Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky
Katedra počítačov a informatiky

Aplikačný rámec pre sprostredkovanie IPFIX správ v nástroji SLAmeter

Diplomová práca

Príloha A

SYSTÉMOVÁ PRÍRUČKA Mediator v1.0

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Vedúci práce:	Ing. Peter Feciľak, PhD.
Konzultant:	Ing. Adrián Pekár

Košice 2013 **Bc. Rastislav Kudla**

Copyright © 2013 Rastislav Kudla. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Text. A copy of the license can be found at http://www.gnu.org/licenses/fdl.html.

Zoznam obrázkov

Zoznam tabuliek

1 Funkcia programu

Program JXColl (Java XML Collector) slúži na zachytávanie a predspracovávanie informácii o tokoch v sieťach získané exportérom. Je súčasťou meracej architektúry BasicMeter, ktorý na základe nastavených parametrov konfiguračného súboru vie dané údaje ukladať do databázy alebo ich sprístupniť pomocou vlastného protokolu pre priame spracovanie (protokol ACP) používateľovi. Údaje uložené v databáze slúžia pre neskoršie vyhodnotenie prídavnými modulmi spomínanej meracej architektúry a sú v súlade s požiadavkami protokolu IPFIX. JXColl tiež generuje účtovacie záznamy, ktoré slúžia na analýzu používania siete konkrétnym používateľom z hľadiska IP adries, protokolov, portov a časových charakteristík. Program bol vytvorený Ľubošom Koščom, neskôr zoptimalizovaný a doplnený novými funkciami Michalom Kaščákom, Adriánom Pekárom a Tomášom Vereščákom.

2 Analýza riešenia

V roku 2011 sa na udalosti DEMONS IPFIX Interoperability Testing Event v Prahe zistilo viacero nedostatkov programu JXColl.

Jedným z nich je absencia podpory viacerých záznamov v akomkoľvek druhu sady. Merací a exportovací proces nástroja BasicMeter exportoval vždy najviac jeden záznam v sade a preto tento problém nebol nájdený skôr. Súčasný stav je neprípustný a je potrebné ho napraviť.

Ďalším problémom je preskakovanie informačných elementov s redukovaným kódovaním. Takéto kódovanie je prípustné pre celočíselné a desatinné dátové typy a na typy dateTimeSeconds a dateTimeMilliseconds. V súčasnom stave ich však JXColl nevie spracovať.

Niektoré dátové typy nie sú v programe JXColl implementované. Príkladom sú dá-

tové typy boolean, macAddress a octetArray. Dátové typy dateTimeMicroseconds a dateTimeNanoseconds nie sú dekódované správnym spôsobom. Tieto typy sú uložené vo formáte časovej známky NTP Timestamp. V JXColl sú v súčasnosti interpretované ako 64-bitové bezznamienkové číslo.

Niektoré informačné elementy môžu mať variabilnú veľkosť. Ide napríklad o elementy s dátovým typom string alebo octetArray. Pri príjme informačných elementov s variabilnou dĺžkou by v JXColl došlo k chybe, pretože taký element má v zázname šablóny uvedenú veľkosť 65535. Veľkosť je vtedy zakódovaná v prvom alebo v prvých troch bajtoch dátového záznamu od jeho aktuálnej pozície.

Problém sa prejavil aj pri dekódovaní organizáciou vytvorených informačných elementov, ktoré majú identifikátor totožný s niektorým štandardným IETF informačným elementom. JXColl mal informácie len o prvom výskyte informačného elementu s daným identifikátorom a nebral do úvahy číslo organizácie. Je potrebné zabezpečiť správne zaobchádzanie s organizáciou definovanými informačnými elementmi.

JXColl umožňuje príjem dát o tokoch len prostredníctvom transportného protokolu UDP. Je potrebné implementovať aj prenos pomocou protokolov TCP a SCTP. V súvislosti s touto úlohou je nutné rátať s odlišnými požiadavkami na správu šablón. Pre protokol UDP je potrebné zaviesť mechanizmus expirácie šabĺón a umožniť správnu funkčnosť viacerých exportérov na jednom počítači, evidenciu aj na základe portu exportéra. V súvislosti s protokolmi TCP a SCTP je potrebné implementovať podporu pre Template Withdawal správy a detekciu poškodených dát. V prípade príjmu poškodených dát, alebo nepovolenej operácie so šablónami je potrebné zrušiť spojenie, resp. asociáciu.

3 Popis programu

Jednotlivé časti programu sú umiestnené v nasledujúcich balíkoch:

- sk.tuke.cnl.bm.JXColl.export - triedy určené na export údajov do databázy alebo protokolom ACP
- sk.tuke.cnl.bm.JXColl.input - triedy slúžiace na príjem dát z exportérov
- sk.tuke.cnl.bm.JXColl.IPFIX - triedy s manuálnou implementáciou IPFIX
- sk.tuke.cnl.bm.JXColl.accounting - triedy účtovacieho modulu
- sk.tuke.cnl.bm.JXColl - hlavné triedy samotného programu
- sk.tuke.cnl.bm.OWD - balík pre modul merania jednosmerného oneskorenia
- sk.tuke.cnl.bm - pomocné triedy a výnimky

3.1 Popis riešenia

V prvom rade bolo potrebné umožniť podporu viacerých záznamov v sade. Trieda NetXMLParser bola rozdelená na dve samostatné jednotky. Jednou je vlákno UDPProcessor, ktorého úlohou je vyberať údaje z PacketCache a časť spracovania bole prepísaná do novej triedy IpfixParser. Parsovanie IPFIX správ bolo zoptimalizované a upravené do prehľadnej podoby. Celkový problém bol rozdelený na menšie úlohy, ktoré sú realizované metódami na parsovanie správy, sady, a všetkých druhov záznamov. V metóde slúžiacej na spracovanie sady bol pre každý druh sady pridaný cyklus, ktorý vyberá záznamy zo sady, až kým sa pozícia v buffri nedostane na koniec sady.

Dekódovanie dát jednotlivých informačných elementov bolo presunuté do samostatnej triedy. Podobne ako v triede IpfixParser je volaná jedna metóda, ktorá na základe dátového typu presunie spracovanie na metódy zaoberajúce sa danou skupinou dá-

tových typov. Údaje z buffra pre dátový typ octetArray sú prevedené do podoby reťazce v kódovaní Base64 aby ich bolo možné uložiť v databáze. Bajt v buffri pre dátový typ boolean je interpretovaný ako true pri hodnote 1 a ako false pri hodnote 0. Všetky ostatné hodnoty znamenajú chybu a je to oznámené volajúcej metóde výnimkou DataException. Okrem implementácie dekódovania dátového typu macAddres bola ešte opravená interpretácia dátových typov dateTimeMicroseconds a dateTimeNanoseconds. Keďže sa tieto hodnoty musia zmestiť do ôsmich bajtov vo formáte NTP Timestamp a analyzujúce aplikácie sú postavené na technológii Java, ktorá nepozná bezznamienkové typy, je vhodné uložiť tieto hodnoty v nezmenenej podobe ako znamienkové 64-bitové číslo (long). Po získaní čísla z databázy je možné jednoducho použiť triedu Timestamp z knižnice Apache Commons Net na jednoduchšiu prácu s touto časovou známkou.

Na podporu redukovaného kódovania bola vytvorená metóda handleReducedSizeEncoding(), ktorá zo skráteného informačného elementu vytvorí buffer plnej veľkosti, ktorý je možné bežným spôsobom interpretovať. Pre bezznamienkové čísla sa do plnej veľkosti zľava doplnia nulové bajty. Pre záporné znamienkové čísla sa doplnia bajty o hodnote 255, aby sa zachovala hodnota čísla. Redukovaný môže byť aj dátový typ float64, a to použitím polovičného počtu bajtov. Vtedy sa dáta interpretujú ako float32.

Organizáciou definované informačné elementy sú definované v súbore ipfixFields.xml. Aby sa uvažovalo číslo organizácie pre informačné elementy, musela byť doplnená trieda IpfixElements o možnosť evidencie informačných elementov nielen na základe ich identifikátora, ale súčasne s číslom organizácie (štandardné informačné elementy sú uvažované s číslom organizácie 0). Spoločne s tým všetky metódy slúžiace na prístup k informáciám o informačných elementoch museli byť rozšírené o parameter čísla organizácie.

Variabilné informačné elementy boli implementované v triede IpfixParser. Pri výbere informačného elementu, ktorého veľkosť je v zázname šablóny definovaná ako 65535

je vybratý prvý bajt z dátovej časti na aktuálnej pozícii. Ak je tento bajt menší ako 255, je vybraných toľko bajtov z buffra, koľko tento bajt uvádza. Ak má bajt hodnotu 255, dĺžka je získaná z ďalších dvoch bajtov.

Súčasťou zmien v parseri je detekcia poškodených správ. Ak je skutočná veľkosť správy iná ako sa uvádza v hlavičke, ide o chybu. Pri každej chybe je vyhodená výnimka DataFormatException. Chyba je indikovaná aj keď je počet zostávajúcich bajtov v buffri menší ako veľkosť sady. Pri výbere údajov z dátového záznamu alebo záznamu šablóny sa testuje náraz na pravú hranicu sady. V prípade jej dosiahnutia sa tiež ohlási chyba.

Na pridanie podpory expirácie šablón musela sa do záznamu šablóny pridať položka času príchod šablóny. Cache bola sprehľadnená a pred parsovaním dátového záznamu je overovaná platosť šablóny. V prípade ak je neplatná, dátová sada sa preskočí. Každých 10 minút sa spúšťa čistiace vlákno, aby vymazalo expirované šablóny a zabránilo tak zaplňovaniu pamäte JXColl. Transportné protokoly TCP a SCTP používajú vlastnú cache na uschovanie šablón s platnosťou len pre konkrétne pripojenie, resp. asociáciu. Umožňená bola podpora pre zrušenie šablón pomocou Template Withdrawal správ. V prípade, ak exportér pošle už kolektorom evidovanú šablónu, alebo ak ruší neexistujúcu šablónu, spojenie, resp. asociácia sa preruší.

Protokoly TCP a SCTP využívajú na každé pripojenie samostatné vlákno. Obidva protokoly počúvajú na nakonfigurovanom porte a počet pripojení, resp. asociácií je ohraničený nastavením v konfiguračnom súbore. Na podporu protokolu SCTP bolo potrebné prejsť na novšiu verziu Javy 1.7.

3.2 Popis tried, členských premenných a metód

Kedže sa počas vývoja a odstraňovania chýb pôvodné triedy a ich metódy nezmenili, v nasledujúcich častiach budú uvedené len tie, ktoré boli doplnené počas riešenia jednotlivých problémov uvedených v analýze riešenia. Popis ostatných tried a metód je uvedený v príručkach predošlých verzií programu.

3.2.1 Trieda SCTPProcessor

Trieda predstavuje vlákno spracúvajúce jednu SCTP ascociáciu.

Konštruktor

public ***SCTPProcessor****(SctpChannel channel, InetSocketAddress exporterAddress, ThreadGroup group)*

Konštruktor uloží predané parametre do členských premenných, vytvorí vlastnú cache typu SingleSessionTemplateCache a vytvorí nový objekt typu IpfixParser, ktorému predá vytvorenú template cache. Nastaví meno vlákna, bude obsahovať IP adresu a port.

Parametre:

channel - Objekt slúžiaci na prácu s asociáciou - príjem a posielanie správ
exporterAddress - primárna adresa SCTP ascociácie obsahujúca aj jeho zdrojový port
group - skupina vlákien, ku ktorej toto vlákno patrí.

Metódy

public void ***run****()*

Hlavná metóda vlákna. Pokiaľ sa vláno nepreruší, dochádza k prijímaniu správ z SCTP asociácie a spracovanie pomocou vlastného parsera. Po prerušení vlákna sa asociácia vypne a vlákno končí svoju činnosť.

3.2.2 Trieda SCTPServer

Slúži ako TCP server. Prijíma požiadavky na vytvorenie SCTP asociácií. Nové asociácie predáva vláknam typu SCTPProcessor.

Konštruktor

public ***SCTPServer****()*

Konštruktor inicializuje SctpServerChannel, nastaví mu blokovací režim a priviaže ho k portu definovanom v konfiguračnom súbore. Nastaví meno vlákna.

Metódy

public void ***run****()*

Hlavná metóda vlákna. Pokiaľ nedôjde k prerušeniu, čaká na vytvorenie pripojenia exportérom. Vytvorený SctpChannel predá novovytvorenému vláknu SCTPProcessor a spustí ho. Dovoľuje pripojiť sa len toľkým exportérom, ako je uvedené v konfiguračnom súbore.

public void ***interruptAllThreads****()*

Táto metóda preruší všetky bežiace vlákna typu SCTPProcessor. Je volaná pri prerušení tohto vlákna.

3.2.3 Trieda TCPProcessor

Trieda predstavuje vlákno spracúvajúce jedno TCP pripojenie.

Konštruktor

public ***TCPProcessor****(Socket socket, InetSocketAddress exporterAddress, ThreadGroup group)*

Konštruktor uloží predané parametre do členských premenných, vytvorí vlastnú cache typu SingleSessionTemplateCache a vytvorí nový objekt typu IpfixParser, ktorému predá vytvorenú template cache. Nastaví meno vlákna, bude obsahovať IP adresu a port.

Parametre:

channel - Objekt slúžiaci na prácu s asociáciou - príjem a posielanie správ

exporterAddress - primárna adresa SCTP ascociácie obsahujúca aj jeho zdrojový port

group - skupina vlákien, ku ktorej toto vlákno patrí.

Metódy

public static void ***run****()*

Hlavná metóda vlákna. Pokiaľ sa vláno nepreruší, dochádza k prijímaniu správ zo socketu a spracovaniu pomocou vlastného parsera. Po prerušení vlákna sa asociácia vypne a vlákno končí. Dôjde k násilnému ukončeniu spojenia pomocou RST segmentu.

public void ***closeConnection****()*

Zatvorí socket.

public void ***resetConnection****()*

Zapne na sockete SO_LINGER a nastaví ho na 0 sekúnd. Potom socket zatvorí. Dôjde k poslaniu RST segmentu.

3.2.4 Trieda TCPServer

Slúži ako TCP server. Prijíma požiadavky na vytvorenie TCP spojenia. Nové asociácie predáva vláknam typu TCPProcessor.

Konštruktor

public ***TCPServer****()*

Konštruktor inicializuje ServerSocket, nastaví mu timeout a priviaže ho k portu definovanom v konfiguračnom súbore. Nastaví meno vlákna.

Metódy

public void ***run****()*

Hlavná metóda vlákna. Pokiaľ nedôjde k prerušeniu, čaká na vytvorenie pripojenia exportérom. Vytvorený socket predá novovytvorenému vláknu TCPProcessor a spustí ho. Dovoľuje pripojiť sa len toľkým exportérom, ako je uvedené v konfiguračnom súbore.

public void ***interruptAllThreads****()*

Táto metóda preruší všetky bežiace vlákna typu TCPProcessor. Je volaná pri prerušení tohto vlákna.

3.2.5 Trieda UDPProcessor

Trieda predstavuje vlákno spracúvajúce UDP pakety, ktoré boli uložené v PacketCache.

Konštruktor

public ***UDPProcessor****()*

Konštruktor nastaví meno vlákna.

Metódy

public void ***run****()*

Hlavná metóda vlákna. Vyberá dáta z PacketCache a predáva ich parseru. V prípade zachytenia výnimky DataFormatException sa aktuálne spracovávaná správa preskočí.

3.2.6 Trieda UDPServer

Slúži ako UDP server. Prijíma UDP datagramy zo siete na nakonfigurovanom porte a ukladá ich do PacketCache.

Konštruktor

public ***UDPServer****()*

Konštruktor inicializuje DatagramChannel, nastaví mu blokovací režim a priviaže ho k portu definovanom v konfiguračnom súbore. Nastaví meno vlákna.

Metódy

public void ***run****()*

Hlavná metóda vlákna. Prijíma pakety zo siete a ukladá ich do PacketCache spoločne s časom prijatia a zdrojového UDP portu. Ak je vlákno prerušené, tak končí.

3.2.7 Trieda TemplateHolder

Trieda sa používa ako úložisko šablón pre jeden UDP exportér identifikovaný pomocou IP adresy, čísla zdrojového portu a identifikátora pozorovacej domény. V prípade použitia spojovo-orientovaného transportného protokolu je identifikátorom len číslo pozorovacej domény.

Konštruktor

public ***TemplateHolder****()*

Vytvorí kolekciu HashMap kde kľúčom je číslo šablóny a hodnotou je záznam šablóny, IPFIXTemplateRecord.

Metódy

public void ***addTemplate****(IPFIXTemplateRecord template)*

Pridá záznam šablóny do evidencie.

Parametre:

template - záznam šablóny

public boolean ***contains****(int templateId)*

Overí dostupnosť záznamu šablóny s daným identifikátorom.

Parametre:

templateId - identifikátor šablóny

public HashMap¡Integer, IPFIXTemplateRecord¿ ***getTemplates****()*

Vráti kolekciu HashMap obsahujúcu všetky šablóny pre tento exportér.

Návratová hodnota:

Kolekcia HashMap obsahujúca všetky evidované šablóny.

public IPFIXTemplateRecord ***get****(int templateId)*

Získa záznam šablóny s daným identifikátorom.

Návratová hodnota:

Objekt záznamu šablóny.

public IPFIXTemplateRecord ***remove****(int templateId)*

Vymaže záznam šablóny s daným identifikátorom.

Návratová hodnota:

Vymazaný záznam šablóny.

public void ***removeAll****()*

Vymaže všetky šablóny z evidencie.

public boolean ***isEmpty****()*

Zistí, či je evidencia šablón pre daný exportér prázdna.

Návratová hodnota:

Pravda, ak exportér neobsahuje žiadnu šablónu, inak nepravda.

3.2.8 Trieda IpfixUdpTemplateCache

Úložisko záznamov šablón pre jednotlivé exportéry. Využíva sa len na správu šablón pri prenose cez protokol UDP. Používa návrhový vzor Singleton.

Konštruktor

public ***IpfixUdpTemplateCache****()*

Vytvorí inštanciu triedy. Inicializuje kolekciu HashMap<ExoporterKey, TemplateHolder> obsahujúcu evidenciu šablón jednotlivých exportérov.

Metódy

public IpfixUdpTemplateCache ***getInstance****()*

Získa jedinú inštanciu tejto triedy. Návrhový vzor Singleton.

Návratová hodnota:

Inštancia tejto triedy

public void ***addTemplate****(IPFIXTemplateRecord template, InetAddress ipfixDevice, int exporterUdpSrcPort, long odid)*

Pridá alebo nahradí záznam šablóny v úložisku daného exportéra, ktorý je identifikovaný IP adresou, zdrojovým UDP portom a číslom pozorovacej domény.

Parametre:

template - záznam šablóny
ipfixDevice - IP adresa exportéra
exporterUdpSrcPort - zdrojový UDP port exportéra
odid - identifikátor pozorovacej domény

public boolean ***addTemplate****(IPFIXTemplateRecord template, InetSocketAddress ipfixDevice, long odid)*

Pridá alebo nahradí záznam šablóny v úložisku daného exportéra, ktorý je identifikovaný IP adresou, zdrojovým UDP portom a číslom pozorovacej domény.

Parametre:

template - záznam šablóny
ipfixDevice - IP adresa a zdrojový UDP port exportéra
odid - identifikátor pozorovacej domény

public TemplateHolder ***getTemplates****(InetAddress exporterAddress, int udpSourcePort, long odid)*

Vráti objekt TemplateHolder obsahujúci všetky šablóny evidované pre daný exportér.

Parametre:

exporterAddress - IP adresa exportéra
udpSourcePort - zdrojový UDP port exportéra
odid - identifikátor pozorovacej domény

Návratová hodnota:

Objekt TemplateHolder obsahujúci všetky evidované šablóny pre daný exportér.

public boolean ***contains****(int templateId, InetSocketAddress ipfixDevice, long odid)*

Zistí, či sa šablóna exportéra nachádza v úložisku pre daný exportér.

Parametre:

templateId - identifikátor šablóny
ipfixDevice - IP adresa a zdrojový port exportéra
odid - identifikátor pozorovacej domény

Návratová hodnota:

True, ak šablóna je evidovaná, inak false.

public boolean ***checkForPresence****(int templateId, InetSocketAddress ipfixDevice, long odid)*

Zistí, či sa šablóna exportéra nachádza v úložisku pre daný exportér a či je platná.

Parametre:

templateId - identifikátor šablóny
ipfixDevice - IP adresa a zdrojový port exportéra
odid - identifikátor pozorovacej domény

Návratová hodnota:

True, ak šablóna je evidovaná a je platná, inak false.

public IPFIXTemplateRecord ***getByID****(int templateId, InetSocketAddress ipfixDevice, long odid)*

Získa šablónu z úložiska daného exportéra danú jej číslom.

Parametre:

templateId - identifikátor šablóny
ipfixDevice - IP adresa a zdrojový port exportéra
odid - identifikátor pozorovacej domény

Návratová hodnota:

Záznam šablóny vyhovujúci zadaniu.

public void ***setUpCleaningTask****()*

Nastaví plánovač, aby spúšťal špeciálne vlákno každých 10 minút. Toto vlákno bude prehľadávať celé úložisko šablón, a ak nájde expirovanú šablónu, tak ju vymaže. Ak to bude posledná šablóna pre daný exportér, vymaže z evidencie aj tento exportér. Toto vlákno zabraňuje prepĺňaniu pamäte JXColl.

public void ***cancelCleaningTask****()*

Zruší vykonávanie čistiaceho vlákna. Volá sa pri ukončovaní programu JXColl.

3.2.9 Trieda IpfixSingleSessionTemplateCache

Úložisko záznamov šablón pre konkrétny exportér. Používa sa pri použití spojovo-orientovaného transportného protokolu.

Konštruktor

public ***IpfixSingleSessionTemplateCache****()*

Prázdny konštruktor.

Metódy

public void ***addTemplate****(IPFIXTemplateRecord template, long odid)*

Pridá alebo nahradí záznam šablóny pre danú pozorovaciu doménu.

Parametre:

template - záznam šablóny
odid - identifikátor pozorovacej domény

public TemplateHolder ***getTemplates****(long odid)*

Vráti objekt TemplateHolder obsahujúci všetky šablóny pre danú pozorovaciu doménu.

Parametre:

odid - identifikátor pozorovacej domény

Návratová hodnota:

Objekt TemplateHolder obsahujúci všetky evidované šablóny tohto exportéra pre danú pozorovaciu doménu.

public boolean ***contains****(int templateId, long odid)*

Zistí, či je evidovaná šablóna s daným identifikátorom a pozorovacou doménou.

Parametre:

templateId - identifikátor šablóny
odid - identifikátor pozorovacej domény

Návratová hodnota:

True, ak šablóna je evidovaná, inak false.

public IPFIXTemplateRecord ***get****(int templateId, long odid)*

Získa šablónu s daným identifikátorom a pozorovacou doménou.

Parametre:

templateId - identifikátor šablóny
odid - identifikátor pozorovacej domény

Návratová hodnota:

Záznam šablóny vyhovujúci zadaniu.

public void ***remove****(int templateId, long odid)*

Zmaže šablónu s daným identifikátorom a pozorovacou doménou. Používa sa pri príjme Template Withdrawal Message

Parametre:

templateId - identifikátor šablóny
odid - identifikátor pozorovacej domény

Návratová hodnota:

Záznam šablóny vyhovujúci zadaniu.

*public void **removeAll**(long odid)*

Zmaže všetky šablóny s danou pozorovacou doménou. Používa sa pri príjme All Data Template Withdrawal Message, alebo All Options Template Withdrawal Message

Parametre:

templateId - identifikátor šablóny
odid - identifikátor pozorovacej domény

Návratová hodnota:

Záznam šablóny vyhovujúci zadaniu.

3.2.10 Trieda ExporterKey

Predstavuje kľúč exportéra. Identifikuje ho na základe IP adresy, čísla zdrojového UDP portu a čísla pozorovacej domény.

Konštruktor

*public **ExporterKey**(InetAddress ipfixDevice, int exporterSrcUdpPort, long observationDomainId)*

Priradí predané premenné atribútom objektu.

Parametre:

ipfixDevice - IP adresa exportéra

exporterSrcUdpPort - číslo zdrojového portu exportéra

observationDomainId - identifikátor pozorovacej domény

Metódy

public InetAddress ***getIpfixDevice****()*

Získa IP adresu exportéra

Návratová hodnota:

IP adresa exportéra.

public InetAddress ***getExporterSrcUdpPort****()*

Získa zdrojový UDP port exporéra.

Návratová hodnota:

Číslo zdrojového portu exportéra.

public InetAddress ***getObservationDomainId****()*

Získa pozorovaciu doménu exportéra.

Návratová hodnota:

Identifikátor pozorovacej domény exportéra.

3.2.11 Trieda IpfixDecoder

Trieda so statickými metódami slúžiacimi na dekódovanie dát z dátového záznamu.

Metódy

public String ***decode****(String type, ByteBuffer buffer)*

Dekóduje dátový typ obsiahnutý v buffri do podoby reťazca podľa špecifikácie IPFIX. Priamo nevykonáva dekódovanie, volá konkrétne metódy podľa kategórie dátového typu.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeUnsignedIntegralType****(String type, ByteBuffer buffer)*

Dekóduje celočíselné bezznamiekové dátové typy unsigned8, unsigned16, unsigned32 a unsigned64. Berie ohľad na skrátené dátové typy.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeSignedIntegralType****(String type, ByteBuffer buffer)*

Dekóduje celočíselné znamienkové dátové typy signed8, signed16, signed32 a signed64. Berie ohľad na skrátené dátové typy.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeFloatType****(String type, ByteBuffer buffer)*

Dekóduje desatinné dátové typy float32 a float64. Berie ohľad na skrátené dátové typy.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeAddressType****(String type, ByteBuffer buffer)*

Dekóduje dátové typy obsahujúce adresy: ipv4Address, ipv6Address a macAddress.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeBooleanType****(ByteBuffer buffer)*

Dekóduje boolean reprezentujúci pravdivostnú hodnotu.

Návratová hodnota:

Reťazec reprezentujúci pravdivostnú hodnotu, ”true” alebo ”false”.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ, alebo ak obsahuje inú hodnotu ako 0 alebo 1
UnsupportedDataException - Ak dátový typ nie je podporovaný

public String ***decodeStringType****(ByteBuffer buffer)*

Dekóduje dáta v buffri ako reťazec v kódovaní UTF-8.

Návratová hodnota:

Reťazec v kódovaní UTF-8.

public String ***decodeOctetArrayType****(ByteBuffer buffer)*

Dáta v buffri prevedie na reťazec do kódu Base64.

Návratová hodnota:

Reťazec predstavujúci binárne dáta zakódované v Base64.

public String ***decodeDateTimeType****(String type, ByteBuffer buffer)*

Dekóduje dátové typy časových známok: dateTimeSeconds, dateTimeMilliseconds, dateTimeMicroseconds a dateTimeNanoseconds.

Parametre:

type - reťazec definujúci dátový typ obsiahnutý v buffri.
buffer - samotné dáta, ktoré sú predmetom dekódovania

Návratová hodnota:

Reťazec reprezentujúci interpretovanú hodnotu buffra na základ predaného typu. Dátové typy dateTimeSeconds a dateTimeMilliseconds predstavujú počet sekúnd, resp. milisekúnd od Unix epochy (00:00 1.1.1970 UTC). Dátové typy dateTimeMicroseconds a dateTimeNanoseconds sú zakódované vo formáte časovej známky NTP Timestamp. Berie sa do úvahy redukované kódovanie prvých dvoch menovaných typov.

Hádže:

DataException - Ak je buffer nesprávnej veľkosti vzhľadom na dátový typ
UnsupportedDataException - Ak dátový typ nie je podporovaný

public ByteBuffer ***handleReducedSizeEncoding****(byte[] input, int arraySize, boolean isSigned)*

Vytvorí zo vstupného poľa bajtov buffer stanovenej dĺžky.

Parametre:

input - vstupné pole bajtov obsahujúce dáta skráteného informačného elementu.
arraySize - veľkosť informačného elementu podľa definície v informačnom modeli IPFIX.
isSigned - ak je dátový typ dát vo vstupnom poli bajtov znamienkové číslo, táto hodnota by mala byť true.

Návratová hodnota:

ByteBuffer obsahujúci informačný dáta informačného elementu o štandardnej veľkosti.

public ByteBuffer ***parseMacAddress****(byte[] input)*

Konvertuje bajty vo vstupnom poli bajtov na reťazec v tvare XX:XX:XX:XX:XX:XX.

Parametre:

input - vstupné pole bajtov obsahujúce dáta skráteného informačného elementu.
arraySize - veľkosť informačného elementu podľa definície v informačnom modeli IPFIX.
isSigned - ak je dátový typ dát vo vstupnom poli bajtov znamienkové číslo, táto hodnota by mala byť true.

Návratová hodnota:

ByteBuffer obsahujúci informačný dáta informačného elementu o štandardnej veľkosti.

3.2.12 Trieda IpfixParser

Táto trieda sa používa na parsovanie IPFIX správ a ich spracovanie.

Konštruktor

public ***IpfixParser****(IpfixUdpTemplateCache templateCache)*

Tento konštruktor indikuje použitie protokolu UDP. Počas spracovania správ sa budú metódy vykonávať touto vetvou.

Parametre:

templateCache - Úložisko šablón pre transportný protokol UDP.

public ***IpfixParser****(IpfixSingleSessionTemplateCache templateCache, TransportProtocol protocol)*

Tento konštruktor indikuje použitie protokolu TCP alebo SCTP. Počas spracovania správ sa budú metódy vykonávať touto vetvou.

Parametre:

templateCache - Úložisko šablón pre transportný protokol TCP alebo SCTP.
protocol - určuje, o aký transportný protokol ide.

Metódy

public IPFIXMessage ***parseIpfixMessage****(ByteBuffer buffer, InetSocketAddress ipfixDevice, long time)*

Spracuje IPFIX správu obsiahnutú v buffri. Pokiaľ sa nedôjde na koniec správy, volá metódu parseIpfixSet().

Parametre:

buffer - buffer obsahujúci samotnú správu
ipfixDevice - IP adresa a port exportéra
time - čas príchodu správy

Návratová hodnota:

Objekt IPFIX správy obsahujúci všetky svoje štruktúry.

Hádže:

DataFormatException - Ak je správa zlej veľkosti alebo ak sú dáta inak poškodené.
TemplateException - Ak nastala nepovolená operácia v mechanizme správy šablón.

public IPFIXSet ***parseIpfixSet****(IPFIXMessage message, ByteBuffer buffer, InetSocketAddress ipfixDevice)*

Spracuje IPFIX sadu obsiahnutú v buffri. Pokiaľ sa nedôjde na koniec sady, volá metódy parseIpfixDataRecord(), parseIpfixTemplateRecord() alebo parseIpfixOptionsTemplateRecord() v závislosti od druhu sady.

Parametre:

message - objekt IPFIX správy
buffer - buffer obsahujúci samotnú správu
ipfixDevice - IP adresa a port exportéra

Návratová hodnota:

Objekt IPFIX sady obsahujúci všetky svoje záznamy.

Hádže:

DataFormatException - Ak je pre sadu nie je dostatok miesta v buffri alebo ak sú dáta inak poškodené.
TemplateException - Ak nastala nepovolená operácia v mechanizme správy šablón.

public IPFIXDataRecord ***parseIpfixDataRecord****(IPFIXMessage message, IPFIXSet ipfixSet, ByteBuffer buffer, InetSocketAddress ipfixDevice)*

Spracuje IPFIX dátový záznam obsiahnutý v buffri. Získa šablónu podľa čísla sady a získava informačné elementy na základe informácií zo šablóny. Ak je šablóna pri prenose cez UDP expirovaná, preskočí danú sadu.

Parametre:

message - objekt IPFIX správy
ipfixSet - objekt IPFIX sady

buffer - buffer obsahujúci samotnú správu
ipfixDevice - IP adresa a port exportéra

Návratová hodnota:

Objekt IPFIX dátového záznamu.

Hádže:

DataFormatException - Ak je pre dátový záznam nie je dostatok miesta v buffri alebo ak sú dáta inak poškodené.
TemplateException - Ak nastala nepovolená operácia v mechanizme správy šablón.

public IPFIXTemplateRecord ***parseIpfixTemplateRecord****(IPFIXMessage message, IPFIXSet ipfixSet, ByteBuffer buffer, InetSocketAddress ipfixDevice)*

Spracuje IPFIX záznam šablóny obsiahnutý v buffri. Získa všetky špecifikátory polí.

Parametre:

message - objekt IPFIX správy
ipfixSet - objekt IPFIX sady
buffer - buffer obsahujúci samotnú správu
ipfixDevice - IP adresa a port exportéra

Návratová hodnota:

Objekt IPFIX záznamu šablóny.

Hádže:

DataFormatException - Ak je pre záznam šablóny nie je dostatok miesta v buffri alebo ak sú dáta inak poškodené.

TemplateException - Ak nastala nepovolená operácia v mechanizme správy šablón.

public IPFIXOptionsTemplateRecord ***parseIpfixTemplateRecord****(IPFIXMessage message, IPFIXSet ipfixSet, ByteBuffer buffer, InetSocketAddress ipfixDevice)*

Spracuje IPFIX záznam šablóny možností obsiahnutý v buffri. Získa všetky špecifikátory polí.

Parametre:

message - objekt IPFIX správy
ipfixSet - objekt IPFIX sady
buffer - buffer obsahujúci samotnú správu
ipfixDevice - IP adresa a port exportéra

Návratová hodnota:

Objekt IPFIX záznamu šablóny.

Hádže:

DataFormatException - Ak je pre záznam šablóny nie je dostatok miesta v buffri alebo ak sú dáta inak poškodené.
TemplateException - Ak nastala nepovolená operácia v mechanizme správy šablón.

4 Preklad programu

4.1 Zoznam zdrojových textov

Zdrojové texty sú k dispozícii v prílohe bakalárskej práce.

Sú k dispozícii tieto zdrojové texty:

- balík sk.tuke.cnl.bm:
  - ACPIPFIXTemplate.java
  - DataException.java
  - DataFormatException.java
  - Filter.java
  - InetAddr.java
  - InvalidFilterRuleException.java
  - JXCollException.java
  - Sampling.java
  - SimpleFilter.java
  - TemplateException.java
  - Templates.java
- balík sk.tuke.cnl.bm.JXColl:
  - Config.java
  - IJXConstants.java
  - IpfixDecoder.java
  - IpfixElements.java
  - JXColl.java
  - NetConnect.java
  - PacketCache.java
  - PacketObject.java
  - RecordDispatcher.java
  - Support.java
- balík sk.tuke.cnl.bm.JXColl.export:
  - ACPServer.java
  - ACPIPFIXWorker.java
  - DBExport.java
  - PGClient.java
- balík sk.tuke.cnl.bm.JXColl.accounting:
  - AccountingManager.java
  - AccountingRecord.java
  - AccountingRecordsCache.java
  - AccountingRecordsExporter.java

```
- balík sk.tuke.cnl.bm.JXColl.accounting:
   OWDCache.java
   OWDFieldSpecifier.java
   OWDFlushCacheABThread.java
   OWDListener.java
   OWDObject.java
   OWDTemplateCache.java
   OWDTemplateRecord.java
   Synchronization.java
- balík sk.tuke.cnl.bm.JXColl.IPFIX:
   ExporterKey.java
   FieldSpecifier.java
   IPFIXDataRecord.java
   IPFIXMessage.java
   IPFIXOptionsTemplateRecord.java
   IPFIXSet.java
   IpfixSingleSessionTemplateCache.java
   IpfixUdpTemplateCache.java
   TemplateHolder.java
```

4.2 Požiadavky na technické prostriedky pri preklade

Preklad programu si vyžaduje nasledovnú hardvérovú konfiguráciu:

- CPU Intel Pentium III 1Ghz alebo ekvivalent
- operačná pamäť 512MB
- pevný disk s 1GB voľného miesta
- grafická karta novej generácie s minimálne 64MB pamäťou
- sieťová karta 100Mb/s

4.3 Požiadavky na programové prostriedky pri preklade

- operačný systém GNU/Linux s verziou jadra 2.6 a vyššou (odporúča sa kvôli podpore SCTP)

- Java Runtime Environment (JRE) verzie 1.7.0_03 a vyššej
- balík lksctp-tools
- knižnice dodávané na inštalačnom médiu

4.4 Náväznosť na iné programové produkty

Program umožňuje ukladanie dát do databázy alebo ich sprístupnenie priamym pripojením, ktoré budú následne vyhodnotené príslušnými prídavnými modulmi. Je implementáciou zhromažďovacieho procesu architektúry BasicMeter. Z toho vyplýva jeho náväznosť na merací a exportovací proces - BEEM, alebo iné implementácie.

4.5 Vlastný preklad

Preklad programu spočíva v nakopírovaní zdrojových súborov a spustení kompilátora jazyka Java s potrebnými parametrami a parametrom classpath nastaveným na prídavné knižnice. Odporúča sa použiť váš obľubený java IDE, kde stačí jednoducho nastaviť verziu JDK na 7.0 alebo vyššie a do cesty classpath pridať cesty ku všetkým potrebným knižniciam. V prostredí Netbeans IDE potom stačí kliknúť na tlačidlo *Clean and Build*.

4.6 Vytvorenie inštalačného DEB súboru

Stačí spustiť skript `buildDeb.sh`, ktorý sa nachádza v priečinku jxcoll/deb.

```
sh buildDeb.sh
```

Výstupom tohto skriptu je súbor s názvom `debian.deb`, ktorý môžme následne premenovať podľa verzie JXColl (napríklad na jxcoll_3.9_i386.deb). Tento skript vykonáva nasledovné:

1. v prípade, ak neexistuje priečinok debian, extrahuje ho z archívu debian.tar.gz, inak tento krok preskočí
2. v prípade, ak neexistuje priečinok debian, extrahuje ho z archívu debian.tar.gz, inak tento krok preskočí
3. skopíruje binárny súbor z projektu do DEB balíčka (predpokladá sa, že bol program kompilovaný v Netbeans IDE pomocou Clean and Build tlačidla)
4. skopíruje konfiguračný súbor z projektu do DEB balíčka
5. skopíruje IPFIX definičný súbor z projektu do DEB balíčka
6. vymaže prípadné dočasné súbory z DEB balíčka
7. vygeneruje MD5 kontrolné súčty pre všetky súbory DEB balíčka
8. zabezpečí maximálnu kompresiu manuálových stránok a changelog súborov
9. skopíruje binárny súbor z projektu do DEB balíčka a nastaví mu práva na vykonávanie
10. vytvorí samotný DEB balíček
11. overí ho pomocou programu lintian - ten vypíše prípadne varovania a/alebo chyby
12. archivuje vytvorený DEB balíček do archívu debian.tar.gz

Pred spustením skriptu je nutné skompilovať JXColl pomocou Netbeans IDE tlačidlom *Clean and Build*. Prípadné zmeny control alebo changelog súboru, manuálových stránok je nutné vykonať ručne. Manuálové stránky je vhodné upraviť pomocou programu *GmanEdit* . Po spustení skriptu sa vytvorí DEB balíček s názvom `debian.deb`. Ten je vhodné premenovať podľa aktuálnej verzie. Vytvorí sa aj archív `debian.tar.gz`, ktorý obsahuje najaktuálnejšiu adresárovú štruktúru DEB balíčka pre budúce využitie (ak neexistuje priečinok debian, vytvorí sa extrakciou z tohto archívu). Ak je potrebné len aktualizovať kód, stačí spustiť skript a ten sa o všetko

postará, pričom vytvorí aj adresár debian. Súbory možno v ňom upravovať až kým nie je všetko podľa predstáv. Ak je všetko hotové, v Netbeans IDE je potrebné vymazať priečinok debian (vykoná sa SVN DELETE, namiesto obyčajného odstránenia zo súborového systému) a projekt "commitnúť".

4.7 Opis známych chýb

V súčasnosti nie sú známe žiadne vážne chyby.

5 Zhodnotenie riešenia

Hlavným cieľom práce bolo zvýšiť interoperabilitu s inými IPFIX riešeniami pomocou zvýšenia konformity so štandardom IPFIX. V práci boli vyriešené problémy, ktoré doteraz znemožňovali dekódovanie viacerých záznamov sade, informačncých elementov s variabilnou dĺžkou, informačných elementov s redukovaným kódovaním alebo niektorých predtým neimplementovaných dátových typov.

Súčasťou práce bolo rozšírenie podpory prenosu údajov o tokoch prostredníctvom transportných protokolov TCP a SCTP, čo zvyšuje možnosti nasadenia nástroja BasicMeter aj v podmienkach s vyššou náchylnosťou na preťaženie v sieti.

Možnosti budúceho vývoja zhromažďovacieho procesu nástroja BasicMeter predstavuje implementácia podpory pre dátové typy umožňujúce prenos štruktúrovaných dát a podpora pre zabezpečené pripojenia od exportérov.

6 Zoznam použitej literatúry

[1] Koščo, M.: Opis sieťových protokolov prostredníctvom jazyka XML, 2005, Diplomová práca, KPI FEI TU, Košice

[2] Kaščák, M.: Príspevok k problematike aplikačného využitia meraní prevádzkových parametrov počítačových sietí, 2007, Diplomová práce, KPI FEI TU, Košice

[2] Pekár, A.: Meranie prevádzkových parametrov siete v reálnom čase, 2009, Bakalárska práca, KPI FEI TU, Košice

[3] Vereščák, T.: Zhromažďovací proces nástroja BasicMeter, 2010, Bakalárska práca, KPI FEI TU, Košice

[4] Pekár, A.: Optimalizácia zhromažďovacieho procesu nástroja BasicMeter, 2011, Diplomová práca, KPI FEI TU, Košice