový čítač uplynulých sekund implicitně nastavený na hodnotu 15. Vlastní nastavení je možné v intervalu 1 až 600 sekund. Při každém zachycení paketu daného toku je vynulován. Pokud během sekundy žádný paket daného toku nepříjde, dojde k jeho inkrementaci. V případě překročení limitu (nastavená hodnota) dochází k vypršení toku a jeho záznam je odstraněn z paměti cache a odeslán společně s dalšími záznamy na kolektor.

Druhým z obou časovačů je tzv. *aktivní časovač (active timer).* Jak již název napovídá, jde o opak neaktivního časovače. Opět jde o jednoduchý časový čítač, avšak k jeho inkrementaci tentokrát dochází v případě, kdy je tok aktivní, čili dokud přichází na exportér nové pakety daného toku. Pokud za celou sekundu žádný nový paket nepříjde, dojde ke spuštění neaktivního a vynulování aktivního časovače. Jestliže je datový tok příliš dlouhý a dojde k překročení limitu, je tento tok ukončen a odeslán na kolektor. Implicitní hodnotu 30 minut je možné libovolně změnit v rámci interavalu 1 až 60 minut. Ukázka vypršení obou časovačů je na obrázku 3.3.



Obrázek 3.3: Znázornění vypršení časovačů

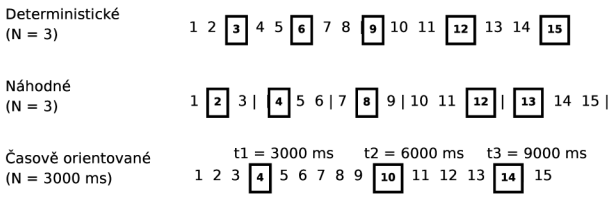
3.7 Vzorkování

Vzorkování neboli *sampling* je technika zpracování přicházejících paketů, která výrazným způsobem snižuje zatížení procesoru. Princip spočívá ve zpracování pouze některých paketů. Pro většinu statistik není potřeba uchovávat informace o každém datovém toku nebo všech jeho paketech. Zejména pro potřeby plánování rozšiřování infrastruktury stačí, když je k dispozici pouze přibližný průběh sledované vlastnosti (například vytížení konkrétního síťového prvku). Implementace firmy Cisco rozlišuje tři metody vzorkování. Ukázka je na obrázku 3.4.

1. *Deterministické vzorkování (deterministic sampling)* spočívá ve zpracování každého *N-tého* paketu, kde N je číslo zvolené uživatelem.

2. *Náhodné vzorkování (random sampling)* vybírá vždy jeden z N paketů, kde N je opět číslo zvolené uživatelem.

3. *Časově orientované vzorkování (time-based sampling)* zpracuje jeden paket za každých N milisekund, kde N volí uživatel. Tato technika je považována za nejlepší ze všech zde uvedených.



Obrázek 3.4: Porovnání vzorkovacích metod

3.8 Verze protokolu NetFlow

Stejně jako každá jiná úspěšná technologie, nevznikl protokol NetFlow za jeden den. Od oficiálního vydání došlo k několika úpravám a rozšířením. V dnešní době (jaro 2010) je možné rozlišovat 10 verzí. Jejich jednoduchý popis je možné nalézt v [3].

1. *Verze 1:* První implementace, dnes již zcela zastaralá, zaměřená pouze na IPv4 (Internet Protocol version 4).
2. *Verze 2:* Pro vnitřní potřeby firmy Cicso. Nikdy nedošlo k její uvolnění.
3. *Verze 3:* Pro vnitřní potřeby firmy Cicso. Nikdy nedošlo k její uvolnění.
4. *Verze 4:* Pro vnitřní potřeby firmy Cicso. Nikdy nedošlo k její uvolnění.
5. *Verze 5:* Nejrozšířenější verze. Nevýhodou je chybějící podpora IPv6 (Internet Protocol version 6). Novinkou oproti předchozím verzím je sekvenční číslo a informace o BGP (Border Gateway Protocol) autonomních systémů. Každý paket obsahuje hlavičku, za kterou následuje sada až 30 NetFlow záznamů.
6. *Verze 6:* Pro vnitřní potřeby firmy Cicso. Dnes nepodporováno.
7. *Verze 7:* Obdoba verze 5. Podpora pro přepínače Cisco Catalyst.
8. *Verze 8:* Mírné rozšíření verze 5. Podpora agregace na směrovačích.
9. *Verze 9:* Devátá verze protokolu NetFlow [7] přinesla dvě hlavní změny. První z nich je podpora IPv6 a druhou možnost využití tzv. *šablon (templates)*. Zatímco v prvním případě jde o nutnost, která umožní využití technologie NetFlow v budoucnu, v druhém případě jde o způsob, jakým lze obsah a strukuru ukládaných dat přizpůsobit konkrétním potřebám.

   Šablona slouží k definici položek, ze kterých se skládá datový záznam. Vyjadřuje jeho strukturu, popis a sémantiku. V případě starších verzí protokolu NetFlow nebylo možné strukturu záznamu měnit. Ne vždy bylo potřeba zpracovávat, přenášet a skladovat všechny položky, které daný formát záznamu podporoval. V takovém případě zabírala tato nepotřebná data prostor v databázi a snižovala užitečný výkon výpočetních prvků.
10. *Verze 10:* Spíše známá jako IPFIX. Jde o rozšíření verze 9, které má sloužit jako standard pro posílání informací o datových tocích.

3.9 Dostupné nástroje

Nástrojů pracujících s technologií NetFlow existuje celá řada. K dispozici jsou komerční i volně dostupné. Mezi nejznámější patří sada aplikací nazývaná *nfdump tools* [20], která je šířena pod BSD licencí. Jde o skupinu programů ovládaných z příkazové řádky podporujících NetFlow ve verzích 5, 7 a 9. Grafické uživatelské rozhraní (GUI – Graphics User Interface) poskytuje nadstavba nfdump-u – *nfsen*. Přes webové rozhraní umožňuje snadné filtrování požadovaných dat a následnou prezentaci ve formě tabulek a grafů. V následujících odstavcích jsou přiblíženy hlavní nástroje ze sady *nfdump tools*, které jsou využívány v rámci aplikace.

NetFlow capture daemon představuje kolektor přijímající datové pakety ze sítě. Získaná data zpracovává a ukládá v binární podobě do souborů. Po uplynutí nastaveného časového limitu periodicky vytváří nové soubory. Implicitní hodnotou je 5 minut. V aplikaci představuje hlavní zdroj dat, kdy běží neustále v pozadí a zpracovává příchozí záznamy, jež jsou okamžitě k dispozici prostřednictvím poskytovaných statistik.

NetFlow dump je určen ke čtení a filtrování dat uložených v souborech, které vytvořil *nfcapd*. Umožňuje výpis řady *top N* statistik jako například nejčastější zdrojové nebo cílové IP adresy, porty nebo používaný protokol. Vše je možné omezit pomocí různých kritérií včetně časového okna vymezující počáteční a koncový čas, ve kterém byly dané toky uloženy. V aplikaci je *nfdump* základním kamenem pro tvorbu všech statistik.

Kapitola 4

Analýza a návrh systému

4.1 Specifikace požadavků

Praktická část této práce spočívá v návrhu a implementaci *informačního systému pro správu FUP* [1]. Aplikace by měla sloužit nejen zaměstnancům firmy poskytující internetové připojení (správci informačního systému) ale i jejich zákazníkům (uživatelé). Zejména z důvodu dostupnosti běžným uživatelům se jako vhodná forma jeví webová aplikace. Vstup do systému bude zabezpečen autorizací pomocí uživatelského jména a hesla.

O každém uživateli budou v databázi uchovány základní informace, které mohou obě strany (správci i uživatel účtu) editovat. Každý uživatel bude mít možnost procházet, filtrovat nebo zobrazovat statistiky o své síťové činnosti. Dále mu systém poskytne přehled provedených plateb, seznam nabízených tarifů či přehled upozornění a novinek ze strany správců. Pro účely komunikace bude sloužit jednoduché diskusní prostředí nebo formulář umožňující odeslání elektronické zprávy. V případech blížícího se splátkového období, zveřejnění nového oznámení, přidání tarifu nebo diskusního příspěvku bude možné nastavit zasílání upozornění v podobě e-mailové zprávy.

Správcům systém umožní editovat zveřejněný obsah (diskusní příspěvky, nabídku tarifů či nástěnku s novinkami), evidovat nové platby, rozšiřovat tabulku známých protokolů (využití ve statistikách), editovat údaje o uživatelích nebo s nimi komunikovat pomocí již zmíněné diskuse. Nejdůležitější funkcí bude filtrace a zobrazování statistik o jednolivých uživatelích i celkové síťové aktivitě.

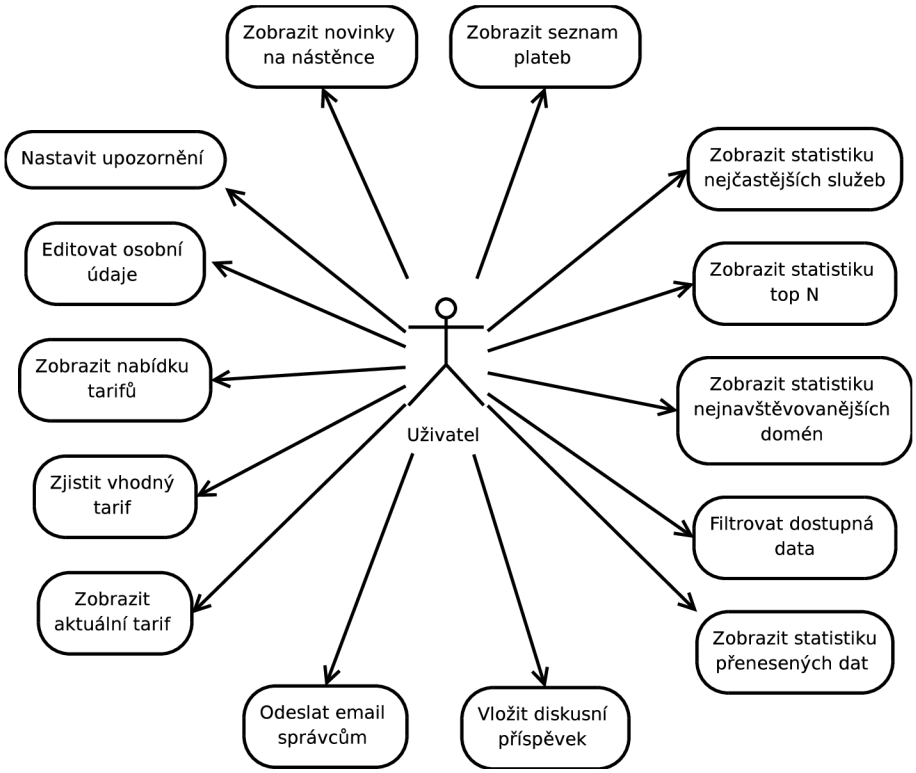
Na pozadí aplikace budou periodicky spouštěny skripty kontrolující překročení datových limitů a hlídající blížící se splátkové datum pro natavené tarify. Částečně bude systém fungovat i jako detektor klasických bezpečnostních hrozeb jako jsou útoky *DoS* (kapitola 2.6.1), *skenování portů* (kapitola 2.6.1) či šíření *červů* (kapitola 2.6.1). Informování o vzniklých událostech proběhne formou e-mailové zprávy odeslané správcům systému (hrozby, překročení limitu) nebo uživatelům (blížící se splátka). Výsledky skriptů budou k dispozici v logovacím souboru. Jeho zobrazení bude možné přes webové rozhraní v administrátorské části.

4.2 Návrh řešení

Pro účely návrhu byl použit grafický modelovací jazyk *UML (Unified Modeling Language) [14]*. Smyslem je srozumitelná grafická prezentace analytické fáze vývoje softwarového pro-

[1] Fair User Policy, více v kapitole 2.7 na straně 12

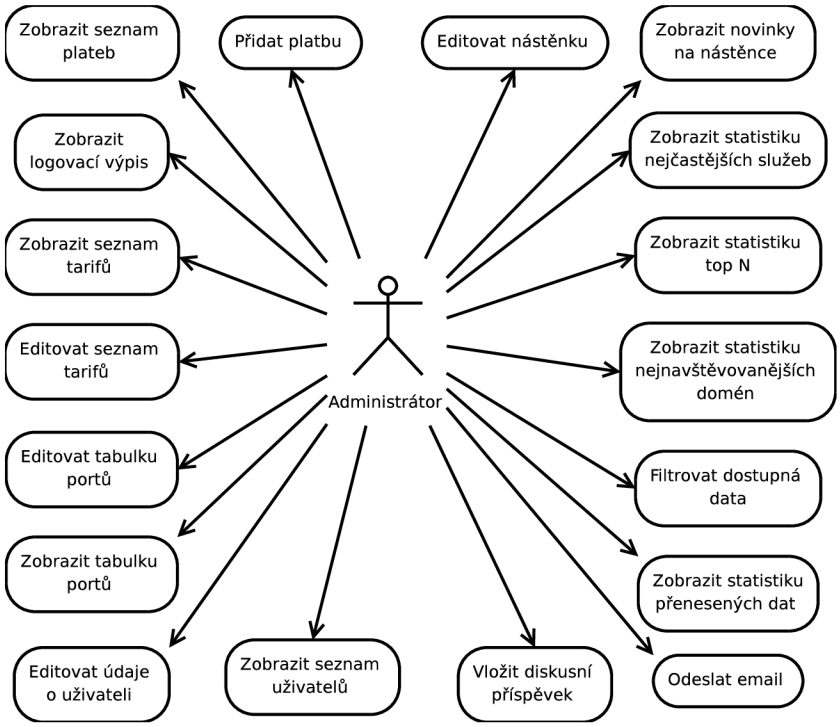
duktu. Funkcionalitu systému z pohledu jeho uživatelů znázorňují diagramy případů užití. Uživatelé se dělí do tří skupin podle práv vymezujících jejich možnosti. Nejvíce omezeni jsou *obyčejní uživatelé*, kteří mají přístup pouze ke svým NetFlow datům, tedy k tokům, v nichž se vyskytuje jejich IP adresa. Možnosti uživatelů jsou na obrázku 4.1.



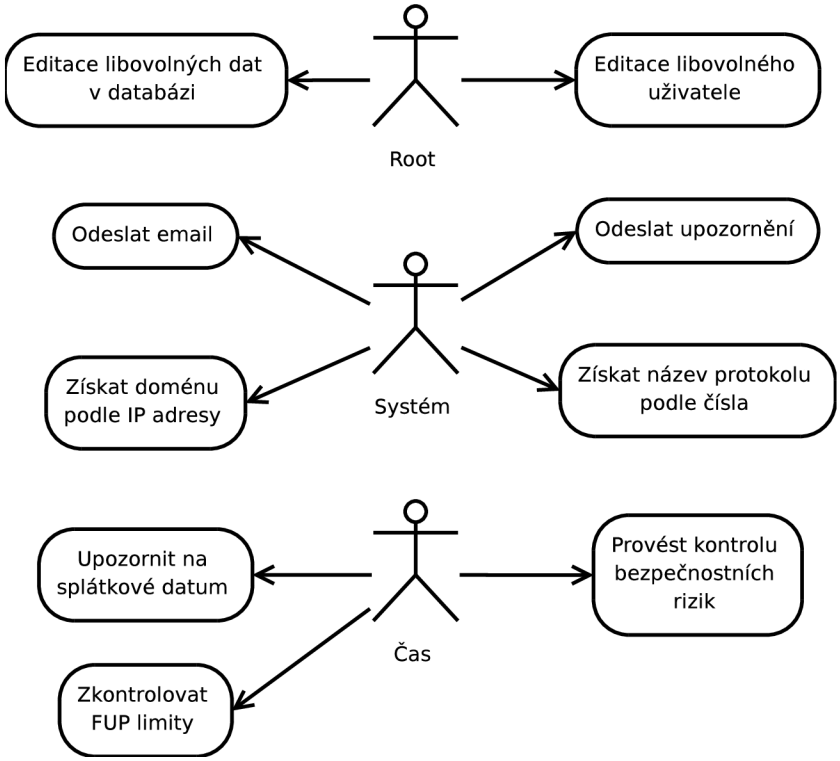
Obrázek 4.1: Diagram případu užití v roli uživatele

Další skupinu tvoří správci systému, kterým jsou k dispozici všechny dostupné NetFlow záznamy. To jim umožňuje sledovat celkovou síťovou aktivitu nebo získat detailní přehled o komunikaci libovolného uživatele. Souhrn poskytované funkcionality je na obrázku 4.2.

Posledním uživatelským profilem je tzv. *superuživatel* neboli hlavní administrátor. Ten má přístup k veškerým systémovým datům i nastavením. Jeho výsadou je hlavně možnost změnit limity charakterizující hlídané bezpečnostní hrozby nebo nastavit datum specifikující den v měsíci, kdy budou uživatelé upozorňováni na blížící se splátku. Společně s možnostmi, které má superuživatel oproti běžným správcům navíc, jsou na obrázku 4.3 také zobrazeni aktéři představující samotný systém a čas. Tito aktéři uzavírají množinu funkcí, které jsou v rámci informačního systému k dispozici.



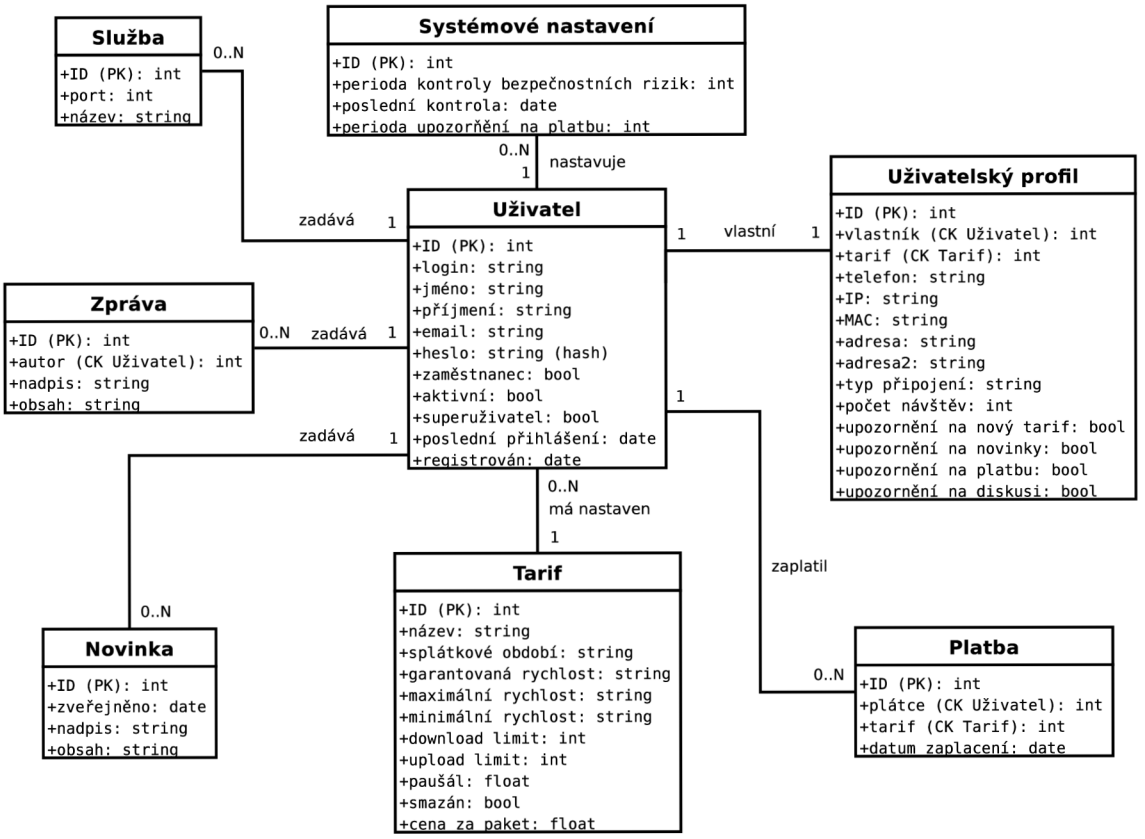
Obrázek 4.2: Diagram případu užití v roli administrátora



Obrázek 4.3: Diagram případu užití v roli systému, hlavního administrátora a času

4.3 Databáze

Stěžejní část dat, se kterými systém pracuje, představují soubory vytvářené NetFlow kolektorem (nfcapd). Vlastní chod systému však také vyžaduje určité datové skladiště. Pro tyto účely byla zvolena databáze *SQLite*. Na obrázku 4.4 je znázorněn *ERD (Entity Relationship Diagram)*, který byl použit pro návrh relační databáze. Jak již bylo uvedeno výše, uživatelé mohou vystupovat ve třech rolích: *uživatelé – zákazníci*, *administrátoři* a *superuživatel*. Rozlišení je realizováno pomocí atributů v entitě *Uživatel*.



Obrázek 4.4: ER diagram použitý při návrhu databáze

Kapitola 5

Implementace

Jako základní implementační prostředek byl zvolen skriptovací jazyk *Python* (http://www.python.org) a webový framework *Django*, který je přiblížen v podkapitole 5.1.1. Data, která jsou nezbytná pro vlastní chod systému, jsou ukládána do databáze *SQLite* (http://sqlite.org). Tvorbu statistik o síťových aktivitách jednotlivých uživatelů zajišťuje open-source nástroj *nfdump* (http://nfdump.sourceforge.net/). Na pozadí aplikace běží softwarový kolektor, který přijímá a ukládá data získaná pomocí protokolu *NetFlow* z externích sond. K tomuto účelu je využito programu *nfcapd*.

Skripty pro kontrolu bezpečnostních incidentů jsou opět pythonovské programy. Jejich spuštění je možné manuálně nebo s využitím prostředků pro automatické periodické spouštění. V systému Linux jde například o program *cron* nebo jeho nadstavbu *anacron*. Více infromací o těchto programech je možné nalézt v manuálových stránkách (*man cron* nebo *man anacron*).

5.1 Použité technologie

5.1.1 Django framework

Django [11] je *webový framework* napsaný v jazyce *Python*. Framework představuje kolekci knihoven a doporučených postupů, které zjednodušují a urychlují vývoj konkrétního druhu aplikací. Poskytuje řadu funkcí řešících běžné nebo opakující se problémy. Programátor se tak může více věnovat vlastní logice programu namísto toho, aby se zabýval již vyřešenými problémy, které by musel sám znovu implementovat.

Samotné Django vzniklo podle [11, kap. Django's History] tak, že jeho vývojáři vytvořili několik aplikací úplně od začátku, bez použití jakéhokoli jiného frameworku. Uvědomili si, že je nemalá část kódu všech aplikací velmi podobná a proto se rozhodli kód zobecnit. Postupně tak začali vytvářet framework. V roce 2005 byl framework Django uvolněn jako open source. Pojmenován byl podle jazzového kytaristy Djanga Reinhardta.

Největší využití umožňuje Django při tvorbě redakčních systémů, neboť poskytuje vlastní administrátorské rozhraní pro správu obsahu a možnost správy uživatelských účtů na základě profilů. Toto zaměření vzniklo z původních aplikací, pro které bylo Django vytvořeno.

MVC

Model View Controller (MVC) [11, kap. The MVC Design Pattern] lze zařadit mezi návrhové vzory, které definují postupy řešící určité problémy. Smyslem MVC je rozdělit logiku

aplikace do tří samostatných částí. Hlavní výhoda je patrná při práci v týmu, kdy je možné programovat všechny části paralelně.

1. *Model* zapouzdřuje způsob uložení a práce s daty. Typicky zajišťuje komunikaci s databází, provádí zápis a čtení souborů nebo volá funkce operačního systému. Dále poskytuje aplikační rozhraní pomocí kterého s nim komunikuje zbytek programu.

2. *View (pohled)* převádí data získaná od modelu na formát, který je vhodný pro prezentaci výsledků uživateli.

3. *Controller (řadič)* představuje řídicí jádro aplikace. Na základě požadavků uživatele získává požadovaná data z modelu a předává je konkrétnímu pohledu k prezentaci.

Základní struktura Django projektu

Každý nově vytvořený projekt obsahuje ve svém kořenovém adresáři následující čtyři soubory.

1. *__init__.py* slouží pro vnitřní potřeby jazyka Python. Označuje adresář, ve kterém se nachází, jako tzv. *balíček (skupina modulů).*

2. *manage.py* představuje nástroj pro práci s projektem. Poskytuje řadu užitečných funkcí jako např. vytvoření superuživatele pro administrační webové rozhraní, spuštění databázové příkazové řádky nebo synchronizaci provedených změn v databázi nebo nastartování vývojového serveru.

3. *settings.py* obsahuje řadu proměnných, pomocí nichž je možné projekt lokalizovat, určit kořenový adresář serveru nebo url adresu, zadat seznam administrátorů či určit, které z podporovaných aplikací budou použity. Framework totiž obsahuje několik předpřipravených balíčků – aplikací, které je možné (a vhodné) použít ve vlastních projektech.

4. *urls.py* definuje přehledný seznam url adres, které se v projektu vyskytují. Mimo jiné lze v tomto souboru definovat, které z pohledů budou vyžadovat autorizaci (přihlášení pomocí loginu a hesla) uživatele.

Vlastní logiku projektu tvoří jednotlivé aplikace. Těch může projekt obsahovat několik. Stačí ale jedna. Název celého projektu (nebo aplikace) se zadává jako argument příkazu pro založení nového projektu (aplikace). Každá aplikace je reprezentována adresářem umístěném v kořenovém adresáři projektu, ve kterém jsou minimálně následující tři soubory. Ve skutečnosti je možné názvy změnit (kromě __init__.py), ale v rámci dodržení konvence je vhodné to nedělat.

1. *__init__.py* slouží pro vnitřní potřeby jazyka Python. Označuje adresář, ve kterém se nachází, jako tzv. *balíček (skupina modulů).*

2. *models.py* obsahuje definici databázové struktury. Jde o výčet a specifikaci jednotlivých tabulek a jejich atributů pomocí jazyka Python. Převod do požadované databázové reprezenatace (MySQL, PostgreSQL, SQLite) obstará Django po zadání příkazu k synchronizaci (parametr *syndb* souboru *manage.py* v kořenovém adresáři projektu). Případné údaje nutné pro přístup k databázi (login, heslo) se nastavují v souboru *settings.py* v kořenovém adresáři projektu.

3. *views.py* slouží pro definici jednotlivých pohledů. Pohledem může být v Djangu libovolná funkce přebírající první parametr typu *HttpRequest (požadavek)* a vracející objekt typu *HttpResponse (odezva)*. HttpRequest je objekt obsahující informace o aktuálním požadavku uživatele. Mimo jiné lze využít pro zjištění použité metody pro přenos parametrů (GET nebo POST) a jejich hodnot. HttpResponse je před odesláním uživateli přeformátován na klasickou *HTTP (Hyper Text Transfer Protocol)* odpověď obsahující hlavičky a tělo odpovědi.

5.1.2 ReportLab Toolkit

ReportLab Toolkit [18] je sada open-source nástrojů umožňující generování *PDF (Portable Document Format)* souborů vyvíjená britskou společností *ReportLab*. Její použití není závislé na operačním systému a jako implementační jazyk byl zvolen Python. Kromě textů je možné generovat tabulky a grafiku v různých bitmapových nebo vektorových formátech. Nejčastějším využitím této knihovny je dynamické generování PDF souborů v rámci webové aplikace.

5.1.3 jQuery

jQuery [13] je volně dostupná knihovna pro jazyk *JavaScript*, která urychluje a zjednodušuje práci s *HTML (Hypertext Transfer Markup Language)* objekty, *CSS (Cascade Style Sheets)*, efekty, animacemi a dalšími dynamickými oživeními webových stránek. Jednou z mnoha výhod je potlačení rozdílů mezi různými prohlížeči na různých operačních systémech. Dále sjednocuje způsob reakce na události, které vznikly na straně klienta (ve webovém prohlížeči). S využitím jQuery je velice snadné dodržovat zásady takzvaného *nevtíravého skriptování*. Jde o techniku programování HTML stránky, kdy kód JavaScriptu není přímo vložen do HTML souboru. Ten obsahuje společně s obsahem pouze definici struktury. Jednotlivé HTML objekty jsou následně dostupné pomocí identifikátorů a je tak možné JavaScriptové soubory dynamicky přidávat nebo ubírat, aniž by se obsah a struktura stránky změnila.

	Verze	**Rok vydání**	**Domovská URL**	**Licence**
Django	1.1.1	2003	http://www.djangoproject.com/	BSD
ReportLab	2.4	2003	http://www.reportlab.com/	BSD
jQuery	1.3.2	2009	http://jquery.com/	MIT, GPL

Tabulka 5.1: Použité frameworky a knihovny

5.2 Zdrojové soubory projektu

Struktura zdrojových souborů odpovídá klasickému projektu ve frameworku *Django*, který je popsán v kapitole 5.1.1 na straně 25. Přehled a základní popis nejdůležitějších souborů je uveden v tabulce 5.2. Statické soubory pro definici vzhledu (*CSS – Cascade Style Sheets*), funkčnosti na straně prohlížeče (JavaScript) a soubory s NetFlow záznamy jsou umístěny ve speciálním adresáři, který vyžaduje nastavení práv přístupu pro všechny uživatele, čímž se odlišuje od zbytku adresářové struktury Django projektu.

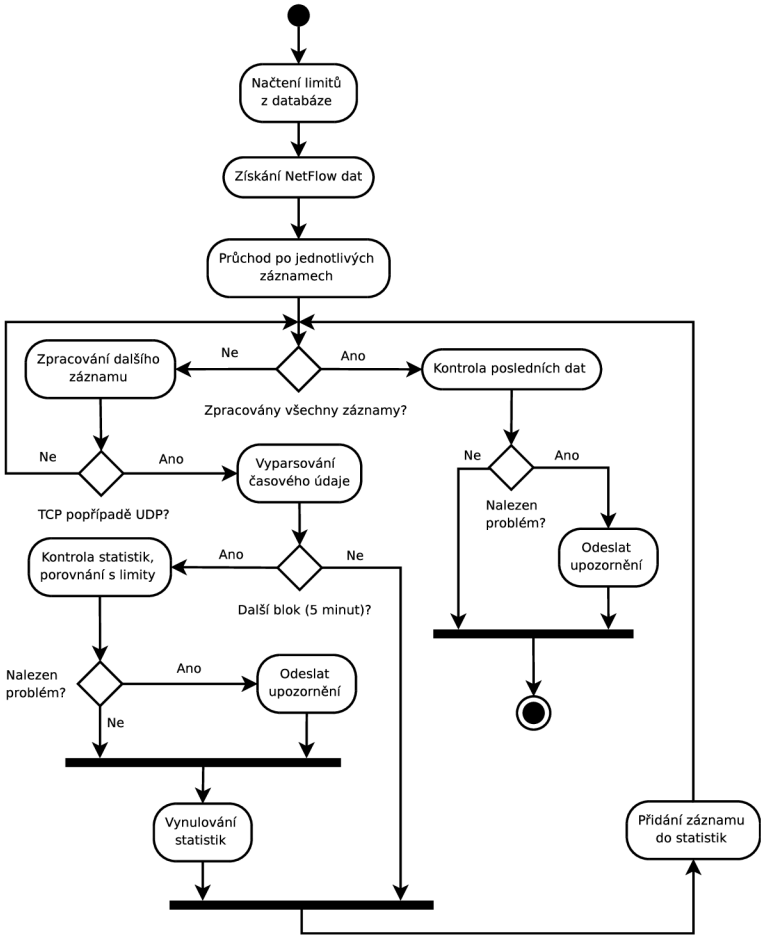
Cesta	Stručný popis
/settings.py	Nastavení projektu z hlediska Djanga. Lokalizace, použité aplikace, globální proměnné obsahující cesty k různým adresářům projektu, ...
/urls.py	Seznam všech URL adres projektu. Dále je zde specifikováno, které stránky vyžadují autorizaci uživatele nebo administrátora.
/public/media/	Adresář se statickými daty (CSS styly, JavaScript, NetFlow data, ...).
/ibp/database/	Adresář, ve kterém je uložen binární soubor představující databázi SQLite.
/ibp/log/	Adresář s textovým logovacím souborem.
/ibp/scripts/	Adresář se skripty pro periodickou kontrolu.
/ibp/templates/	Adresář obsahující strukturu HTML šablon.
/ibp/templatetags/	Adresář se soubory tzv. filtrů. Jde o speciální funkce, které ovlivňují výsledný vzhled prezentovaných dat. Například formát data nebo převod datových jednotek.
/ibp/admin_views.py	Definice všech pohledů v administrátorské části aplikace.
/ibp/alerts.py	Funkce zajišťující odesílání e-mailových upozornění na vzniklé události.
/ibp/forms.py	Definice všech použitých formulářů.
/ibp/graphs.py	Definice všech použitých grafů. Zobrazovaná data se předávají přes globální proměnné v příslušném pohledu.
/ibp/helpers.py	Soubor pomocných funkcí, zejména sestavování příkazů pro nfdump a zpracování odeslaných formulářů.
/ibp/log.py	Funkce pro práci s logovacím systémem.
/ibp/models.py	Definice databázových tabulek v jazyce Python.
/ibp/pdfs.py	Definice funkcí pro dynamickou tvorbu PDF souborů.
/ibp/user_views.py	Definice všech pohledů v uživatelské části aplikace.
/ibp/views.py	Definice pohledů mimo administrátorskou i uživatelskou část. Jde zejména o úvodní obrazovku s přihlašovacím formulářem a funkce pro autorizaci na základě loginu a hesla.

Tabulka 5.2: Krátký popis nejdůležitějších souborů

5.3 Periodické kontroly

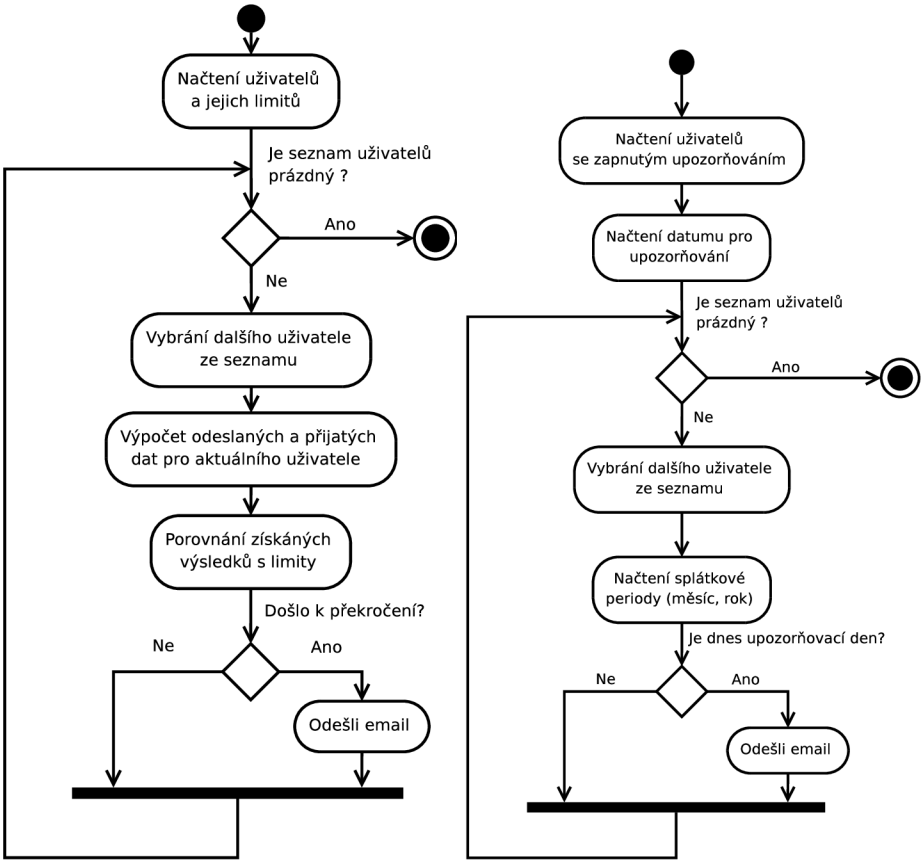
Na pozadí aplikace dochází k periodickému spouštění skriptů, které provádějí určité kontroly. Jedná se o kontrolu překročení datových limitů, upozorňování uživatelů na blížící se splátku a detekci nejčastějších bezpečnostních ohrožení: DoS útok, šíření červů a skenování portů. Blíže jsou tyto incidenty popsány v kapitole 2.6 na straně 10. Výsledky uvedených operací jsou ukládány do logovacího souboru, který je administrátorům k dispozici přes webové rozhraní. Hlavní smysl ale mají odesílaná upozornění ve formě elektronické zprávy.

Na obrázku 5.1 je ve zjednodušené verzi ukázán algoritmus, který je použit pro kontrolu bezpečnostních rizik. Skript načítá limitní hodnoty z databáze a prochází všechna NetFlow data od poslední kontroly. Pro všechny pětiminutové bloky záznamů provádí statistiky, které nakonec porovná s načtenými limitními hodnotami. V případě překročení limitů dojde k odeslání upozorňovacího e-mailu a kontrola pokračuje.



Obrázek 5.1: Diagram kontroly bezpečnostních rizik

Levý diagram na obrázku 5.2 zobrazuje algoritmus kontroly překročení nastavených datových limitů. Z databáze se načtou základní data o uživatelích (např. IP adresa) a jejich limitech v rámci nastavených tarifů. Poté dojde ke zpracování uložených NetFlow záznamů, z nichž se vypočítají součty odeslaných a přijatých dat. Tyto výsledky se porovnají se získanými limity a v případě překročení je správcům systému odesláno upozornění ve formě e-mailové zprávy.

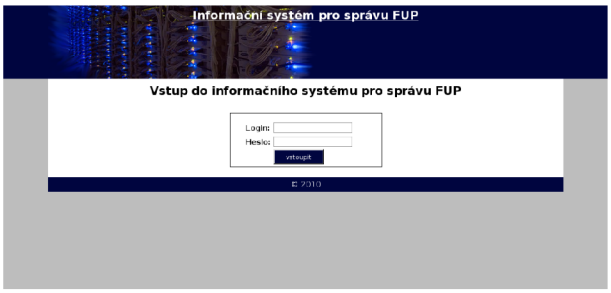


Obrázek 5.2: Odesílání upozornění na blížící se splátku (vlevo) a kontrola FUP limitů (vpravo)

Pravý diagram na obrázku 5.2 pak ukazuje algoritmus pro upozornění uživatelů na blížící se splátkové datum. Princip je obdobný jako v případě kontroly limitů, ale místo výpočtu součtů odeslaných a přijatých dat se pouze zjišťuje, zda je vhodné datum pro odeslání upozornění. K tomu se využívá perioda splátkového období tarifu uživatele a aktuální datum.

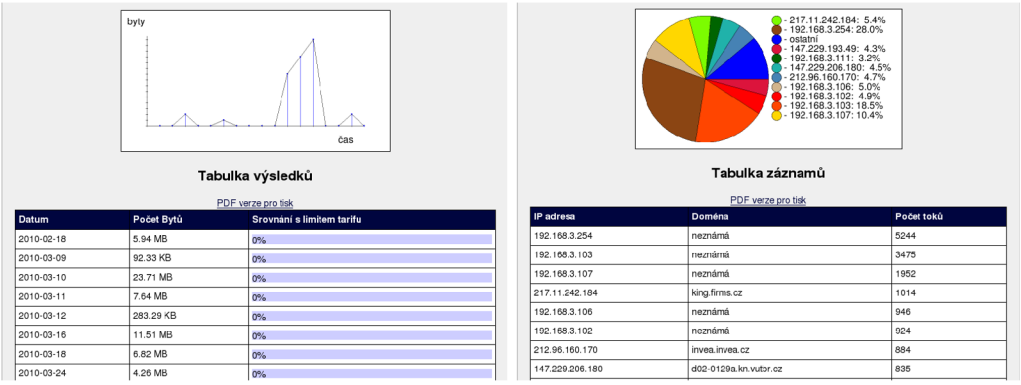
5.4 Vzhled aplikace

Na následujících obrázcích jsou ukázky výsledného vzhledu uživatelského rozhraní aplikace. Obrázek 5.3 ukazuje úvodní obrazovku umožňující vstup do systému. Na obrázku 5.4 je jeden z formulářů pro specifikaci požadované statistiky. Poslední obrázkek 5.5 pak ukazuje grafický doprovod při prezentaci výsledků.



Obrázek 5.3: Přihlašovací obrazovka

Obrázek 5.4: Formulář pro upřesnění dotazu



Obrázek 5.5: Grafická prezentace statistik

Kapitola 6

Závěr

6.1 Dosažené výsledky

Výslednou aplikaci je možné nasadit do provozu na produkčním serveru. Umožňuje přístup třem skupinám uživatelů, které jsou rozlišeny podle profilů. *Běžným uživatelům* systém umožňuje procházet a analyzovat údaje o jejich síťové aktivitě. Výpis statistik je možné stáhnout ve fromátu PDF. K dispozici je dále evidence provedených plateb, nabídka tarifů, diskuse pro komunikaci se správci či možnost nastavit si zasílání upozornění na blížící se splátku pomocí e-mailové zprávy.

Administrátorům jsou k dispozici všechny uložené NetFlow záznamy, z nichž lze získat přehled o probíhající komunikaci na síti jako celku nebo vše filtrovat podle jednotlivých uživatelů, času, portů či protokolů. Dodržování nastavených tarifů v rámci FUP hlídají periodicky spouštěné skripty, které v případě překročení limitů správce upozorní. Dále systém umožňuje správu veřejného obsahu (diskuse, nástěnka, seznam tarifů) a editaci uživatelů.

Posledním z profilů je *superuživatel (root)*, který má přístup k libovolnému nastavení systému včetně časových period pro opakované spouštění skriptů, limitů pro detekci potenciálních bezpečnostních hrozeb nebo editaci administrátorů.

Díky použitému frameworku (Django) je velmi snadné přidávat do projektu další moduly (v rámci frameworku nazývány aplikacemi), které možnosti systému rozšíří.

6.2 Další rozšíření

Vylepšení aplikace je možné zejména v oblasti specifikace výsledné funkcionality. Uměle vytvořené požadavky pravděpodobně neodpovídají reálným potřebám poskytovatele internetového připojení.

V případě reálného nasazení do provozu se nabízí podpora více jazyků. Pro tyto účely disponuje framework Django vlastními prostředky, které implementaci ulehčují.

Dále je možné propracovat grafický vzhled nebo rozšířit možnosti detekce hrozeb. Vyhledávání bezpečnostních incidentů představuje samostatné téma. V rámci této práce šlo o rozšíření naznačující způsob využití.

Literatura

[1] Bigelow, S. J.: *Mistrovství v počítačových sítích.* Computer Press, 2001, ISBN 80-251-0178-9.

[2] Bouška, P.: Začínáme s monitoringem sítě [online]. `http://www.samuraj-cz.com/clanek/zaciname-s-monitoringem-site/`, 2009-09-01 [cit. 2010-03-03].

[3] Caligare: Netflow Export Format [online]. `http://netflow.caligare.com/netflow_format.htm`, 2006-05-10 [cit. 2010-04-08].

[4] Cisco: NetFlow Services Solutions Guide [online]. `http://www.cisco.com/en/US/docs/ios/solutions_docs/netflow/nfwhite.html`, 2007-01-22 [cit. 2010-03-28].

[5] Cisco: Introduction to Cisco IOS NetFlow – A Technical Overview [online]. `http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/prod_white_paper0900aecd80406232.html`, 2007-10 [cit. 2010-04-01].

[6] Cisco: Introduction to Cisco IOS Flexible NetFlow [online]. `http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6555/ps6601/ps6965/prod_white_paper0900aecd804be1cc.html`, 2008-09 [cit. 2010-04-07].

[7] Claise, B.: RFC 3954 — Cisco Systems NetFlow Services Export V9 [online]. `http://www.ietf.org/rfc/rfc3954.txt`, 2004-10 [cit. 2010-04-14].

[8] Claise, B.: RFC 5101 — Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information [online]. `http://www.ietf.org/rfc/rfc5101.txt`, 2008-01 [cit. 2010-03-10].

[9] Endorf, C.; Schultz, E.; Mellander, J.: *Intrusion Detection & Prevention.* McGraw-Hill, 2004, ISBN 0072229543.

[10] Gong, Y.: Detecting Worms and Abnormal Activities with NetFlow [online]. `http://www.symantec.com/connect/articles/detecting-worms-and-abnormal-activities-netflow-part-2`, 2004-09-22 [cit. 2010-04-20].

[11] Holovaty, A.; Kaplan-Moss, J.: *The Definitive Guide to Django: Web Development Done Right.* Apress, 2008, ISBN 978-1-4302-0331-5.

[12] HP: Looking for HP OpenView? [online]. `https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-36657_4000_100`, 2010 [cit. 2010-03-04].

[13] Jadrný, T.: Co je to jQuery [online]. `http://jquery.ic.cz/`, 2010 [cit. 2010-04-17].

[14] Kanisová, H.; Müller, M.: *UML srozumitelně*. Computer Press, 2004, ISBN 80-251-0231-9.

[15] Nagios: About Nagios [online]. `http://www.nagios.org/about/overview`, 2009 [cit. 2010-03-04].

[16] Phaal, P.; Panchen, S.; McKee, N.: RFC 3176 — InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks [online]. `http://www.ietf.org/rfc/rfc3176.txt`, 2001-09 [cit. 2010-03-20].

[17] Quittek, J.; Zseby, T.; Claise, B.: RFC 3917 — Requirements for IP Flow Information Export (IPFIX) [online]. `http://www.ietf.org/rfc/rfc3917.txt`, 2004-10 [cit. 2010-03-17].

[18] ReportLab: ReportLab's Open Source Libraries [online]. `http://www.reportlab.com/software/opensource/`, 2009 [cit. 2010-04-17].

[19] Scambray, J.; Stuart McClure, G. K.: *Hacking bez tajemství*. Computer Press, 2002, ISBN 80-7226-644-6.

[20] Sourceforge: NFDUMP [online]. `http://nfdump.sourceforge.net/`, 2009-05-11 [cit. 2010-04-15].

[21] Tobola, J.: FlowMon – inovace v oblasti monitorování a bezpečnosti počítačových sítích. *Sdělovací technika: telekomunikace – elektronika – multimédia*, , č. 8, 2009: str. 22, ISSN 0036-9942.

[22] Vyleťal, M.: Není FUP jako FUP [online]. `http://www.lupa.cz/clanky/neni-fup-jako-fup/`, 2006-07-17 [cit. 2010-04-17].

[23] Zabbix: About Us [online]. `http://www.zabbix.com/about.php`, 2010 [cit. 2010-03-04].

[24] Zennos: About [online]. `http://community.zenoss.org/community/about`, 2010 [cit. 2010-03-04].

Dodatek A

Popis instalace

V této příloze bude popsána instalace a inicializace informačního systému. Uvedený postup je zaměřen na nově nainstalovanou linuxovou distribuci *Kubuntu verze 9*, nicméně by podle něho neměl být problém zprovoznit systém i na jiné novější distribuci.

A.1 Předpoklady

V další části se bude předpokládat, že jsou splněny požadavky uvedené v tabulce A.1. Pokud tomu tak není (v případě čisté instalace splněny být nemusí), bude nutné je doinstalovat. Možný postup je popsán v tabulce.

Požadavek	**Provedení**
Aktualizovaný systém	`sudo aptitude update` `sudo aptitude dist-upgrade`
Překladač jazyka C	`sudo apt-get install build-essential`
Python (verze 2.4–2.6)	`sudo apt-get install python`
Webový prohlížec (firefox)	`sudo apt-get install firefox`

Tabulka A.1: Požadavky na systém

A.2 Nfdump

Systém pracuje s vlastní kopií binární spustitelné verze programu nfdump, ale v systému musí být k dispozici nfcapd (verze 1.6+), který bude na pozadí v reálném čase ukládat nové NetFlow záznamy (kolektor).

1. Instalace vyžaduje *flex* a *bison*: `sudo apt-get install flex bison`
2. Stažení zdrojových kódů z `http://www.djangoproject.com/download/`
3. Rozbalení do libovolného adresáře: `např. /home/ibp/nfdump-1.6.1`
4. Přesun do zvoleného adresáře: `cd /home/ibp/nfdump-1.6.1/`
5. Konfigurace: `sudo ./configure`

6. Příprava instalace: `sudo make`
7. Instalace: `sudo make install`

A.3 Framework Django

1. Stažení zdrojových kódů z http://www.djangoproject.com/download/
2. Rozbalení do libovolného adresáře: `např. /home/ibp/Django-1.1.1`
3. Přesun do zvoleného adresáře: `cd /home/ibp/Django-1.1.1/`
4. Spuštění instalace: `sudo python setup.py install`

A.4 ReportLab

1. Stažení zdrojových kódů z http://www.reportlab.com/ftp/ReportLab_2_4.tar.gz
2. Rozbalení do libovolného adresáře: `např. /home/ibp/ReportLab_2_4`
3. Přesun do zvoleného adresáře: `cd /home/ibp/ReportLab_2_4/`
4. Spuštění instalace: `sudo python setup.py install`

A.5 Zprovoznění APACHE serveru

1. Instalace programu Synaptic: `sudo apt-get install synaptic`
2. Spuštění programu Synaptic: `sudo synaptic`
3. Instalace LAMP serveru
   (a) Zvolit: `Akce/Vybrat balíky podle účelu`
   (b) Zaškrtnout `LAMP server`
   (c) Potvrdit: `OK`
   (d) Během instalace je nutné zadat heslo administrátora pro přístup k MySQL serveru (pro účely informačního systému však MySQL není potřeba).
4. Ověření funkčnosti
   (a) Dotaz na úvodní stránku localhostu: `např. firefox http://localhost`
   (b) Měla by se zobrazit stránka s oznámením funkčnosti (`It works!`).

A.6 Instalace webové aplikace

1. Zkopírování adresáře *web* na disk: `např. do /home/ibp/web/`
2. Zkopírování statických souborů na APACHE server:
   `sudo cp -r /home/ibp/web/public/media /var/www/`
3. Spuštění Django serveru
   (a) `cd /home/ibp/web/`
   (b) `make runserver`
4. Vytvoření databáze: `make syndb`
5. Inicializace databáze: `make data`
6. Spuštění aplikace ve webovém prohlížeči: `firefox http://localhost:8000/login/`

A.7 Přihlášení do systému

Pokud byla databáze vytvořena a inicializována (viz. předchozí sekce A.6), jsou k dispozici účty uvedené v následující tabulce.

Role	**Login**	**Heslo**	**E-mail**	**Heslo (pro e-mail)**
Uživatel	user1	user1	ibpUser01@gmail.com	ibpUser01Pass
Uživatel	user2	user2	ibpUser02@gmail.com	ibpUser02Pass
Uživatel	user3	user3	ibpUser03@gmail.com	ibpUser03Pass
Uživatel	user4	user4	ibpUser04@gmail.com	ibpUser04Pass
Uživatel	user5	user5	ibpUser05@gmail.com	ibpUser05Pass
Uživatel	user6	user6	ibpUser06@gmail.com	ibpUser06Pass
Administrátor	admin1	admin1	ibpUsername@gmail.com	ibpPassword
Administrátor	admin2	admin2	ibpUsername@gmail.com	ibpPassword
Superuživatel	root	root	ibpUsername@gmail.com	ibpPassword

Tabulka A.2: Systémové a e-mailové účty uživatelů

A.7.1 Přístup superuživatele

Superuživatel má možnost měnit libovolná data v databázi. Pro tyto účely je k dispozici speciální administrační rozhraní implicitně poskytované frameworkem Django. Přihlašovací obrazovka je na adrese `/admin/`, tedy v našem případě `http://localhost:8000/admin/`.

A.8 Spuštění kolektoru

Pro on-line zpracování příchozích dat je nutné spustit proces představující NetFlow kolektor. Jde o součást balíku nfTools, konkrétně *nfcapd*. Popis parametrů je možné najít v manuálových stránkách (man nfcapd).

```
Příkaz: nfcapd -l /home/ibp/web/public/media/nf/data/
                -b <IP adresa> -p 9996
```

IP adresou se zde myslí lokální adresa počítače, na kterém je nfcapd spuštěn. Aby bylo možné data na kolektor zasílat ze vzdálených míst (exportérů), bude potřeba, aby zadaná IP adresa byla veřejná.

A.9 Spuštění periodických skriptů

Pro zajištění kontroly dodržování FUP limitů, detekci potencionálních hrozeb nebo upozorňování uživatelů na blížící se splátku je nutné nastavit v systému pravidelné spouštění skriptů. Manuální spuštění je možné přes webové rozhraní v administrátorské části systému.

Pro automatické periodické spouštění je možné využít programu *cron*, resp. *anacron*. Po zadání příkazu `crontab -e` se otevře soubor, do kterého je možné zapsat požadované parametry spouštěných skriptů. Příklad je v tabulce A.3. Položky v záhlaví tabulky znamenají po řadě: minuta, hodina, den v měsíci, měsíc, den v týdnu a požadovaný příkaz. V prvním a druhém případě je tedy příkaz proveden každý den v 0 hodin 0 minut. Poslední příkaz je spouštěn na začátku každé hodiny.

m	**h**	**dom**	**mon**	**dow**	**cmd**
0	0	*	*	*	`cd /home/ibp/web/; make fup`
0	0	*	*	*	`cd /home/ibp/web/; make pay`
0	*	*	*	*	`cd /home/ibp/web/; make threat`

Tabulka A.3: Nastavení automatického periodického spouštění skriptů