Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Hĺbková analýza paketov prostredníctvom protokolu IPFIX

Príloha B

Systémová príručka

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	Ing. Adrián Pekár, PhD.
Konzultant:	Ing. Ján Juhár

Košice 2015 **Bc. Dávid Farkas**

Obsah

1 Funkcia programu MyBeem

Program MyBeem reprezentuje najnižšiu vrstvu architektúry BasicMeter. Predstavuje monitorovací a exportovací proces. Tieto procesy slúžia na monitorovanie sieťovej prevádzky a jej parametrov s následným exportovaním nameraných hodnôt do vyššej vrstvy. Program exportuje tieto hodnoty vo formáte konfrontujúcim so štandardmi IPFIX a PSAMP. Je to konzolová aplikácia a nemá žiadne grafické rozhranie. Rôzne výpisy programu je možné sledovať priamo v konzole. Program bol vyvíjaný použitím open-source technológií.

1.1 Popis modulov a podprogramov

Program MyBeem obsahuje nasledujúce moduly:

- **aggregation.c** - súbor obsahujúci väčšinu funkcionality procesu agregácie.
  - *void flow_cache_rework(void)* - funkcia pre reorganizáciu vyrovnávacej pamäte tokov, ktorá spúšťa presun tokov medzi jednotlivými buffer-ami
  - *struct cache_item *packet_to_flow(packet_info_t *packet)* - funkcia priradenia paketu do jemu prislúchajúceho toku

    Vstupné premenné:
    packet - jednotlivé polia hlavičky paketu.

    Výstupná hodnota:
    0 v prípade, že nieje nájdený záznam o toku pre daný paket,
    *cache_item ** - Tok, do ktorého patrí daný paket.
  - *void merge_flows(struct cache_item *from, struct cache_item *to)* - funkcia pre spájanie dvoch tokov v procese agregácie tokov

    Vstupné premenné:
    from - smerník na dátovú štruktúru toku, ktorý bude agregovaný a násled-

ne vymazaný.

to - smerník na dátovú štruktúru toku, ktorej záznam má zhodný identifikátor toku s tokom, ktorý sa má agregovať, a do ktorej budú nahraté jednotlivé položky štruktúry *from*.

– *void move_flows_to_next_buff(struct cache_item *item, int i, int buff_nmbr)* - samotná funkcia presunu tokov medzi jednotlivými buffer-ami pri agregácii

Vstupné premenné:

item - smerník na dátovú štruktúru toku, ktorej hodnoty budú kontorlované, či spĺňa alebo nespĺňa agregačné podmienky.
i - hash, na základe ktorého je daná štruktúra umiestnená v hash liste.
buff_nmbr - číslo hash listu, v ktorom sa nachádza daná dátová štruktúra.

– *uint64_t fwd_packet_flow_identificator(packet_info_t *packet)* - funkcia, ktorá vytvára jednoznačný identifikátor paketu v smere *forward*.

Vstupné premenné:

packet - jednotlivé polia hlavičky paketu.

Výstupná hodnota:

Premenná obsahujúca výsledný číselný identifikátor paketu.

– *uint64_t bckwd_packet_flow_identificator(packet_info_t *packet)* - funkcia, ktorá vytvára jednoznačný identifikátor paketu v smere *backward*.

Vstupné premenné:

packet - jednotlivé polia hlavičky paketu.

Výstupná hodnota:

Premenná obsahujúca výsledný číselný identifikátor paketu.

– *void flow_keys_reduction(struct cache_item *item, int buff_nmbr)* - funkcia na redukciu kľúčových hodnôt toku

Vstupné premenné:

item - smerník na dátovú štruktúru toku, ktorej kľúčové hodnoty majú byť redukované.

buff_nmbr - číslo hash listu resp. buffer-a, do ktorého má byť presunutý smerník na danú dátovú štruktúru.

- **beem.c** - hlavný súbor obsahujúci základnú funkciu main().
  - *int main (int argc, char **argv)* - základná funkcia celého programu

    Vstupné premenné:

    argc - počet argumentov povelového riadku.

    argv - smerník na pole obsahujúce argumenty povelového riadku.

    Výstupná hodnota:

    0 v prípade úspechu, *-1* v prípade neúspechu.

- **cache.c** - modul vyrovnávacej pamäte cache. Obsahuje taktiež funkcie pre klasifikáciu paketov a identifikáciu paketových párov.
  - *int compare_keys(struct flow_key *fk, packet_info_t *pkt)* - funkcia, ktorá porovná kľúčové hodnoty toku s príslušnými hodnotami polí hlavičky paketu.

    Vstupné premenné:

    fk - kľúčové hodnoty toku.

    pkt - jednotlivé polia hlavičky paketu.

    Výstupná hodnota:

    0 ak sa kľúčové hodnoty toku nezhodujú s príslušnými hodnotami polí paketu,

    1 ak nastala zhoda kľúčových hodnôt a jedná sa o tok od odosielateľa k príjemcovi,

    2 ak nastala zhoda kľúčových hodnôt a jedná sa o spätný tok.

– *struct cache_item *cache_get_item(packet_info_t *packet)* - funkcia, ktorá identifikuje toky pre pakety

Vstupné premenné:

packet - jednotlivé polia hlavičky paketu.

Výstupná hodnota:

*T*ok, do ktorého patrí daný paket.

– *struct _packet_info_t *find_packet_pair(struct list *list, struct _packet_info_t *packet)* - funkcia, ktorá nájde odpovedajúce paketové páry

Vstupné premenné:

list - zoznam paketov pre vyhľadávanie paketových párov .

packet - jednotlivé polia hlavičky paketu.

Výstupná hodnota:

0 v prípade úspechu.

– *struct dpi_item *dpi_get_item(packet_info_t *packet, uint64_t key)* - funkcia, ktorá identifikuje dpi toky pre pakety.

Vstupné premenné:

packet - jednotlivé polia hlavičky paketu. *key* - identifikátor dpi toku.

Výstupná hodnota:

DPI tok, do ktorého patrí daný paket.

– *uint64_t flow_identificator(struct flow_key *fk)* - funkcia pre výpočet číselného identifikátora toku.

Vstupné premenné:

fk - kľúčové hodnoty toky.

Výstupná hodnota:

Premenná obsahujúca výsledný číselný identifikátor toku.

– *void add_packet_to_flow(struct cache_item *item, struct _packet_info_t *packet)* - funkcia, ktorá vkladá pakety do tokov.

Vstupné premenné:
item - príslušný tok.
packet - jednotlivé polia hlavičky paketu.

– *int flowCacheInit(void)* - funkcia pre inicializáciu vyrovnávacej pamäte tokov.

Výstupná hodnota:
0 v prípade úspechu, *-1* v prípade neúspechu.

– *void flowCachePacketProcessing(void)* - funkcia pre správu pamäte tokov pri spracovávaní paketu.

– *void pcache_list_expire(struct list *pcache, uint64_t current_time)* - funkcia pre kontrolu aktuálnosti prvkov vyrovnávacej pamäte tokov.

Vstupné premenné:
pcache - prvky vyrovnávacej pamäte.
current_time - aktuálny čas.

– *void flowAggregationProcess(void)* - funkcia, ktorá spúšťa proces agregácie tokov v prípade, že je splnená podmienka uplynutia stanoveného časového intervalu.

– *void flowCacheTimeExpireInit(void)* - funkcia, ktorá prechádza celú vyrovnávaciu pamäť tokov a spúšťa funkciu kontorly expirácie toku v každom z hash listov.

– *void flowCacheTimeExpire(struct cache_item *item, int i, int mark)* - funkcia, ktorá kontroluje, či daný tok expiroval,ak áno, presunie ho do pamäte expirovaných tokov. **Vstupné premenné:**
item - smerník na dátovú štruktúru toku, ktorá bude kontrolovaná pre

splnenie alebo nesplnenie podmienok pasívnej expirácie.
i - hash, na základe ktorého je daná štruktúra umiestnená v hash liste.
mark - číslo hash listu, v ktorom prebieha kontrola expirácie tokov.

- *void flowCacheExport(void)* - funkcia pre export vyrovnávacej pamäte tokov.
- *uint64_t getCurrentTime(int precision)* - funkcia pre získanie aktuálneho času.

  Vstupné premenné:
  precision - presnosť, s akou chceme získať aktuálny čas.

  Výstupná hodnota:
  0 v prípade neúspechu,
  current - aktuálny čas v prípade úspechu.
- *void debugFlowCache(void)* - funkcia zabezpečujúca kontrolné výpisy pamäte tokov.
- *void cache_status(void)* - funkcia opisujúca stav vyrovnávacej pamäte.

- **capture.c** - modul odchytávania paketov. Tento modul využíva knižnicu libpcap, pomocou ktorej odchytáva pakety zo sieťovej prevádzky, vyberá z nich potrebné informácie a ukladá ich do pamäte cache.
  - *void* threadTimeExpire(void *arg)* - funkcia vlákna časovej expirácie tokov.

    Vstupné premenné:
    arg - smerník na argument typu void.

    Výstupná hodnota:
    Vracia nulovú hodnotu.
  - *void* threadExport(void *arg)* - funkcia vlákna exportu tokov.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

Vracia nulovú hodnotu.

– *void* threadPacketProcessing(void *arg)* - funkcia vlákna pre spracovanie paketov.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

Vracia nulovú hodnotu.

– *void* threadFlowAggregation(void *arg)* - funkcia vlákna pre agregáciu tokov.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

Vracia nulovú hodnotu.

– *void *setsignal (int sig, void (*func)(int))* - funkcia, ktorá nastavuje reakciu na signál.

Vstupné premenné:

sig - signál.

func - smerník na obslužnú funkciu pre signál.

Výstupná hodnota:

SIG_ERR v prípade výskytu chyby alebo *old.sa_handler*, čo je pôvodná reakcia na signál.

– *void* threadPacketCapturing(void *arg)* - funkcia vlákna pre odchytávanie

paketov.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

Vracia nulovú hodnotu.

– *struct ndpi_detection_module_struct *setup_ndpi()* - funkcia na vytvorenie detekčného modulu pre hĺbkovú analýzu paketov.

Výstupná hodnota:

NULL, ak služba hĺbkovej analýzy paketov je vypnutá, v opačnom prípade smerník na detekčný modul.

– *void terminate_ndpi(struct ndpi_detection_module_struct *mod)* - funkcia na zatvorenie detekčného modulu pre hĺbkovú analýzu paketov.

Vstupné premenné:

mod - smerník na detekčný modul.

– *void *malloc_wrapper(unsigned long size)* - alokačná funkcia.

Vstupné premenné:

size - veľkosť.

Výstupná hodnota:

Smerník na začiatok vymedzený pamäťový priestor.

– *void free_wrapper(void *freeable)* - funkcia na uvoľnenie pamäťového miesta.

Vstupné premenné:

freeable - smerník na pamäťové miesto.

– *void addPacket(packet_info_t *packet_info)* - funkcia identifikuje tok pre daný paket a odošle ho na spracovanie do pamäte tokov.

Vstupné premenné:

packet_info - smerník na jednotlivé polia hlavičky paketu.

– *void processPacket(struct packet_capturing_thread *t, const u_char *packet)* - funkcia zabezpečujúca vlastné spracovanie paketu a odoslanie dát na klasifikáciu.

Vstupné premenné:

t - vlákno na odchytávanie paketov.

packet - binárne dáta paketu.

– *void catchInt(int sig_num)* - funkcia na odchytenie signálu INT.

Vstupné premenné:

sig_num - číslo signálu.

– *void catchInt2(int sig_num)* - funkcia na odchytenie druhého signálu INT. Po jeho odchytení program definitívne končí.

Vstupné premenné:

sig_num - číslo signálu.

– *void doCallback(u_char *argument, const struct pcap_pkthdr *pkthdr, const u_char *packet)* - callback funkcia pcap knižnice zabezpečujúca odoslanie dát v pakete na spracovanie.

Vstupné premenné:

argument - argumenty, predávané do callback funckie posledným argumentom funkcie pcap_loop.

pkthdr - smerník na hlavičku paketu obsahujúcu timestamp, odchytenú dĺžku paketu a reálnu dĺžku paketu.

packet - smerník na vlastné odchytené dáta paketu.

– *void startCapture()* - funkcia, ktorá inicializuje odchytávanie paketov: nastavuje BPF filter pre pcap knižnicu, a spúšťa vlastný cyklus od-

chytávania.

- **config.c** - modul konfigurácie programu MyBeem. Využíva podporné knižnice libxml2.
  - *void printConfig(xmlDocPtr doc)* - funkcia vypíše na štandardný výstup momentálnu konfiguráciu.

    Vstupné premenné:

    doc - ukazovateľ na XML konfiguráciu, ktorá sa má vypísať.
  - *xmlXPathObjectPtr getNodeSet(xmlDocPtr doc, xmlChar *xpath)* - funkcia, ktorá podľa výrazu xpath vzhľadá prvý vyhovujúci uzol v konfiguračnom XML súbore.

    Vstupné premenné:

    doc - ukazovateľ na XML konfiguráciu.

    flow - ukazovateľ na xpath výraz.

    Výstupná hodnota:

    NULL ak sa nenájde cesta k požadovaným dátam v konfiguračnom súbore, alebo *result* v prípade úspešného získania dát.
  - *int readConfigInterfaces(xmlDocPtr doc, struct packet_capturing_thread **pcts, int *pct_count)* - funkcia pre načítanie informácii o rozhraniach z konfiguračného súboru

    Vstupné premenné:

    doc - ukazovateľ na XML konfiguráciu.

    pcts - pakety z daného vlákna.

    pct_count - počítadlo paketov.

    Výstupná hodnota:

    0 v prípade úspešného načítania údajov, *-1* v prípade chyby.

– *xmlDocPtr readConfigFile (const char *docname)* - funkcia do pamäte načíta zadaný konfiguračný súbor.

Vstupné premenné:

docname - názov konfiguračného súboru.

Výstupná hodnota:

doc, čo je ukazovateľ na konfiguračný súbor.

– *char* getConfigElement(xmlDocPtr doc, const char* xpath)* - funkcia získa z XPath výrazu prislúchajúci obsah elementu, ktorý je cieľom tohoto výrazu.

Vstupné premenné:

doc - ukazovateľ na XML konfiguráciu, ktorá sa bude prehľadávať.
xpath - ukazovateľ na xpath výraz.

Výstupná hodnota:

keyword, čo je samotný požadovaný obsah elementu.

– *void getConfigTemplates(xmlDocPtr doc, template_t templates[])* - funkcia pre získanie poľa nakonfigurovaných šablón.

Vstupné premenné:

doc - ukazovateľ na XML konfiguráciu.
templates - pole šablón, ktoré bude naplnené.

– *void cleanShutdown(int exitc, xmlDocPtr configuration)* - funkcia korektne ukončí činnosť programu s príslušným statusom. Uzatvorí XML parser a dealokuje pamäť s nim asociovanú.

Vstupné premenné:

exitc - status programu pri skončení.
configuration - ukazovateľ na XML konfiguráciu, ktorá sa má uvoľniť.

  - *void syslog_server_config_file()* - funkcia na vytvorenie konfiguračného súboru pre aplikáciu syslog-ng

- **debug.c** - logovací modul, zabezpečuje tvorbu a výpis logovacej správy na štandardný výstup.

  - *void log_message(char message[],int code)* - funkcia, ktorá sa stará o tvorbu a výpis samotnej log správy.

    Vstupné premenné:

    message - pole, ktoré reprezentuje samotnú log správu.

    code - kód príslušnej log správy.

- **export.c** - exportovací modul, zabezpečuje export údajov do kolektora. Export využíva knižnicu ipfix.

  - *void initExport()* - inicializačná funkcia exportéra. Nastavuje všetky náležitosti pre úspešné spojenie.

  - *void closeExport()* - deinicializačná funkcia exportéra. Nastavuje všetky náležitosti pre úspešné ukončenie spojenia.

  - *int exportFlow (struct cache_item *flow)* - exportná funkcia. Naplní IPFIX správu konkrétnymi údajmi a odošle ju po sieťovom transporte zhromažďovaču.

    Vstupné premenné:

    flow - tok, ktorý sa bude exportovať.

    Výstupná hodnota:

    0 v prípade úspechu, *-1* v prípade neúspechu.

- **ipfix.c** - modul ipfix potrebný pre export údajov v štandarde IPFIX.

  - *ipfix_field_type_t findIEField(int ie)* - v generovanom poli v hlavičkovom súbore ipfix_fields.h nájde záznam konkrétneho informačného elementu.

Vstupné premenné:

ie - identifikátor informačného elementu.

Výstupná hodnota:

Štruktúru obsahujúcu informácie o nájdenom informačnom elemente.

– *void updateIeLenghts()* - funkcia na nastavenie dĺžky informačných elementov. Využíva sa pre účely protokolu NetFlow v5.

– *static int do_writen(int fd, char *ptr, int nbytes)* - funkcia, ktorá zapíše do súboru 'n' bajtov. Pokiaľ je výstupný súbor stream socket , funguje ako write.

Vstupné premenné:

fd - deskriptor súboru.

ptr - buffer, ktorý sa do súboru zapíše.

nbytes - Počet bajtov ktorý sa má do súboru zapísať.

Výstupná hodnota:

Veľkosť zapísaných dát v prípade úspechu, *-1* v prípade neúspechu.

– *static int _connect_nonb(int sockfd, struct sockaddr *saptr, socklen_t salen, int sec)* - eportná funkcia. Naplní IPFIX správu konkrétnymi údajmi a odošle ju po sieťovom transporte zhromažďovaču.

Vstupné premenné:

sockfd - deskriptor socketu.

saptr - smerník na adresu soketu.

salen - dĺžka socketu.

sec - timeout pre spojenie.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *iobuf_t *_ipfix_getbuf (void)* - funkcia na získanie ďalšieho bufferu.

Výstupná hodnota:

Ďalší buffer.

– *void _ipfix_freebuf(iobuf_t *b)* - funkcia na uvoľnenie bufferu.

Vstupné premenné:

b - Buffer.

– *void _free_field_types(ipfix_field_t **flist)* - funkcia uprace v pamäti tabuľku typov polí.

Vstupné premenné:

flist - smerník na tabuľku typov polí.

– *int ipfix_encode_int(void *in, void *out, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

in - informačný element v host byte order.

out - zakódovaný informačný element v network byte order.

len - rozsah informačného elementu.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_decode_int(void *in, void *out, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

in - zakódovaný informačný element v network byte order.

out - nezakódovaný informačný element v host byte order.

len - rozsah informačného elementu.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_snprint_int(char *str, size_t size, void *data, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

str - zakódovaný informačný element vo forme retazca.

size - veľkosť výsledného reťazca.

data - informačný element v tvare int.

len - rozsah informačného elementu.

Výstupná hodnota:

Veľkosť zapísaných dát.

– *int ipfix_snprint_uint(char *str, size_t size, void *data, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

str - zakódovaný informačný element vo forme retazca.

size - veľkosť výsledného reťazca.

data - informačný element v tvare unsigned int.

len - rozsah informačného elementu.

Výstupná hodnota:

Veľkosť zapísaných dát.

– *int ipfix_encode_bytes(void *in, void *out, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

in - vstupná reprezentácia informačného elementu.

out - výstupná reprezentácia informačný element.

len - rozsah informačného elementu.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_decode_bytes(void *in, void *out, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

in - vstupná reprezentácia informačného elementu.

out - výstupná reprezentácia informačný element.

len - rozsah informačného elementu.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_snprint_bytes(char *str, size_t size, void *data, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

str - zakódovaný informačný element vo forme retazca.

size - veľkosť výsledného reťazca.

data - informačný element v tvare byte.

len - rozsah informačného elementu.

Výstupná hodnota:

Veľkosť zapísaných dát.

– *int ipfix_snprint_string(char *str, size_t size, void *data, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

str - zakódovaný informačný element vo forme retazca.

size - veľkosť výsledného reťazca.

data - informačný element v tvare ip adresy.

len - rozsah informačného elementu.

Výstupná hodnota:

Veľkosť zapísaných dát.

– *int ipfix_snprint_uint(char *str, size_t size, void *data, size_t len)* - funkcia na kódovanie a dekódovanie informačných elementov.

Vstupné premenné:

str - zakódovaný informačný element vo forme retazca.

size - veľkosť výsledného reťazca.

data - informačného elementu v tvare unsigned int.

len - rozsah informačného elementu.

Výstupná hodnota:

Veľkosť zapísaných dát.

– *void ipfix_free_unknown_ftinfo(ipfix_field_t *f)* - funkcia na upratanie neznámych položiek tabuľky polí.

Vstupné premenné:

f - tabuľka polí.

– *ipfix_field_t *ipfix_create_unknown_ftinfo(int eno, int type)* - funkcia na vytvorenie neznámych položiek tabuľky polí.

Vstupné premenné:

eno - enterprise číslo.

type - typ poľa.

Výstupná hodnota:

ftinfo z globálneho zoznamu alebo NULL.

– *ipfix_field_t *ipfix_get_ftinfo(int eno, int type)* - funckia vracia konkrétnu položku zo zoznamu neznámych položiek.

Vstupné premenné:

eno - enterprise číslo.

len - typ poľa.

Výstupná hodnota:

ftinfo z globálneho zoznamu alebo NULL.

– *int ipfix_init (void)* - funkcia zabezpečujúca inicializáciu IPFIX exportéra.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *ipfix_add_vendor_information_elements(ipfix_field_type_t *fields)* - funkcia pridá informačné elementy definované v ipfix_def.h súbore do globálneho zoznamu typov polí.

Vstupné premenné:

fields - pole o veľkosti nfields+1. Posledný člen má ftype = 0.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *void ipfix_cleanup (void)* - funkcia uprace pamäťové alokácie.

– *int _ipfix_connect (ipfix_collector_t *col)* - funkcia pridá informačné elementy definované v ipfix_def.h súbore do globálneho zoznamu typov polí.

Vstupné premenné:

col - Deskriptor zhromažďovača.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *void _ipfix_disconnect(ipfix_collector_t *col)* - funkcia odpája IPFIX exportér od zhromažďovača.

Vstupné premenné:

col - Deskriptor zhromažďovača.

– *int _ipfix_send_msg(ipfix_t *ifh, iobuf_t *buf)* - funkcia odošle IPFIX správu podľa vstupných parametrov a buffera.

Vstupné premenné:

ifh - štruktúra obsahujúca kompletné informácie o spojení.

buf - buffer obsahujúci dáta na odoslanie.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_reconnect(ipfix_t *ifh)* - funkcia sa pokúsi o znovupripojenie k definovanému zhromažďovaču.

Vstupné premenné:

ifh - štruktúra obsahujúca kompletné informácie o spojení.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_add_template(ipfix_t *ifh, iobuf_t *buf)* - funkcia zapíše na začiatok bufferu šablónu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompletné informácie o spojení.

buf - buffer, ktorý bude hlavičku obsahovať.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int _ipfix_write_hdr(ipfix_t *ifh, iobuf_t *buf)* - funkcia zapíše do bufferu hlavičku IPFIX správy.

Vstupné premenné:

ifh - štruktúra obsahujúca kompletné informácie o spojení.

buf - buffer, ktorý bude hlavičku obsahovať.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_open(ipfix_t **ipfixh, int sourceid, int ipfix_version)* - funkcia otvorí ipfix export.

Vstupné premenné:

ipfixh - štruktúra obsahujúca kompltné informácie o spojení.

sourceid - identifikátor meracieho procesu.

ipfix_version - verzia IPFIX protokolu (0x09 - NetFlow, 0xa0 - IPFIX).

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *void ipfix_close(ipfix_t *h)* - funkcia uzavrie ipfix export.

Vstupné premenné:

h - štruktúra obsahujúca kompltné informácie o spojení.

– *void _drop_collector(ipfix_collector_t **list, ipfix_collector_t *node)* - pomocná funkcia na odstránenie zhromažďovača zo zoznamu zhromažďovačov.

Vstupné premenné:

list - zoznam zhromažďovačov.

node - zhromažďovač, ktorý má byť odstránený.

– *void _ipfix_drop_collector(ipfix_collector_t **col)* - funkcia na odstránenie zhromažďovača zo zoznamu zhromažďovačov.

Vstupné premenné:

col - zhromažďovač, ktorý má byť odstránený.

– *int ipfix_add_collector(ipfix_t *ifh, char *host, int port, ipfix_proto_t prot, int refreshTmpl, int reconnectFreq, int connTimeout)* - funkcia pridá zhromažďovač do zoznamu zhromažďovačov.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

host - adresa zhromažďovača.

port - port na ktorom tento zhromažďovač očakáva dáta.

prot - protokol prenosu.

refreshTmpl - čas obnovenia šablóny.

reconnectFreq - frekvencia znovupripájania k zhromažďovaču.

connTimeout - čas vypršania spojenia.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_new_template(ipfix_t *ifh, ipfix_template_t **templ, int nfields, int id)* - funkcia pridá šablónu do zoznamu šablón.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

nfields - počet polí v pridávanej šablóne.

id - identifikátor šablóny.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_new_data_template(ipfix_t *ifh, ipfix_template_t **templ, int nfields, int id)* - funkcia pridá šablónu do zoznamu šablón.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

nfields - počet polí v pridávanej šablóne.

id - identifikátor šablóny.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_new_option_template(ipfix_t *ifh, ipfix_template_t **templ, int nfields, int id)* - funkcia pridá šablónu do zoznamu šablón.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

nfields - počet polí v pridávanej šablóne.

id - identifikátor šablóny.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_add_field(ipfix_t *ifh, ipfix_template_t *templ, uint32_t eno, uint16_t type, uint16_t length)* - funkcia pridá nové pole do aktuálnej šablóny.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

eno - enterprise číslo.

type - dátový typ poľa.

length - dĺžka poľa.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_add_scope_field(ipfix_t *ifh, ipfix_template_t *templ, uint32_t eno, uint16_t type, uint16_t length)* - funkcia pridá nové pole do aktuálnej šablóny.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

eno - enterprise číslo.

type - dátový typ poľa.

length - dĺžka poľa.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_get_template(ipfix_t *ifh, ipfix_template_t **templ, int nfields)* - funkcia získa novú šablónu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

nfields - počet polí vo vytváranej šablóne.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_get_template_array(ipfix_t *ifh, ipfix_template_t **templ, int nfields, int *types, int *lengths)* - funkcia získa novú šablónu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

templ - ukazovateľ na šablónu ktorá sa má do zoznamu pridať.

nfields - počet polí vo vytváranej šablóne.

types - types nových vytvaraných polí.

lengths - dĺžky nových vytváraných polí.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *void ipfix_free_template(ipfix_template_t *templ)* - funkcia uvoľní pamäť vytvorenej šablóny.

Vstupné premenné:

templ - dĺžky nových vytváraných polí.

– *void ipfix_delete_template(ipfix_t *ifh, ipfix_template_t *templ)* - funkcia vymaže šablónu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.
templ - dĺžky nových vytváraných polí.

– *void ipfix_release_template(ipfix_t *ifh, ipfix_template_t *templ)* - funkcia vymaže šablónu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.
templ - dĺžky nových vytváraných polí.

– *int ipfix_export(ipfix_t *ifh, ipfix_template_t *templ, void *buffer)* - funkcia na export dát pomocou protokolu IPFIX cez IPFIX správu podľa daných parametrov spojenia.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.
templ - dĺžky nových vytváraných polí.
buffer - buffer obsahujúci binárne dáta na export.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_export_array(ipfix_t *ifh, ipfix_template_t *templ, int nfields, void **fields, uint16_t *lengths)* - funkcia vykonáva vlastný export dát pomocou protokolu IPFIX cez IPFIX správu podľa daných parametrov spojenia.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.
templ - ukazovateľ na šablónu ktorá sa použije pri exporte.

nfields - počet polí vo vytváranej šablóne.

fields - smerníky na exportované polia.

lengths - pole dĺžok jednotlivých exportovaných polí.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

– *int ipfix_export_flush(ipfix_t *ifh)* - funkcia vykonáva odoslanie správy a uzavretie exportu.

Vstupné premenné:

ifh - štruktúra obsahujúca kompltné informácie o spojení.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu.

- **list.c** - modul implementujúci údajovú štruktúru jednosmerného zoznamu.

  – *int list_size(struct list *list)* - funkcia vráti počet prvkov v zozname.

  Vstupné premenné:

  list - zoznam, ktorého veľkosť je zisťovná.

  Výstupná hodnota:

  veľkosť určeného zoznamu.

  – *void list_add_first(struct list *list, struct list_item *item)* - funkcia vloží prvok na začiatok zoznamu.

  Vstupné premenné:

  list - zoznam, do ktorého je vkladaný prvok.

  item - vkladaný prvok.

  – *void list_add_last(struct list *list, struct list_item *item)* - funkcia vloží prvok na koniec zoznamu.

  Vstupné premenné:

list - zoznam, do ktorého je vkladaný prvok.

item - vkladaný prvok.

– *struct list_item *list_remove_first(struct list *list)* - funkcia odstráni prvý prvok zo zoznamu.

Vstupné premenné:

list - zoznam, z ktorého bude prvok odstránený.

Výstupná hodnota:

smerník na prvok, ktorý bol zo zoznamu odstránený, alebo *0* v prípade, že zoznam je prázdny.

– *struct list_item *list_remove_last(struct list *list)* - funkcia odstráni posledný prvok zo zoznamu.

Vstupné premenné:

list - zoznam, z ktorého bude prvok odstránený.

Výstupná hodnota:

smerník na prvok, ktorý bol zo zoznamu odstránený, alebo *0* v prípade, že zoznam je prázdny.

– *struct list_item *list_remove_item(struct list *list, struct list_item *item)* - funkcia odstráni daný prvok zo zoznamu.

Vstupné premenné:

list - zoznam, z ktorého bude prvok odstránený.

item - prvok, ktorý má byť zo zoznamu odstránený.

Výstupná hodnota:

smerník na prvok, ktorý bol zo zoznamu odstránený, alebo *0* v prípade, že zoznam daný prvok neobsahuje.

- **MurmurHash64.c** - modul, korý obsahuje hash funkciu na tvorbu číselného

identifikátora toku flow_id.

- *uint64_t MurmurHash64B (const void * key, int len, unsigned int seed)* - funkcia, ktorá sa stará o tvorbu číselného identifikátora toku.

  Vstupné premenné:

  key - ukazovateľ na reťazec pozostávajúci z kľúčových hodnôt toku.

  len - dĺžka reťazca, ktorý pozostáva z kľúčových hodnôt toku.

  seed - premenná použitá pri výpočte hash-u.

  Výstupná hodnota:

  h, čo je výsledný hash, a teda číselný identifikátor toku .

- **packetIdent.c** - modul umožňujúci vytváranie identifikátora paketu.
  - *struct identificator_s* ip_packet_identificator(const unsigned char* packet,const unsigned char length_of_identificator)* - funkcia pre vytváranie identifikátora paketu zadanej dĺžky.

    Vstupné premenné:

    packet - vstupný paket, z ktorého sa vytvára identifikátor.

    length_of_identificator - veľkosť identifikátora paketu.

    Výstupná hodnota:

    identifikátor daného paketu.

  - *void ip_packet_identificator_MD5(uint8_t *m_digest, const unsigned char* packet)* - funkcia pre vytváranie zašifrovaného identifikátora paketu použitím MD5.

    Vstupné premenné:

    m_digest - šifrovaný identifikátor paketu dĺžky 16 byte-ov.

    packet - vstupný paket, z ktorého sa vytvára identifikátor.

  - *void copy_char_array(const unsigned char *from,unsigned char *where,int*

length) - funkcia pre kopírovanie reťazcov.

Vstupné premenné:

from - zdroj kopírovania.

where - cieľ kopírovania.

length - rozsah kopírovaných dát.

- **queue.c** - modul implementujúci údajovú štruktúru jednosmerného cyklického zoznamu (front).
  - *void inline insert_after(struct list_item *item, struct list_item *newitem)* - funkcia vloží prvok bezprostredne za určený prvok vo fronte.

    Vstupné premenné:

    item - prvok, za ktorý je vkladaný nový prvok vo fronte.

    item - prvok, ktorý je do frontu vkladaný.
  - *void queue_add_last(struct queue *queue, struct list_item *item)* - funkcia vloží nový prvok na koniec frontu.

    Vstupné premenné:

    queue - front, do ktorého je prvok vkladaný.

    item - prvok, ktorý je do frontu vkladaný.
  - *struct list_item *queue_remove_first(struct queue *queue)* - funkcia odstráni prvý prvok z frontu.

    Vstupné premenné:

    queue - front, z ktorého má byť prvok odstránený.

    Výstupná hodnota:

    smerník na prvok, ktorý bol z frontu odstránený, alebo *0* v prípade, že front je prázdny.
- **sampling.c** - modul vzorkovania. Umožnuje vzorkovanie sieťovej prevádzky.

– *int is_sampled(int type, long int param1, long int param2)* - funkcia vyvoláva funkciu vzorkovania podľa konfiguračného parametra.

Vstupné premenné:

type - typ vzorkovania, ktoré má byť na spracovávaný paket použité.
param1 - prvý parameter pre použitie vo vzorkovacom algoritme.
param2 - druhý parameter pre použitie vo vzorkovacom algoritme.

Výstupná hodnota:

informácia o tom, či bol paket zvoleny

– *int systematic_count_based_sampling(long int param1, long int param2)* - funkcia implementujúca systematické vzorkovanie podľa počtu.

Vstupné premenné:

param1 - prvý parameter určujúci počet za sebou vybraných paketov.
param2 - druhý parameter určujúci počet za sebou nevybraných paketov.

Výstupná hodnota:

0 ak paket nebol vybraný pre spracovanie, *1* v opačnom prípade

– *int systematic_time_based_sampling(long int param1, long int param2)* - funkcia implementujúca systematické vzorkovanie podľa času.

Vstupné premenné:

param1 - prvý parameter určujúci dĺžku intervalu, v ktorom budú pakety vybrané.
param2 - druhý parameter určujúci dĺžku intervalu, v ktorom nebudú pakety vybrané.

Výstupná hodnota:

0 ak paket nebol vybraný pre spracovanie, *1* v opačnom prípade

– *int n_of_N_sampling(long int param1, long int param2)* - funkcia implementujúca vzorkovanie "n z N". Vybrané pakety sú dopredu určené stúpa-

júcou postupnosťou náhodných čísel, ktoré učujú poradie vybraných paketov v súbore pozorovaných paketov.

Vstupné premenné:

param1 - prvý parameter určujúci veľkosť množiny vybraných paketov.
param2 - druhý parameter určujúci veľkosť množiny paketov, z ktorých sa bude vyberať (max. 1000).

Výstupná hodnota:

0 ak paket nebol vybraný pre spracovanie, *1* v opačnom prípade

– *int uniform_probability_sampling(long int param1)* - funkcia implementujúca náhodné vzorkovanie s uniformnou pravdepodobnosťou.

Vstupné premenné:

param1 - parameter určujúci hodnotu pravdepodobnosti s akou bude každý paket vybraný.

Výstupná hodnota:

0 ak paket nebol vybraný pre spracovanie, *1* v opačnom prípade

– *int non_uniform_probability_sampling(long int param1, long int param2)* - funkcia implementujúca náhodné vzorkovanie s neuniformnou pravdepodobnosťou. Pravdepodobnosť je funkcia rozdielu, medzi aktuálnym časom a spodnou alebo hornou hranicou "sure sampleïntervalu.

Vstupné premenné:

param1 - prvý parameter určujúci spodnú hodnotu *sure sample* intervalu.
param2 - druhý parameter určujúci vrchnú hodnotu *sure sample* intervalu.

Výstupná hodnota:

0 ak paket nebol vybraný pre spracovanie, *1* v opačnom prípade

- **sync.c** - modul synchronizacie meracích bodov voči kolektoru JXcoll.

– *void mns_init()* - funkcia pre inicializáciu aproximácie metódou najmenších štvorcov.

– *void mns_insert(double x, double y)* - funkcia vloží dvojicu do štatistickej vzorky.

Vstupné premenné:

x - aktuálny čas, ktorý má byť do štatistickej vzorky vložený.

y - posun hodín zodpovedajúci aktuálnemu času.

– *void mns_remove(double x, double y)* - funkcia odstráni dvojicu zo štatistickej vzorky.

Vstupné premenné:

x - hodnota času, pre ktorý majú byť údaje zo štatistickej vzorky odstránené.

y - posun hodín zodpovedajúci času x.

– *uint64_t getCurrentTime(int precision)* - funkcia pre získanie aktuálneho času.

Vstupné premenné:

precision - presnosť, s akou chceme získať aktuálny čas.

Výstupná hodnota:

0 v prípade neúspechu,

current - aktuálny čas v prípade úspechu.

– *void *senderThread(void *arg)* - funkcia vlákna odosielania synchronizačných správ.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu. .

– *void *threadSync(void *arg)* - funkcia synchronizačného vlákna, vytvára vlákno pre odosielanie správ a prijíma synchronizačné správy od synchronizačného servera.

Vstupné premenné:

arg - smerník na argument typu void.

Výstupná hodnota:

0 v prípade úspechu, *-1* v prípade neúspechu. .

Každý modul má aj svoje hlavičkové súbory, v ktorých sa nachádzajú definície štruktúr a jednotlivých funkcií.

1.2 Popis vstupných a výstupných súborov

Vstupom programu je samotná sieťová prevádzka a výstup tvoria exportované dáta. Program je konfigurovateľný pomocou konfiguračného súboru **config.xml**. Konfigurovateľné parametre znázorňuje tabuľka 1 – 1.

Tabuľka 1 – 1: Zoznam značiek konfiguračného súboru config.xml

configuration	koreňová značka, ktorá ohraničuje všetky konfiguračné parametre konfiguračného súbor
messageLogLevel	úroveň výpisu logovacích správ na štandardný výstup
observationPointId	jedinečný identifikátor pozorovacieho bodu (celočíselná kladná hodnota 1-32767)
observationDomainId	jedinečný identifikátor pozorovacej domény

sync_port	port, použitý pri synchronizácii nástrojom MyBeem
readfile	ak true, príznak čítania zo súboru. Ak false, "číta"sa zo zvoleného sieťového rozhrania
dumpFile	názov súboru z ktorého sa v prípade nastavenia readFile na true bude čítať
interface	sieťové rozhranie, z ktorého sa majú odchytávať pakety
pcapFilter	typ BPF filtra pre filtrovanie paketov
flows	značka ohraničujúca parametre ovplyvňujúce nastavenie tokov
biflows	prepínač na zapnutie/vypnutie podpory obojsmerných tokov exportérom, false-uniflow true-biflow
passiveTimeout	nastavenie času v milisekundách pre pasívny timeout. Pasívny timeout je čas, za ktorý keď pre príslušný tok nie je obdržaný žiaden paket, tak daný tok je expirovaný.
activeTimeout	nastavenie času v milisekundách pre aktívny timeout. Aktívny timeout je čas, po uplynutí ktorého je príslušný tok expirovaný a údaje exportované aj napriek tomu, že pakety pre príslušný tok sú stále zachytávané. Musí byť väčší ako pasívny timeout.
sampling	značka ohraničujúca nastavenia týkajúce sa vzorkovania

type	celočíselná hodnota z intervalu 0 až 5 špecifikujúca spôsob vzorkovania
parameter1	prvý parameter pre vzorkovacie funkcie.
parameter2	druhý parameter pre vzorkovacie funkcie.
templates	značka ohraničujúca nastavenia týkajúce sa šablón
template	značka ohraničujúca nastavenia týkajúce sa jednej konkrétnej šablóny
field	definícia poľa v rámci jednej šablóny prostredníctvom identifikačného čísla informačného elementu *elementID* . Ak je tento element skupinovo (enterprise) špecifický, značka field sa zadáva spolu s atribútom enterprise.
mediator	značka ohraničujúca nastavenia pre mediátor
doMediation	prepínač na zapnutie/vypnutie služby
collector	značka ohraničujúca nastavenia pre kolektor
version	špecifikácia verzie kolektora/mediátora (verzia IPFIX protokolu)
host	internetová adresa zhromažďovača/mediátora, prípadne localhost
port	port, na ktorom kolektor/mediátor očakáva IPFIX správy
protocol	transportný protokol, ktorý sa použije pri odosielaní IPFIX správ

refreshTemplateTime	čas, po ktorom má byť používaná šablóna opätovne preposielaná kolektoru/mediátoru. (Nastavenie zavisí od nastavenia "default template lifetime"v kolektore.)
reconnectFrequency	počet sekúnd, po ktorých sa nástroj MyBeem bude pokúšat o znovupripojenie ku zhromažďovaču/mediátoru
connectionTimeout	počet sekúnd, po ktorých vyprší timeout spojenia so zhromažďovačom/mediátorom
synchronization	značka ohraničujúca nastavenia pre synchronizačný server
doSync	prepínač na zapnutie/vypnutie synchronizácie voči synchronizačnému serveru, false-zapnutá synchronizácia true-vypnutá synchronizácia
port	port, na ktorom MyBeem očakáva synchronizačné správy
serverAddress	internetová adresa synchronizačného servera
serverPort	port, na ktorom synchronizačný server očakáva synchronizačné správy
logging	značka ohraničujúca nastavenia pre správu logov
sendingProtocol	protocol, pomocou ktorého budú zasielané výpisy na syslog server
syslogServIP	internetová adresa syslog servera
syslogServPort	port, na ktorom bude komunikovať exportér so syslog serverom

messageLogLevel	nastavenie úrovne výpisov programu
aggregation	značka ohraničujúca nastavenie pre agregačný proces
aggregationTrigger	časový interval, po ktoreho ubehnutí bude stále prechádzaná vyrovnávacia pamäť tokov
octetTotalCountForAggregation	minimálny počet oktetov, ktorý ak tok spľňa, tak nedôjde k jeho agregácii
doAggregation	prepínač na zapnutie/vypnutie procesu agregácie
automaticAggregation	prepínač na zapnutie/vypnutie procesu automatickej agregácie
first	kľúčový element s najvyššou hodnotou priority, ktorý bude agregovaný
second	kľúčový element s druhou najvyššou hodnotou priority, ktorý bude agregovaný
third	kľúčový element s treťou najvyššou hodnotou priority, ktorý bude agregovaný
fourth	kľúčový element so štvrtou najvyššou hodnotou priority, ktorý bude agregovaný
dpi	značka ohraničujúca nastavenia pre hĺbkovú analýzu paketov
doDPI	prepínač na zapnutie/vypnutie služby hĺbkovej analýzy paketov
protofile	názov súboru vlastných aplikačných protokolov

2 Funkcia programu Syncserv

Progam Syncserv reprezentuje synchronizačný server nástroja BasicMeter. Je určený pre synchronizáciu hodín nástroja MyBeem. Takisto ako nástroj MyBeem je tento program konzolová aplikácia a jeho výpisy je možné sledovať v konzole. Program bol vyvíjaný použitím open-source technológií.

Program Syncserv obsahuje nasledujúce moduly a funkcie:

- **list.c** - modul implementujúci údajovú štruktúru jednosmerného zoznamu. Obsahuje rovnaké funkcie ako zodpovedajúci modul programu MyBeem, uvedené v sekcii 1.1.
- **main.c** - hlavný súbor obsahujúci základnú funkciu *main()*.
  - *void signal_int(int signum)* - funkcia nastaví riadiacu premennú programu na *0*, čo spôsobí ukončenie vykonávania.

    Vstupné premenné:
    signum - číslo signálu, pri ktorého obsluhe je funkcia volaná.
  - *int main(int argc, char **argv)* - hlavná funkcia programu, ktorá spustí vykonávanie synchronizácie.

    Vstupné premenné:
    signum - číslo signálu, pri ktorého obsluhe je funkcia volaná.

    Výstupná hodnota:
    Vracia nulovú hodnotu.
- **sync.c** - modul vykonávania synchronizacie meracích bodov.
  - *uint64_t getCurrentTime(int precision)* - funkcia pre získanie aktuálneho času.

    Vstupné premenné:

precision - presnosť, s akou chceme získať aktuálny čas.

Výstupná hodnota:
0 v prípade neúspechu,
current - aktuálny čas v prípade úspechu.

– *int sync_thread(char *port)* - funkcia v cykle pre každú prijatú správu vloží svoju časovú známku a odošle späť synchronizačnému klientovi.

Vstupné premenné:
port - číslo portu, ktoré bude program Syncserv používať pre komunikáciu.

Výstupná hodnota:
0 v prípade úspechu, *-1* v prípade neúspechu.

3 Funkcia skriptu beem_adjuster.sh

Beem_adjuster.sh je skript typu bash a sluzi na generovanie mensej verzie, takzvanej verzie "šitej na mieru"programu MyBeem. Účelom tohto skriptu je zabezpečenenie možnosti prispôsobenia programu MyBeem k potrebám používateľa alebo k schopnostiam počítačového systeému.

Požiadavky nástroja adjuster:

- nainštalovaný balík libxml2-utils
- nástroj musí mať k dispozícii všetky zdrojové kódy, Makefile a config.xml

Proces generovania verzie "šitej na mieru"prebieha nasledovne:

- používateľ prostredníctom konfiguračného súboru config.xml si nastaví informačné elementy a doplňujúce služby (mediácia, synchronizácia, logovanie), ktoré v upravenej verzie chce mať k dispozícii. Za vypnutý informačný element

sa považuje ten, ktorého položka v konfiguračnom súbore je vykomentovaná alebo vymazaná. Za vypnutú službu sa považuje tá služba, ktorej prvý riadok položky je nastavený na hodnotu false.

- Po uložení zmien konfiguračného súboru sa má spustiť skript beem_adjuster.sh, ktorý pre každý zapnutý informačný element a zapnutú službu vygeneruje makro do hlavičkového súboru ipfix_infelems.h. V zdrojových kódoch časti prislúchajúce k jednotlivým informačným elementom a službám sú ohraničené podmienkou pre predspracovanie (#ifdef a #endif). Po skončení zápisu makier skript ešte spustí Makefile, aby sa skompilovali nové kódy programu MyBeem. Ak niektoré makro nie je definované v hlavičkovom súbore ipfix_infelems.h, preprocessor pri preklade časť vypnutého elementu alebo vypnutej služby vynechá.

Časti na vynechanie sa nachádzajú v nasledujúcich moduloch:

- **cache.c** – v tomto module sa vypočíta hodnota každého informačného elementu. Keď niektoré informačné elementy sú "vypnuté", tak sa časť na výpočet ich hodnôt vynechá.
- **export.c** – tento modul slúži na vytvorenie IPFIX správ a na ich export. Ak niektoré informačné elementy sú vypnuté, tak vloženie ich hodnoty do IPFIX správy sa nemôže vykona, preto predspracovač aj túto časť zdrojového kódu vynechá.
- **capture.c** – v tomto module sa inicializujú vlákna pre doplňujúce služby. Ak niektoré zo služieb sú vypnuté, ich časti sa vynechajú.
- **sync.c** – v tomto module sa nachádza sada funkcií, ktoré sa používajú len vtedy, keď služba synchronizácie je zapnutá.
- **debug.c** – v tomto module sa posielajú logovacie správy na server.
- **config.c** – v tomto module sa nachádza metóda, ktorým sa nakonfiguruje

logovací server.

4 Preklad programov

Táto kapitola predstavuje prekladov programu SyncServ a programu MyBeem a ich požiadavky na technické a programové prostriedky.

4.1 Zoznam zdrojových textov programu MyBeem

Program MyBeem obsahuje nasledujúce zdrojové texty:

`aggregation.c`

`aggregation.h`

`beem.c`

`beem.h`

`cache.c`

`cache.h`

`capture.c`

`capture.h`

`config.c`

`config.h`

`debug.c`

`debug.h`

`export.c`

`export.h`

`ipfix.c`

`ipfix.h`

`ipfix_def.h`

`ipfix_fields.h`

ipfix_infelems.h

list.c

list.h

MurmurHash64.c

MurmurHash.h

packetIdent.c

packetIdent.h

queue.c

queue.h

sampling.c

sampling.h

sync.c

sync.h

4.2 Zoznam zdrojových textov programu Syncserv

Program Syncserv obsahuje nasledujúce zdrojové texty:

list.c

list.h

main.c

main.h

sync.c

sync.h

4.3 Požiadavky na technické prostriedky pri preklade

Na preklad zdrojových textov postačuje počítač s procesorom architektúry i386/i686 alebo amd64. Výkon neovplyvňuje úspešnosť prekladu.

4.4 Požiadavky na programové prostriedky pri preklade

Pre úspešné skompilovanie zdrojových súborov a správne fungovanie programu sú potrebné:

- operačný systém s linuxovým jadrom verzie 2.4 a vyššej
- kompilátor zdrojových súborov jazyka C - gcc prípadne g++ verzie 3.3 alebo vyššej
- knižnica libpcap-dev verzie 0.8.3 alebo vyššej
- knižnica libxml2-dev verzie 2.6.23 alebo vyššej
- knižnica openssl verzie 0.9.1 alebo vyššej
- knižnica libsctp-dev verzie 1.0.9 alebo vyššej
- knižnica libxml2-utils verzie 2.7.8 alebo vyššej
- knižnica libndpi verzie 1.5.2 (vyššie verzie zatiaľ nie sú podporované)

4.5 Vlastný preklad

Pre korektnú kompiláciu zdrojových kódov programov je potrebné najprv nainštalovať podporné knižnice, pokiaľ ešte niesú v systéme nainštalované. Pre distribúciu Debian GNU/Linux je postup inštalovania podporných knižníc nasledovný:

```
$ apt-get install libpcap0.8 libpcap08-dev libxml2 libxml2-dev
$ apt-get install libssl-dev libsctp-dev libxml2-utils
```

Okrem týchto knižníc je potrebné aninštalovať aj knižnicu libndpi verzie 1.5.2 z SVN repozitára spoločnosti ntop. Inštalácia môže byť vykonaná nasledovne:

```
$ svn co https://svn.ntop.org/svn/ntop/trunk/nDPI/
apt-get install gawk gcc autoconf build-essential libtool
```

```
cd nDPI
sh autogen.sh
make
make install
export LD_LIBRARY_PATH="/usr/local/lib:$LD_LIBRARY_PATH"
```

Pre úspešné nainštalovanie knižníc je potrebný prístup k internetu a zároveň sú potrebné administrátorské práva k systému na ktorom prevádzame inštaláciu.

```
$ make -f Makefile
```

Predchádzajúcim príkazom sa vykonáva kompilácia oboch programov. Výsledkom je spustiteľný súbor s názvom **beem**, respektíve **syncserv**.

5 Náväznosť na iné programové produkty

Program MyBeem je implementáciou najnižšej vrstvy architektúry BasicMeter. Z toho vyplýva náväznosť na programy, ktoré sú implementáciami druhej alebo tretej vrstvy spomínanej architektúry. Program MyBeem má taktiež návaznosť na synchronizačný server nástroja BasicMeter, ktorým je program Syncserv. Táto náväznosť je obojstranná. Samostatne bežiaci program MyBeem, alebo Syncserv nemá veľký význam. Účel spĺňajú až v spojení s ostatnými vrstvami architektúry. Taktiež má MyBeem náveznosť na program syslog-ng, pri ktorého použití je MyBeem schopný zasielať celú paletu výpisov na syslog server, avšak tento program musí byť spustený zadaním parametra príkazového riadku MyBeem-u pri jeho spustení. Jedine tak je celý proces zasielania log správ na server zautomatizovaný bez ďalšej potreby zásahu do konfiguračného súboru syslogd klienta.

Referencie

[1] MONICA: *SLAmeter.* 2012. Dostupné na internete: <`http://wiki.cnl.sk/Monica/SLAmeter`>

[2] TREMKO, S.: *Meracie body pre nástroj SLA Meter: Bakalárska práca.* Košice: KPI FEI TUKE, 2012. 62 s.

[3] KECSEY, T.: *Konformita nástroja BasicMeter s architektúrou IPFIX: Bakalárska práca.* Košice: KPI FEI TUKE, 2010. 59 s.

[4] HUSIVARGA, Ľ.: *Identifikácia paketových párov: Bakalárska práca.* Košice: KPI FEI TUKE, 2008. 43 s.

[5] HUSIVARGA, Ľ.: *Meranie časových charakteristík sieťovej prevádzky: Diplomová práca.* Košice: KPI FEI TUKE, 2010. 64 s.

[6] KECSEY, T.: *Optimalizácia meracieho a exportovacieho procesu nástroja BasicMeter: Diplomová práca.* Košice: KPI FEI TUKE, 2012. 90 s.

[7] TREMKO, S.: *Redukcia informácií o IP tokoch za účelom zníženia záťaže monitorovacích systémov: Diplomová práca.* Košice: KPI FEI TUKE, 2014. 85 s.

[8] FARKAS, D.: *Hĺbková analýza paketov prostredníctvom protokolu IPFIX: Diplomová práca.* Košice: KPI FEI TUKE, 2015. 78 s.