VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PROFILOVANÍ SÍŤOVÉ KOMUNIKACE

NETWORK PROFILING

PROJEKTOVÁ PRAX 1

AUTOR PRÁCE **ROMAN DOBIÁŠ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. PETR MATOUŠEK, Ph.D.,MA**
SUPERVISOR

BRNO 2017

Obsah

Kapitola 1

Úvod

Táto práca sa zaoberá problematikou sieťového profilovania systémov z hľadiska návrhu, implementácie a využitia vlastného softwarového riešenia k vytvoreniu profilov rôznych počítačových systémov.

Vzhľadom na historický vývoj, počítačové (a vrámci nich aj sieťové) systémy sa postupom času stávali komplexnejšie, robustnejšie a tým sa z pohľadu ľudského faktora znižovala prehľadnosť pri spravovaní takýchto sietí. Tento vývoj prirodene viedol k požiadavke na nové nástroje. V oblasti sietí vniklo sieťové profilovanie – vytváranie akého si logu, popisujúceho deje v sieti medzi uzlami.

Informácie následne môžu využívať bezpečnostní správcovia, ktorých záujmom môže byť odhalenie podozrivej alebo nežiadúcej aktivity v sieti, manažéri, ktorí podľa rozloženia aktivity v sieti môžu lepšie naplánovať rozšírenie siete [4].

Nástroje, ktoré pre tieto účely vznikli, pristupovali k sieti len ako k množine komunikujúcich uzlov bez ohľadu na ich vnútorné procesy, ktoré stoja za komunikáciou. Pridanie hlbšej úrovne sledovania uzlov by ale mohlo byť výhodou v rôznych aplikáciach, napríklad pri detekcií útokov alebo šíreniu škodlivých súborov už na základe prítomnosti istých aplikácii v sieťovej komunikácii.

Cieľom tejto práce je preto využiť už existujúce riešenia a rozšíriť ich o zaznamenávanie procesov, pracujúcich na vybraných uzloch.

V úvodnej kapitole je definovaná problematika sieťového profilovania a predstavené existujúce dostupné riešenia, ktoré sa využívajú pre sledovanie sieťovej aktivity procesov.

V nasledujúcej kapitole je stručne popísaná technológia *NetFlow* od spoločnosti *Cisco* a možnosti jej využitia pre potreby profilovania.

Predposledná kapitola sa zaoberá návrhom a implementáciou softwarového riešenia pre účely profilovania.

Posledná kapitola predstavuje myšlienky spojené s rozšírením a zdokonalením systému a s prípadnými plánmi do budúcnosti.

Kapitola 2

Profilovanie

Profilovanie sieťovej komunikácie je proces, pri ktorom sa pre konkrétnu počítačovú sieť vytvorí jej profil - akýsi log udalostí, ktoré sa vyskytli počas prevádzky siete, asociované ku konkrétnym zariadeniam (uzlom) v sieti a ich parametre.

Pre vytvorenie profilu je nutné analýzovať jednotlivé prenosy, ktoré tvoria **toky dát** (*ang. Flow*) medzi jednotlivými uzlami v sieti. Pre účely profilovania nie sú podstatné samotné dáta, ktoré sú prenášané tokmi, ale kľúčové informácie, popisujúce charakter samotného toku - **metadáta**. Medzi skúmané vlastnosti patria napr. komunikujúce adresy a porty, typ prenosového protokolu (obvykle na úrovni L4 [3]) a veľkosť prenesených dát.

Metadáta je následne možné použiť pre rôzne účely:

- účtovanie prenesených dát jednotlivých uzlov v sieti
- detekcia škodlivého správania a útokov v sieti (DDOS [1])
- monitorovanie záťaže a dimenzovanie siete

2.1 Existujúce nástroje

Počas štúdia sme sa oboznámili už s existujúcimi nástrojmi pre platformu Windows, ktorých funkcionalita aspoň čiastočne pokrýva profilovanie procesov.

2.1.1 Microsoft Message Analyzer

Program[2] umožňujúci odchytávanie a analýzu packetov, vznikajúcich na úrovni sieťovej karty. Nástroj umožňuje prehliadať užitočné dáta packetov a detekovať aplikačný protokol prenosu. Nástroj nie je vhodný na vytváranie profilov, pretože pracuje na úrovni transakcií, z ktorých sa skladajú toky a neumožňuje pohľad na systém z vrstvy tokov.

2.1.2 NetLimiter4

NetLimiter[3] je nástroj primárne určený k limitovaniu objemu dát prenesených jednotlivými aplikáciami. Okrem toho nástroj umožňuje sledovať počet prenesených dát jednotlivými procesmi v aktuálnom a celkovom čase, generovať jednoduché grafy a exportovať data do

[1] Distributed denial-of-service
[2] MS Message Analyzer - www.microsoft.com/en-us/download/details.aspx?id=44226
[3] NetLimiter4 - www.netlimiter.com/

`CSV` formátu. Takto vytvorené dáta sú ale príliš stručné a nie sú dostatočné pre vytváranie profilov (napr. neposkytujú množinu komunikačných protokolov, komunikujúcich adries pre každý proces).

2.1.3 GlassWire

Program [4]poskytuje obdobnú funkcionalitu ako program NetLimiter, z pohľadu profilovania ale umožňuje vytvárať jednoduchú štatistiku komunikujúcich adries a protokolov k daným procesom.

2.1.4 Zhodnotenie nástrojov

Vyššie spomenuté nástroje je možné charakterizovať nasledujúcou tabuľkou:

Vlastnosť	MS Message Analyzer	NetLimiter4	GlassWire
Podpora rôznych operačných systémov	MS Windows	MS Windows	MS Windows
Licencia	Freeware	Shareware	Shareware
Sledovanie sieťovej aktivity procesov	Obmedzene	Áno	Áno
Export štatistík	Áno	Áno	Áno

Nástroje, zmienené v tejto kapitole, sú nevyhovujúce k vytváraniu komplexných profilov sietí. Neposkytujú podbrobné informácie o procesoch a majú tendenciu vytvárať grafové štatistiky, ktoré môžu byť využiteľné v obchodných stykoch, ale sú nepraktické pre automatizovanú prácu. Zároveň sú to neslobodné riešenia, ktorých funkcionalita a prenositeľnosť je značne limitovaná na konkrétnu platformu.

2.2 NetFlow

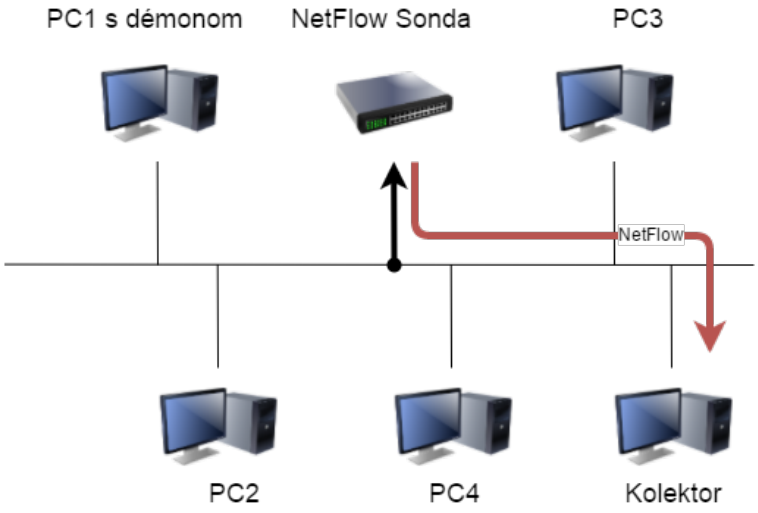
Jedným z voľne dostupných a populárnych riešení pre profilovanie siete je technológia *NetFlow* od spoločnosti *Cisco* [1]. V nej je tok definovaný ako jednosmerné prúdenie dát medzi zdrojovým a koncovým uzlom na úrovni *L4*, popísaný vlastnosťami ako je typ protokolu, čas začatia a ukončenia toku, celkový počet prenesených bajtov [2]. Každý tok je rozlíšený podľa trojice (adresa, port, protokol) zdrojového a koncového uzlu. Od verzie *NetFlow* v9 je technológia implementovaná s dôrazom na možnosť rozšíriť základné štandardné metadáta o vlastné položky a z tejto verzie vychádza štandard *IPFIX*, popisujúci proces vytvorenia, prenosu a uchovania záznamov o tokoch.

V praxi je systém *NetFlow* tvorenými dvoma nie nutne rôznymi uzlami – **sondou** a **kolektorom**. Sonda je softwarový alebo hardwarový podsystém, realizujúci zber metadát z toku, ktorý prechádza jej sieťovým rozhraním. Získané dáta sú preposielané v štruktúrovanej podobe k ľubovoľnému počtu kolektorov. Kolektor ako podsystém archivuje, prípadne spracúva prijaté dáta.

Nevýhodou vyššie popísaného riešenia je pohľad na uzol v sieti ako na čiernu skrinku (*ang. black box*), ktorej vnútorne procesy sú pre okolie skryté. Pre komplexnú znalosť a profilovanie systému by bolo vhodné preniknúť v metadátach až na úroveň jednotlivých procesov vo sledovaných uzloch a zohľadniť ich v štatistikách. Keďže technológia *NetFlow*

[4]GlassWire - www.glasswire.com

Obr. 2.1: Priklad komunikácie podsystémov *NetFlow*, kde sonda preposiela informácie kolektoru.



umožňuje flexibilne definovať typ metadát, je výhodné využiť jej existujúce rozhranie a doplniť chýbajúce z informácie z dodatočného zdroja. Podľa miesta, z ktorého zasiahneme do procesu *NetFlow*, môžeme rozlišovať nasledujúce scenáre:

2.2.1 Rozšírenie *NetFlow* sondy

Najjednoduchšie riešenie z pohľadu zložitosti realizácie spočíva v doplňaní informácii už pri vzniku metadát, priamo v *NetFlow* sonde. Vybrané softwarové riešenia sond (napr. *NProbe*) poskytujú možnosť rozšíriť funkcionalitu sondy na základe **zásuvných modulov** a predom definovaných rozhraní. V praxi by doplnenie dát spočívalo vo vytvorení špeciálneho pluginu, ktorý by pri vzniku toku doplnil tok o názov procesu, pripojeného na konkrétny port a protokol. Medzi nevýhody tohto prístupu patrí naviazanosť na konkrétne (často neslobodne šírené) softwarové riešenie, nedostatok kvalitnej dokumentácie sond a hotových rozšírení pre jednotlivé sondy, a prenositeľnosť viazaná na dostupnosť sondy na jednotlivých operačných systémoch.

2.2.2 Proxy *NetFlow* collector

Koncepcia sonda-kolektor nabáda k riešeniu vložením medzičlánku – kolektora, ktorý by preposielal upravené dáta k finálnemu kolektoru. Realizácia uvedeného postupu je podmienená vlastnou implementáciou komunikačného protokolu kolektora.

2.2.3 Úprava archivovaných *NetFlow* záznamov

Táto metóda je založená na paralelnom uchovávaní metadát o procesoch v separátnom systéme a *NetFlow* metadát, a následnom spojení oboch záznamov do jedného na základe časových známok. Nevýhoda riešenia spočíva vo zvýšenej réžii spojenej s dočasným uchovávaním dát o procesoch v separátnom súbore. Je takisto nutné navrhnúť systém na automatizované doplnenie chýbajúcich údajov do záznamov.

Kapitola 3

Návrh systému

3.1 Popis systému

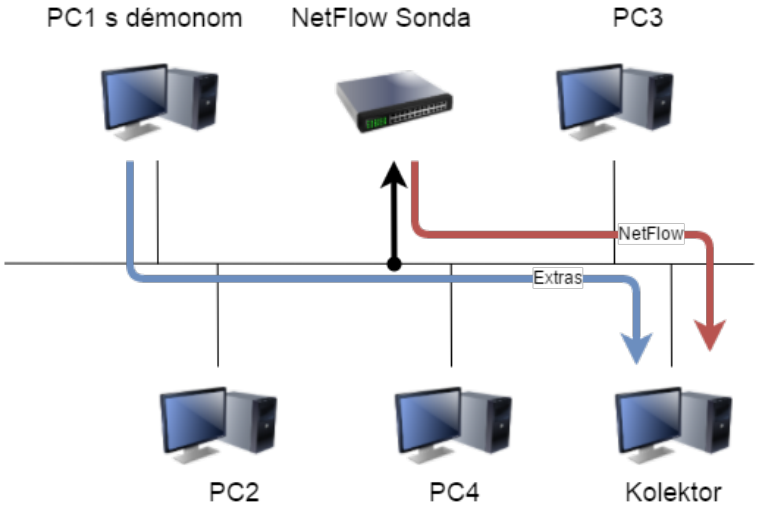
Naše riešenie vychádza zo scenára úpravy dát až po ich archivovaní. Zvolená metóda sa javí ako najvýhodnejšia, pretože navrhnutý systém nie je bytostne závislý neslobodných produktov tretích strán a je prakticky nasaditeľný pri použití ľubovoľnej NetFlow sondy.

Systém je postavený na klasickom *NetFlow* modelu sonda-kolektor, doplnenom o podsystémy pre zber dát (*ProcessTracker*) a doplňovanie do *NetFlow* záznamov (*FlowUpdater*). Komunikácia oboch podsystémovje realizovaná na sieťovej schéme server-klient.

ProcessTracker je serverový démon, rezidentný na konkrétnom sieťovom uzli, ktorého procesy nás zaujímajú, ktorý vytvára snímky (*ang. Snapshot*) sieťového rozhrania v periodických intervaloch. Snapshot predstavuje množinu usporiadaných štvoríc typu adresa, port, protokol, názov programu, reprezentujúcu procesy viazané na sieťové rozhrania konkrétneho počítača. V časovom horizonte uchováva démon usporiadanú množinu snímkov, zoradených podľa času výskytu.

FlowUpdater je démon typu klient, pracujúci popri *NetFlow* kolektore. Jeho úlohou je priebežne alebo v dobe doplnenia získavať dáta od *ProcessTrackera* pomocou sieťového rozhrania a rozširovať záznamy kolektoru.

Obr. 3.1: Znázornenie systému *NetFlow*, rozšíreného o dodávanie názvov procesov.



Záznamy vytvorené kolektorom sú rozšírené o data, pochádzajúce z *ProcessTrackera*. Snímky, ktoré *FlowUpdater* získa od *ProcessTrackera* aj NetFlow záznamy, uložené v súboroch na disku, majú charakter lineárnych zoznamov, kde jednotlivé položky nasledujú tak, ako nastali v čase. Činnosťou *FlowUpdatera* je potom sekvenčne prechádzať oba zoznami tak, aby približne sedel čas snímku s časom výskytu události a týmto spôsobom dopĺňať záznami o ich priliehajúce názvy procesov zo snímku.

Navrhnutý systém v sebe nesie niekoľko parametrov, ktoré ovplyvňujú výkonnosť a náročnosť jeho chodu. Na úrovni *ProcessTrackera* môžeme uvažovať o časovom rozlíšení, ktoré je definované frekvenciou záznamu snímky. Príliš vysoká frekvencia snímkovania vedie ku zvýšeným pamäťovým a výpočtovým nárokom. Naopak, príliš nízka frekvencia môže mať za následok stratu krátkodobých tokov, ktorých trvanie je kratšie ako perióda snímkovania. Obdobne môžeme uvažovať o frekvencii dopĺňania na strane *ProcessTrackera*. Príliš nízka frekvencia môže spôsobiť hromadenie redundantných dát v pamäti a tým pádom zvýšené nároky na pamäť. V praxi nám ale nízke rozlíšenie snímkov neprekáže, pretože významné toky v sieti z povahy trvajú dlšie, ako niekoľko sekúnd.

Nie príliš zrejmý je dopad komunikácie medzi *ProcessTracker* a *FlowUpdater*. Komunikácia medzi oboma prvkami môže byť vedená po samostatnom kanáli, v praxi je ale súčasťou bežnej sieťovej komunikácie a v prípade rozsiahlych prenosov môže mať negatívne dopady na výkonnosť nie len aplikácii na sledovanej stanici ale aj zvyšných uzlov v sieti. Najsilnejšie sa tento neduh prejaví pri skokovom prenose nahromadených dát, preto je vhodné uvažovať o rozložení komunikácie na menšie, čiastočné prenosy.

3.2 Implementácia riešenia

Pre implementáciu riešenia sme si vybrali sondu *NProbe* od spoločnosti *NTop*, ktorá je v demo verzii verejne prístupná na stránkach spoločnosti a pre účely kolektoru aplikáciu *nfcapd* z knižnice *flow-tools*. Pre prácu so archivovanými záznamami slúži nástroj *nfdump* [1], využitý na zobrazovanie a filtrovanie dát a knižnica *libnf* [2], vytvorená na fakulte, použitá pre úpravu záznamov. Vlastné programovanie podsystémov je realizované v jazyku C. Riešenie je prioritne vyvíjané pre operačný systém *Linux* a jeho štandardné rozhranie.

3.2.1 Komunikácia podsystémov

Podsystémy spolu komunikujú cez sieť pomocou prenosového protokolu *UDP*, ktorý umožňuje vytvoriť jednoduchý komunikačný model so schémou požiadavka-odpoveď. V medziach interpretácie je požiadavkou ľubovoľný diagram, ktorý klient (*FlowUpdater*) pošle na odpovedajúcu adresu a port servera (*ProcessTracker*). Server následne odpovedá diagramom, ktorého dáta začínajú hlavičkou (serializovaná binárna štruktúra `struct query_msg`, obsahujúca údaj o počte záznamov v diagrame) a polom serializovaných inštancii štruktúry `data_t`.

Pre zvýšenie abstrakcie a prehľadnosti kódu bol vytvorený jednoduchý modul (*ang. wrapper*), ktorý obaluje štandardné systémové volania sieťových funkcii do zjednodušeného rozhrania, umožňujúceho vytvorenie inštancii *UDP* servera a klienta, čítanie a zápis dát. Jednotlivé funkcie rozhrania sú neblokové, čo umožňuje kombinovať sieťovú komunikáciu s ostatnými činnosťami bez využitia ďalších vlákien.

[1] http://nfdump.sourceforge.net/
[2] libnf - wwww.github.com/VUTBR/libnf

3.2.2 Implementácia ProcessTrackeru

Podsystém *ProcessTracker* je server postavený na protokole `UDP`, ktorý vykonáva nekonečnú slučku s funkciou `serverClient()` a periodicky volá funkciu `updateBindings()`.

Inicializácii nekonečnej smyčky predchádza spustenie skriptu `hosts.sh`, ktorý slúži pre zistenie dostupných IPv4 a IPv6 adries sieťových rozhraní. Skript je založený na invokácii programu `ip`, ktorého výstup sa pomocou programu *grep* pretriedi tak, aby zostali len riadky s adresami. Následne sa výstup transformuje na adresy, oddelené znakom nového riadku. *AWK*. Volanie skriptu z programu prebieha pomocou štandartnej funkcie `popen` [3], ktorá umožňuje pracovať s výstupom skriptu ako s UNIXovým súborom.

Pre každé volanie funkcie `updateBindings()` je volaný skript `bindings.sh`, ktorý prevedie výstup programu *netstat*[4] do formátu `CSV` [5] opätovne pomocou *AWK*. Výstupom skriptu je sekvencia riadkov v usporiadanom tvare $(protokol, adresa, port, program)$. Výstup skriptu je získaný volaním funkcie `popen` a spracuváva sa po riadkoch.

Úlohou funkcie `updateBindings()` je detekovať, či sa nezmenilo priradenie programov pre jednotlivé porty rozhraní. Pre uchovanie informácii o procesoch je použitá vyhľadávacia tabuľka, implementovaná ako hash tabuľka, ktorej kľúčom je trojica $(adresa, port, protokol)$ a má aktualizačnú sémantiku (uchováva unikátne záznamy). V každom volaní funkcie `updateBindings()` je pre každú štvoricu, získanú skriptom `bindings.sh` indexovaná tabuľka a následne sa porovnáva reťazec, reprezentujúci názov programu priradeného ku kľúču. V prípade nezhody reťazcov je tabuľka aktualizovaná a na koniec fronty sa pridá nová entita, popisujúca zmenu.

Na koniec fronty sa pridá záznam, obsahujúci štvoricu $(adresa, port, protokol, program)$. V prípade, že program naslúchal na ľubovoľné sieťové rozhranie, čo je možné detekovať ak adresa má tvar `0.0.0.0`, potom je potrebné záznam replikovať pre každú dostupnú sieťovú adresu, ktorú sme získali pomocou skriptu `hosts.sh`, a samostatne vložiť do fronty.

Vo funkcii `serverClient()` sú pri požiadavke na server informácie z fronty serializované do statického pola s hlavičkou a prepreposlané klientovi.

3.2.3 Implementácia FlowUpdateru

Podsystém *FlowUpdater* je realizovaný formou `UDP` klienta, ktorý je rezidentný počas behu kolektora. Podsystém má vlastnú inštanciu fronty a vyhľadávacej tabuľky a po štarte má obe dátové štruktúry prázdne. Pri každej rotácii súborov, ktorú vykoná kolektor, smeruje na *ProcessTracker* požiadavka pre záznamy, ktorými sa doplní fronta. Pomocou knižnice *libnf* sa sekvenčne prechádza súbor, obsahujúci *NetFlow* záznamy a z vrcholu fronty sa odoberajú prvky tak, aby čas záznam na vrchole fronty mal neskoršie časové známku ako práve doplňovaný *NetFlow* záznam. Záznamy odstraňované z fronty sa využijú na aktualizovanie vyhľadávacej tabuľky. Následne sa údajé z doplňovaného prvku použijú pre vyhľadanie názvu programu, ktorý sa doplní do *NetFlow* záznamu. V prípade, že sa pre daný kľúč v tabuľke položka nenachádza, použije sa konštantný reťazec označujúci absenciu procesu. Pre uchovanie názvu procesu sa používa pole `LNF_FLD_USERNAME`, definované v knižníci *libnf*, ktoré je typu reťazec, nutný pre uchovanie názvu.

Pretože *NetFlow* záznamy programu *nfdump* sú binárne súbory, ktorých položky sú bloky dát uložené sekvenčne v súbore, nie je možné pridať k položke nový stĺpec priamo

[3] popen - www.man7.org/linux/man-pages/man3/popen.3.html
[4] netstat - wwww.linux.die.net/man/8/netstat
[5] Comma separated values

v originálnom súbore, ale je potrebné pri dopĺňaní záznamov vytvárať nový výstupný súbor. Z dôvodu označenia už spracovaného súboru je nutné pôvodný súbor zmazať alebo premenovať.

V súčasnosti je implementácia projektu v integračnej a ladiacej fáze.

3.2.4 Využitie systému

Výsledkom práce podsystému *FlowUpdater* je nový stĺpec `USERNAME` v záznamoch kolektora, obsahujúci názov procesu alebo reťazec XXX v prípade chýbajúcej asociácie.

Keďže program *nfdump* v štandardnom prevedení podporuje len fixné polia záznamu, pre zobrazovanie a filtráciu záznamov sa využíva upravená verzia programu *nfdumpp* z projektu *libnf*. Výstupom programu je text, preto pre spracovanie záznamov je možné využiť klasické nástroje, napr. *grep* z balíčka *GNU Coreutils* pre vyhľadanie záznamov obsahujúcich daný proces.

Obr. 3.2: Rozšírené záznamy *NetFlow* o stĺpec `USERNAME`, obsahujúce názov procesu, zodpovedom za komunikáciu.

username	proto	srcip	srcport	dstip	dstport
XXX	17	8.8.8.8	53	10.0.2.15	18122
939/dnsmasq	17	10.0.2.15	18122	8.8.4.4	53
XXX	17	8.8.4.4	53	10.0.2.15	18122
XXX	1	10.0.2.15	0	8.8.4.4	0
939/dnsmasq	17	10.0.2.15	6152	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	6152
939/dnsmasq	17	10.0.2.15	1103	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	1103
939/dnsmasq	17	10.0.2.15	33412	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	33412
939/dnsmasq	17	10.0.2.15	28609	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	28609
939/dnsmasq	17	10.0.2.15	26217	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	26217
939/dnsmasq	17	10.0.2.15	7509	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	7509
939/dnsmasq	17	10.0.2.15	48090	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	48090
939/dnsmasq	17	10.0.2.15	40743	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	40743
939/dnsmasq	17	10.0.2.15	10963	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	10963
939/dnsmasq	17	10.0.2.15	16526	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	16526
939/dnsmasq	17	10.0.2.15	42519	8.8.8.8	53
XXX	17	8.8.8.8	53	10.0.2.15	42519

3.3 Plány do budúcna

Napriek funkčnosti vyššie popísaného riešenia existuje veľa problémov, ktoré by zabraňovali potencionálnemu nasadeniu aplikácie v neexperimentálnom prostredí.

3.3.1 Spracovanie získaných dát

Pre otestovanie funkčnosti riešenia by bolo vhodné nasadiť systém v reálnej sieťovej prevádzke. Testovanie by spočívalo v meraní prenesených dát jednotlivých aplikácii a v následnom porovnávaní s výsledkami dosiahnutými iným riešením. Zaujímavým parametrom by mohla byť miera dopĺňania, definovaná ako počet doplnených ku celkovému počtu NetFlow tokov.

3.3.2 Zabezpečenie komunikácie

V aktuálnom štádiu predstavuje systém možné bezpečnostné riziko vzhľadom na komunikáciu medzi *ProcessTrackerom* a *FlowUpdater*, ktorá prebieha nešifrovane (*ang. Plaintext*). Ľubovoľný uzol v sieti, cez ktorý by prechádzala sieťová komunikácia, by bol schopný pasívne tieto informácie odchytiť a po krátkom rozbore pravdepodobne aj interpretovať. Z tohto dôvodu je nutné zaviesť šifrovanie medzi podsystémami.

Prípadný problém predstavuje aj absencia autentifikácie medzi *ProcessTrackerom* a serverom. Keďže *ProcessTracker* v tomto momente obslúži ľubovoľného klienta, pri odhalení naslúchajúceho *ProcessTrackera* môže dochádzať k úniku informácii o procesoch a taktiež k cieľavedomému narušaniu procesu dopĺňania. Jednoduchým riešením je obmedziť rozsah, prípadne zaviesť zoznam povolených (*ang. Whitelist*) sieťových adries. Obecným riešením je rozšírenie komunikácie o autentifikačné prvky, konkrétne overenie prístupového hesla.

Z hľadiska stability komunikácie je súčasný systém náchylný na chyby vzhľadom na absenciu zabezpečenia dát a nespolahlivosť protokolu *UDP*. Problém je možné riešiť pridaním zabezpečovacieho kontrolného súčtu (napr. CRC32 alebo MD5 checksum), pre detekciu konzistentnosti paketov po prijatí *FlowUpdaterom*.

3.3.3 Prenositeľnosť

Súčasné riešenie je obmedzené na beh na platforme Linux. Obmedzenie je spôsobené odlišnosťou operačných systémov pri práci s procesmi a absenciou univerzálneho aplikačného programovacieho rozhrania (*ang. API*) pre ich identifikáciu. Pre zabezpečenie prenositeľnosti je nutné do podsystému zaviesť štandardné rozhranie pre získanie informácii o procesoch a následne implementovať toto rozhranie pre jednotlivé operačné systémy. V poslednom rade môže existovať problém s licenciami pri použití nášho riešenia v aplikáciach. V súčasnosti je pre *NetFlow* sondu využitý propertiarny program *NProbe* od spoločnosti *NTop*.

3.3.4 Efektívnosť

Pri nasadení reálnych programov sa skúma ich náročnosť na systémové zdroje a vplyv na funkčnosť a flexibilnosť. Obdobne je možné uvažovať o efektívnosti a náročnosti nášho riešenia.

Pre zníženie objemu prenášaných dát medzi *ProcessTrackerom* a *FlowUpdaterom* je možné využiť komprimáciu dát. K tomuto účelu je vhodné využiť už existujúce otvorené knižnice pre kompresiu a dekompresiu dát, medzi ktoré patrí napr. *zlib* [6] alebo *lz4* [7].

[6] zlib - www.zlib.net
[7] lz4 - www.lz4.github.io/lz4/

Kapitola 4

Záver

Výsledkom tejto práce je implementácia vlastného informačného systému, rozširujúcom systém *NetFlow*, ktorý je vhodný pre vytváranie profilov sieťovej aktivity zariadení s dôrazom na aktivitu vnútorných procesov vybraného zariadenia. V dokumente je popísaná samotná technológia *NetFlow*, nutnosť aj spôsob jej rozšírenia.

Práca podrobne popisuje návrh z pohľadu kooperácie systémov a následne približuje podstatné aspekty z vytvorenej implementácie systému.

Vzhľadom na fakt, že výsledok práce je skorej funkčný koncept než reálny systém, sú na konci dokumentu predostrené možnosti možného pokračovania práce, hlavne s dôrazom na zvýšenie kvality a spolahlivosti riešenia, rozvíjajúc nutné opatrenia pre možné reálne nasadenie systému.

V prípade pokračovania práce je následne možné vytvoriť profily počítačových systémov na existujúcich systémoch s dôrazom na účtovanie prenesených dát jednotlivých systémov.

Literatúra

[1] *Introduction to Cisco IOS NetFlow - A Technical Overview*. [Online; navštíveno 26.01.2017].
URL http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[2] *NetFlow - Wikipedia*. [Online; navštíveno 26.01.2017].
URL http://en.wikipedia.org/wiki/NetFlow

[3] *Transport layer - Wikipedia*. [Online; navštíveno 26.01.2017].
URL http://en.wikipedia.org/wiki/Transport_layer

[4] Whisnant, A.; Faber, S.: Network Profiling Using Flow. Technická zpráva, Carnegie Mellon University, 2012, [Online; navštíveno 26.01.2017].
URL http://www.sei.cmu.edu/reports/12tr006.pdf