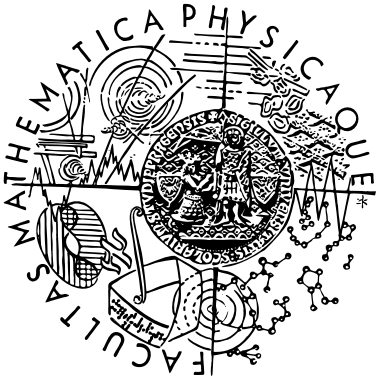
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta



Bakalářská práce

Peter Sabolčák

FlowIDS

Středisko informatické sítě a laboratoří

Vedoucí bakalářské práce:

Dan Lukeš

Studijní program: Informatika, Správa počítačových systémů

2006

Na tomto mieste by som rád poďakoval vedúcemu Bakalárskej práce, Danovi Lukešovi, za cenné rady a podporu v priebehu tvorby bakalárskej práce. Rád by som poďakoval ľuďom, ktorí mi pomáhali s týmto neľahkým projektom, za neutíchajúcu podporu a to hlavne svojim rodičom a mojej Beruške. Dodatočne by som rád poďakoval Ing. Jánu Chadimovi za praktické rady a poskytnutie testovacieho hardware.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím s požičaním práce.
V Prahe dňa 22.5.2006 Peter Sabolčák

Obsah

1 Úvod **6**
- 1.1 Cieľ . . . 6
- 1.2 Čo je IDS . . . 6
- 1.3 NetFlow . . . 7
- 1.4 Porovnanie s ostatnými implementáciami . . . 9
- 1.5 Niečo o FlowIDS . . . 11

2 Užívateľská dokumentácia **12**
- 2.1 Vlastnosti . . . 12
- 2.2 Inštalácia . . . 12
- 2.3 Konfiguračné nástroje . . . 13
  - 2.3.1 Príkazová riadka . . . 13
  - 2.3.2 Konfiguračný súbor . . . 14
  - 2.3.3 Príklad konfiguračného súboru . . . 17
- 2.4 Pravidlá . . . 18
  - 2.4.1 Špecifikácia prúdu . . . 18
  - 2.4.2 Dodatočné vlastnosti . . . 20
  - 2.4.3 Príklad súboru s pravidlami . . . 23
- 2.5 Post-detekcia . . . 24
  - 2.5.1 Flow premenné . . . 25
- 2.6 Používanie programu . . . 25

3 Programátorská dokumentácia **27**
- 3.1 main.c . . . 27
- 3.2 main.h . . . 28
- 3.3 ConfParser . . . 29
- 3.4 RulesParser . . . 30
- 3.5 log . . . 33
- 3.6 process . . . 33
- 3.7 DetectionEngine . . . 36
- 3.8 StringMatch . . . 37

3.9 Cisco 37
3.10 Util 38

4 Záver **39**
4.1 Výsledok práce 39
4.2 Záverečné testovanie 39

Názov práce: FlowIDS
Autor: Peter Sabolčák
Katedra (ústav): Středisko informatické sítě a laboratoří
Vedúci bakalárskej práce: Dan Lukeš
e-mail vedúceho: dan@obluda.cz

Abstrakt: FlowIDS je systém umožňujúci detekciu nežiadúceho dátového toku v počítačovej sieti (nežiaducim môže byť napríklad vírová aktivita ako aj nadmerné vyťažovanie zdrojov sieťovej infraštruktúry) a následne vykonať proti takýmto tokom protiopatrenia. Informácie o dátových tokoch nám poskytuje hardware sieťovej infraštruktúry. Následná eliminácia nežiaducich aktivít sa vykonáva zmenou nastavenia tohto hardware. FlowIDS sa zameriava hlavne na sieťový hardware Cisco. Využíva jeho protokol NetFlow k obstarávaniu dát. Obstarávanie však môže vykonávať priamo naše sieťové rozhranie, ktoré nielen že nahradzuje NetFlow ale obstaráva podrobnejšie informácie podľa, ktorých môžeme vykonať hlbšiu analýzu. Analýza a protiopatrenia sa určujú na základe pravidiel definovaných v súbore, ktorý je načítaný pri spustení.

Kľúčové slová: IDS, NetFlow, IPS, Cisco

Title: FlowIDS
Author: Peter Sabolčák
Department: Network and Labs Management Center
Supervisor: Dan Lukeš
Supervisor's email address: dan@obluda.cz

Abstract: FlowIDS is system which can detect some of the undesirable traffic in computer networks (undesirable traffic could be also ie. virus activity or overloading of network) and mostly doing counteraction which are set by administrator. Information about data flows are provided by hardware of network infrastructure, eliminitaion of undesirable activity is done through changes in network hardware settings. Regarding the quantity of solution which are provided on market, I decided to focus on Cisco network hardware, which is now popular and mostly spread. FlowIDS also benefit from NetFlow protocol (introduced by Cisco) to gather easily necessary data. NetFlow is not the only source of gathering data. FlowIDS can also operate with local interface to gather data directly from network.

Keywords: IDS, NetFlow, IPS, Cisco

Kapitola 1

Úvod

1.1 Cieľ

Hlavným cieľom tejto práce je vytvoriť program, ktorý bude pomáhať systémovým administrátorom k lepšiemu zabezpečeniu ich servera ako aj ich sieti ako takej. FlowIDS v prvom rade umožňuje detekciu prieniku za pomoci užívateľom definovaných pravidiel, ktoré sú pre jednoduchosť podobné s pravidlami programu snort (viz 2.4). Za pomoci týchto pravidiel určujeme, ktorý prenos je pre nás neprijateľný. Určujúcim elementom pre toto rozhodovanie je najmä hlavička paketu, ale môžu sa zvažovať aj telá a taktiež množstvo a veľkosť paketov. Týmto sme dosiahli vysokú mieru flexibility pri definovaní jednotlivých typov útokov a tým k vyššej pravdepodobnosti úspešnej detekcií prieniku.

Oproti podobným nástrojom ponúka FlowIDS aj možnosť post detekčnej aktivity a to buď vo forme záznamu v log súboru, syslogu, alebo varovanie cez email. Zaujímavou voľbou je aj prekonfigurovanie sieťového prvku na základe konfiguračného súboru definovaného v pravidlách. Tento súbor obsahuje príkazy, ktoré majú byť odoslané sieťovému prvku.

1.2 Čo je IDS

IDS alebo Intrusion Detection System, je už ako samotný názov napovedá systém na detekciu prienikov. Tieto systémy sa delia na dve skupiny: Network based IDS a Host based IDS. Host based IDS sa snaží detekovať prieniky za pomoci analýzy lokálnych systémových súborov a ďalších lokálnych parametrov systému, číže je viazaný na operačný systém ako taký. Zatiaľ čo hlavnou úlohou Network based IDS je takéto prieniky detekovať na základe vyhodnocovania toku dát v monitorovanej sieti. Tieto dáta obstaráva sieťové

rozhranie, ktoré sa nachádza na serveru kde je nainštalovaný náš program. Vzhľadom k tomu, že cez router prechádza celý tok, ktorý ide do našej siete, tak pri snahe ochrániť celú našu sieť je najvhodnejšie odchytávať tieto pakety (dáta) priamo na ňom. Aby sme odľahčili router od ďalšej úlohy, nášho detekovania, je dobré tieto dáta preposielať na náš vyhodnocovací stroj, ktorý bude analyzovať pakety.

Vzhľadom k tomu, že sa v dnešnej dobe čoraz viac stretávajú systémový administrátori s rôznymi nástrahami ako červy, vírusy, trójske kone a pod., nástroje, ktoré znižujú riziko napadnutia systému sú vyhľadávané, najmä vďaka možnosti detekovať najnovšie útoky, keďže pravidlo na zamedzenie daného útoku nie je ťažké vytvoriť. Na druhú stranu potrebná systémová záplata nemusí byť k dispozícií ani po týždni od oznámenia danej chyby a tak IDS je občas jediným možným a účinným protiopatrením pred útokmi. Mimo to umožňujú taktiež detekovať špeciálne, málo rozšírené útoky, ktoré nemajú veľa podobných vlastností. Tieto nástroje nielen že ponúkajú možnosť detekcie útoku ale aj efektívnejšie vystopovať útočníka a to za pomoci logov, ktoré priebežne vytvárajú.

1.3 NetFlow

NetFlow je technológia vyvinutá firmou CISCO za účelom umožnenia lepšej správy siete na základe ich sieťových prvkov. Túto technológiu neimplementuje iba CISCO, ale taktiež aj firma Juniper, ktorá si ju pomenovala ako *cflowd export*. Technológia sa dá uplatniť v niekoľkých oblastiach sieťovej správy:

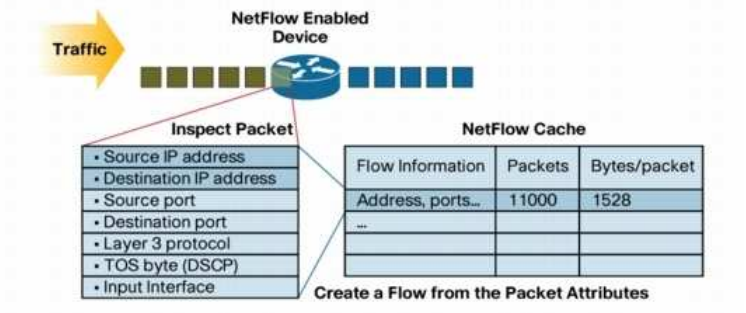
- Sieťové plánovanie
- Aplikácia politiky na prenos
- Zúčtovanie a evidencia prenosu
- Bezpečnostný monitoring

Samozrejme, že pri konkrétnejších požiadavkách sa môžeme zamerať na monitoring užívateľov ako takých, ako aj aplikácií a tým si vytvoriť štatistiky pre určenie zaťaženia protokolu a pod.

NetFlow technológia pracuje na princípe kategorizácie prúdoch (ang. flow) dát. Prúd je jednosmerná sekvencia paketov medzi zdrojom a cieľom. To znamená, že pri obojsmernej komunikácií sú definované 2 prúdy. Prúdy sú jednoznačne identifikované na základe týchto kľúčov:

- Zdrojová IP adresa
- Cieľová IP adresa
- Zdrojové číslo portu
- Cieľové číslo portu
- Typ protokolu (definovanom na 3. vrstve)
- Type of Service (Typ služby)
- Vstupné logické rozhranie

Týchto sedem vlastností jednoznačne určujú prúd. Ak má prúd odlišnú čo by len jednu vlastnosť a nie je definovaný žiaden prúd, ktorý by mal rovnaké vlastnosti, tak je tento prúd novým prúdom v systéme. Ak takýto prúd existuje tak sa jeho doterajšie hodnoty patrične inkrementujú. To sa deje až do chvíle kým sa flow-y neexportujú. Export flow-ov je tiež špecifická vec, nakoľko sa deje len za určitých okolností. Export môže nastať v prípade ak daný flow je neaktívny určitý čas. Neaktívny v zmysle, že v danom časovom úseku neprišiel žiadny nový paket pre tento flow. Flow môže byť označený na export tiež v prípade ak jeho životnosť presiahne určitý časový úsek. Užitočné pri dlho trvajúcich FTP prenosoch. Poslednou možnosťou je keď daný paket má flagy, ktoré oznamujú ukončenie spojenia (napr. FIN, RST). Prednastavený čas pre časovač, ktorý označuje jednotlivé flow-y je v prípade 'inactive' 15 sekúnd a pre 'active' 30 minút. Tieto dáta sa potom exportujú na počítač, ktorý sa nazýva NetFlow Collector. Ten môže dané dáta patrične agregovať a upravovať. V našom prípade funguje ako collector FlowIDS.



NetFlow paket, obsahuje dve oblasti: hlavičku a telo. Tie sú však odlišné od hlavičiek, ktoré definujú protokoly ako IP. Telo aj hlavička majú pevne daný formát.

Postupom času sa NetFlow vyvíjal a tak pribúdali nové vlastnosti, ktoré dokáže daný NetFlow zaznamenať a uložiť. NetFlow je momentálne v 5 verziách. Špecifikáciu týchto verzií môžte nájsť v tabuľke 1.1.

NetFlow je podporovaný už takmer na všetkých sieťových prvkoch od CISCO. K jeho chodu potrebujete mať IOS 11.1 a vyššie. K tomu aby sme mohli exportovať NetFlow dáta na váš server, kde beží NetFlow Collector (napr. FlowIDS) musíme urobiť tieto kroky:

1. Prihlásenie do systému, prechod do enable módu a aktivácia config módu (`configure terminal`)
2. `ip flow-export version 5`
3. `ip flow-export destination X.X.X.X YYYY` - kde X.X.X.X značí IP adresu nášho servera, kde nám beží FlowIDS a YYYY je UDP port, na ktorom očakáva export pakety.
4. `ip flow-cache timeout active 1` - zaručí nám exportovanie paketov po uplynutí active času (tu je to konkrétne 1 minúta).
5. `ip route-cache flow` - aktivácia NetFlow pre IP routing

Tieto príkazy boli testované na systémoch IOS vo verzii 12.2 a 12.3. Na iných systémoch sa môžu líšiť, a tak konzultujte prípadné problémy s dokumentáciou k vašim sieťovým prvkom. Vypnutie NetFlow sa deje cez CISCO typický príkaz `no` za ktorým nasleduje príkaz, ktorý chcem deaktivovať.

1.4 Porovnanie s ostatnými implementáciami

FlowIDS by sa dal porovnať s niekoľkými rozdielnymi programami. Avšak asi najbližšie má k nemu program snort. Tento program je vyvíjaný už viac ako 7 rokov a za tieto roky si vybudoval adekvátne renomé na trhu. Snort je ako sa sám pasuje, štandardom na trhu IDS a aj preto som zvolil čo najpodobnejšie pravidlá. Program však aj napriek tomu, že je tak rozšírený nemá funkcie, ktoré sú v dnešnej dobe čoraz viac potrebnejšie. Hlavnou z funkcií, ktorú momentálne neobsahuje je podpora IPv6, ktorá začína byť čoraz viac potrebnejšia. Ďalšou z neimplementovaných funkcií je podpora technológie NetFlow (viz sekciu 1.3), ktorá je v dnešnej dobe implementovaná na takmer každom hlavnom sieťovom prvku. Tieto a ďalšie "nedostatky" som

Verzia	Popis
1	Základná verzia, obsahuje export základných údajov, ako sú samotné kľúče a počet prenesených dát a paketov.
5	Momentálne najrozšírenejšia verzia, ktorá sa líši od predošlej v dvoch nových poliach v tele a v jednom v hlavičke. Konkrétne sa jedná o AS a Masky (ako pre zdrojovú tak aj pre cieľovú stanicu) a v hlavičke je pridané sekvenčné číslo, ktoré je každým novým exportom inkrementované. Týmto spôsobom sa dá jednoducho zistiť koľko sme stratili paketov.
7	Špecifická pre CISCO Catalyst 6500 a 7600 switche. Takmer úplne rovnaká s verziou 5, až na niektoré rezervované políčka, ktoré dostali svoje určenie.
8	Možnosť výberu až z 11 agregačních schém.
9	Od tejto verzie je možnosť si flexibilne navoliť čo exportovať a čo nie, a tak už nie je nutnosť kvôli nejakej novej požiadavke na export navrhnúť novú verziu NetFlow.

Tabuľka 1.1: Verzie NetFlow

sa pokúsil v mojom programe odstrániť a pridať niekoľko ďalších rozširujúcich funkcií, ktoré umožňujú ešte väčšiu kontrolu nad sieťou.

Vzhľadom k tomu, že FlowIDS nie je bežným IDS, je na mieste definovať prípadne ďalší program, ktorý má obdobné funkcie a ďalšie nadštandardné funkcie FlowIDS by sa dali s ním porovnávať. Keďže FlowIDS nie je len IDS postavené nad lokálnym rozhraním, ale aj analyzátor NetFlow exportov a zároveň je schopný vykonávať na ich základe evidenciu dátového toku. K tomuto účelu bol vytvorený program flow-tools, ktorý svojimi funkciami predčil svojich konkurentov a momentálne je najvyužívanejším programom na analýzu NetFlow exportov. Tomuto programu však chýbajú vlastnosti ako možnosť analýzy na základe pravidiel a samotná hlbšia analýza odchytených dát. Tieto prvky sa dajú v niektorých prípadoch nahradiť jednoduchým využitím funkcii `bash`u a ďalších externých programov. Je to však nielenže pomalé, ale taktiež obmedzujúce, nakoľko týmto potrebujeme využívať funkcie externých programov, ktoré nemusia byť k dispozícií.

FlowIDS sa snaží všetky tieto nevýhody odstrániť a vlastnosti týchto dvoch programov spojiť v jeden a postaviť nový systém, ktorý sa svojimi vlastnosťami blíži IPS (Intrusion Prevention System).

1.5 Niečo o FlowIDS

FlowIDS je Network based IDS, ktoré sa snaží nielen o detekciu ako najrozšírenejší IDS v dnešnej dobe snort, ale aj využiť prostriedky siete k zabráneniu zneužívania našich zdrojov. Takéto systémy sa už nazývajú IPS (Intrusion Prevention System), keďže sa nesprávajú pasívne, ale práve naopak, pokúšajú sa prípadného útočníka zneškodniť už v zárodku.

Vzhľadom na širokú škálu ponúkaných sieťových riešení bolo nutné si vybrať jedno, na ktoré sa bude projekt zameriavať. Ja som sa rozhodol zamerať na sieťové prvky od firmy CISCO, keďže práve tá je momentálne najrozšírenejšou firmou v tejto oblasti. Jej sieťové prvky obsahujú operačný systém IOS, ktorý ponúka možnosť flexibilnej konfigurácie a viacero možností monitorovania. Práve tieto vlastnosti CISCO prvkov boli rozhodujúce pri výbere zariadení, pre ktoré bude projekt optimalizovaný. Dodatočná optimalizácia bola aj v smere odchytávania paketov a to za pomoci filtra, ktorý je možné nastaviť tak aby sa k vyhodnoteniu dostali len tie pakety, ktoré spĺňajú určité kritéria. Ostatné sa budú zahadzovať.

Kapitola 2

Užívateľská dokumentácia

Pri čítaní tejto časti sa očakáva od čitateľa znalosť základných sieťových prvkov a ich vlastností ako aj základy fungovania služieb v počítačových sieťach.

2.1 Vlastnosti

Hlavnou funkciou FlowIDS je detekcia neželaného toku dát. Takýto tok dát si definujeme za pomoci pravidiel (viz. 2.4). Príkladom takého toku môže byť pokus o napadnutie nášho počítača. Tento útok má typické niektoré vlastnosti, ktoré sa dajú špecifikovať a to za pomoci našich pravidiel. Taktiež môžeme označiť za neželaný tok dát aj prípad keď konkrétna stanica v sieti prekročí určený objem dát. Takéto a ešte mnoho ďalších možností môžete nastaviť za pomoci pravidiel FlowIDS. Je treba podotknúť, že program nedokáže spájať fragmenty paketov, a tak nám môžu uniknúť niektoré podstatné dáta. Preto je dobré buďto na našich sieťových prvkoch zahadzovať fragmentované pakety, alebo využiť možnosti nášho programu a minimalizovať tak škody, ktoré tým môžu byť spôsobené. Jednou s možností je určiť minimálnu veľkosť fragmentovaného paketu za pomoci `minfrag` parametru v pravidlách (2.4).

2.2 Inštalácia

Inštalácia programu je veľmi jednoduchá. K zdarnému prekladu zdrojových súborov a vytvoreniu binárneho súboru je nutné mať nainštalované v systému knižnicu lipcap. Program sa dá nainštalovať dvomi spôsobmi a to za prvé spustením príkazu '`scons`' v hlavnom adresári programu. `sconsj`e ekvivalent k autoconf-u a automake a snaží sa ich nahradiť, jedinou nevýhodou je, že k

spusteniu tohto programu, a tým pádom k zdarnému prekladu musíme mať nainštalovaný samotný program `scons` a taktiež `python`. Z tohto dôvodu bola pridaná neskôr podpora pre obvyklú cestu inštalácie programu zo zdrojových súborov a to konkrétne: `./configure && make`. Po tomto príkaze bude na vás čakať v adresáru src/ binárny súbor flowids, ktorý už môžete spustiť. Program je momentálne portovaný na Linux a BSD systémy.

2.3 Konfiguračné nástroje

Nastavenia programu je možné meniť za pomoci príkazového riadku ako aj cez konfiguračný súbor. Tento súbor sa špecifikuje cez príkazový riadok. Pri jeho nešpecifikovaný sa hľadá súbor s názvom "`flowids.conf`" v pracovnom adresári programu. V prípade, že dané nastavenie definujeme v konfiguračnom súbore ako aj v argumentoch, tak sa uprednostní hodnota uvedená v argumentoch.

2.3.1 Príkazová riadka

Pre prepínače neexistujú dlhé varianty. Niektoré parametre nemôžu byť definované v jednom volaní programu a to konkrétne tie, ktoré nám špecifikujú zdroj dát (Local, NetFlow, SPAN). Pri nešpecifikovaní akýchkoľvek parametrov sa program spustí s prednastavenými hodnotami, bude odchytávať na lokálnom sieťovom rozhraní, ktoré sa pokúsi nájsť sám. Avšak ak nebude mať definovaný súbor s pravidlami, program sa ukončí. Ostatné hodnoty sa nastavia na ich prednastavenú hodnotu, tá je buď špecifikovaná v tabuľke a ak tam nie je tak to znamená, že daná hodnota bude 0 (prípadne prázdny reťazec). Nasleduje samotný zoznam parametrov (prepínačov) pre príkazový riadok:

-c súbor definovanie konfiguračného súboru. (viz 2.3.2)

-d naštartuje nám program v daemon móde. Je vhodné nastaviť kam má program logovať, keďže pri nešpecifikácií log súboru sa nám informácie vypisujú priamo na konzolu čo je však v tomto prípade nevhodné, keďže sa od danej konzoly odpojujeme a tak prídeme o informácie ohľadom FlowIDS a jeho činnosti.

-D debug mód, určený k analýze programu samotného a to za pomoci obsiahlejších výpisov.

-f filter parameter určuje filter, ktorý sa pošle knižnici, ktorá obstaráva pakety. Nad rovnakou knižnicou pracuje tcpdump a tak aj formát tohto filtra je rovnaký. (viz. [4]

-i zariadenie špecifikácia zariadenia, cez ktoré bude program operovať.

-l log_file log_file obsahuje názov súboru kde sa budú zapisovať logy z programu.

-L nastavenie formy obstarávania dát na lokálnu, tj. k odchytávaniu paketov bude slúžiť sieťový adaptér nachádzajúci sa na serveru kde beží FlowIDS.

-N nastavenie formy obstarávania dát na NetFlow. Momentálne sú podporované NetFlow v1 a v5. Pri detekovaní exportu mimo tieto verzie sa pakety zahadzujú a tak treba adekvátne nastaviť sieťový prvok na vhodnú verziu. (viz 1.3)

-p nastaví nám náš adaptér na promiskuitný mód (Promiscuos mode). Prednastavený je nepromiskuitný mód.

-r súbor načíta a pošle do programu súbor so zaznamenanými paketmi. Súbor je v tcpdump formáte, ktorý dokáže vytvoriť aj FlowIDS a to za pomoci -w prepínača.

-S nastavenie formy obstarávania dát na SPAN rozhranie. Táto forma sa prakticky nelíši od lokálnej, je to iba konkrétnejšie spresnenie pre program, keďže SPAN sa nastavuje na sieťových prvkoch mimo nášho servera a program ako taký funguje rovnako ako pri odchytávaní na lokálnom rozhraní.

-w súbor zápis zaznamenaných paketov do súboru. Vhodné k neskoršej analýze.

2.3.2 Konfiguračný súbor

V prvom rade, kľúčové slová v konfiguračnom súbore sú case-insensitive, tzn. že `hodnota` a `HoDnOtA` je pre program to isté. Cestu ku konfiguračnému súboru a jeho názov môžeme definovať cez príkazový riadok, pomocou prepínača *c*. Ak ju nešpecifikujeme, tak sa súbor bude hľadať v aktuálnom adresári programu pod názvom "`flowids.conf`".

Konfiguračný súbor umožňuje nakonfigurovať program komplexnejšie a pre niektoré jeho vlastnosti neexistuje ekvivalent pre príkazový riadok. Komentáre sú definované ako text začínajúci znakom '#'. Komentár končí až

na novom riadku. Niektoré kľúčové slová vyžadujú ako hodnotu čísla, iné zas reťazce. Reťazce a čísla nesmú obsahovať medzery. Zoznam kľúčových slov:

Názov	**Typ**	**Popis**
BufSize	Číslo	veľkosť BPF bufferu, do ktorého sa ukladajú odchytené pakety. Z neho priamo putujú do nášho programu. V prípade ak máte veľký počet zahodených paketov, pokúste sa túto hodnotu zvýšiť.
Daemonize	Číslo	0 pre nedeamonizovanie a 1 pre deamonizáciu programu.
DataSrc	Reťazec	definujeme zdroj, ktorý nám bude poskytovať informácie k vyhodnocovaniu. `LOCAL` pre lokálne odchytávanie, `SPAN` pre odchytávanie cez SPAN port a `NETFLOW` pre analýzu dát cez NetFlow export.
Debug	Číslo	v prípade ak je hodnota 1, program bude vypisovať obšírnejšie údaje o svojej činnosti. Na vypnutie tejto vlastnosti slúži hodnota 0.
FlowTime	Číslo	udávané v sekundách. Udáva ako dlho majú byť jednotlivé záznamy udržované. Tejto funkcii sa hovorí aj plávajúci čas. Ak daná časť flow-u (určená za pomoci TimeSeg) je staršia ako FlowTime, tak sa zmaže. V prípade, že je už celý flow starý, maže sa. Čím viac chcete mať uložených informácií v programu o dianí vo vašej sieti, tým vyššia hodnota musí byť definovaná. Prednastavená hodnota je 604800 (tj. 1 týždeň).
If	Reťazec	názov zariadenia, na ktorom budeme odchytávať pakety.
LogFile	Reťazec	súbor, do ktorého sa budú zapisovať logy. Je možné špecifikovať aj relatívnu cestu k súboru, vzhľadom k aktuálnemu adresáru, kde sa nachádza náš program.
Nf-Port	Číslo	port na ktorom bude FlowIDS očakávať NetFlow export pakety od routrov.
NmbPkt	Čislo	počet paketov, ktoré sa majú odchytiť. Po tomto počte program skončí.
PidFile	Reťazec	súbor, v ktorom je uložené PID nášho programu.
Priority	Číslo	`nice` priorita, ktorá sa má nastaviť
Promisc	Číslo	0 znamená vypnutý promiskuitní mód a 1 znamená zapnutý.
RouterAny	Čislo	Pri hodnote 1 program prijíma NetFlow pakety od ľubovoľného serveru, ktorý exportuje NetFlow. Pri 0 prijíma iba od špecifikovaných serverov zo zoznamu, ktorý sa vytvára sa pomoci kombinácii RouterIP a RouterPort.

RouterEnablePass	Reťazec	heslo pre prihlásenie do enable módu na routru, ktorý definujeme. Ak tento parameter neuvedieme tak si program bude myslieť, že daný router nemá na tejto vrstve autentifikáciu, a preto ju program bude preskakovať a pokračovať vo vykonávaní príkazov.
RouterIP	IP adresa	adresa routru, s ktorým môže náš program komunikovať. Tento parameter slúži na povolenie routra, od ktorého môžeme prijímať a vyhodnocovať NetFlow dáta ale taktiež tento router je využívaný aj pri post detekcii, kedy sa routrom odosielajú príkazy. V prípade ak chceme aby náš program získaval NetFlow dáta od ľubovoľného routra tak buďto tento parameter vôbec neuvádzame alebo pri nutnosti uvedenia môžeme prinútiť prijímať všetky pakety za pomoci voľby RouterAny. Môžeme definovať viacero routrov, kde sa každému bude prideľovať číslo a to vzhľadom k tomu, ktorý v poradí sa objavil v konfiguračnom súbore. Potom sa môžme na tieto čísla odkazovať v pravidlách (viz 2.4). Pri definície každého jedného routra je dôležité zachovať poradie, v ktorom definujete jeho vlastnosti. Poradie je: RouterIP, RouterPort, RouterUserPass, RouterEnablePass.
RouterPort	Číslo	konfiguračný port pre RouterIP, cez ktorý môžme daný router obsluhovať. Na tomto porte musí byť umiestnený telnet daemon. Za každým kľúčovým slovom RouterIP musí nasledovať táto definícia portu, ktorá sa vzťahuje k tomuto routru.
RouterUserPass	Reťazec	heslo pre prihlásenie do user módu na routru, ktorý si definujeme. Ak tento parameter neuvedieme tak si program bude myslieť, že daný router nemá na tejto vrstve autentifikáciu a pri pokuse o spojenie bude rovno vykonávať príkazy v user móde.
RuleFile	Reťazec	súbor, z ktorého sa načítajú pravidlá.
Syslog	Číslo	0 pre nelogovanie do syslogu a 1 pre logovanie doň.

TimeSeg	Číslo	udávané v sekundách. Určuje nám segmentáciu vnútornej štruktúry, ktorá si práve v každom časovom úseku definovanom za pomoci tejto premennej, agreguje príslušné dáta pre jeden flow. A tak ak chceme mať čo najviac podrobnejšie dáta pre flow, tak tento čas má byť čo najmenší, ale potom sa nám zväčšuje štruktúra. Ak nám však nevadí väčšia agregovanosť a s tým spojená väčšia nepresnosť pri zisťovaní vlastností toku v určitom čase, tak túto premennú môžme nastaviť na vyššiu hodnotu. Prednastavená hodnota je 3000 sekúnd.
TimeTrf	Číslo	časový interval udávaný v sekundách. Interval určuje ako často sa má spúšťať Traffic Accounting funkcia, ktorá zisťuje či toky neporušujú Traffic Accounting pravidlá. Čím presnejšie chceme byť informovaní o porušení Traffic Accounting pravidiel v našej sieti tak tým nižšia hodnota musí byť definovaná. Prednastavená hodnota je 3000 sekúnd.

Tabuľka 2.1: Kľúčové slová pre konfiguračný súbor

2.3.3 Príklad konfiguračného súboru

```
If=eth0
Promisc=0
Rulefile=flowids.rules
#Debug=1
datasrc=LOCAL
syslog=0
RouterAny=1
RouterIP=147.32.114.199
RouterPort=23
RouterUserPass=user_heslo
RouterEnablePass=en_heslo
flowTime=60
```

Toto nastavenie nám nakonfiguruje náš program takto:

1. nastaví hlavné rozhranie, určené na zber údajov, na eth0
2. explicitne deaktivuje promiskuitní mód

3. nastaví cestu k súboru s pravidlami
4. príklad komentára, bez znaku '#' by to znamenalo aby bol program v debug móde
5. nastaví formu odchytávania a spracovania na lokálnu
6. deaktivuje logovanie do syslog-u
7. program môže spracovať NetFlow pakety od ľubovoľného routra
8. definujeme prvý router, konkrétne jeho prvú časť a to IP
9. druhá časť definície routra, port, na ktorom sa budú vykonávať príkazy
10. tretia časť definície routra, je nepovinná, v našom prípade znamená, že pre prístup na router 1 je treba heslo `user_heslo`.
11. štvrtá časť definície routra, je nepovinná, v našom prípade znamená, že pre prístup do enable módu na routru 1 je treba heslo `en_heslo`.

2.4 Pravidlá

Pravidlá tvoria neodmysliteľnú súčasť programu keďže špecifikujú vlastnosti, ktoré má mať paket k tomu aby bol označený za podozrivý. Taktiež nám určujú čo sa po takejto detekcii ma vykonať. Pri tvorbe pravidiel si musíme uvedomiť, že každé ďalšie pridané pravidlo nám spomaľuje celý program a tak ostáva iba na vás akou mierou bude zvládať server spracovávať jednotlivé pakety. Tvar pravidiel vychádza z pravidiel definovaných v snortu. Je to hlavne kvôli uľahčeniu zápisu pravidiel pre nových užívateľov FlowIDS, ktorý už majú skúsenosť so snortom. Vďaka tomuto je prepis pravidiel zo snortu do FlowIDS triviálna záležitosť. Na jednom riadku môžeme definovať najviac jedno pravidlo. Za znakom '#' definujeme komentár, ktorý končí až na novom riadku. Pravidlo pozostáva z dvoch hlavných komponent: špecifikácia prúdu a dodatočné vlastnosti.

2.4.1 Špecifikácia prúdu

V tejto časti pravidla špecifikujeme hlavné špecifiká prúdu dát a to konkrétne protokol, zdrojovú a cieľovú IP adresu ako aj port.

`Syntax pravidiel:`

```
Protokol Zdrojová_IP Zdrojový_Port Cieľová_IP Cieľový_Port ( Ďalšie vlastnosti )
```

Ako protokol môžme špecifikovať tieto protokoly:

- icmp / icmp6
- igmp / igmp6
- tcp / tcp6
- udp / udp6
- any / any6

kde *any* znamená ľubovoľný protokol. Pre každý typ protokolu existujú dve varianty, kde jedna má naviac číslo '6'. Toto číslo reprezentuje verziu IP, na ktorú sa to vzťahuje, v tomto prípade je to IP verzia 6. Pochopiteľne protokol, ktorý neobsahuje '6' reprezentuje IP verziu 4. Napríklad 'tcp' reprezentuje protokol nad IPv4 a 'tcp6' nad IPv6. Takže IPv4 a IPv6 sú v týchto pravidlách separátne prúdy dát. V prípade ak chcete definovať pre jedno pravidlo viac ako jeden protokol a pritom nie všetky cez kľúčové slovo *any* tak je treba definovať pre každý protokol zvlášť pravidlo, tj v jednom pravidle nie je možné zreťazovať protokoly, definovať viac ako jeden. Kľúčové slovo *any* môže byť troška zavádzajúce keďže toto kľúčové slovo zahrňuje v sebe iba detekciu IPv4 paketov, pre IPv6 pakety slúži *any6*, ktoré je opäť iba pre verziu IPv6.

IP adresy môžeme zadávať buď v bežnom formátu alebo v CIDR zápisu ak chcem definovať masku siete. Pri IP adresách môžeme využiť operátor negácie '!', za ktorým musí hneď nasledovať samotná IP adresa a ta môže byť opäť zapísaná v jednom z dvoch ponúkaných formátov. Tzn. že môžeme vytvoriť pravidlo typu, všetko čo nie je naša sieť a ide do našej siete:

```
any !192.168.1.1/24 any 192.168.1.1/24 any (...)
```

V prípadne ak chceme aby pravidlo platilo pre ľubovoľnú IP adresu, aplikujeme kľúčové slovo '`any`'. Toto kľúčové slovo je rovnaké jak pre IPv4 tak aj pre IPv6 adresy, nakoľko otázku týchto dvoch verzií rieši prvý parameter pravidla.

Ďalšou z vecí, ktoré si môžeme definovať sú porty. Tie môžeme zadať buďto ako jeden, alebo ako rozsah. Opäť máme k dispozícií operátor negácie '!', ktorý môžeme aplikovať aj na rozsah. Rozsah sa zapisuje vo formátu:

```
x:y
```

Kde x a y sú hraničné čísla portov a platí x¡=y. Hraničné porty sú taktiež súčasťou rozsahu. Pri negácií hľadáme porty mimo daný rozsah (port). Za operátorom negácie musí nasledovať rozsah alebo samotný port, nesmie byť medzi nimi žiadna medzera. Opäť ako pri IP adresách aj tu môžeme definovať celý možný rozsah a to buď pomocou kľúčového slova '`any`' alebo taktiež jednoduchým definovaným celého rozsahu `0:65535`.

2.4.2 Dodatočné vlastnosti

K tomu aby sme bližšie špecifikovali dodatočné vlastnosti paketu ako aj toku dát ako takého slúži druhá časť, ktorá sa snaží za pomoci ďalších dodatočných kľúčových slov dodefinovať vlastnosti, ktoré majú označovať podozrivé pakety. V tejto časti definujeme aj post detekčné akcie. Tvar pravidiel vyzerá nasledovne:

Kľúčové_slovo: hodnota;

Dodatočné vlastnosti musia byť uvedené v rámci jedných zátvoriek a zátvorka '(' musí byť oddelená od špecifikácie prúdu dát aspoň jednou medzerou. Reťazec môže obsahovať aj medzery ale v tom prípade musí byť uvedená v úvodzovkách. V textu, ktorý definujeme v `msg` alebo aj v súbore, ktorý sa má posielať routru, sa môžu použiť formátovacie reťazce `\n \t`. Tento text však nesmie obsahovať znak ':'.Kľúčové slová, tak ako v prípade konfiguračného súboru, sú case-insensitive. Nasleduje zoznam kľúčových slov a ich význam

Ack Hľadá konkrétne ACK číslo pri TCP paketoch.

Acl Súbor s príkazmi, ktoré sa majú poslať routru v prípade zhody. Súbor môže obsahovať flow premenné (viz. 2.5.1). Tento súbor musí existovať už pri samotnom spustený a pri jeho zmene musíme reštartovať program. K tomu aby sme vedeli, ktorému routru pošleme príkazy, musíme definovať parameter `Router`.

Alert Úroveň, ktorá určuje vážnosť log správy. Úrovne sa označujú číslami tak ako ich označuje syslog. Preto je táto vlastnosť ideálna pri logovaní do syslogu. Jednotlivé stupne sú:

1 = LOG_DEBUG Správy na debug úrovni

2 = LOG_INFO Správy určené k informovaniu o udalosti

3 = LOG_NOTICE Normálny stav ale nastala podstatná vec

4 = LOG_WARNING Varovanie

5 = LOG_ERR Chyba

6 = LOG_CRIT Kritická chyba

7 = LOG_ALERT Opatrenia musia byť vykonané čo najskôr

8 = LOG_EMERG Systém je nepoužiteľný

Conn Maximálny počet spojení, ktoré môže byť inicializované v danom časovom úseku.

Conn-time Časový úsek pre splnenie požiadavky maximálneho počtu spojení. Udáva sa v sekundách. V prípade, že tento parameter neuvedieme, tak sa bude brať nekonečno.

Content Definujeme obsah paketu. Ten môže byť binárny ale aj znakový. Pre písanie binárnych častí paketu uvedieme túto časť v '|'. Táto binárna a znaková časť sa môže ľubovoľne krát striedať. V binárnej časti oddeľujeme každé dva znaky medzerou. Príklad:

```
content: "Ahoj| 00 AF 01 |, Peto!| AA BB |".
```

Medzi textom a znakom '|' nie je medzera, jedine v prípade ak je aj ona sama hľadaná.

Depth K tomuto parametru musíme mať definovanú vlastnosť `content`. Určuje vzdialenosť, od počiatočného bodu hľadania, do ktorej má pokračovať v hľadaní.

Dsize Veľkosť dátovej časti paketu. Platí vždy len pre jeden paket a nenapočítava sa to.

Email Email, na ktorý sa v prípade úspešnej detekcie pošle správa uvedená v msg časti.

Flags TCP flagy, ktoré musia byť v paketu. Definujeme ich za pomoci týchto znakov: `F, S, , R, P, A, U, 2, 1`. Môžeme uviesť viac znakov za sebou. V tom prípade nesmú byť oddelené medzerami. Ak bude paket obsahovať niektoré flagy naviac, ale požadované sa v ňom vyskytujú tak, aj napriek tomu je táto vlastnosť splnená.

ID ID číslo IP paketu, ktorý má byť označený ako nevhodný.

Icode Ak je to ICMP paket tak overí či sa zhoduje jeho ICMP kód. Je možné použiť operátor negácie `!`, za ktorým musí nasledovať číslo.

Itype Ak je to ICMP paket tak overí či sa zhoduje jeho ICMP typ. Je možné použiť operátor negácie `!`, za ktorým musí nasledovať číslo.

Logto Súbor, do ktorého sa budú zaznamenávať všetky správy úspešne detekovaného toku dát na základe pravidla, kde túto možnosť špecifikujeme. Konkrétne sa tam zapíše správa definovaná cez parameter `msg`, ktorý môže obsahovať flow premenné (viz 2.5.1).

Minfrag Aká môže byť minimálna veľkosť fragmentovaného paketu.

Msg Správa, ktorá sa zapisuje do log súboru, prípadne sa posiela na email. Môže obsahovať flow premenné (viz 2.5.1).

Nmb_packets Počet paketov, ktoré sa môžu v danom toku dát preniesť za určitý čas.

Nmb_packets-time Čas, v ktorom sa meria počet prenesených paketov. Udáva sa v sekundách. V prípade, že nie je definovaný, berie sa nekonečno.

Offset K tomuto parametru musíme mať definovanú vlastnosť `content`. Určuje od akej vzdialenosti v dátovej časti paketu má program začať hľadať požadovaný reťazec.

Router Poradové číslo routra, ktorý je definovaný v konfiguračnom súbore (viz 2.3.2). Musí byť špecifikovaný v prípade, že chcem posielať príkazy routru.

Seq SEQ číslo v TCP paketu.

Size Maximálna veľkosť dátového toku za určitý čas. V tejto veľkosti sú zahrnuté aj veľkosti hlavičiek paketov.

Size-time Čas, v ktorom sa meria veľkosť dátového toku. Udáva sa v sekundách. V prípade, že nie je definovaný je to nekonečno.

TTL TTL hodnota, ktorú mať paket, ktorý je označený ako nevhodný. Platí aj pre IPv6 pakety kde to je pomenované ako hop-limit.

Táto sekcia označovaná ako dodefinovacia musí obsahovať aspoň jednu dodefinovaciu vlastnosť. V tejto časti sú jednotlivé vlastnosti oddelené ';' a v rámci vlastnosti je kľúč oddelený od hodnoty znakom ':'. Niektoré kľúčové slová majú charakter Traffic Analysing-u, tj analýza jednotlivých dát ako takých, ale taktiež sa tam vyskytujú kľúčové slová definujúce Traffic Accounting, tj napočítavanie toku dát a udržiavanie ich hodnôt v rámci programu.

Niektoré kľúčové slová sú však neutrálne a to `acl, alert, email, logto, msg, router`. Pod Traffic Accounting spadajú tieto kľúčové slová: `size, size-time, conn, conn-time, nmb_packets, nmb_packets-time`. Zvyšné pravidlá sú určené pre Traffic Analysing.

2.4.3 Príklad súboru s pravidlami

V tejto časti by som rád ukázal, príklad možného nastavenia FlowIDS, tak aby zvládalo rutinné operácie. Avšak tento príklad je len ukážkou základných možností a pri požiadavkách na komplexnejšie pravidlá užívateľ môže z nich vychádzať, ale nemusí sa nimi obmedzovať.

```
tcp any any 10.0.0.1 22 (logto:ssh_transfer.log;msg:"SSHtrf over 200MB, IP %h \n";size:200000;)
any any any 10.0.0.1 any (logto:ftp.log;msg:"%H get over 10MB in 10 sec \n";size:10000;size-time:10;)
udp any any 10.0.0.1 53 (logto:dns.log;msg:"DNS slam  \n ";nmb_packets:100;nmb_packets-time:100;)
tcp any any any any any  (logto:tcp.log;seq:31337; ack:31337; msg:"31337??IP %h \n";flags:S;)
any 10.0.0.1/24 any !10.0.0.1/24 1000:2000 (logto:p2p.log;msg:"P2P IP %H \n";email:bondoer@urtax;)
any 10.0.0.1/24 any 10.0.0.1/24 any (logto:siet.log;offset:5;depth:20;content:"/bin/sh";msg:"RmtExec, IP %h \n";)
any !10.0.0.1   any 10.0.0.1  any (logto:portscan.log;msg:"Portscan>IP %h \n";conn:100;conn-time:5;acl:send.cisco;router:1;)
```

Pravidlá sú jak z Traffic accounting oblasti tak aj z Traffic analysing. Rozoberieme si ich postupne.

1. každý paket, ktorý má na tretej vrstve protokol TCP a cieľovou adresou je 10.0.0.1, port 22, sa bude napočítavať a ako náhle prekročí 200MB limit, tak sa zapíše text uvedený v msg do súboru ssh_transfer.log, pritom nahradí %h za zdrojovú IP adresu.

2. každý paket, ktorý je smerovaný cez ľubovoľný prokotol (nad IPv4) a jeho cieľ je IP adresa 10.0.0.1, bude zaznamenaný a ak v čase 10 sekúnd prenesie viac ako 10MB tak sa zaznamená správa s textom msg do ftp.log súboru.

3. tento tok sa zaznamená v prípade, ak za 100 sekúnd príde z ľubovoľného počítača do počítača s IP adresou 10.0.0.1 na port 53 viac ako 100 paketov.

4. typické pravidlo na analýzu, ktoré berie priamo pakety a rovno vyhodnotí, bez nutnosti čakať, konkrétne v tomto prípade nájde taký paket, ktorého seq a ack číslo je rovné 31337 a paket má nastavený aspoň SYN flag, avšak toto sa musí odohrávať na TCP protokole.

5. prvé pravidlo kde využívame masku siete a definujeme tým viacero IP adries, pre ktoré toto pravidlo platí. Pri cieľovej adrese sme opäť zadefinovali rozsah ale s tým rozdielom, že sme pred neho umiestnili '!', tj že sme ho znegovali. V praxi to znamená, že chceme nájsť všetok dátový tok pochádzajúci mimo našej siete a smerujúci do nasej na

určitý rozsah portov. Tým rozsahom môžu byť definované napríklad Peer to Peer služby, ktoré sú zväčša nežiaduce. Logovanie neprebieha v tomto prípade iba do súboru ale taktiež aj vo forme odoslania emailu na našu adresu.

6. pravidlo je typické pre monitorovanie našej vlastnej siete a prenosu v rámci nej. nakoľko berie v úvahu iba pakety, ktoré pochádzajú a smerujú do nej. Mimo to, pakety musia mať vo svojom tele ešte reťazec "/bin/sh", a FlowIDS bude tento reťazec hľadať od 5 pozície (počíta sa od nuly!) až kým neprejde 20 znakov na pozíciu 25. V prípade ak na tomto mieste nájde našu vzorku tak sa zaznamená táto udalosť

7. za pomoci posledného pravidla si môžeme ľahko postaviť systém, ktorý bude detekovať či nás niekto nescanuje, v prípade, že dostaneme viac ako 100 konexií v priebehu 5 sekúnd, tak sa daná správa zaznamená do portscan.log a tiež sa pošlú príkazy uvedené v send.cisco súbore routru číslo jeden. Tento router sme si museli predtým definovať v konfiguračnom súbore, ináč toto volanie zlyhá.

2.5 Post-detekcia

Ako som už naznačil v sekcií Pravidlá, po detekovaní podozrivého paketu je možné vykonať protiopatrenia. Tieto protiopatrenia sa priamo definujú v pravidlách a to za pomoci kľúčových slov `acl, alert, email, logto, msg, router`. Ich základný popis bol spomenutý v sekcií 2.4. Jednotlivé protiopatrenia sa dajú kombinovať a tak nám vznikne komplexný IPS (Intrusion Prevention System) schopný ubrániť sa alebo adekvátne reagovať na takmer akýkoľvek útok alebo správcom nepovolený dátový tok.

Užívateľ má po úspešnej detekcii nevhodného dátového toku na výber z niekoľkých možností protiopatrení, ktoré siahajú od žiadnej, cez notifikáciu na email alebo zápis správy do log súboru, až po zakázanie príslušného toku na konkrétnom routri. Vďaka kombinácií sa priblížime k stavu, kde sa o našu sieť tak povediac stará program sám a my sme o tom všetko informovaní cez emaily. Avšak aj v takýchto prípadoch je nutný občasný zásah administrátora, keďže tieto techniky sa dajú jednoducho zneužiť a útočník ich môže využívať proti nám. Hlavným pilierom tejto časti pravidiel bude asi bezpochyby `logto, msg, email`. Za pomoci `logto` špecifikujeme kam sa majú prípadné logy ukladať, tj. nie je nutné mať všetky záznamy a poplašné varovania v jednom log súbore ale môžeme si ich vhodne rozdeliť, podľa jednotlivých pravidiel. Pri nešpecifikovaní tohto parametru bude program automaticky vypisovať varovanie do konzoly (konkrétne do stdout). Vďaka definovaniu

Identifikátor	Popis
%h	Zdrojová IP adresa
%H	Cieľová IP adresa
%p	Zdrojové číslo portu
%P	Cieľové číslo portu
%T	Protokol
%h	Zdrojová IP adresa
%i	Vstupné rozhranie
%f	Verzia IP protokolu

Tabuľka 2.2: Flow premenné a ich význam

vlastnosti `email` povieme programu, aby nám pri pozitívnej zhode rovno poslal email s varovnou správou. Obe zo spomínaných vlastností využívajú atribútu `msg`, v ktorom si definujeme aký text sa má uložiť do špecifikovaného log súboru, respektíve aký text sa má poslať emailom. V tomto texte sa môžu vyskytovať tzv. Flow premenné (viz. 2.5.1). Všetky tieto vlastnosti môžu mimo Flow premenných obsahovať taktiež operátory formátovania: `\n \t` kde prvý operátor nám odriadkuje a druhý vloží tabulátor.

2.5.1 Flow premenné

V textu, s ktorým pracujeme v našom programe je možné špecifikovať tzv. flow premenné. Text, ktorý môže obsahovať tieto premenné je buďto definovaný v "msg" alebo v "acl" časti pravidiel. Flow premenné, sú reťazce, ktoré sa nahradzujú parametrami paketu, ktorý bol označený ako nevhodný za pomoci detekčných funkcií. Premenná je definovaná ako reťazec začínajúci znakom '%'. Ak však chceme využiť tento znak na iné účely ako na definovanie premennej, čiže chceme zrušiť jeho význam, tak stačí napísať "%%". Tento reťazec sa nám nahradí jedným takým znakom. Flow premenné sú uvedené v tabuľke 2.2: Po napísaný niektorej z tých premenných do svojho textu, tak tá premenná sa nahradí reťazcom podľa typu, ktorý je špecifikovaný v tabuľke.

2.6 Používanie programu

Program sa v prvom rade nastaví za pomoci konfiguračného súboru a potom dodatočne za pomoci prepínačov definovaných pri spustený. Po tomto

spustený sa nahrajú pravidla do pamäti, takže pri zmene pravidla je nutné program reštartovať. Taktiež môžeme pri spustení definovať filter pravidlo, ktoré nám vo väčšine prípadov zvýšiť výkon nášho systému a to tak, že pakety, ktoré nespĺňajú podmienky určené v tomto filtru sa zahadzujú a nedostanú sa do nášho programu, viac informácií o tom ako tieto pravidlá zapisovať nájdete na [4]. NetFlow modul ako taký nepodporuje IPv6, vzhľadom k tomu, že NetFlow verzie 1 a 5 ho nepodporujú tiež. Pri potrebe obšírnejších údajov o činnosti programu musíme program nastaviť do debug módu, v ktorom sa zaznamenáva každá informácia. Avšak debug mód slúži skôr na odhalenie prípadných chýb v pravidlách alebo na zistenie čo konkrétne náš program spracováva a preto by program nemal byť bežne spustený v tomto režime, keďže tieto výpisy sú natoľko rozsiahle, že pri dostatočnom veľkom dátovom toku mohli prísť o naše zdroje (CPU, Pamäť). Ak sa detekuje niečo čo je označené za pomoci pravidiel ako nevhodné, tak sa vykonajú patričné protiopatrenia a program po ich aplikácií pokračuje ďalej v behu. Výsledky počínania programu môžme vidieť v `logto` súboroch alebo v našich mailových schránkach. Program počas behu zaznamenáva všetok traffic do svojej vnútornej štruktúry, ktorá je segmentovaná za pomoci premennej TimeSeg a všetky dáta, ktoré sú staršie ako FlowTime maže. Premazávacia funkcia sa spúšťa každých TimeSeg sekúnd, a tak je vhodné túto premennú nastaviť vhodne vzhľadom k FlowTime. Po ukončení nám program vypíše štatistiku odchytených paketov a stratených paketov.

Kapitola 3

Programátorská dokumentácia

V tejto kapitole by som postupne analyzovať jednotlivé súbory, v ktorých sú začlenené moduly. Celý program je postavený na knižnici libpcap, ktorá je určená k vývoji aplikácií odchytávajúcich dátový tok. Táto knižnica je nielen multiplatformová, ale má aj nesmierne množstvo užitočných funkcií, ako je filter na pakety, vďaka nemu nie je nutné prijímať každý paket, ale môžeme si vyberať, taktiež má rozhranie na ukladanie a čítanie dátových tokov a popritom je dostatočne rýchla, aby nás nelimitovala aj v tých najvyťaženejších sieťach. Pri nasadení do vyťažených sietí je najväčším problémom vyvolávanie prerušení, ktoré nás limitujú. Avšak jednou z jej nevýhod je nedefragmentácia paketov. Keďže sa táto knižnica snaží byť čo "najsurovejšia" tak dostávame každý paket tak, ako ho dostane systém, čiže aj s hlavičkou a nedefragmentovaný. Je len na nás, aby sme ho defragmentovali. Ja som sa v mojej práci rozhodol defragmentáciu vynechať a to z výkonnostných dôvodov a vzhľadom k náročnosti implementovať danú vlastnosť.

3.1 main.c

Hlavný súbor, v ktorom sa nachádza vstupná funkcia `main()`, v ktorej sa spracovávajú vstupné argumenty a na základe nich sa rozhoduje. V prípade ak sa nedefinuje v argumentoch programu konfiguračný súbor, tak sa zavolá funkcia `LoadConf()` s parametrom "`flowids.conf`", tj bude sa hľadať konfiguračný súbor flowids.conf v aktuálnom adresári. V prípade log súboru, je to obdobné až na to, že pri nedefinovaný log súboru sa budú záznamy vypisovať do stderr. Pre súbor s pravidlami nie je definovaná žiadna prednastavená hodnota, a preto pri ak tento parameter nedefinujeme ani v argumentoch a ani v príslušnom konfiguračnom súbore, tak program skončí. To platí aj pre zle definované hodnoty, ako cesta k súborom a pod. Jedinou výnimkou je sieťové

rozhranie, ktoré ak zle definujeme, tak sa program pokúša nájsť ľubovoľné rozhranie a to za pomoci libpcap funkcie `pcap_lookupdev()`. K inicializácií pcap zariadenia slúži funkcia `init_pcap()`. V tejto funkcii sa mimo hľadania zariadenia, inicializuje prípadný súbor špecifikujúci dátový tok, ktorý sme si uložili. Táto funkcia sa programu aktivuje za pomoci parametru -w resp. -r a samozrejme špecifikácie príslušného súboru.

K nastaveniu signálov slúži funkcia `handle_signals()`, v ktorej si nastavíme funkcie, ktoré bude obstarávať signál SIGINT,SIGTERM, SIGHUP. Pre tieto signály nastavujeme obstarávaciu funkciu `Exitus()`, ktorá zavolá `Exit()`. Funkcia `Exit()` je určená na opustenie programu a to z ľubovoľnej pozície. Táto funkcia prijíma niektorý z vnútorných exit kódov na základe, ktorých sa ešte rozhoduje či zaradí nejaký dodatočný výstup a potom sa spustí funkcia `CleanUp()`, ktorá dealokuje všetky naalokované štruktúry a polia. Taktiež zavrie všetky súbory a zavolá knižnicovú funkciu `exit()` s príslušným error kódom, ktorý dostala funkcia Exit ako argument.

Pre zistenie či dané pravidlo neporušuje Traffic Accounting pravidlá slúži funkcia `TraffAcc()`. Táto funkcia slúži k počítaniu a vyhodnocovaniu toku ako takého a nielen jeho aktuálnych elementov (paketov). Táto funkcia sa volá každých iTimeTrf sekúnd. Túto hodnotu môžeme definovať iba cez konfiguračný súbor.

Poslednou funkciou definovanou v tomto súbore je funkcia `Daemonize()` a s ňou spojené funkcie `PidGet()`,`PidWrite()`,`PidCheck()` a `PidRemove()`, ktoré slúžia na prácu s PID súborom. Funkcia `Daemonize()` nám, v prípade ak si to užívateľ zvolí, daemonizuje náš program, tj odpojí ho od kontrolného terminálu, nastaví jeho pracovný adresár na "/" a stdout, stdin, stderr na "/dev/null". Následne program zavolá funkciu fork(), nastaví umask na 027 a uloží svoj PID do PID súboru. Ak sa však na začiatku tohto procesu zistí, že už existuje PID súbor pre náš program, tak sa ukončí. Čiže v prípade, že budeme chcieť spustiť viacero inštancií nášho programu, tak nám neostáva nič iné než si vytvoriť dve konfiguračné súbory, ktoré sa budú líšiť aspoň v názve PID súboru.

3.2 main.h

Tento hlavičkový súbor obsahuje kostru celého programu, ktorá v sebe drží všetky údaje o aktuálnej konfigurácií programu ako aj informácie o priebehu niektorých procesov. Táto štruktúra sa volá IDSSetting a je definovaná takto:

```
struct IDSSetting
{
  struct bpf_program Filter; /*filter pre pcap */
  pcap_t* sniff;     /* sniffing zariadenie */
  char* szInterface;  /* rozhranie, z ktorého získavame dáta */
  char* szLogFile;  /* Náš Log súbor*/
  char* szRuleFile; /* Náš súbor s pravidlami */
  char* szConfigFile; /* Náš Konfiguračný súbor */
  char* szPidFile;  /* PID súbor */

  int iPromisc; /* 0=Non-promiscuous; 1=Promiscuos */
  int iPacketCnt; /* Koľko paketov máme odchytiť. 0 = nekonečno */
  int iSyslog;  /* Chceme logovať do syslogd? 1= ÁNO; 0=NIE*/
  int iDaemonize; /* Má sa FlowIDS spustiť ako daemon? */

  int iNfPort;  /* Port, na ktorom budeme dostávať pakety od nášho routru */

  int dataSrc;  /* Odkiaľ dostaneme dáta 0=Local,1=Span,2=Netflow */
  int iPktGetMthd;  /* LIBPCAP odchytávanie*/
  int iLogfd;   /* Deskriptor log súboru*/
  int   iRulefd;  /* Deskriptor súboru s pravidlami*/
  int   iRulefpos;  /* Pozícia v súbore s pravidlami */
  int iDebug;   /* Debug mód? */
  int iDatalink;  /* Rozhranie */
  int lostPacketCnt;  /* Koľko sme stratili paketov pri NetFlow */
  char* save_filename; /* súbor na uloženie zaznamenaných dát */
  char* read_filename;  /* súbor na načítanie zaznamenaných dát */

  int iFilePos;   /* riadok na ktorom sme v súbore s pravidlami */
  int iLinePos; /* znak na ktorom sme v súbore s pravidlami */
  pthread_t thread; /* Thread ID */
  struct Router *routers; /* Routre */
  int   iRoutersIP;   /* Počet Routrov */

  time_t  iTimeSeg; /* Časová segmentácia vo flow */
  time_t  iTimeTrf; /* Interval v ktorom bude púšťaná TrafficAcc() funkcia */
  time_t  iFlowTime;  /* Čas reprezentujúci, kedy sú dané dáta staré a mali by byť
      zmazané */
  time_t  lastCas;  /* Čas kedy nastala posledná kontrola Traffic Acc pravidiel */
};
```

Výpis kódu 3.1: Štruktúra IDSSetting

V tejto štruktúre sú uložené všetky potrebné premenné, ktoré musia byť viditeľné v každej časti programu za každého okamžiku, ináč povedané, ktoré majú mať vlastnosti globálnych premien. Na tomto mieste nebudem vysvetľovať význam jednotlivých premien, ale v prípade ak vám nie je jasná funkcia tej ktorej premennej, tak ak je dôležitá tak sa bude v ďalších častiach dodatočne definovať a vysvetľovať jej význam. Táto štruktúra sa resetuje na prednastavené hodnoty v `initStruct()` funkcii definovanej v `main.c`.

Ďalej v tomto súbore sú definované všetky hlavičky protokolov, ktoré tento program podporuje(až na NetFlow protokoly tie sú definované v súboru cisco.h, 3.9). Nachádzajú sa tu definície týchto hlavičiek: ethernet, ipv4, ipv6, tcp, udp, icmp, icmp6, igmp. Mimo to definujeme štruktúru curr_packet, do ktorej si ukladáme rozparsované hlavičky aktuálneho paketu.

3.3 ConfParser

Nakoľko má konfiguračný súbor jednoduchý zápis, tak aj tento súbor relatívne jednoduchý. Sú v ňom definované dve funkcie. Prvou z nich, ktorú si rozoberieme je `LoadConf()`. Táto funkcia prijíma ako argument názov kon-

figuračného súboru. V prípade, ak žiadny konfiguračný súbor nemáte alebo sme uviedli zlú cestu, tak táto funkcia zlyhá a program sa ukončí. Ďalej nasleduje samotné parsovanie, kde sa ignorujú medzery a prípadne všetky znaky za '#' až do konca riadku. Ako náhle obdržíme prvé slovo tak čakáme na znak '='. V prípade ak sa tam nenachádza tak sa program ukončí s chybou v konfiguračnom súbore. V opačnom prípade už len budeme čakať na hodnotu danej premennej, po jej obdržaní zavoláme druhú z funkcií, ktorá je v tomto súbore a to funkciu `InsertValue()`, ktorej dáme príslušný názov premennej a jej hodnotu a ona si ju uloží v globálnej štruktúre IDSSetting, ktorá je definovaná v `main.c` (3.1). Ak by sme chceli pridať nejaký ďalší parameter do vášho konfiguračného skriptu, tak stačí len pridať príslušný kód do funkcie `InsertValue()`, ktorý pri zhode mien premennej na vstupe a vašej, si danú hodnotu niekam uloží.

3.4 RulesParser

Tento súbor je jedným z najobsiahlejších a to vďaka tomu, že užívateľ ma na výber z radu možností ako pravidla definovať ako aj počet kľúčových slov a ich hodnôt je vysoký. Hlavnou funkciou, ktorá sa volá ako prvá je InitRuleParser(). Táto funkcia nám otvorí súbor s pravidlami a uloží si jeho deskriptor do štruktúry IDSSetting. Po tomto zavolaní už môžeme zavolať funkciu ParseRules().Tá si ako prvé alokuje potrebné štruktúry a to štruktúru pre uloženie IP adries(v4 alebo v6), portov a ďalej troch netriviálnych štruktúr a to moreOpt, postDetection a traffAcc. Každá z týchto štruktúr je deklarovaná v `RulesParser.h`. Štruktúra moreOpt je určená k bližšej špecifikácii paketu a to za pomoci premenných, ktoré reprezentujú hodnoty ako ACK a SEQ TCP paketu, flagy, ale taktiež obsah daného paketu a pod. V štruktúre postDetection sú definované post detekčné údaje ako správa, ktorá sa má v súvislosti s úspešnou zhodou pravidla s paketom poslať. Ďalej sa tam nachádza email, na ktorý sa má príslušná správa poslať, ako aj súbor, do ktorého sa majú zaznamenať tieto správy. V neposlednej rade tu môžeme nájsť súbor, ktorý sa má poslať sieťovému prvku. Poslednou štruktúrou, ktorú nám ostáva zadefinovať v súvislosti s pravidlami je štruktúra traffAcc. Táto štruktúra je úzko spätá s Traffic Accountingom, čiže evidenciou prenosu. Táto evidencia je vhodná najmä v sieťach, kde sú obmedzenia na prenos, ale nielen to. Vďaka tejto funkcii sa dajú detekovať vírusy, keďže tie sú špecifické veľkou snahou o rozšírenie a tým pádom veľkým počtom inicializovaných spojení. Definujeme si tu napríklad veľkosť prúdu ako aj čas, za ktorý sa bude táto veľkosť počítať. Ďalej tu definujeme počet paketov prenesených za určitý čas a posledná premenná nám udáva počet spojení za určitý čas.

Po úspešnom alokovaní a obstaraní riadku zo súboru pravidiel za pomoci funkcie `GetLine()`, začneme postupne parsovať príslušný riadok a to obstaraním si prvého tokenu funkciou `NextToken()`. Po obstaraní tokenu sa na základe premennej "part" zistí, ktorá časť má nasledovať a podľa toho sa zistí či daný token je validný pre našu časť. Takto sa postupne zaplnia štruktúry až dôjdeme k časti INSERT, kde sa tieto štruktúry uložia do štruktúry s pravidlami.

Základným pilierom štruktúry pre ukladanie pravidiel je štruktúra RulesParser:

```
struct RulesParser
{
  struct Rules* AnaHead;
  struct Rules* AccHead;

  int iActive;  /* 0-UNDEF; 1-Iba Analyzovanie; 2-Iba Accounting; 3-Oboje */
  int iRulesCnt;  /* využívame ju v TRAFFIC_ACCOUNTING */
};
```

Výpis kódu 3.2: Štruktúra RulesParser

V tejto štruktúre rozdeľujeme pravidlá na dve časti a to na základe ich definícií. Ak má pravidlo časť kde sa definuje čo by len jedno z accounting pravidiel definovaných v traffAcc, tak sa štruktúra zaraďuje do accounting pravidiel, ináč sa nachádza v Analyzing pravidlách. Týmto sme vzali v úvahu fakt, že pravidiel definujúcich čisto iba accounting časť bude málo, a preto nie je nevyhnutné tvoriť skupinu, ktorá by definovala pravidlá, ktoré majú obe časti naraz.

Jednotlivé pravidlá sú definované takto:

```
struct Rules
{
  struct Proto     *Prot;
  struct rulIP   *srcIPv4, *dstIPv4;
  struct rulIP6    *srcIPv6, *dstIPv6;
  struct rulPort   *srcPort, *dstPort;
  struct moreOpt   *options;
  struct postDetection *postDet;
  struct traffAcc *trf_acc;
  int iRuleID;

  struct Rules *next, *prev;
};
```

Výpis kódu 3.3: Štruktúra Rules

Pre každé pole, ktoré je definované syntaxou pravidiel (viz 2.4) je definovaná zvlášť štruktúra udržujúca informácie o vlastnostiach toho konkrétneho poľa. Prvým z polí je pole protokol:

```
struct Proto
{
  int iFamily;
  int iProtocol;
  int isAny;
};
```

Výpis kódu 3.4: Štruktúra Proto

Ako vidieť, práve tu je definovaná ?rodina?, ktorá nám určí, ktorú verziu IP protokolu využívame. Ďalej tu je samotný protokol a ako posledná je tu premenná obsahujúca informáciu o tom či bol definovaný špeciálny protokol `any`, ktorý znamená ľubovoľný protokol. Táto premenná je obsiahnutá v každej štruktúre kde je možné definovať kľúčové slovo `any`.

Za protokolom nasleduje definícia IP adresy a portu

```
struct rulIP
{
  struct in_addr IPAddr;
  struct in_addr Netmask;
  int isAny;
  int isNegation;
};
```

Výpis kódu 3.5: Štruktúraída rulIP

Vzhľadom k tomu že IP adresa môže byť vo verzii 4 alebo 6 tak existujú dva verzie štruktúry rulIP. Prvá definuje IPv4 a volá sa rulIP a druhá, ktorá definuje IPv6 sa volá rulIP6. Tieto dve štruktúry sa líšia iba typom IPAddr a Netmask, kde miesto pre IPv4 definovanej štruktúry struct in_addr, IPv6 obsahuje štruktúru struct in6_addr. Je možné špecifikovať aj netmasku, ktorá sa skonvertuje CIDR zápisu do bežnej IP adresy, reprezentujúcej masku siete. Taktiež tu je opäť definovaná premenná isAny, ktorá má rovnaký význam ako v štruktúre protokolu. Obdobnou premennou je premenná isNegation, ktorá obsahuje informáciu či sa daná hodnota má negovať.

```
struct rulPort
{
  u_short loport, hiport;
  int isAny;
  int isNegation;
};
```

Výpis kódu 3.6: Štruktúra rulPort

Pre port platí, že v prípade definovania exaktného portu, tj. žiadneho rozsahu, tak premenné loport a hiport obsahujú totožné čísla portov.

Poslednými štruktúrami, ktoré sa vyskytujú v tejto časti sú postDetection, moreOpt a traffAcc, ktoré sa napĺňajú informáciami špecifikovanými v časti v zátvorkách.

Štruktúry sú po vyplnení a skontrolovaní poslané funkcii `InsertRule()`, ktorá uloží toto pravidlo do globálnej štruktúry. Funkcia je v niekoľkých variantách, a to pre vkladanie pravidla, ktoré obsahuje IPv6 adresy (InsertRule6), pravidiel, ktoré sa majú začleniť do AccHead, čiže do accounting časti (InsertRuleTrfAcc) a obdobne definovanej funkcii pre IPv6 (InsertRuleTrfAcc6).

3.5 log

Táto časť programu sa stará o logovanie. Hlavnou funkciou v tejto časti je `LogMsg()`, ktorá prijíma ako prvý parameter stupeň logovania. Tie sú odvodené od stupňov definovaných v syslog-u. Jednotlivé stupne sú:

LOG_NOTIME Správa neobsahuje časové pole.

LOG_DEBUG Správy na debug úrovni

LOG_INFO Správy určené k informovaniu o udalosti

LOG_NOTICE Normálny stav, ale nastala podstatná vec

LOG_WARNING Varovanie

LOG_ERR Chyba

LOG_CRIT Kritická chyba

LOG_ALERT Opatrenia musia byť vykonané čo najskôr

LOG_EMERG Systém je nepoužiteľný

Tieto stupne nemajú v rámci programu momentálne takmer žiadny význam. Jeho hlavne uplatnenie je v prípade, že logujeme do syslog-u. V tomto prípade sa patričná správa zaloguje pod zadaným stupňom. Tento stupeň si môžeme pri pravidlách definovať za pomoci kľúčového slova `Alert`. Druhým parametrom je deskriptor súboru, do ktorého chceme zapisovať. Tu môžeme využiť aj deskriptory 0,1,2, ktoré sú v systéme UNIX priradené stdout, stdin, stderr. Za týmto nasleduje samotná správa, ktorá môže obsahovať aj premenné, a to vďaka tomu, že využívame premenný počet argumentov, ktoré sa potom substituujú v tejto správe.

Ďalšou funkciou, ktorá sa definuje v tejto časti programu, je `InitLog()`. Funkcia vytvára deskriptor na hlavný log súbor a otvorí ho pre čítanie a v prípade ak neexistuje ho vytvorí, v opačnom bude pridávať na koniec súboru. Tento súbor ostáva otvorený počas celej doby behu programu.

3.6 process

Dalo by sa povedať, že práve tento súbor je hlavným súborom pre odchytávanie paketov na lokálnom rozhraní. Najmä vďaka tomu, že práve on jednotlivé pakety rozoznáva a analyzuje a napĺňa štruktúru, ktorá reprezentuje

každý odchytený paket. K tomu, aby sme mohli daný paket spracovať potrebujeme vedieť ako sú jednotlivé pakety a protokoly definované. K tomu nám poslúži RFC. Napríklad pre IPv4 [3] a IPv6 [1]. Tieto štruktúry sú definované v `main.h`. K tomu aby sa jednotlivé časti paketu správne spracovali slúžia funkcie `processEthernet()`, `processIPv4()`, `processIPv6()`, `processTCP()`, `processUDP()`, `processICMP()`, `processIGMP()`. Tieto funkcie na vďaka pretypovaniu na patričné štruktúry postupne získavajú dáta a ukladajú do štruktúry Flow, ktorá je definovaná takto:

```
struct Flow
{
  struct Packet *head, *tail;

  time_t  lastMod;  /* Čas poslednej modifikácie */
  time_t  creat;    /* Čas vytvorenia */
  u_char  family;
  union
  {
    struct  in6_addr srcIPv6;
    struct  in_addr srcIPv4;
  }src;
  union
  {
    struct  in6_addr dstIPv6;
    struct  in_addr dstIPv4;
  }dst;

  u_short   sport, dport;

  u_char  inter;    /* rozhranie, z ktorého dostávame flow */
  u_char  prot;     /* TCP, UDP, ICMP, IGMP, ICMP6 */
  tcp_seq seq;    /* sequence číslo */
  tcp_seq ack;    /* acknowledgement číslo */
  u_char  flags;

  u_char  isFrg;    /* ak 1 tak je to fragment */

  u_short id;   /* ID */
  u_char  ttl;    /* time to live */

  char  ICMPtype;
  char  ICMPcode;

  u_int iDataLen; /* veľkosť tela paketu */

  u_int   total_nmb_packets;  /* počet paketov, ktoré sú uložené v tomto flow */
  u_int   total_len;  /* veľkosť celého flow-u, aj hlavičiek */

  struct Flow *prevFlow, *nextFlow; /* Kolízna doména */
};
```

Výpis kódu 3.7: Štruktúra Flow

Táto štruktúra je najhlavnejšou štruktúrou celého programu, a preto sa jej budeme patrične venovať. Do tejto štruktúry sa postupne načítavajú jednotlivé parametre paketu, a takto sa nám kryštalizuje. V tejto štruktúre nie je definovaná dátová časť, nakoľko ta sa nikde neukladá a využíva sa len raz a to priamo pri detekcii, potom sa zahadzuje. V prvom rade štruktúra Flow je elementárny prvok hash tabuľky, ktorá tvorí základný kameň pre Traffic Accounting. Táto štruktúra nám reprezentuje náš flow (prúd dát), ktorý sme zaznamenali, a to buď cez NetFlow rozhranie alebo lokálne rozhranie. Z dôvodu možných kolízií sú definované premenné `prevFlow` a `nextFlow`. Za pomoci týchto premenných si vytvárame našu oblasť pretečenia, ktorá je reprezentovaná obojsmerným spojovým zoznamom. Štruktúra ako taká však

nemôže byť nápomocná v prípade ak chceme monitorovať dáta a priebežne ich mazať, tj. vytvoriť si tzv. plávajúci týždeň. Každý uložený prúd (flow) sa segmentuje na jednotky `struct Packet`.

```
struct Packet
{
  time_t  createTime; /* Čas vytvorenia */

  u_int   nmb_packets;  /* Počet paketov v tomto záznamu */
  u_int   payload_len;  /* veľkosť tela v bytoch, v tomto zázname */

  struct Packet* prvPckt;
  struct Packet* nxtPckt;
};
```

Výpis kódu 3.8: Štruktúra Packet

Táto štruktúra, vzhľadom k požadovaným vlastnostiam užívateľov po takomto programe nie je príliš obsiahla. Obsahuje iba čas vzniku, počet prenesených paketov, resp. dát a odkaz na ďalšie prvky v spojovom zoznamu.

K účelu segmentácie hlavnej Flow štruktúry nám slúžia premenné `head` a `tail`, ktoré obe ukazujú na začiatok respektíve koniec jednosmerného spojového zoznamu, ktorý si v sebe udržiava informácie čo sa stalo v jednotlivých časových úsekoch. Vďaka tejto štruktúre vieme adekvátne znižovať hodnoty pri premazávaní nášho prúdu dát a tak udržiavať náš plávajúci týždeň. Segmentáciu nášho prúdu nastavujeme cez premennú *TimeSeg* v konfiguračnom súbore. Jednotky, v ktorých sa udáva hodnota tejto premennej sú sekundy. V prípade ak túto premennú nešpecifikujeme, tak sa použije prednastavená hodnota 600 sekúnd (5 minút). Ďalšou podstatnou premennou je *FlowTime*, ktorá nám určuje dĺžku času počas ktorého máme udržiavať záznamy, t.j. ako dlho má trvať náš flow čas. Pri nešpecifikovaní sa používa pred nastavená hodnota 1 týždeň. Jednotky, v ktorých sa táto hodnota udáva sú opäť sekundy. V prípade ak niektorí zo záznamov prekročil hodnotu *FlowTime* začne funkcia *CleanData()* (definovaná v 3.10) premazávať jednotlivé pakety. Ak sú všetky staršie ako náš FlowTime, tak sa zmaže celý flow z hash tabuľky. Funkcia, ktorá volá premazávania sa spustí raz za TimeSeg čas. Takže je treba dbať na vhodné zvolenie pomeru hodnôt TimeSeg a FlowTime. Štruktúra Flow mimo iného ešte obsahuje všetky potrebné informácie o flowu, ktoré sa príchodom ďalších paketov obnovujú. Konkrétne to sú premenné `total_nmb_packets` a `total_len`, z ktorých sa pri zisťovaní či niekto neporušil Traffic Accounting pravidlo, prechádzajú a overujú. Táto štruktúra nie je používaná len na ukladanie flow-u tvoreného z viacerých paketov. Taktiež slúži pri spracovaní paketu a priamo do nej sa ukladajú hodnoty prijatého paketu. Vzhľadom k tomu, že aj 1 paket môže tvoriť flow je toto riešenie bezproblémové a v prípade, že sa nachádza takýto flow v našej hash tabuľke, tak sa vezmú hodnoty `total_nmb_packets` a `total_len` a zväčšia sa v príslušnom flowu. Ak však tento flow neexistuje, tak vzhľadom k tomu, že je to všetko rozparsované tak ako má tak sa len vytvorí prvý segment (struct

Packet), ktorý bude obsahovať rovnaké údaje ako samotný paket a celý flow sa začlení do tabuľky.

Štruktúra `struct idx` definovaná v tejto časti je využívaná v prípade Traffic Accounting pravidiel kde sa musí prechádzať celá naša štruktúra a keďže to je rozsiahla hash tabuľka, tak najrýchlejším spôsobom pri minimálnom využití pamäte je si uložiť indexy na všetky indexy tabuľky kde sa nachádza nejaký flow.

```
struct idx
{
  int index;
  struct idx *next;
};
```

Výpis kódu 3.9: Štruktúra idx

Tým pádom ak potrebujeme prejsť celú štruktúru stačí sa nasmerovať na začiatok tejto index štruktúry a potom sa môžeme posúvať v nej až nakoniec. Tento smer nám spravuje štruktúra FlowHeader, ktorá si pamätá nasmerovanie na prvý a posledný prvok a taktiež počet flow.

```
struct FlowHeader
{
  int FlowCnt;
  struct idx *IndexHead, *IndexTail;
};
```

Výpis kódu 3.10: Štruktúra FlowHeader

3.7 DetectionEngine

Tento modul zisťuje či daný paket spĺňa niektoré z pravidiel. Pre zistenie či daný paket splňuje Traffic Analysing časť pravidla slúži funkcia `Detection()` a pre zistenie Traffic Accounting časti pravidiel je `TrafficAcc()` funkcia.

Jednotlivé pravidla sa kontrolujú pomocou vhodne usporiadaných podmienok, ktoré testujú každú vlastnosť, ktorú má paket a je definovaná v pravidlu. V prípade ak sa daná vlastnosť nebude zhodovať, tak sa funkcia ihneď ukončí. Ak však nastane v každom prípade zhoda, tak funkcia vráti 1. Rozhranie nad touto funkciou poskytuje funkcia Detect(), ktorá volá na každé pravidlo zvlášť Detection() funkciu a napočítavá počet pozitívnych nálezov a potom ich vracia.

Pre Traffic Accounting slúži funkcia TrafficAcc(), ktorá sa volá každých iTimeTrf sekúnd. Takto si môžeme zrýchliť beh programu ak chcem podrobné údaje o tokoch, ale nechceme ich často kontrolovať. Pri tomto kontrolovaní sa prv prejdú časti s pravidlami určujúcimi vlastnosti toku ako aj ich Traffic Analysing časť. Keď u oboch častiach bude zhoda, tak sa spustí druhá časť

testov, ktorá bude overovať či daný flow neporušil niektorý z Traffic Accounting pravidiel a to tak, že si ich hodnoty bude porovnávať voči vlastnostiam flow-u.

Tento modul mimo iné obsahuje aj funkciu Preparse(), ktorá je využívaná pri nahradzovaní premenných v texte definovaných v pravidlách, prípadne v súbore, ktorý sa má posielať na router. Tieto premenné sa nahrádzajú parametrami paketu, ktorý bol označený ako nevhodný za pomoci detekčných funkcií. Premenná je definovaná ako reťazec začínajúci znakom '%'.

3.8 StringMatch

Modul určený k nájdeniu zhody dát v paketu vzhľadom k textu špecifikovanom v pravidlách. Tento text môže obsahovať aj binárne, netlačitelné znaky, ktoré môžu byť zadávané za pomoci špeciálneho operátora '|'. V tomto modulu sú definované tri hlavné funkcie a to `searchStr()`, ktorá vykonáva samotné hľadanie za pomoci Boyer-Moorovho algoritmu, a pomocné funkcie, ktoré definujú Boyer-Moorovskú štruktúru a to `MakeShift()` a `MakeSkip()`, ktoré vytvárajú Shift a Skip tabuľku pre viac informácií viz. [2].

3.9 Cisco

Tento modul zabezpečuje komunikáciu programu so zariadeniami CISCO ako aj obstaráva NetFlow dáta a spracúva ich. Prvou funkciou, na ktorú by sme mali naraziť je `InitNetFlow()`, ktorá nám inicializuje premenné a vytvorí nám spojenie na NetFlow router. Avšak hlavnou funkciou je pochopiteľne `GetNetFlow()`, ktorá obstaráva NetFlow záznamy a spracováva ich. Ako náhle ich spracuje spustia sa na nich detekčné funkcie. Prvou z nich je `DetectTrafficAcc()`, ktorá vkladá náš flow do hash tabuľky. Ďalšou je už spomínaná funkcia `Detect()`. Momentálne program podporuje NetFlow vo verzii 1 a 5.

Ďalšou významnou funkciou je `ConfigureRouter()`, ktorá je určená k posielaniu príkazov routru. Funkcia sa najprv napojí na router a potom sa pri výskyte požiadavky na password mu pošle užívateľské heslo, ktoré sme definovali v konfiguračnom súbore. Ak to heslo nie je definované, tak funkcia si myslí, že nie je potrebná autorizácia a tak pokračuje. Takto to pokračuje aj pre prechod do enable módu. Ak však nebudete mať špecifikované enable heslo, tak si program opäť myslí, že nie je potrebné heslo pre prihlásenie do enable módu, a tak pokračuje. Pri vstupe do enable módu sa začnú posielať a vykonávať jednotlivé príkazy. Po poslaní všetkých príkazov sa zavrie spojenie.

Poslednou funkciou v tomto module je `PrintNetFlow()`, ktorá je skôr určená na debug ako na voľný beh. Táto funkcia nám ukazuje súhrn zaznamenaného toku dát a to prostredníctvom prejdenia hash tabuľky a výpisu jednotlivých flow. Táto funkcia sa spúšťa iba na konci programu a iba v prípade ak sme v debug móde.

3.10 Util

Obsahuje pomocné funkcie ako aj funkcie vykonávajúce operácie, ktoré nie sú dostatočne veľké na to aby boli začlenené do samotných modulov. Je to napríklad posielanie emailov, práca s reťazcami, kontrola vstupu, ale taktiež najdôležitejšia funkcia v tomto module a to `CleanData()`. Táto funkcia nám prečisťuje našu vnútornú štruktúru umiestnenú v hash tabuľke. Funkcia vymazáva iba záznamy, ktoré sú staršie ako FlowTime. Je spúšťaná z funkcie `TrafficAcc()`, čo jej dáva vlastnosť že je pustená každých TimeSeg sekúnd. Táto funkcia ide cez všetky indexy, k tomu využíva listHead, ktorá indexy spravuje, a po zistení, že nejaký paket je starší, začína mazať.

V tejto funkcii sa nachádza posledná z funkcií, ktorú by som rád v tejto časti spomenul a to `hashfunc()`. Ta je definovaná aj pre IPv6 adresy ako `hashfunc6()`. Tieto hash funkcie nám zahashujú kľúče protokol, zdrojová, cieľová IP adresa a port a vrátia nám index do našej hash tabuľky.

Kapitola 4

Záver

4.1 Výsledok práce

Nakoľko podobné systémy sa vyvíjali niekoľko rokov a spolupracovalo na nich ohromné množstvo vývojárov, tak potešujúcim faktom pre mňa je myšlienka, že sa aj mne podarilo začleniť mojim programom medzi týchto ľudí. Aj keď sa mi nepodarilo implementovať a otestovať všetky náležitosti, tak si myslím, že tento program nájde svoje uplatnenie v širšej sfére, nakoľko funkčnosťou poskytuje pohodlie a flexibilitu pri správe sieti. Prvotný zámer programu bol vytvoriť čo najvýkonnejší systém schopný obstarávať obrovský dátový tok a to tak aby sa nemuselo zahadzovať väčšie množstvo paketov. Neskôr som však zistil, že metóda obstarávania paketov, ktorú som si zvolil, ktorá využívala priamo sieťové rozhranie (read() na raw sockete), nie je oproti libpcap až tak rýchla. Obstarávacia knižnica libpcap, prakticky vydrží aj pôvodne požadovaný sieťový tok 1GB/s a obmedzujúcim faktorom v týchto podmienka boli prerušenia, ktoré sa vyvolávali príchodom paketov. Samozrejme, v týchto prípadoch je dôležité použiť adekvátny hardware a to nielen pre server, na ktorom pobeží FlowIDS, ale aj pre samotný router, ktorý by mal byť výkonnostne čo najlepší. FlowIDS ako taký ponúka širokú škálu moností, ktoré s kombináciou Netflow Collectoru posúvajú možnosti tohto programu omnoho ďalej.

4.2 Záverečné testovanie

Nakoľko u programov typu FlowIDS je dôležité porovnať výkon daného systému v určitých prostrediach, tak som sa rozhodol vykonať testy odchytávania a analýzy paketov na lokálnom rozhraní. NetFlow som nepovažoval za dôležitú časť na otestovanie, nakoľko väčšina výpočetných operácií sa vyko-

	Záťaž CPU (%)	**Zahodené pakety (%)**	**Dátový tok (MBit/s)**
Pravidlo1	14	1,2	50,4
Pravidlo5	17	1,8	50,7
Pravidlo10	41	10,5	52,0
Pravidlo25	40	30,1	50,8
Pravidlo50	42	27,3	70,0
Pravidlo100	43	20,4	56,6

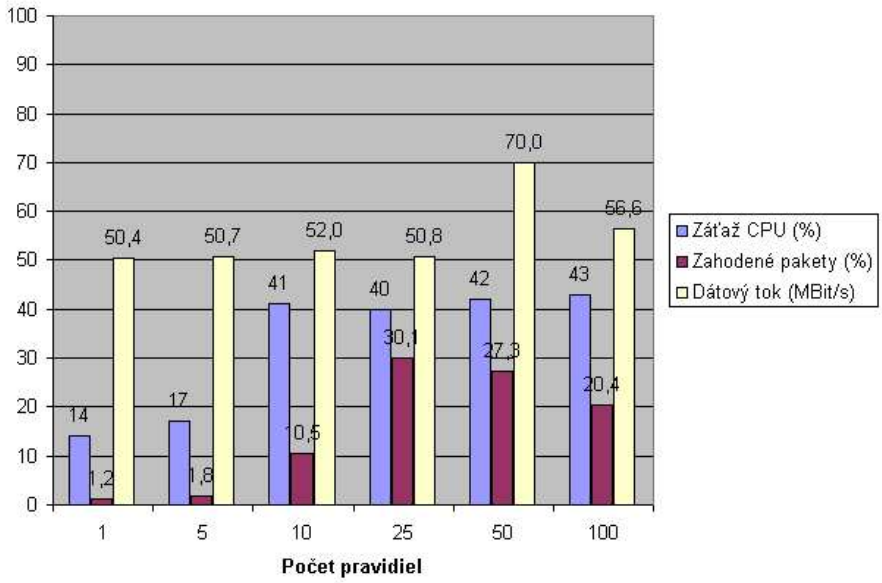
Tabuľka 4.1: Tabuľka výsledkov pre 1. testovaný systém

náva na routri, ktorý FlowIDS nemôže ovplyvňovať. Pre účely porovnávania som pripravil sadu pravidiel, ktoré sú priložené na CD, ktoré je súčasťou tejto práce a nazývajú sa: pravidlo1, pravidlo5, pravidlo10, pravidlo25, pravidlo50, pravidlo100. Čísla v názvu súboru znamenajú počet pravidiel umiestnených v danom súboru. Keďže vykonávanie testov v sieti kde by som si mohol patrične škálovať tok dát nemohli byť z technických príčin uskutočnené tak boli vykonané testy v sieťach kde je tok premenlivý a keďže aj tok je jedným z faktorov určujúci výkon systému tak som ho taktiež uviedol pre porovnanie v tabuľke.

Prvý z testovaných systémov sa nachádzal v gigabitovej sieti a preto je pre nás najzaujímavejší, nakoľko táto oblasť bola a je pre program určujúca. Nakoľko sa nám nepodarilo vygenerovať maximálny dátový tok tejto siete, tak výsledky majú iné hodnoty priemerných dátových tokov. Program bol mal v prvom prípade nastavenú hodnotu pre príjmací buffer 8192 a aj tu môžeme vidieť aký to má dopad na rýchlosť spracovania a na počet zahodených paketov. Priorita nebola nijakým spôsobom zvyšovaná Systém mal takúto konfiguráciu:

Konfigurácia
OS : FreeBSD 4.11-RELEASE-p16
CPU : Intel(R) Pentium(R) 4 CPU 2.00GHz (2004.57-MHz 686-class CPU)
Mem : 256MB
Net : Intel(R) PRO/1000 Network Connection; 1000 Mbps full duplex

Vzhľadom k tomu, že v tomto teste sa vyskytla najsilnejšia konfigurácia tak sa dali očakávať aj najlepšie výsledky, ktoré sú zobrazené v grafe 4.1. Avšak niektoré z nich boli až prekvapivo slabé, a to hlavne vo výsledkoch počtu zahodených paketov. Bolo to hlavne zapríčinené nedostatočne veľkým príjmacím bufferom, ktorý sa dá nastaviť pomocou kľúčového slova `BufSize` v konfiguračnom súbore. Taktiež nebolo nastavená nejaká vyššia priorita, tá sa dá pre zmenu nastaviť pomocou parametru `Priority` v konfiguračnom



Obr. 4.1: Graf prvého merania

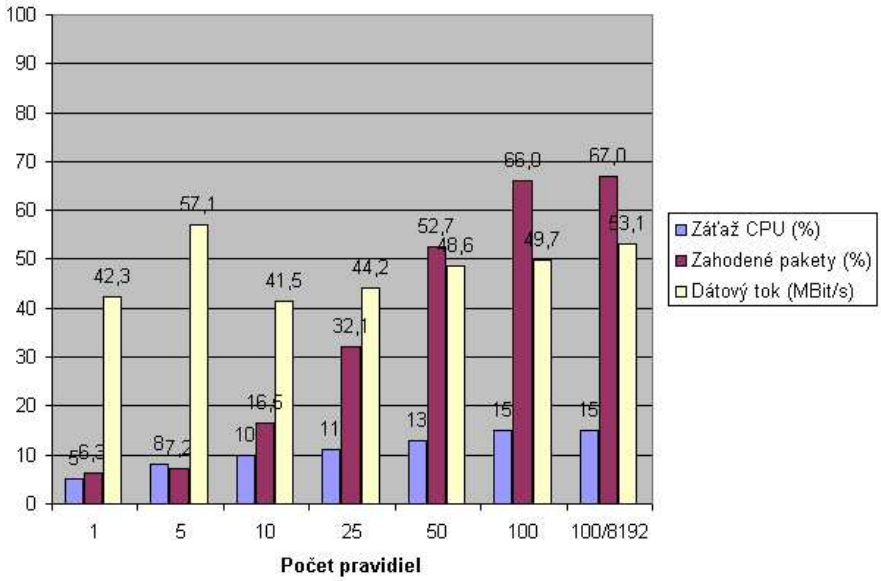
súbore. Všetky merania sa odohrali v približnom časovom intervale 5 minút počas, ktorého sa generoval umelý tok dát, zväčša cez FTP alebo iné služby poskytujúce ľahko uskutočniteľný prenos dát. V priloženom grafe je vidieť ako počet pravidiel zvyšuje záťaž na procesor, ale je zaujímavé pozorovať kedy sa buffer očividne začína preplňovať a FlowIDS začína mať problémy. Vzhľadom k tomu, že už nebolo možné vykonať opätovné meranie na tomto serveru, ostáva otázkou ako moc by sa zlepšili výsledky. Na ostatných testovaných serveroch ponúkam už riešenia, ktoré majú buffer veľkosti 65535 bytov a pre porovnanie udávam spustený test s nastavením bufferom o veľkosti 8192 a bez nastavenia akejkoľvek priority.

Druhý server je pomerne nevýkonný a tak je určený hlavne na určenie výkonu pre low-end servere. S týmto serverom však boli počas merania problémy a nedosahovali sa ani zďaleka také rýchlosti ako pri ostatných. Problémom bola dozaista aj konfigurácia servera a nielen jeho slabší výkon. Prehľadnejšie to môžeme vidiet na obrázku 4.2.

	Konfigurácia
OS	: FreeBSD 5.4-RELEASE
CPU	: Pentium III/Pentium III Xeon/Celeron (448.97-MHz 686-class CPU)
Mem	: 256MB
Net	: 3Com 3c905B-TX Fast Etherlink XL; 100Mbps full duplex

	Záťaž CPU (%)	**Zahodené pakety (%)**	**Dátový tok (MBit/s)**
Pravidlo1	5	6,3	42,3
Pravidlo5	8	7,2	57,1
Pravidlo10	10	16,5	41,5
Pravidlo25	11	32,1	44,2
Pravidlo50	13	52,7	48,6
Pravidlo100	15	66,0	49,7
Pravidlo100/8192	15	67,4	53,1

Tabuľka 4.2: Tabuľka výsledkov pre 2. testovaný systém



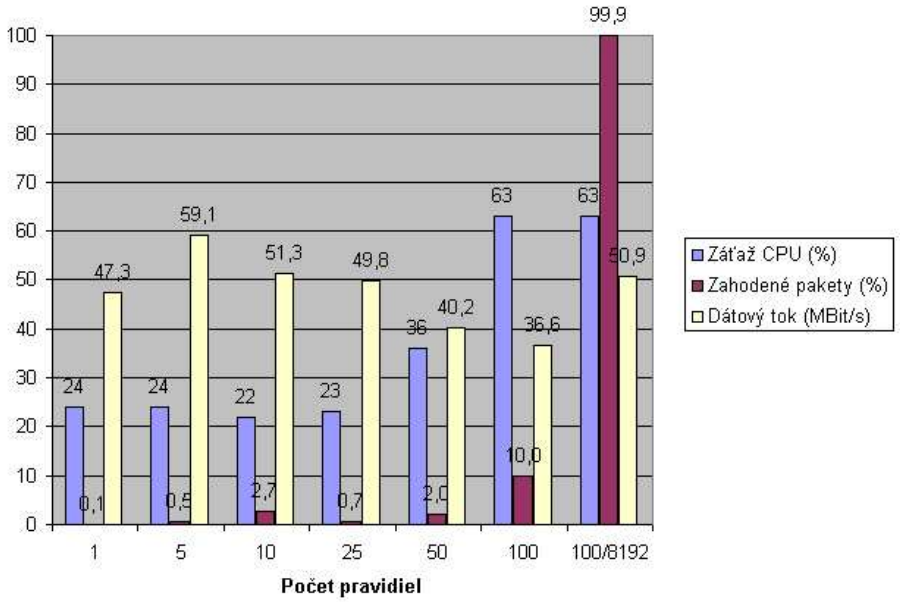
Obr. 4.2: Graf druhého merania

	Záťaž CPU (%)	**Zahodené pakety (%)**	**Dátový tok (MBit/s)**
Pravidlo1	24	0,1	47,3
Pravidlo5	24	0,5	59,1
Pravidlo10	22	2,7	51,3
Pravidlo25	23	0,7	49,8
Pravidlo50	36	2,0	40,2
Pravidlo100	63	10,0	36,6
Pravidlo100/8192	63	90,8	80,0

Tabuľka 4.3: Tabuľka výsledkov pre 3. testovaný systém

Graf na obrázku 4.3 reprezentuje tretí server, ktorý je výkonnejší a konečne postavený na Linuxe, takže mohlo byť vykonané testovanie aj na inom sieťovom stacku ako ponúka BSD.

Konfigurácia
OS : GNU/Linux (Debian) 2.6.8-2-686
CPU : Intel Celeron (Coppermine)(797.0324 MHz 686-class CPU)
Mem : 128MB
Net : Intel(R) PRO/100 Network Connection; 100Mbps full duplex



Obr. 4.3: Graf tretieho merania

Zaujímavou hodnotou bol výkon systému pri nastavený bufferu na veľkosť 8192. Vzhľadom k výkonu, k veľkosti RAM a ďalším parametrom bol tento buffer príliš malý na to aby stíhal uspokojiť relatívne akceptovateľné množstvo paketov. Takže v prípade takéhoto počtu stratených paketov je dobré sa pokúsiť a zmenu v nastavení tejto hodnoty. Nielen že jej zvýšenie ale aj zníženie môže prinášať patrične rozdiely vo výkone. Je to len na vás do akej miery to budete skúšať a hľadať ideálnu hodnotu pre váš systém.

Literatúra

[1] Deering, S.; Hinden, R.: "RFC 2460", *Internet Protocol, Version 6 (IPv6) Specification.*

[2] Gusfield, Dan: "Boyer Moore," *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology* (Cambridge University Press, January 15, 1997).

[3] Information Sciences Institute: "RFC 791", *Internet Protocol, Protocol Specification.*

[4] Jacobson, Van; Leres, Craig; McCanne, Steven: "tcpdump", *tcpdump man page, http://www.ethereal.com/docs/man-pages/tcpdump.8.html*