VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

KORELACE IPFIX ZÁZNAMŮ Z PROVOZU PROXY SERVERŮ

CORRELATING IPFIX RECORDS OF PROXY SERVER TRAFFIC

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. MICHAL KRŮL**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. ONDŘEJ RYŠAVÝ, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Krůl Michal, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Počítačové sítě
Název: **Korelace IPFIX záznamů z provozu proxy serverů**
Correlating IPFIX Records of Proxy Server Traffic
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s principy proxy serverů a nastudujte si relevantní protokoly (např. SOCKS5, HTTP).
2. Nastudujte si technologii NetFlow/IPFIX pro monitorování síťového provozu s přihlédnutím na jeho implementaci v systémech Flowmon.
3. Vytvořte datovou sadu obsahující reprezentativní vzorek záznamů z komunikace před a za proxy serverem.
4. Navrhněte způsob, jakým je možné IPFIX záznamy korelovat. Uvažujte i případ zřetězení více proxy serverů za sebou.
5. Navrhnuté řešení implementujte a otestujte jeho funkčnost.
6. Proveďte vyhodnocení vytvořeného řešení a diskutujte jeho parametry.

Literatura:

- R. Hofstede et al., "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," in IEEE Communications Surveys & Tutorials, vol. 16, no. 4, pp. 2037-2064, Fourthquarter 2014, doi: 10.1109/COMST.2014.2321898.
- V. Bajpai and J. Schönwälder, "Network Flow Query Language-Design, Implementation, Performance, and Applications," in IEEE Transactions on Network and Service Management, vol. 14, no. 1, pp. 8-21, March 2017, doi: 10.1109/TNSM.2016.2633511.
- X. Zeng et al., "Flow Context and Host Behavior Based Shadowsocks's Traffic Identification," in IEEE Access, vol. 7, pp. 41017-41032, 2019, doi: 10.1109/ACCESS.2019.2907149.
- Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D., and L. Jones, "SOCKS Protocol Version 5", RFC 1928, DOI 10.17487/RFC1928, March 1996.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Ryšavý Ondřej, doc. Ing., Ph.D.**
Konzultant: Holkovič Martin, Ing., UIFS FIT VUT
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 14. října 2021

Abstrakt

V této práci je rozebrán problémem korelace záznamů síťových toků proxy serverů. Hledáno je řešení, které by dovolilo automatizovaným způsobem určovat související toky na obou stranách proxy serveru. Pro tyto účely je vytvořena datová sada s odchycenou síťovou komunikací, nad kterou je následně provedena analýza. Na výsledcích analýzy je následně vystavěno řešení, které je dále testováno a diskutováno.

Abstract

This thesis engages the problem of correlation the network flow records. It tries to find solution, which would allow to automatically pinpoint correlating flows on both sides of the proxy server. For this purpose, a dataset containing captured network traffic is created, which then serves as a base for analysis. Based on the results of the analysis a solution is presented, which is consequently tested and discussed.

Klíčová slova

proxy server, síťový tok, záznam síťové komunikace, export síťového toku, IPFIX, korelace síťového provozu, korelační metody

Keywords

proxy server, network flow, network communication record, export of network flow, IPFIX, correlation of the network traffic, correlation methods

Citace

KRŮL, Michal. *Korelace IPFIX záznamů z provozu proxy serverů.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Ondřej Ryšavý, Ph.D.

Korelace IPFIX záznamů z provozu proxy serverů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Ondřeje Ryšavého, Ph.D. Zadání, další informace a software pro vypracování mi poskytl Ing. Martin Holkovič a firma Flowmon Networks. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Krůl
18. května 2022

Poděkování

Tímto bych chtěl poděkovat panu doc. Ondřeji Ryšavému za vedení práce a věcné připomínky. Dále také panu Ing. Martinu Holkovičovi za praktický dohled nad prací, časté konzultace a vedení projektu. V neposlední řadě bych také rád poděkoval firmě Flowmon Networks za poskytnutí zadání.

Obsah

Kapitola 1

Úvod

Proxy servery jsou jednou z možností, jak skrýt a chránit komunikaci ve velkých sítích typu Internet. Díky své povaze mohou být velmi účinným nástrojem pro ochranu identity uživatelů koncových sítí, zamezení útoků na takové sítě, ale také pro efektivní hospodaření s přiděleným adresovým prostorem. Případně slouží jako mezistupeň pro rozložení zátěže na koncové servery, které nezvládnou obsloužit větší množství konkurenčních připojení. V moderní síťové struktuře jsou zařízení, které fungují jako proxy, neopomenutelným účastníkem síťových komunikací. Na trhu existuje velké množství řešení, které nabízí software fungující jako proxy, buďto jednoduchý, s určitým řízením přístupu anebo kombinovaný s firewallem. Proxy servery mohou pro různé účely využívat domácnosti, firmy, vzdělávací instituce a také poskytovatelé internetových služeb, aplikací anebo vůbec přístupu k internetu. Může se však také stát, že budou využity pro přesměrování provozu a provádění útoků z praktické anonymity. V každém případě se mohou vyskytnout důvody, kvůli kterým by bylo užitečné moci spojit k sobě části komunikace před a za proxy serverem.

V této práci je blíže rozveden problém korelace záznamů toků, které jsou exportovány z odchycené síťové komunikace vznikající při využití proxy serverů. Korelací je v tomto kontextu myšleno vzájemné přiřazení různých záznamů toků, které se vyskytují na různých stranách proxy serverů. Problémem korelace záznamů toků je tedy myšlen problém nalezení záznamů toků, u kterých je možné nalézt logickou spojitost komunikace mezi dvěma koncovými body v síti. Řešeny jsou i případy, kdy komunikace prochází přes více proxy serverů. Obecně je tedy hledána série vzájemně logicky navazujících záznamů síťových toků.

Záznamy síťové komunikace jsou v této práci uměle vytvořeny a dále zkoumány s cílem zjistit, zda je možné pouze na základě informací v nich obsažených přiřadit jednotlivé záznamy toků k sobě a vytvořit z nich záznam odpovídající celé komunikaci mezi koncovými stanicemi. Řešení problému korelace je hledáno za účelem jeho následného použití v praxi, a tudíž by mělo být nejlépe automatizované. Jakmile navrženo, je řešení v této práci algoritmizováno, implementováno a následně testováno. Pokud se ukáže jako kvalitní, mělo by být řešení využito v produktech firmy Flowmon, která projekt pro práci zadávala.

Výsledkem práce je tedy aplikace, které zvládá korelovat záznamy toků proxy serverů, které figurují v síťové komunikaci za využití takových protokolů jako HTTP a SOCKS. Je možné díky ní nalézt korelaci mezi záznamy jednoho i více proxy serverů, které stojí v komunikaci logicky za sebou. Aplikace byla navržena aby zvládala korelaci na určité úrovní bez jakéhokoliv uživatelského vstupu, přičemž při poskytnutí uživatelských informací o zpracovávané komunikaci (přítomnost privátních rozsahů, typ proxy serverů, apod.) je korelace zpřesňována.

Žádné předchozí řešení, které by řešilo tento konkrétní problém nalezeno nebylo. Existují obecné principy pro korelaci záznamů síťových toků a práce [8], které se soustředily například na korelaci komunikace procházející sítí Tor, což je pro problém řešený v této práci dobrý odrazový můstek. Každopádně o aplikaci tohoto přístupu čistě na proxy servery nebyly nalezeny žádné informace.

Text této práce je rozvržen do šesti kapitol, úvodu a závěru. V kapitole 2 probíhá seznámení se s technologiemi proxy serverů. Jsou zde popsány jejich typy, využití a jiné vlastnosti. Dále jsou také uvedeny síťové protokoly, které jsou s proxy servery úzce spjaty. V kapitole 3 jsou popsány využívané standardy pro zaznamenávání síťových toků a jejich reálná implementace v některých systémech. V kapitole 4 je popsán proces získávání datové sady pro analýzu a vytvoření řešení. Je zde rozebráno prostředí, ve kterém probíhal sběr dat, které by měly být využity k analýze provozu a návrhu řešení korelace. Je zde popsána vytvořená síťová struktura a následně vytvořená datová sada. V kapitole 5 je popsáno navržené řešení a následně vytvořený algoritmus. Je zde popsán postup, jakým byl analyzován získaný síťový provoz proxy serverů a myšlenky, se kterými byla analýza prováděna. Následně je popsáno navržené řešení problému korelace a je prezentována algoritmizovaná podoba tohoto řešení. V kapitole 6 je detailně popsána implementace algoritmu. Konečně v kapitole 7 je popsán způsob testování vytvořeného řešení a aplikace. Je popsána datová sada využívaná pro testování, dále testování jako takové. Na konci jsou dosažené výsledky zhodnoceny a je diskutována úspěšnost vyřešení problému korelace.

Kapitola 2

Použití proxy serverů

Dříve než bude možné začít zpracovávat záznamy a použít z nich získané informace pro vytvoření návrhu jejich korelování, je nutná trocha teorie. Nejprve je tedy potřeba seznámit se s technologiemi, nad kterými bude prováděn výzkum. V této kapitole jsou popsány proxy servery jako takové, je detailněji rozebrána jejich činnost a fungování. Následně jsou zmíněny typy proxy serverů, které je možné nejčastěji najít v reálném síťovém provozu a v neposlední řadě protokoly, které s jejich fungováním úzce souvisí.

2.1 Proxying

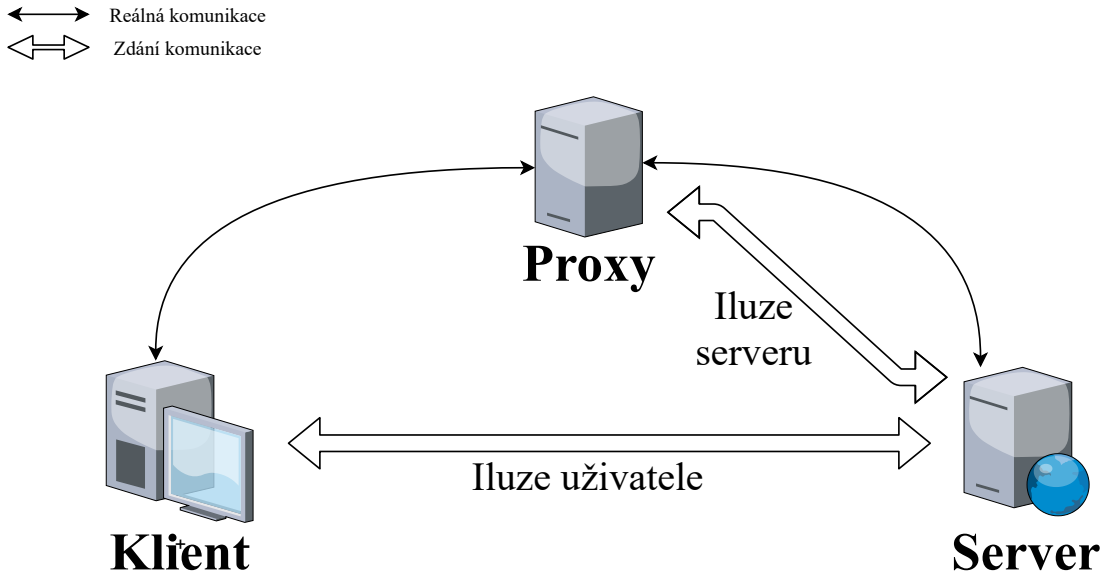
Proxying je způsob, jak ve velké síti, například v Internetu, umožnit přístup pouze k jednomu nebo malé skupině strojů z podsítě, přičemž je zachováno zdání, že je umožněn přístup ke všem strojům dané podsítě[6]. *Proxy server* je specializovaná aplikace, která zajišťuje komunikaci mezi vnitřní sítí a Internetem[17].

V minulosti byly proxy servery využívány pro ukládání často navštěvovaných Internetových stránek do vyrovnávací paměti (cache). V době, kdy byl Internet poměrně malý a publikované stránky byly statické, byla tato funkčnost výhodná. Proxy server stahoval stránky do své vyrovnávací paměti a minimalizoval tím zbytečné připojování k Internetu. S nárůstem počtu uživatelů Internetu a také množstvím a charakterem navštěvovaných stránek, se od užívání proxy serverů upustilo. Většina stránek v Internetu je dynamická, tudíž její platnost vyprší hned po odeslání a uživatelé navštěvují obrovské množství stránek, takže ukládání do mezipaměti již nebylo tak výhodné. Objeveno byla ovšem jiné využití, související s jejich vlastností skrýt všechny uživatele za jednu adresu, čímž proxy server prakticky poskytuje anonymitu při komunikaci. Další využitelnou vlastností je možnost filtrování síťového provozu v obou směrech. Nyní je tedy prvořadým účelem většiny proxy serverů funkce firewallu[15].

Proxy server přistupuje jménem klienta či uživatele k určité síťové službě a fakticky tím zakrývá přímé spojení mezi oběma partnery[17]. Ze své podstaty, kdy jsou proxy servery využívány jako firewally, jsou sice proxy servery schopné monitorovat a filtrovat komunikaci, která přes ně prochází, nicméně pro účely monitorování síťového toku přes proxy server nejsou takové funkce důležité. Protože proxy servery vznikly především jako nástroj používaný pro komunikaci v Internetu, je proxy servery nejčastěji zpracovávaný protokol HTTP[15]. Vznikly ovšem i další protokoly, které poskytují zapouzdření a větší zabezpečení při komunikaci skrz proxy server (například protokol SOCKS)[13].

2.2 Fungování proxy serverů

Proxy server je nastaven pro nějaký protokol anebo skupinu protokolů, které zpracovává. Pokud má proxy server zpracovávat nějaký protokol, je nutné daný protokol definovat v jeho nastavení anebo nastavit proxy server, aby rozuměl širší skupině protokolů[6]. Více jsou jednotlivé typy proxy serverů rozebrány v dalším textu. Základní schéma je možné vidět na obrázku 2.1.



Obrázek 2.1: Fungování proxy serveru, schéma

Pokud má probíhat komunikace klienta s cílovým serverem přes proxy server, je nutné provést nastavení také na straně klienta. Existuje několik způsobů, jak nastavit klienta tak, aby využíval pro komunikaci s Internetem proxy server.

1. *Aplikace*, jejíž nastavení ji umožňuje využívat proxy server. Aplikační software musí vědět, jak kontaktovat proxy server a jak mu předat informaci, ke kterému cílovému serveru je třeba se připojit.

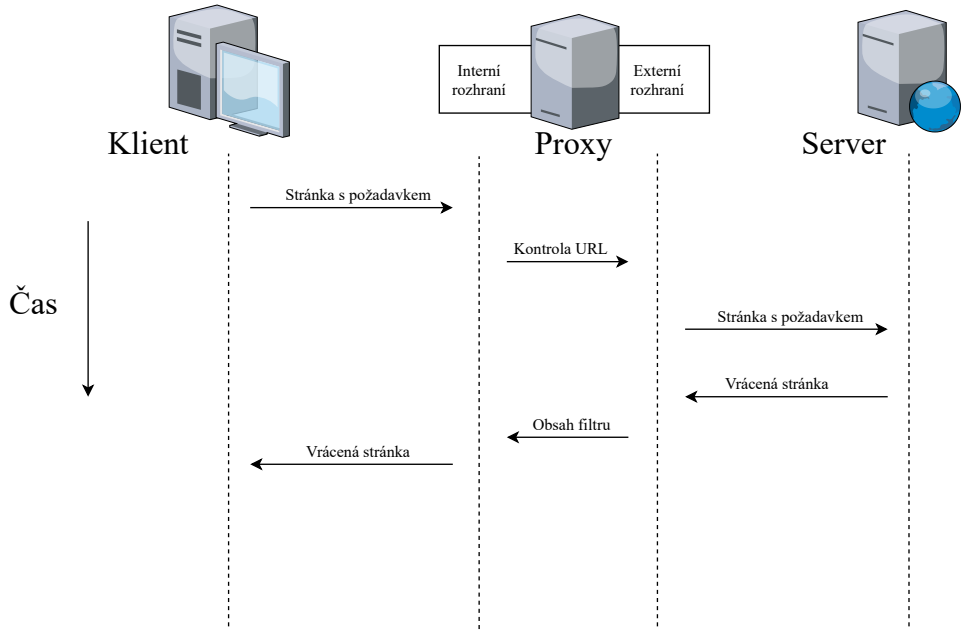
2. *Operační systém*, jehož nastavení umožňuje využívat proxy server. Operační systém je upraven, aby na základě kontroly IP adres rozhodl, zda nemá být spojení uskutečněno přes proxy server.

3. *Uživatel*, který využívá software bez funkce využití proxy serveru, aby se přímo připojil k proxy serveru a určil cílový server pro komunikaci.

4. *Router*, který odchytává procházející provoz, který odesílá na proxy server, anebo přímo funguje jako proxy server. Na straně klienta přitom nedochází k žádným úpravám[6].

Příklad jednoduché komunikace s využitím proxy serveru je následující. Nejdříve klient naváže potřebné spojení s proxy serverem a vyžádá si určitou internetovou službu (HTTP,

FTP, apod.). Klientský software odesílá požadavky v souladu se zásadami zabezpečení. Proxy server se poté spojí s cílovým serverem. Dále přeposílá proxy server data od klienta cílovému serveru a naopak. Proxy server tedy funguje jako brána. Z pohledu klienta je komunikace s proxy serverem stejná jako komunikace s cílovým serverem. Z pohledu cílového serveru probíhá komunikace s proxy serverem, kterého považuje za klienta, přičemž o skutečné identitě klienta cílový server neví[6, 17]. Příklad jednoduché komunikace je na obrázku 2.2.



Obrázek 2.2: Fungování proxy serveru, stavový diagram

Proxy server většinou funguje na takzvaném opevněném hostiteli s dvěma rozhraními (dual-homed, dvoudomý, duální hostitel). Opevněný hostitel (bastion host) je systém s posílenou bezpečností, který slouží k přístupu na Internet. Jedno rozhraní hostitele vede do vnitřní chráněné sítě, druhé potom do externí sítě. Mezi těmito sítěmi není povolen přímý provoz, přeposílání IP paketů je vypnuté, což umožňuje softwaru běžícím na tomto hostiteli úplnou kontrolu nad procházejícím provozem[17].

Při použití proxy serveru jsou stanice, které jsou umístěné v síti 'schované' za proxy serverem, nedosažitelné pro externí systémy. Jak již bylo zmíněno, nedochází k přeposílání a směrování paketů, tudíž sítě na obou stranách proxy serveru nemusí být vůbec kompatibilní. Takové funkcionality je u proxy serverů dosaženo tím, že u požadavků na úrovní služeb nedochází pouze ke změně a přepočítání hlavičky paketu, ale požadavky se kompletně znovu generují[15].

Další funkcí, důležitou při popisu proxy serveru pro účely této práce, je dělení přenosového pásma při připojení. Proxy server může fungovat jako jediný bod připojení k Internetu s jedinou IP adresou, přes který se připojuje k Internetu celá vnitřní síť. Takové proxy servery jsou oblíbené v domácnostech a malých podnikových sítích, kde je k dispozici pouze jedno připojení[15].

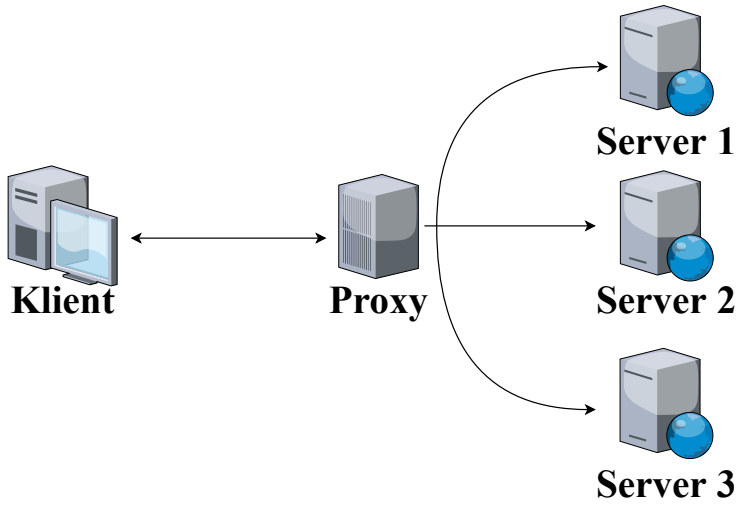
2.3 Typy proxy serverů

Proxy servery lze rozdělit na mnoho typů. Zde je uvedeno několik nejběžnějších. Rozdělení závisí například na jejich umístění v rámci sítě anebo na způsobu, jakým je přistupováno k předávání zpráv různých protokolů.

Dopředné a reverzní proxy servery

Jak již bylo zmíněno, proxy server je v síťové komunikaci zároveň serverem i klientem. Rozdíl mezi dopředným a reversním proxy serverem tedy není úplně jednoznačný. Nicméně jako *dopředný proxy* je většinou označována komunikace interního uživatele, který se skrz server snaží dostat k nějaké Internetové službě. Popis proxy serveru, jak byl zatím uveden, lze tedy přiřadit k dopřednému proxy serveru[17].

Oproti tomu *reverzní proxy* využívá trochu jinou technologii. Není využíván jako firewall, ale spíše jako bezpečný obsahový server určený pro vnější klienty. Tento server zabraňuje přímému a nekontrolovanému přístupu k datům a službám serverů vnitřní sítě z vnější sítě. Také mohou být využity pro urychlení činnosti sítě, neboť před server lze umístit jeden nebo více proxy serverů, mezi kterými můžeme zátěž vyrovnat. Kromě již dříve zmíněného ukládáni do vyrovnávací paměti, můžeme také proxy server využít pro směrování klientským požadavků mezi více koncových serverů. Předpoklad je že tyto servery jsou identické a pro náročnost aplikací, které na nich běží, by normálně nebyly schopny obsloužit větší množství požadavků[17, 15]. Schéma fungování reverzního proxy je možné vidět na obrázku 2.3.



Obrázek 2.3: Reverzní proxy server

Jeden proxy server může poskytovat dopředný i reverzní proxying, záleží pouze na jeho nastavení[17].

Proxy na úrovni aplikací a okruhů

Aplikační proxy (application-level proxy) je specializovaný program, který je přesně implementován pro určitou službu, rozumí a provádí příkazy aplikačního protokolu. Klasickým příkladem je například propouštění HTTP provozu ve směru z vnitřní sítě do vnější sítě. Aby mohla využívat proxy server, musí každá služba mít vlastní program, který ji obsluhuje.

Oproti tomu *okruhové proxy* (circuit-level proxy) vytváří spojení mezi klientem a serverem bez porozumění a interpretace aplikačního protokolu. Tento typ proxy poskytuje obsluhu pro širokou škálu různých protokolů, prakticky můžou být nastaveny pro obsluhu skoro jakéhokoliv protokolu. U některé protokolů je však potřeba určitá intervence aplikační vrstvy, a tudíž nemůžou být tímto typem proxy serveru obslouženy (například protokol FTP z důvodu předávání informací o portech). Hlavní nevýhodou tohoto typu proxy serveru je velmi malá možnost kontroly procházejícího toku, jejich funkcionalita se dá přirovnat k filtrům paketů[17, 6].

Relevantní protokoly

Ze své podstaty může být proxy server nastaven pro obsluhu jakéhokoliv protokolu. Každý protokol má svá specifika, od čehož se poté odvíjí také způsob, jakým proxy server navazuje a udržuje spojení. V této sekci je soupis protokolů, se kterými je nejčastěji možné se setkat při fungování proxy serverů v reálné síti. Protokoly zde vypsané byly dále sledovány v rámci vytváření datové sady.

HTTP a HTTPS

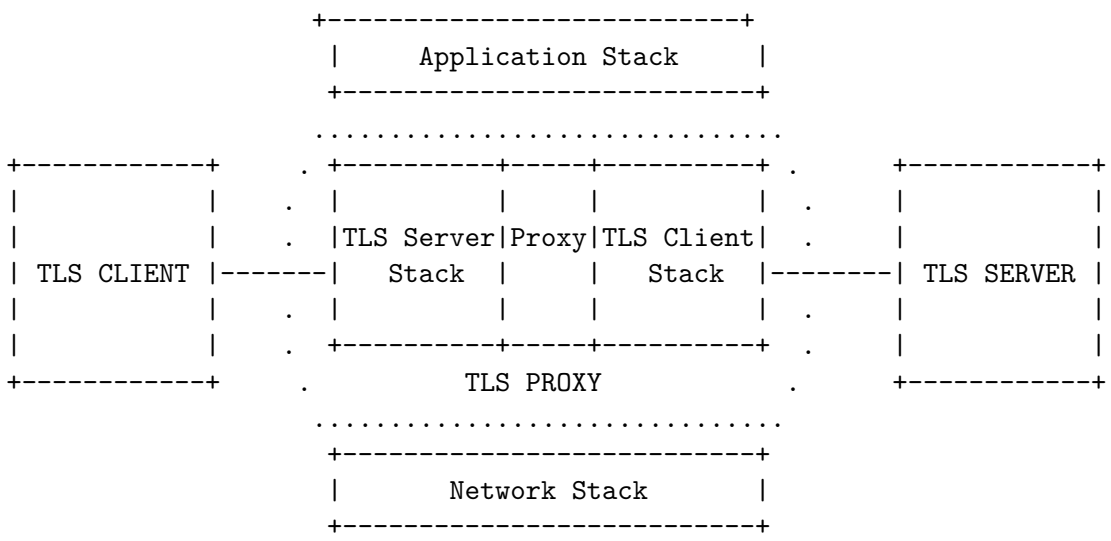
Vzhledem k tomu, že proxy servery jsou nástroje využívané především při webové komunikaci, je úzká souvislost s protokolem HTTP zřejmá. Proxy server obsluhující protokol HTTP (HTTP proxy server) musí dodržet všechna pravidla pro připojení skrz HTTP protokol. Server zprávy většinou pouze přeposílá, ovšem může je i upravovat, anebo úplně měnit[16, 10].

Skrz proxy server je možná i zašifrovaná komunikace za pomoci TLS/SSL. K tomu je zapotřebí TLS proxy. Jeho přesné fungování závisí na organizační politice, nicméně běžný je model, kdy proxy server odchytí TLS handshake klienta, který poté sám naváže s cílovým serverem. Proxy server má tedy možnost volně šifrovat a dešifrovat zprávy z obou stran. Může tedy provádět kontrolu obsahu, i když je zašifrovaný[22]. Schéma můžeme vidět na obrázku 2.4.

SOCKS

SOCKS není čistě protokol, je to souprava proxy nástrojů, která umožňuje zprostředkovanou komunikaci bez nutnosti vytváření speciálních proxy modulů. Referenční implementace je volně dostupná a stala se de facto standardním balíčkem pro proxying v Internetu.

SOCKS se skládá ze dvou částí. SOCKS serveru, který funguje na aplikační úrovni, a SOCKS klienta, který funguje mezi aplikační a transportní vrstvou. Aplikace, které mají komunikovat skrz SOCKS proxy server, musí být upraveny. SOCKS proxy server zavádí komunikační kanály pro libovolné aplikace a poté zajišťuje jejich správu a ochranu. Je tedy s jeho pomocí možné obsloužit jakýkoliv požadavek. Jsou k dispozici dvě verze SOCKS protokolu. Těmi jsou SOCKSv4 a SOCKVv5, přičemž verze 5 obsahuje od svého předchůdce podporu pro protokol UDP a ICMP a možnost autentizace.



Obrázek 2.4: Schéma TLS proxy serveru[22]

Vzhledem k jeho generalitě, neprovádí protokol SOCKS žádnou specifickou kontrolu nebo logování pro jednotlivé protokoly, které obsluhuje. Nicméně umožňuje logování příchozích spojení a může být nakonfigurován, aby zasílal upozornění při odmítnutých připojeních[17, 6].

Kapitola 3

Síťový provoz a jeho monitorování

V této kapitole je definován síťový provoz tak, jak k němu bude přistupováno v této práci a jsou popsány způsoby jeho monitorování a záznamu, které jsou pro tuto práci relevantní. Detailněji jsou popsány technologie Netflow a IPFIX, které slouží pro vytváření záznamů o síťových tocích. Je také věnována pozornost způsobu jejich využití a implementace v systémech, které vytváří a vydává firma Flowmon.

Monitorování síťového provozu můžeme rozdělit do dvou hlavních kategorií, rozdělených dle přístupu: aktivní a pasivní. *Aktivní* přístupy samy naplní síť provozem, aby mohly provádět různé typy měření. Mezi nástroje, které jsou k tomuto přístupu využívány, patří například `Ping` nebo `Traceroute`. *Pasivní* přístupy sledují existující provoz, a tudíž sledují provoz vytvářený uživatelem. Do této kategorie spadá například zachytávání síťových paketů, které však může v sítích s propustností až 10 Gbps vyžadovat velmi drahý hardware a náročné prostředí pro zpracování. Další příklad z této kategorie, který nabízí lepší využití ve vysokorychlostních sítích, je export síťového toku[11].

Pro pojem 'tok' anebo 'síťový tok' se v internetové komunitě vyskytuje více definicí. V této práci je tok definován dle standardu RFC 7011, který zpracovává standard IPFIX jako:

> Tok je soubor paketů nebo rámců, které prochází kolem určitého bodu pozorování v síti v určitém časovém intervalu.

Pro všechny pakety v toku platí, že mají soubor společných vlastností. Tyto vlastnosti jsou definovány jako výsledek aplikace nějaké funkce na následující hodnoty:

1. jedna nebo více hlaviček paketů, hlaviček transportních protokolů nebo hlaviček aplikačních protokolů,
2. jedna nebo více charakteristik samotného paketu,
3. jedno nebo více polí odvozených ze zpracování paketu.

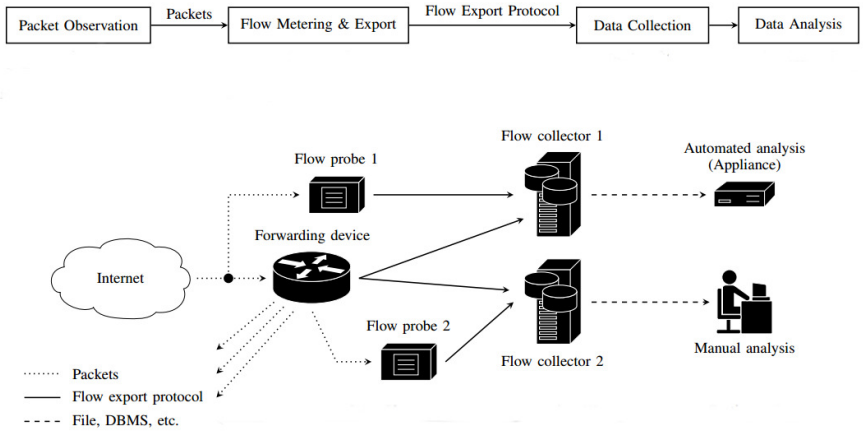
Paket náleží toku pouze, pokud splňuje všechny definované vlastnosti[4].

3.1 Proces sledování síťového provozu

Sledování síťového toku probíhá pouze v určitých místech sítě, které se nazývají *pozorovací body* (observation points). Jako pozorovací bod je možné určit síťové linky, ke kterým je připojena sonda; sdílené médium, jako třeba Ethernetové LAN; jeden port routeru; rozhraní

anebo soubor rozhraní routeru. Jeden pozorovací bod také může být složen z více pozorovacích bodů. Každý pozorovací bod je přiřazen k nějaké *pozorovací doméně* (observation domain). Ta je definována jako největší soubor všech pozorovacích bodů, ze kterých můžou být agregovány informace pro jeden tok[4].

Typicky se proces sledování síťového toku skládá ze čtyř fází. Tyto fáze jsou znázorněny na obrázku 3.1 a dále jsou popsány.

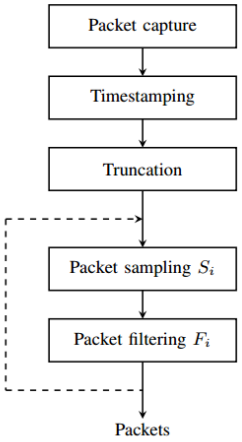


Obrázek 3.1: Proces sledování a exportu síťového toku

Fáze jsou následující:

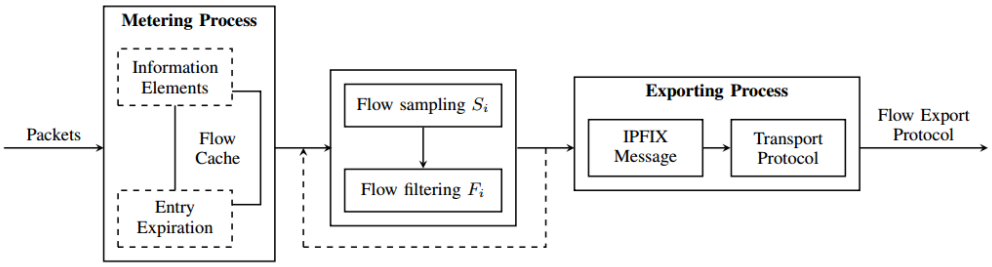
- **Zachytávání síťových paketů**. Jde o proces, u kterého jsou zachytávány a předzpracovány síťové pakety. První krok této fáze, zachycení paketu, je vlastně čtení paketu ze síťové linky, které většinou provádí síťová karta (NIC). Ještě dřív, než jsou pakety uloženy do mezipaměti síťové karty a odtamtud dál do paměti hostitele, musí projít několika kontrolami, například porovnáním `checksum` hodnoty. Druhý krokem je označení časovou známkou (timestamp). Tento krok je esenciální pro následující zpracovávající funkce a analytické aplikace. Podle časové známky jsou organizovány pakety, které jsou z různých pozorovacích bodů agregovány do jednoho datasetu. Následuje oříznutí paketu, což znamená vybrání pouze tolika bitů, jaká je přednastavená délka snímku (snapshot length). Tento krok redukuje objem dat, které aplikace provádějící sledování zpracovává a v konečném důsledku může velmi urychlit analýzu, zvlášť pokud jsou důležité například pouze hlavičky paketů a ne jejich obsah. Posledním krokem je vzorkování a filtrování paketů. Motivací pro vzorkování je vybrání podmnožiny paketů, tak aby bylo stále možné určit všechny vlastnosti celého toku. Filtrování se provádí pro odstranění paketů, které nejsou pro další zpracování zajímavé. Všechny kroky kromě zachycení paketu a označení časovou značkou jsou nepovinné. Celý proces této fáze je znázorněn na obrázku 3.2[11].

- **Měření a export síťového toku**. V této fázi jsou pakety seskupovány do síťových toků, ze kterého je následně exportován *záznam o síťovém toku*. Ten obsahuje informace o specifickém toku, především tedy měřené vlastnosti toku a charakteristické vlastnosti toku. Seskupení (nebo agregace) paketů do síťového toku probíhá při takzvaném *Procesu měření* podle *informačních elementů*, které definují strukturu toku. To jsou zjednodušeně pole, které mohou být exportovány do záznamu o síťovém toku



Obrázek 3.2: Proces zachytávání síťových paketů

a budou rozebrány dále. Po seskupení je tok ukládán ve *flow cache*. To je tabulka, ve které jsou uloženy informace ohledně všech aktivních síťových toků. V každém toku je určena jedna hodnota, která se nazývá *flow key*, podle které je určeno, zda nově příchozí paket patří do již existujícího nebo nového toku. Informace jsou ve flow cache drženy do chvíle, kdy je tok považován za ukončený. Následně jsou informace o toku opět vzorkovány a filtrovány a následně zapouzdřeny ve zprávách. Pro formát těchto zpráv je vytvořeno několik standardů, v této práci jsou využívány formáty NetFlow a IPFIX, které jsou blíže popsány dále. Celý proces je znázorněn na obrázku 3.3[4, 11].



Obrázek 3.3: Proces měření a exportu síťového toku[11]

- **Sběr a analýza dat**. Tyto dvě fáze nejsou pro tuto práci tak důležité, a proto jsou zde prezentovány pouze zběžně. Sběr dat je prováděn kolektory, které přijímají, ukládají a provádí předzpracování dat od jednoho nebo více exportérů v síti. Typickými úkony předzpracování je komprese, agregace nebo anonymizace dat. Analýza dat je poté poslední fází celého procesu sledování síťového provozu. Rozlišuje se mezi třemi hlavními oblastmi pro analýzu dat: Analýza toku a Hlášení, Detekce hrozeb, Monitorování výkonu[11].

Jak již bylo zmíněno, pro záznamy o síťovém toku existují standardy, podle kterých jsou vytvářeny. V následujícím textu jsou popsány standardy, které jsou relevantní pro tuto práci.

3.2 NetFlow v9

Celý název tohoto standardu zní *Cisco Systems NetFlow Services Export Version 9*. Je to velmi populární nástroj pro sledování síťového toku založeného na IP komunikaci. Má velmi podobnou funkčnost jako předchozí verze NetFlow v5, která ve své době byla nejpoužívanějším nástrojem pro monitorování sítí. Hlavní rozdíl mezi verzemi je, že verze 9 využívá šablonu, která definuje kolekci datový polí s korespondujícím popisem struktury a sémantiky. To poskytuje flexibilní a rozšiřitelný přístup ke sledování síťových toků. Je totiž možné přidat nová datová pole, aniž by bylo nutné měnit strukturu exportovaného formátu. Tyto nová pole obsahují vlastní popis, takže i kolektory, které jim nerozumí, mohou záznam o toku interpretovat[7, 19, 12].

Záznam NetFlow v9 se skládá ze série exportních paketů. Obsah takového paketu je složen z hlavičky paketu a poté jedné nebo více šablonových nebo datových FlowSets. Šablonové FlowSets poskytuje popis datových polí, které jsou obsaženy v datových FlowSets. Tyto data se mohou objevit ve stejném paketu, anebo v dalších příchozích paketech[2, 7]. Ilustraci poskytuje obrázek 3.4.

Obrázek 3.4: NetFlow v9 exportní paket[2]

FlowSet je obecný pojem pro kolekci záznamů, které jsou obsaženy v exportním paketu hned za hlavičkou. Šablonový FlowSet je kolekce jedné nebo více šablon záznamů, které byly sdruženy dohromady v exportním paketu. Každá šablona záznamu obsahuje unikátní ID, kterým je odlišena od ostatních šablon, které vyprodukovalo jedno zařízení. Datový FlowSet je kolekce jednoho nebo více záznamu dat, které byly sdruženy dohromady v exportním paketu. Každý záznam dat se odkazuje na nějaké ID šablony záznamu, podle kterého je určena šablona k přečtení dat[2, 7].

Možné kombinace šablon a dat, které se mohou objevit v exportním paketu, jsou vypsány dále.

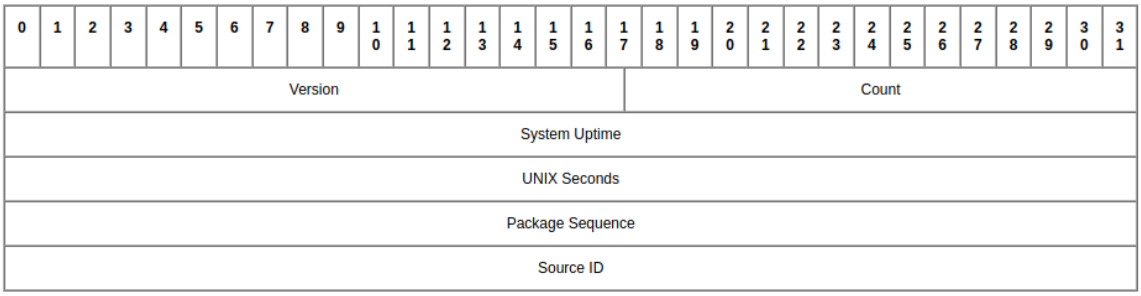
- Vzájemně proložené šablonové a datové FlowSets.
- Paket obsahující pouze šablonové FlowSets.
- Paket obsahující pouze datové FlowSets.

V případě, že se v jednom paketu vyskytuje kombinace datových a šablonových FlowSets, nemusí platit, že mají spolu nějakou souvislost. Souvislost mezi šablonami a daty ve FlowSets je dána pouze pomocí definovaných ID. Ve chvíli kdy byly definovány a odeslány všechny šablony, skládají se exportní pakety už pouze z datových FlowSets. Pakety obsahující pouze šablonové FlowSets nejsou obvyklé, ale mohou se vyskytnout[2, 7].

Formát hlavičky

Formát hlavičky je stejný jako v předchozích verzích NetFlow[2]. Hlavička je ilustrována na obrázku 3.5.

- *Verze (Version)* je verze formátu záznamu toku, pro NetFlow v9 tedy vždy 9.



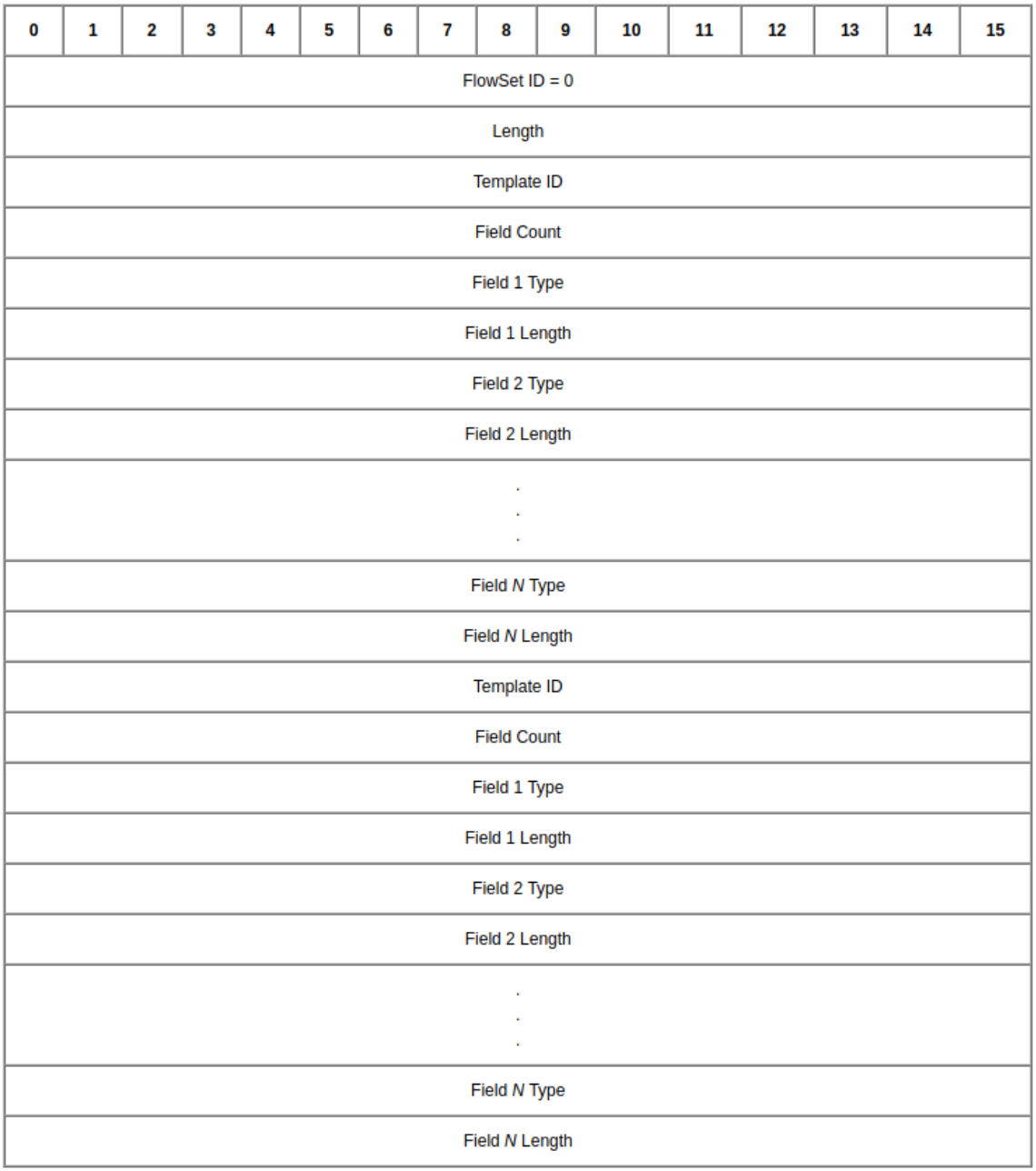
Obrázek 3.5: Hlavička exportního paketu NetFlow v9[2]

- *Počet (Count)* je celkový počet záznamů v exportním paketu. Je to suma všech záznamů FlowSet všech typů.
- *Čas od startu (System Uptime)* čas v milisekundách od prvního startu zařízení.
- *UNIX sekund (UNIX Seconds)* čas v sekundách od začátku UNIX epochy, ve kterém paket opouští exportér.
- *Sekvenční číslo (Sequence number)* sekvenční číslo všech exportních paketů v rámci jedné sledovací domény, které byly zaslány exportérem. Číslo je postupně se inkrementující a kolektor podle něj může zjistit, které exportní pakety jsou ztraceny.
- *ID zdroje (Source ID)* je 32 bitové číslo, které identifikuje sledovací doménu exportéru. Spolu s IP adresou může být využíváno k identifikaci jednotlivých záznamů toků ze stejného exportéru[7].

Formát šablonového FlowSet

Šablony jsou klíčovým elementem v NetFlow v9 formátu. Umožňují kolektorům anebo jiným dalším aplikacím zpracovat data, aniž by předem museli nutně vědět formát těchto dat. Šablony jsou využívány pro popis typu a délky jednotlivých polí v rámci datových záznamů, se kterými mají shodující se ID[2]. Ilustrace je na obrázku 3.6.

- *FlowSet ID* je hodnota, která slouží k rozlišení šablony záznamu a samotných datových záznamů. Šablona záznamu obsahuje vždy FlowSet ID hodnotu v rozmezí 0-255. Momentálně platí, že šablona záznamu popisující pole záznamu má hodnotu 0. Datové záznamu mají ID hodnotu nenulovou a větší jak 255.
- *Délka (Length)* je celková délka FlowSet. Je to suma délky FlowSet ID, hodnoty délky samotné a všech šablon záznamů ve FlowSetu.
- *ID šablony (Template ID)* je unikátní identifikátor každé šablony záznamu. Unikátnost platí pouze v rámci sledovací domény. Platí pro ně stejná pravidla jako pro FlowSet ID.
- *Počet polí (Field count)* je počet polí v šabloně záznamu. Protože jeden FlowSet může obsahovat více záznamů, může tato hodnota sloužit k rozlišení začátku a konce dvou různých záznamů.

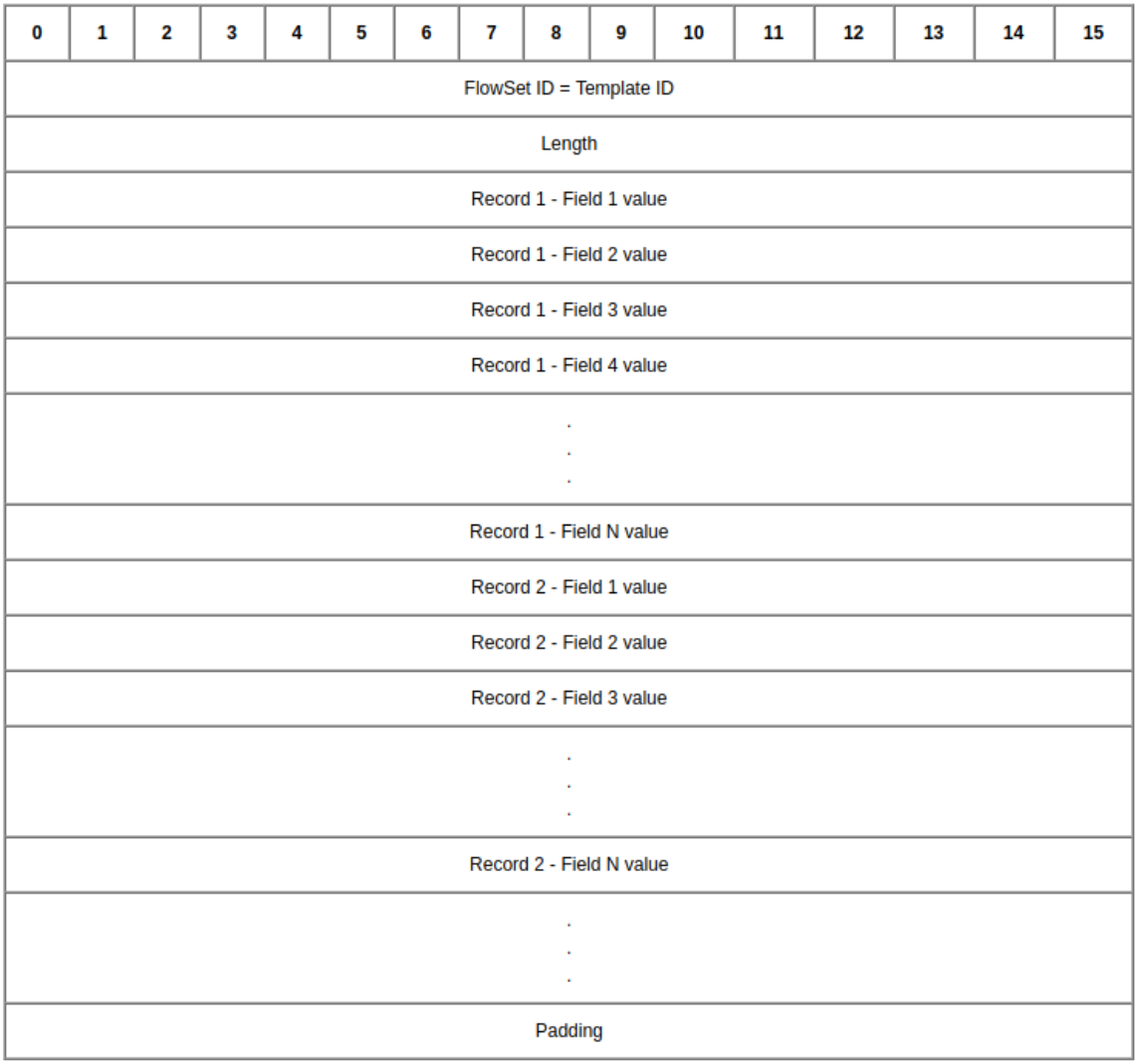


Obrázek 3.6: Formát šablonového FlowSetu[2]

- *Typ pole (Field type)* numerická hodnota, která určuje typ pole. Výpis všech typů polí je v příloze A.
- *Délka pole (Field legth)* je délka pole daného typu v bytech[7, 2].

Formát datového FlowSet

Ilustrace datového FlowSet je na obrázku 3.7.



Obrázek 3.7: Formát datového FlowSetu[2]

- *FlowSet ID* je ID hodnota datového FlowSet. Mapuje se na nějaké ID šablony. Pomocí tohoto ID kolektor nalezne šablonu záznamu, podle které dekóduje záznamy o toku z FlowSet dat.
- *Délka (Length)* je celková délka FlowSet. Je to suma délky FlowSet ID, hodnoty délky samotné, všech záznamů o tocích a doplňujících bitl, pokud nějaké jsou.
- *Záznam - Pole (Record - Field)* je kolekce dat záznamu o toku, kde každý záznam obsahuje skupinu datových polí. Typ a délka každého pole je definována v odpovídající šabloně.
- *Doplnění (Padding)* je nepovinné doplnění záznamu FlowSet, aby byl každý zarovnán na 32 bitů. Doplnění by mělo být pomocí nul[7, 2].

Důležitý pro formát NetFlow je ještě jeden typ záznamu, a to šablona nastavení a k tomu korespondující data nastavení. Tento typ záznamu je využíván pro předání metadat o samotném NetFlow procesu[2].

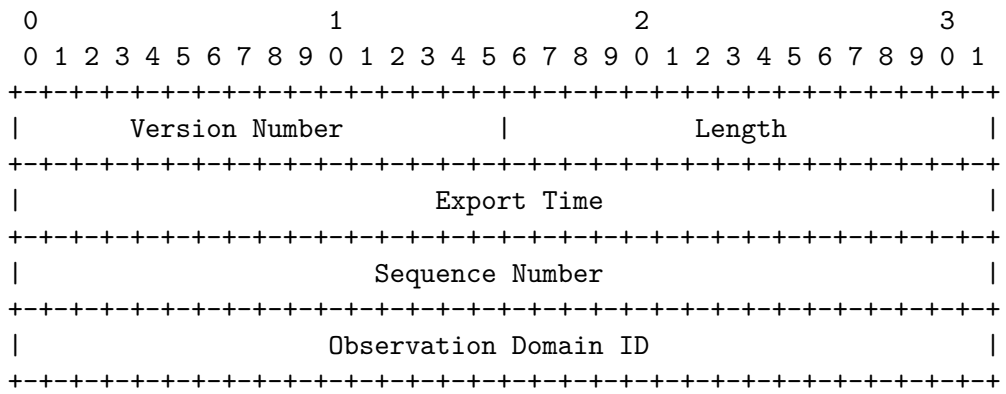
3.3 IP Flow Information Export (IPFIX)

IPFIX je výsledkem snahy standardizovat exportní protokol pro záznam síťového toku. Z části je založen na NetFlow v9, nicméně oproti tomu poskytuje mnoho nových funkcí. Formát IPFIX vznikl za účelem zlepšení interoperability v měření síťového provozu. Protokol je navržen jako bez směrový, transportně nezávislý s flexibilní reprezentací dat a pokrývá majoritu potřeb pro síťovou správu na vrstvách L3 a L4. V tuto chvíli je IPFIX prakticky internetových standardem pro zpracovávání záznamů síťových toků[4, 21].

IPFIX zpráva je složena z hlavičky a setů, kterých může být více, ale také nemusí být žádný. Setů jsou tři typy: datový set, šablonový set a set šablon možností. Každý set v sobě obsahuje záznamy. Podobně jako u NetFlow je několik možností kombinací těchto typů v jedné zprávě (proložené všechny tři typy, pouze datový set nebo pouze šablony)[4].

Se standardem IPFIX souvisí pojem **Informační elementy** (IE). To jsou na kódování nezávislé popisy atributu, které se mohou vyskytnout v IPFIX záznamu. Jejich standardizovaný seznam udržuje IANA. Typ, asociovaný s IE určuje hranice, co všechno může daný element obsahovat a také určuje správný kódovací mechanismus, který má být využit v IPFIX formátu. Informační elementy obsahují jméno, numerické ID, popis, typ, délku a status. Kromě standartizovaných elementů je možné zavést nové informační elementy na úrovní firem, aniž by bylo nutné měnit IANA záznamy[4, 11].

Formát hlavičky zprávy



Obrázek 3.8: Formát hlavičky IPFIX zprávy[4]

- *Verze (Version)* je použitá verze IPFIX protokolu. Jeho hodnota je 10 (o jedna více než u NetFlow v9)
- *Délka (Length)* je celková délka zprávy měřená v oktetech.
- *Čas exportu (Export Time)* je čas, ve kterém zpráva opustila exportér. Čas je vyjádřen jako 32 bitové číslo značící počet sekund od začátku UNIX epochy.
- *Sekvenční číslo (Sequence Number)* je inkrementující se sekvenční číslo omezené na 32 bitů, které označuje datové záznamy (šablonové ne). Číslování je unikátní vždy

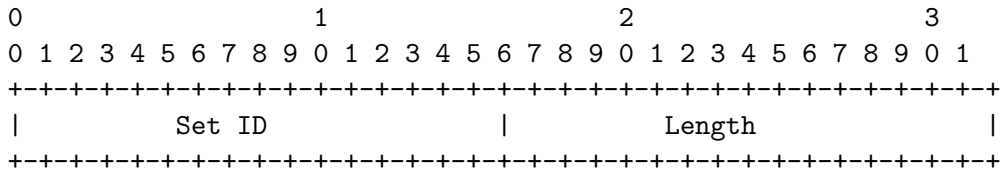
v rámci jednoho exportního toku z určité sledující domény. Sekvenční číslo může sloužit ke kontrole, které záznamy se ztratily při přenosu.

- *ID sledující domény (Observation Domain ID)* je 32 bitové číslo unikátní v rámci jednoho exportního procesu. Identifikuje sledující doménu, ve které byl zachytáván síťový tok. V případě, že tok nemůže být přiřazen k žádné doméně, měla by být hodnota 0[4].

Formát setu

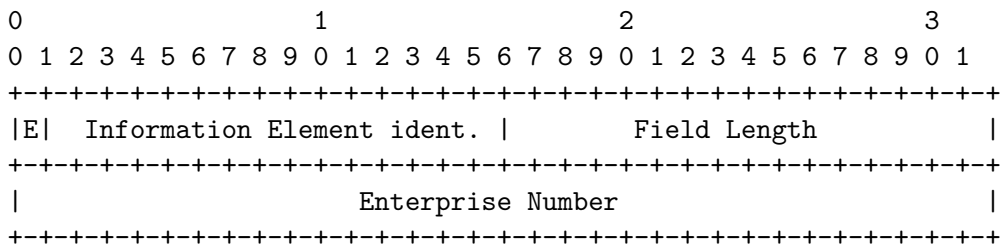
Set je generický pojem pro kolekci záznamů, které mají podobnou strukturu. Každý typ setu se skládá z hlavičky a jednoho nebo více záznamů.

Hlavička setu obsahuje dvě hodnoty. Ilustraci je možné vidět na obrázku 3.9.



Obrázek 3.9: Formát hlavičky setu[4]

- *ID setu (Set ID)* identifikuje set. Hodnota 2 je rezervována pro šablonu, hodnota 3 pro šablonu možností. Ostatní hodnoty menší než 256 jsou rezervovány pro budoucí využití, všechny vyšší hodnoty jsou využívány pro datové sety. Hodnoty 0 a 1 se nevyužívají.
- *Délka (Length)* je celková délka setu v oktetech.

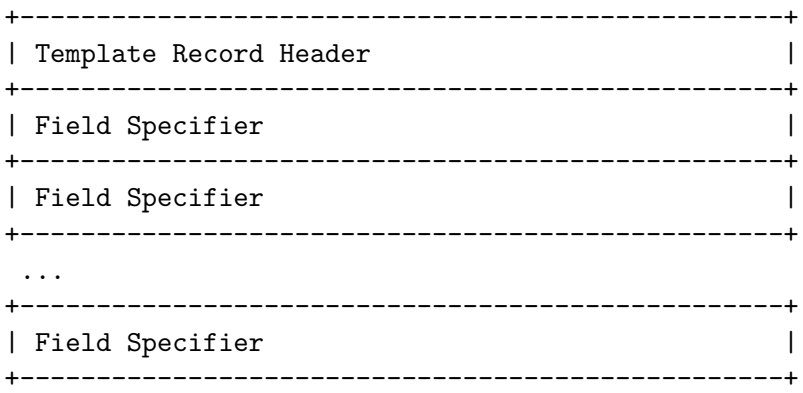


Obrázek 3.10: Formát specifikátoru pole[4]

Záznam setu má 3 typy, které odpovídají typům setu. Pro jejich popis je ale nutné definovat pojem specifikátor pole. *Specifikátor pole* je způsob, jakým jsou v IPFIX záznamech popsány informační elementy. Jeho strukturu je možné vidět na obrázku 3.10. Informační elementy jsou v něm identifikovány pomocí *identifikátoru*. Pokud je enterprise bit (označení E) nastaven na 0, jde o informační element, který definuje IANA. Pokud ne, jde o specifický informační element nějaké firmy, jejíž identifikátor musí být dále přítomen (jinak je pole nepovinné). Specifikátor obsahuje také délku korespondujícího informačního elementu.

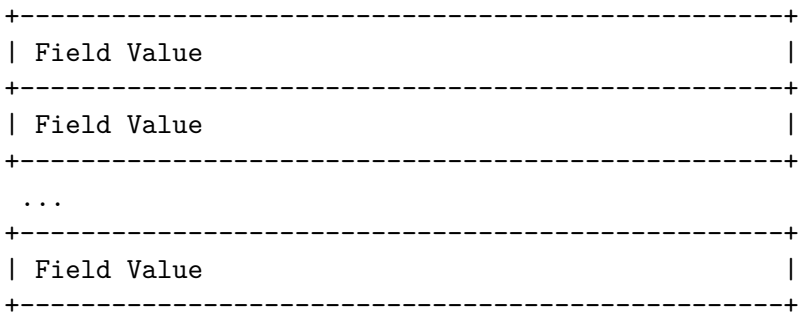
Dále jsou popsány typy záznamů.

- *Šablonový záznam.* Stejně jako u NetFlow formátu, šablony umožňují kolektoru zpracovávat záznam o toku, aniž by nezbytně znal interpretaci všech datových záznamů. Tento typ záznamu se skládá z hlavičky a kombinace specifikací polí pro informační elementy, které definuje IANA, nebo informační elementy definované firmou. Na obrázku 3.11 je možné vidět strukturu záznamu. Hlavička obsahuje ID, dle kterého jsou přiřazovány datové záznamy, a celkovou délku záznamu.



Obrázek 3.11: Formát šablonového záznamu[4]

- *Datový záznam.* Strukturu záznamu je možno vidět na obrázku 3.12. Skládá se pouze z jednoho nebo více datových polí. ID šablony, která má být použita pro interpretaci tohoto typu záznamu, je přítomno v hlavičce setu. Platí tedy, že ID datového setu je stejné jako ID nějaké šablony. Pokud není šablona k dispozici, nemůže být datový záznam interpretován.



Obrázek 3.12: Formát datového záznamu[4]

- *Záznam šablony možností.* Tento typ záznamu předává kolektoru dodatečné informace, které nelze předat jen pomocí záznamů o toku. Popisuje ostatní datová pole, která nenesou pouze informace o toku[4, 21].

3.4 Implementace v systémech Flowmon

Implementace NetFlow a IPFIX v systémech firmy Flowmon majoritně odpovídá popsaným doporučením dle standardů. Navíc ke standardu IPFIX firma k záznamu přidává 4 pole, která jsou zobrazena na obrázku 3.13. Na levé straně je označení tohoto pole a na pravé jeho význam. Na následujícím obrázku 3.14 je popořadě vypsán IPFIX kód exportovaného pole a identifikátor firmy.

IPFIX_INVEA_HOST	HTTP hostname
IPFIX_INVEA_URL	HTTP URL
IPFIX_INVEA_METHOD_ID	HTTP method
IPFIX_INVEA_STATUS_CODE	HTTP result code

Obrázek 3.13: Exportované pole v systémech Flowmon

IPFIX	Ent. ID
1	39499
2	39499
4	39499
12	39499

Obrázek 3.14: Identifikační kódy

Kapitola 4

Datová sada

Následující kapitola obsahuje popis datové sady, která byla využívána v této práci pro potřeby analýzy provozu proxy serverů. Pro vytvoření datové sady byla vystavěna virtuální síť, obsahující virtuální servery a koncové stanice. Datová sada, využívaná v této práci, se skládá pouze ze síťových paketů odchycených v rámci této sítě.

V této kapitole je nejdříve popsáno prostředí, ve kterém byla virtuální síť vystavěna, následně architektura sítě jako takové a také role přidělené strojům v této síti. Následně je popsána vytvořená datová sada.

4.1 Prostředí pro vytvoření datové sady

Pro vytvoření datové sady byla vystavěna virtuální síť. V rámci této sítě byly spuštěny stroje, z nichž některé fungovaly jako koncové klientské stanice, některé jako proxy servery a některé jako koncové servery. Celá síť byla také připojena k Internetu, aby mohl být odchytáván reálný webový provoz. V následující sekci je nejdříve popsán využívaný software a typy virtuálních strojů. Dále je blíže popsána vytvořená síťová struktura, která byla využívána pro vytvoření a odchytávání síťového provozu.

Použité nástroje

Pro účely vytvoření datové sady byla vystavěna virtuální síť, složená z virtuálních strojů různých typů. Celá virtualizace probíhala na jednom stroji Lenovo Ideapad M15, který ovšem nebyl dedikovaný pouze pro tuto práci. Pro vystavění sítě byl využit software EVE-NG[1]. Jde o emulátor síťového prostředí, který podporuje virtualizaci různých strojů s různými operačními systémy. Výhodou tohoto softwaru je jeho podpora pro různá virtuální zařízení, jeho možnost běžet naprosto izolovaně od vnější sítě a také velká uživatelská jednoduchost při vytváření síťových struktur. Nástroj byl vybrán také z důvodu, že je využíván na Fakultě Informačních Technologií při výuce, a tedy šlo o známé prostředí. Pro účely této práce byla vybrána bezplatná edice.

Nástroj je dostupný ve formě OVF obrazu připraveného pro virtualizaci anebo ISO obrazu připraveného pro instalaci. Pro účely práce byla vybrána verze určená pro virtualizaci. Obraz softwaru byl spouštěn ve virtualizačním softwaru VMWare Workstation Player. Po spuštění virtuálního EVE-NG softwaru a konfiguraci uživatelů je možné k běžícímu soft-

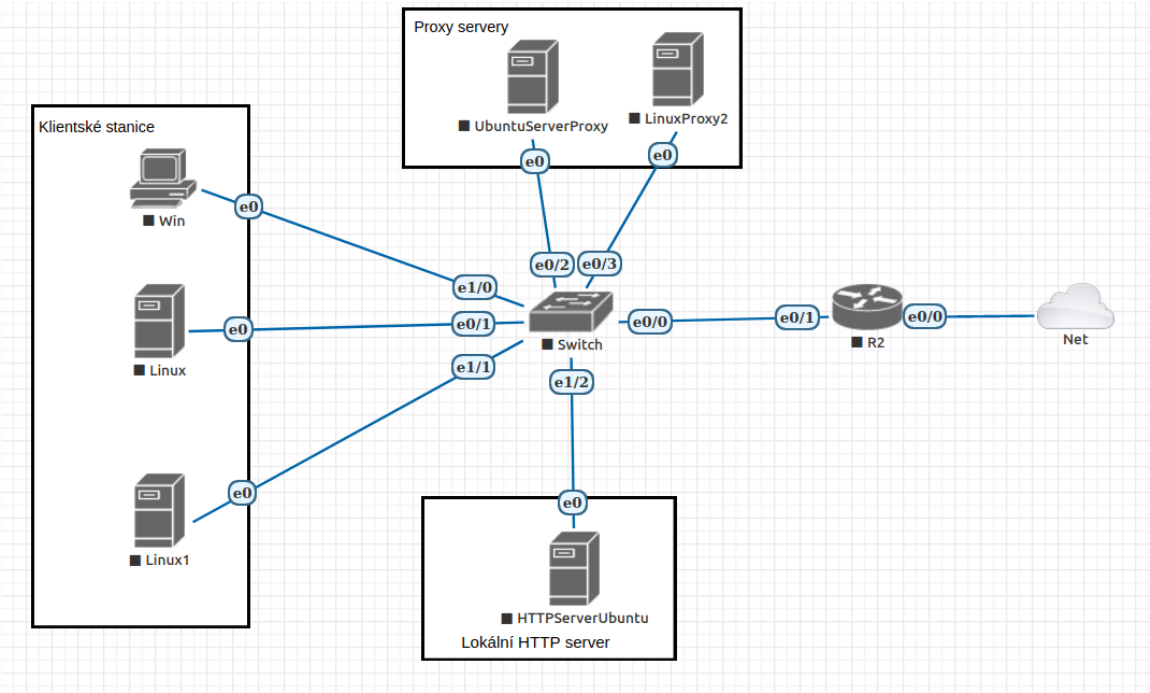
[1] https://www.eve-ng.net/

waru přistoupit pomocí webové aplikace. V té je po přihlášení možné vytvářet jednotlivé projekty se síťovými strukturami, ale také kontrolovat momentální stav síťového emulátoru.

Na obrázku 4.1 lze vidět snímek obrazovky z prostředí aplikace. Jde o stránku s otevřeným projektem se síťovou strukturou. Pro vytváření síťových struktur je zde přehledné grafické prostředí. Pomocí funkcí v levém panelu je možné přidávat a spravovat jednotlivé uzly sítě, provádět akce související se všemi uzly v struktuře (např. zapnout všechny uzly, vypnout všechny uzly nebo resetovat všechny uzly).

Jednotlivé stroje běží jako vlastní virtuální obrazy v rámci emulátoru. Pro jejich spuštění je nutné do EVE-NG softwaru nahrát a na správné místo uložit jejich obrazy. Pro účely této práce byly na internetu nalezeny, staženy a použity následující virtuální obrazy:

- L2-ADVENTERPRISE-M-15.1-20140814,
- L3-ADVENTERPRISE9-15.5.2T,
- linux-ubuntu-18.04-server,
- Win10_21H2_Czech_x64.



Obrázek 4.1: Screenshot prohlížečové aplikace EVE-NG(vložit aktuální)

K jednotlivým spuštěným strojům se lze poté připojit několika způsoby. Je možné se ke strojům připojit pomocí konzolové aplikace `telnet` anebo pomocí nástroje pro vzdálené připojení ke grafickému uživatelskému rozhraní `vnc`. Dále je možné spustit EVE-NG webovou aplikaci v módu HTML5 konzole, díky čemuž je možné se připojit ke strojům přímo ve webovém prohlížeči.

Síťová struktura

Jak bylo popsáno v kapitole 3, typů proxy serverů je velké množství. Při vytváření datové sady bylo nutné myslet na co největší pestrost, aby po následná analýze bylo možné vytvořit co nejuniverzálnější řešení. Bylo nutné vzít v úvahu, že různé operační systémy mohou využívat různé způsoby možností připojování se k proxy serverům. Následně bylo nutné vzít v úvahu různé typy proxy serverů a mít na paměti, že implementační detaily mohou být rozdílné u každého jednoho proxy serveru dostupného na trhu, ať už poskytovaného zdarma anebo placeného. Vv neposlední řadě bylo důležité myslet na to, že komunikace s různými koncovými servery může probíhat rozdílně pro každý proxy server. Byla tedy snaha, aby vystavěná síťová struktura byla velmi různorodá, bylo vyzkoušeno více operačních systémů, ať už v roli klientské stanice nebo proxy serveru a více přístupů k proxying, tak jak byly popsány. Dle operačního systému bylo zkoušeno více aplikací, které fungují jako proxy servery a byly zkoušeny jak dopředné, tak reverzní proxy servery. Byl spuštěn také koncový server pro HTTP dotazy v rámci sítě, ale většina datové sady se skládá z odchycené komunikace z dotazů na Internet. Komunikace probíhající pouze uvnitř sítě poskytla dobrý prvotní náhled na fungování proxy serverů. Nicméně vytvořená síťová komunikace byla oproti reálné moderní komunikaci v Internetu, kdy je při přístupu na jednotlivé webové stránky stahován obsah z několika serverů, poněkud chudá.

Dále jsou popsány jednotlivé stroje a software, který na nich byl využit. Byla vystavěna síť obsahující 6 virtuálních strojů. Každému stroji byla staticky přidělena IP adresa z rozsahu 10.0.0.0/24, definována výchozí brána a DNS server (adresa 8.8.8.8, DNS server Google). Dále byl v síti funkční L2 switch a L3 router. Výpis strojů je uveden dále. Pouze pro případ zřetězení proxy serverů za sebe byl kvůli problémům s NAT překladem rozsah adres změněn a v datové sadě je tedy u tohoto případu rozsah 192.168.8.0/24.

Koncová uživatelská stanice

Jde o stroj s operačním systémem Ubuntu Server 20.04. Pro tento stroj nebyl potřeba žádný speciální software, stačil webový prohlížeč, který byl v obrazu operačního systému již zakomponován. Využívaný prohlížeč byl Mozzila Firefox. Nastavení cesty k proxy serveru probíhalo v nastaveních prohlížeče, využit byl tedy přístup, kdy si je aplikace vědoma, že probíhá proxying. Tento stroj byl v síťové struktuře vytvořen dvakrát, z důvodu spojování síťových toků z dvou stejných zařízení.

Koncový server

Stroj s operačním systémem Ubuntu Server 20.04, který sloužil pro testování spojení skrz proxy server a pro jednoduchou komunikaci. Na tomto serveru byl nainstalován balíček Apache 2, takže který sloužil jako jednoduchý HTTP server. Podpora pro HTTPS byla také definována pomocí OpenSSL aplikace dostupné v základní výbavě operačního systému.

Proxy server Linux

Stroj s operačním systémem Ubuntu Server 20.04, který sloužil jako proxy server. Oproti popisu v kapitole 2 bylo ve fázi vytváření datové sady stroji přiděleno pouze jedno rozhraní. Na stroj byly nainstalované 3 aplikace fungující jako dopředné HTTP proxy servery (Squid, Privoxy a Tinyproxy) a jedna fungující jako reverzní HTTP proxy (Haproxy). U dopředných proxy serverů stačilo nastavit port, na kterém bude server poslouchat. Tyto porty byly poté

důležité při analýze síťové komunikace, protože signalizovaly využití proxy serveru. Porty byly nastaveny následovně:

- Squid - 3128,
- Privoxy - 8118
- Tinyproxy - 8888

U reverzního bylo záhodno zachovat zdání HTTP, případně HTTPS serveru, tudíž port byl nastaven na 80 a 443. Bylo však také nutné nastavit podporu pro HTTPS a server, na který budou požadavky přeposílány (definován byl Koncový server popsaný výše).

Byl vyzkoušen také dopředný SOCKS proxy server, který šlo jednoduše vytvořit pomocí `ssh` příkazu s dynamickým předáváním portů. Příkaz byl následující: `ssh -N -D 0.0.0.0:1080 localhost`, kde příznak 'D' znamená dynamické předávání portů na portu 1080, které využívá SOCKS protokol, příznak 'N' znamená, že ssh příkaz zůstane nečinný.

Klientská stanice Windows

Šlo o stroj s operačním systémem Windows 10. Stroj byl využíván jako klientská stanice při komunikaci přes proxy server. V systému Windows lze nastavit využívání proxy serveru jednoduše v nastavení. Využíván byl prohlížeč Microsoft Edge.

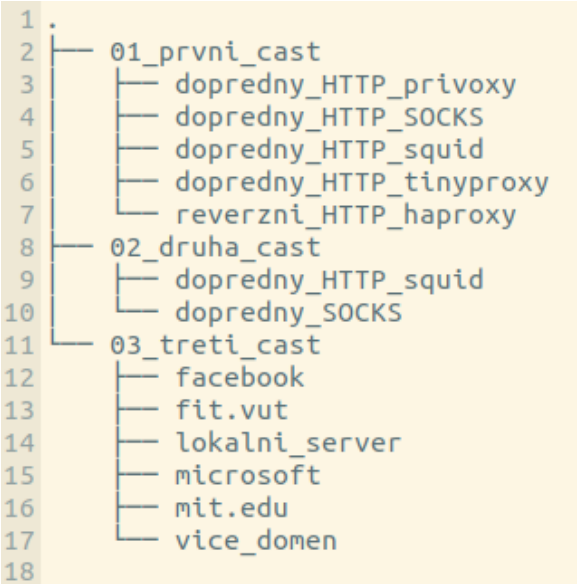
4.2 Popis datové sady

Provoz v síti byl uměle vytvářen a následně odchytáván. Odchytávání probíhalo vždy na síťovém rozhraní proxy serveru pomocí programu Wireshark. Z uživatelské stanice byly za využití proxy serveru vytvářeny požadavky na různé webové stránky a z odchycené komunikace vzešlé z těchto požadavků je vytvořena datová sada.

Datová sada se skládá z kolekce `pcap` souborů odchycených v prostředí vytvořené sítě. Tvoří ji záznamy komunikace mezi koncovou uživatelskou stanicí, proxy serverem a koncovými servery, na kterých jsou uloženy dotazované webové stránky. Výpis domén, na které cílily dotazy, je uveden v následujícím seznamu:

- Lokální server s nainstalovaným Apache,
- Facebook.com,
- Google.com,
- Seznam.cz,
- Wikipedia.org,
- Microsoft.com,
- MIT.edu,
- Fit.vut.cz.

Je samozřejmé, že v této době dotaz na webovou adresu neznamená dotaz pouze na jeden server. Lze předpokládat, že jsou společnostmi, které vlastní výše vypsané stránky, využívány proxy servery pro rozložení zátěže. Dále že DNS záznamy odkazují na více serverů a že finální webová stránka se skládá z dat z více různých serverů, například reklamních, a podobně. Každý dotaz na webovou doménu tedy často znamená několik spojení na různé IP adresy a v konečném důsledku tedy několik záznamů o síťových tocích, které bude potřeba brát v potaz.



Obrázek 4.2: Souborová struktura datové sady

Vytvořená datová sada může být rozdělena na tři části. První část sloužila pro základní analýzu a otestování teorií o fungování proxy serverů. Druhá část poté byla vytvořena pro účely vývoje a testování implementace navrženého řešení. Třetí část obsahuje komunikaci využívající více proxy serverů. Vypsaná struktura datové sady je na obrázku 4.2.

V první části vytváření datové sady byla pro každou webovou doménu vytvořena sada tří `pcap` souborů. Takto byla vytvořena datová sada za využití tří dopředných HTTP, jednoho dopředného SOCKS a reverzního HTTP proxy serveru. Každý z těchto souborů obsahuje kompletní komunikaci jedné klientské stanice s jednou webovou doménou, která následuje zadání URI do adresního řádku prohlížeče. Aby bylo dosaženo odchycení kompletní komunikace při každém pokusu, byla pokaždé vymazána kompletní cache paměť prohlížeče. Takto byla vytvořena sada pro každý výše zmíněný typ proxy serveru, přičemž pro reverzní proxy server existuje datová sada pouze pro komunikaci s lokálním serverem, na kterém běželo Apache. Z této části datové sady byly následně vytvářeny záznamy o síťových tocích, které sloužily pro první návrh algoritmu pro řešení korelace záznamu síťových toků.

Soubory, které patří do této části nebyly po vytvoření nijak upravovány. Obsahují tedy pro nás důležitou odchycenou síťovou komunikaci, ale také nějaké režijní komunikace systému, které nemusí být vůbec důležité. Samozřejmě je přítomna také DNS komunikace, která ale není analyzována, protože se vyskytuje pouze na jedné straně proxy serveru. V tabulce 4.1 jsou vypsány některé statistické hodnoty související s první částí datové sady. Vypsána je vždy průměrná hodnota parametru. Průměr hodnot byl získán z 12 souborů, protože každou doménu byly vytvořeny tři soubory s kompletní komunikací a byly zkoušeny

tři dopředné HTTP a jeden dopředný SOCKS server. Výjimkou je lokální webový server, kde byl využit reverzní proxy server, protože podpora pro SOCKS nebyla na serveru instalována. Celková velikost této části je 267 MB.

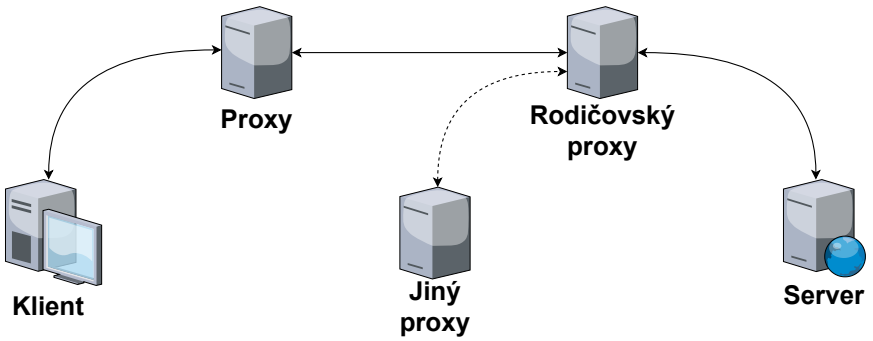
V druhé části vytváření datové sady byly vytvořeny větší pcap soubory, které obsahují komunikace z více klientských stanic, které probíhaly ve stejný čas. V těchto souborech se také nachází komunikace s různými webovými servery z různých domén. Pro vytvoření této části sady byl využíván pouze jeden dopředný proxy server, a to Squid. Z této sady byly následně vytvářeny záznamy o síťových tocích, které sloužily pro další vývoj a testování navrženého algoritmu. Soubory v této části jsou rozděleny dle kombinací operačních klientských stanic, které se v komunikaci vyskytují. Celková velikost této datové sady je 285 MB.

Cílová doména	Průměrná velikost souboru (MB)	Počet souborů	Průměrný počet TCP toků do proxy	Průměrný počet TCP toků z proxy
Lokální server	0,03	12	4	14
facebook.com	1	12	8	16
google.com	0,5	12	7	15
seznam.cz	11	12	55	117
wikipedia.org	0,18	12	2	6
microsoft.com	1,8	12	13	63
mit.edu	4,4	12	14	30
fit.vut.cz	3,3	12	9	13

Tabulka 4.1: Statistické udaje první části datové sady

Zřetězení více proxy serverů za sebou.

V potaz byly vzaty také případy, kdy je více proxy serverů provázáno za sebe a tvoří hierarchickou strukturu. To v principu znamená, že proxy server, který přijímá požadavek od klienta, ho nezasílá na koncový server, ale přeposílá na další určený proxy server, který je nazýván rodičovským. Tato stromová struktura může dále napomáhat k rozložení zátěže v síti. Vizualizaci takto zřetězených proxy serverů je možné vidět na obrázku 4.3.



Obrázek 4.3: Vizualizace zřetězených proxy serverů

Pro vytvoření datové sady s tokem přes více proxy serverů, byla využita aplikace Squid, která jednoduše dovoluje nastavit rodičovský proxy server, na který bude přeposílat požadavky, a úplně zakázat přímý přístup ke koncovému serveru. Takto provázány byly dva proxy servery. Větší strukturu již za prvé nedovolovaly hardwarové zdroje a za druhé byl předpoklad, že obecný princip pro výsledné řešení bude stejný pro jakékoliv množství provázaných proxy serverů.

Datová sada s tímto typem komunikace za použití proxy serverů byla vytvořena podobným způsobem jako první část, s tím že ale pro dotaz na každou z domén byla komunikace odchytávána pouze jednou. Pro každý případ je vytvořena sada 2 pcap souborů, které byly vytvořeny odchytáváním komunikace na rozhraních jednotlivých proxy serverů. Statistické informace ohledně této části datové sady jsou obsaženy v tabulce 4.2. Celkem třetí část obsahuje 88 MB dat.

Cílová doména	Celková velikost obou souborů (MB)	Počet TCP toků před prvním proxy	Počet TCP toků mezi proxy	Počet TCP toků za druhým proxy
facebook.com	2,4	17	17	27
fit.vut.cz	8	16	21	31
microsoft.com	4,4	24	23	45
mit.edu	6,3	26	23	37
Lokální server	0,17	9	15	13
Více různých domén	66,8	84	240	371

Tabulka 4.2: Statistické udaje druhé části datové sady

Celková velikost datové sady je tedy 641 MB dat. Do toho není započítána velikost testovací datové sady, protože ta byla vytvořena výběrem specifických datových toků. V kapitole 5 je popsán způsob exportu záznamu o síťovém toku. Protože je pro to potřeba specifický software, jsou exportované záznamy také přiloženy k datové sadě, ale do celkové velikosti datové sady započítány nejsou.

Kapitola 5

Navržené řešení a algoritmus

V této kapitole je popsáno navržené řešení problému korelace záznamů síťových toků proxy serverů. V rámci popisu řešení jsou nejdříve popsány a diskutovány již existující obecné korelační metody. Následně je popsána vlastní analýza záznamů komunikace a dále také využití atributů v exportovaných záznamech, které je možné získat pomocí softwaru firmy Flowmon.

Následně je v kapitole popsán navržený algoritmus, na základě kterého by mělo být možné provádět automatizovanou korelaci záznamů síťových toků. Algoritmus je popsán nejdříve v čistém základu, jak může fungovat bez jakéhokoliv uživatelského vstupu, a následně s možnými rozšířeními, které zpřesňují jeho výsledky.

5.1 Současné korelační metody

Obsah této sekce je inspirován diplomovou prací s názvem *Korelace dat na vstupu a výstupu sítě Tor*. V té je řešen problém, který má podobné parametry jako problém řešen v této práci. V této sekci jsou přiblíženy existující metody, které se v současné době využívají. Metody nebyly implementovány ani testovány, ale byly využity pro návrh řešení problému popsaného v této práci.

Následující text je přepsán z výše zmíněné diplomové práce [8] a upraven aby odpovídal řešenému problému.

Definice požadovaných vlastností metody

Wang et al. [23] definovali matematicky korelační metodu pro určení závislosti dvou datových toků. Od korelační metody je požadováno, aby co nejpřesněji určila závislost dvou datových toků. Označím-li $Proxy_{in}$ jako množinu všech datových toků na jedné straně proxy serveru, $Proxy_{out}$ jako množinu všech datových toků na druhé straně proxy serveru, tak ideální korelační funkce je definována jako $CF : Proxy_{in} \times Proxy_{out} \rightarrow 0, 1$.

$$CF = \begin{cases} 1 & \Leftrightarrow \text{datové toky } \textbf{jsou} \text{ v korelaci} \\ 0 & \Leftrightarrow \text{datové toky } \textbf{nejsou} \text{ v korelaci} \end{cases}$$

Charakteristiku datového toku lze modelovat metrickou funkcí $M : F \times P \rightarrow Z$, kde F je doména datových toků, P je vybraná doména parametrů datových toků a Z je doména korelační metody. Na základě metriky lze postavit ohodnocenou korelační funkci $CVF : Z \times Z \rightarrow \langle 0, 1 \rangle$, kde výsledkem je reálné číslo mezi 0 a 1. Touto funkcí CVF , viz vzorec 5.1, lze aproximovat CF zavedením prahové hodnoty δ takové, že $0 \leq \delta \leq 1$.

$$CVF = (M(f_i, p), M(f_j, p)) \geq \delta \Rightarrow f_i \text{ je v korelaci s } f_j \tag{5.1}$$

K tomu je potřeba nalézt M, p, CVF a δ takové, které splňují podmínky formule 5.2.

$$\forall f_i, f_j \in F : CVF(M(f_i, p), M(f_j, p)) \geq \delta \Rightarrow f_i \text{ je v korelaci s } f_j \tag{5.2}$$

Klíčem je nalézt takové unikátní charakteristiky datového toku, které jsou invariantní při průchodu skrz proxy server, neovlivněné šifrováním a unikátní pro každý datový tok. Na těchto je následně možné postavit spolehlivou korelační metodu. Důležité je, aby výběr unikátních vlastností datových toků zajistil, že bude míra správných pozitivních výsledků (*true positives*) vysoká a míra falešných pozitivních výsledků (*false positives*) nízká.

Výpis existujících metod

V následujícím textu jsou stručně vypsány některé metody, které jsou využívány pro korelaci síťových toků. Metody byly převzaty ze zmíněné práce [8], ale jsou přiloženy odkazy na původní zdroje.

Počítání paketů pro samostatný datový tok

Serjantov et al. [20] analyzovali možnost korelace pomocí počítání paketů, kdy na vstupní a výstupní lince mixu lze pozorovat osamělý datový tok. Využívají počítání paketů v rámci subintervalů, které jsou na sobě vzájemně nezávislé. Pokud je míra výskytu paketů vstupního datového toku téměř stejná jako u výstupního datového toku, je mezi nimi závislost.

Daneziho algoritmus

Danezi navrhuje [9] koeficient míry pravděpodobnosti, že se vstupní datový tok nachází v agregátu výstupních datových toků. K výpočtu koeficientu je nutné mít informaci o ostatních výstupních datových tocích. Metoda využívá modelu zpoždění celé sítě a funguje na principu, kdy útočník pozoruje datový tok např. odpověď webového serveru zpět k uživateli, který je autorem požadavku. Tento datový tok může být reprezentován jako funkce objemu datového toku v čase. Tato funkce je konvoluována s exponenciální funkcí zpoždění, výsledkem je šablona, která předpovídá, jak bude datový tok vypadat v anonymizující síti. Všechny linky v síti jsou pak porovnány s tímto odhadem a je určena míra podobnosti s danou šablonou datového toku. Podle toho lze uzly klasifikovat jako potenciální první, druhý a třetí uzel na cestě daného spojení.

Vzájemná informace a frekvenční analýza datových toků

Zhu et al. navrhují dvě třídy korelačních metod [24], jmenovitě metody postavené na časové doméně a metody postavené na frekvenční doméně. První kategorie metod využívá ke korelaci datových toků statistické informace o míře výskytu počtu paketů v čase a druhá kategorie metod využívá ke korelaci datových toků spektrální funkci počtu výskytu paketů v čase, tedy jaké jsou frekvence počtu výskytů paketů v rámci časového intervalu.

5.2 Analýza datové sady

V této sekci je popsána analýza, která byla provedena nad vytvořenou datovou sadou. Nejdříve je popsán software, který byl využit k vytváření záznamů síťových toků ze záznamů komunikace. Následně je popsán záznam jako samotný a jeho jednotlivé komponenty a atributy. Dále jsou diskutovány možnosti, jakými by mohlo být možné vytvořit řešení problému korelace.

Flowmon sonda

Flowmon sonda je jedním z hlavních produktů firmy Flowmon. Jde o exportér záznamů o síťovém toku, který podporuje jak formát NetFlow tak IPFIX[1]. Mimo to má rozsáhlou analytickou funkcionalitu, kterou je pro účely této práce zbytečné popisovat. Důležitá pro tuto práci je funkce exportu záznamu síťového toku z `pcap` souboru. Operační systém sondy je CentOS 7.

Flowmon sonda byla od firmy Flowmon poskytnuta ve formě OVF obrazu pro virtualizaci. Stejně jako v předchozích případech byl využit software VMware Workstation Player pro spuštění virtuálního stroje. Po spuštění a počáteční konfiguraci, která zahrnovala ruční nastavení IP adresy, protože na sondě není implicitně povoleno DHCP, je možné se k sondě připojit skrz webovou aplikaci. Tato aplikace nabízí širokou škálu konfiguračních a analytických možností, které však nebylo nutné využívat. Pro přístup k sondě stačilo SSH spojení.

Export záznamů o síťovém toku

Pro exportování síťového toku bylo možné využít konzolovou aplikaci dostupnou v prostředí Flowmon sondy. Aplikaci bylo možné spustit buď přímo v okně programu VMware anebo za pomocí SSH spojení. Aplikace měla název `flowmonexp5`, jsou pro ni nutná root oprávnění, proto je v systému udělena výjimka pro spouštění aplikace s příkazem `sudo`. Příkaz pro export síťového toku z `pcap` souboru je:

```
sudo flowmonexp5 -I pcap-replay:file=[INPUT_FILE],speed=0 \
-E csv > [OUTPUT_FILE.csv]
```

V příkazu je na místě `INPUT_FILE` nutné doplnit cestu ke vstupnímu souboru ve formátu `pcap` nebo `pcapng`, a na místě `OUTPUT_FILE` jméno souboru, do kterého bude zapsán výstup ve formátu csv. Je možné měnit parametr `speed`, který může nabývat hodnot 0 anebo 1. Hodnota 0 znamená, že exportér nebude respektovat pořadí záznamů ve vstupním souboru a bude je zpracovávat jak bude možné. Při hodnotě 1 bude exportér respektovat i pořadí a časovou hodnotu záznamů a bude soubor zpracovávat podle toho, nicméně zpracování v tu chvíli trvá stejnou dobu jako trvalo odchytávání souboru. Informace o časovém pořadí toků však zůstává zachována ve výstupu v obou případech, i když při hodnotě 0 nejsou záznamy na výstupu podle času seřazeny.

Výstupem výše uvedeného příkazu je základní záznam o síťovém toku. Obsahuje důležité informace jako verze protokolů na L3 a L4, zdrojové a cílové IP adresy, zdrojové a cílové porty a časovou informaci. Neobsahuje ovšem informace o aplikačním protokolu, který je v daném toku používán. Tuto informaci je nutné do záznamu explicitně přidat.

Vzhledem k tomu, že analýza probíhala nad záznamy síťové komunikace, ve které figurovaly proxy servery, dalo se předpokládat, že velká majorita záznamů bude obsahovat HTTP

anebo HTTPS komunikaci. Kvůli tomu bylo jasné, že bude potřeba přidat do exportovaného záznamu také informace o HTTP a TLS protokolech. To je možné docílit přidáním následujících parametrů do předchozího příkazu:

```
-X /usr/local/lib/flowmonexp/plugin-tls.so -P tls
-X /usr/local/lib/flowmonexp/plugin-http.so -P http
```

Přidáním těchto parametrů do příkazu zajistíme exportování dalších atributů souvisejících s těmito protokoly. Seznam některých těchto atributů, které byly považovány za důležité, je v tabulce 5.1.

HTTP	**TLS**
HTTP_METHOD_MASK	TLS_CONTENT_TYPE
HTTP_REQUEST_HOST	TLS_SERVER_VERSION
HTTP_REQUEST_URL	TLS_SERVER_RANDOM
HTTP_REQUEST_URL_SHORT	TLS_CIPHER_SUITE
HTTP_REQUEST_REFERER	TLS_SNI
HTTP_REQUEST_AGENT	TLS_CLIENT_VERSION
HTTP_RESPONSE_STATUS_CODE	TLS_CIPHER_SUITES
HTTP_RESPONSE_CONTENT_TYPE	TLS_CLIENT_RANDOM

Tabulka 5.1: Exportované informační elementy HTTP a TLS

Nyní již byl k dispozici dostatek atributů, aby mohla být provedena analýza. Velké množství informací, se kterými bylo možné pracovat, poskytl protokol TLS. Tento protokol posílá v rámci handshake velké množství dat, které jsou patrně unikátní v rámci různých síťových komunikací. Bylo tedy předpokládáno, že díky exportovaným atributům tohoto protokolu bude možné spolehlivě rozdělit a korelovat jednotlivé síťové toky, ve kterých se protokol vyskytuje.

Komunikace skrz proxy server, která probíhala zapouzdřená v protokolu SOCKS, vypadala velice podobně jako komunikace normální. Protokol přidal do komunikace pouze režijní zprávy, nicméně základ komunikace byl nezměněný. Záznamy exportované z těchto toků také obsahovaly stejné množství atributů jako záznamy toků, které SOCKS nevyužívaly. Protokolu SOCKS jako takovému tedy nebylo již dále nutné věnovat větší pozornost.

Možnosti řešení

V sekci 5.1 této kapitoly byly popsány metody korelace síťových provozů. Tyto metody navrhují určité způsoby jak korelaci řešit, jako například počítání paketů anebo korelaci dle časových informací. Co se týče druhého z návrhů, časové korelace, jde rozhodně rozhodně o užitečný nástroj. Na druhou stranu ale je počítání paketů v našem případě celkem nepoužitelné. Při pohledu na exportované záznamy toků je vidět, že počty paketů v tocích nejsou tak diverzní mezi jednotlivými toky, aby mohla být dle nich jednoznačně určena korelace.

Naštěstí jednoduchý proxy server předává mnohem více informací v nezměněné podobě, než například síť Tor, pro kterou jsou korelační metody převážně určeny. Výhodou je například velké množství atributů exportovaných z protokolu TLS a to, že máme stejný přístup k oběma stranám proxy serveru. Při hledání řešení byl tedy nejvíce sledován tento protokol a byla snaha z něj získat dostatečné množství unikátních charakteristik toku, jak bylo popsáno v sekci 5.1.

Techniky využití TLS protokolu

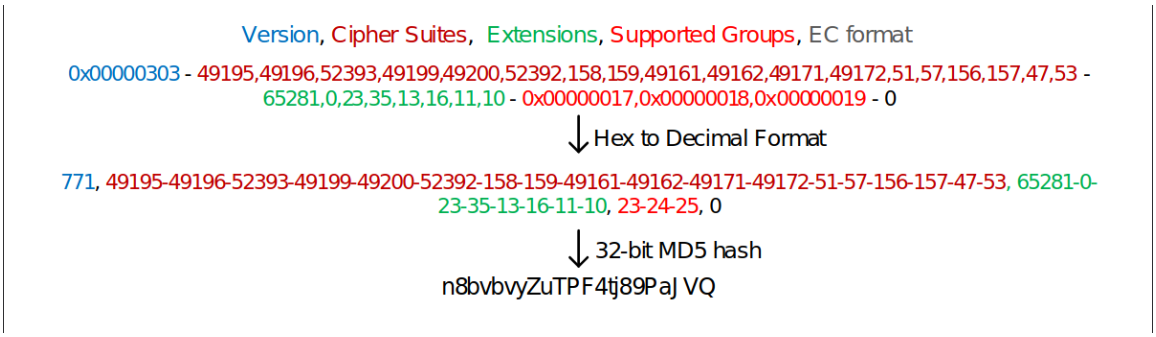
Velké množství atributů v záznamu síťového toku znamená velký potenciál pro využití. Důležité je však pro rozdělení nesouvisejících a spojení souvisejících toků vybrat pouze ty atributy, které zaručí ve své kombinaci jedinečnost záznamu. Jednou ze známých technik, která takto využívá atributy exportované z toků obsahujících TLS protokol je metoda určování 'otisku prstu' aplikace pomocí *JA3 hashe*.

Techniku JA3 hashe vyvinula trojice John B. Althouse, Jeff Atkinson a Josh Atkins v roce 2017. Byla publikovaná s opensource licencí pod cloudovou softwarovou společností Salesforce.com. Od té doby byla podpora pro tuto techniku implementována do několika softwarů a byla vyvinuta také technika JA3S hashe pro určování 'otisku prstu' serveru[5].

JA3 hash se vypočítává z parametrů zprávy *Client Hello*, která je zasílána jako iniciátor TLS spojení. Tato zpráva je zasílána v čisté podobě, na rozdíl od zbytku TLS komunikace která je šifrovaná. Struktura této zprávy a způsob jejího vytvoření jsou přímo závislé na implementaci klientské aplikace. Z detailů zprávy je tedy možné otisk prstu určit. Postupem metody JA3 hashe je shromáždit decimální hodnoty bajtů z následujících polí v Client Hello zprávě:

- verze,
- přijímané typy šifer,
- seznam rozšíření,
- eliptické křivky,
- formát eliptických křivek.

Tyto hodnoty jsou následně spojeny do jednoho řetězce, přičemž je použit symbol ',' jako rozdělovník jednotlivých polí a symbol '-' jako rozdělovník jednotlivých hodnot v rámci jednoho pole. Řetězec je následně zpracován MD5 hash funkcí, která na výstup vyprodukuje 32 bitů dlouhý řetězec. Tento řetězec je JA3 hash TLS otisk prstu[5]. Příklad vytvoření JA3 hashe je na obrázku 5.1.



Obrázek 5.1: Příklad vypočítání JA3 hashe[14]

Bohužel z exportovaných atributů v záznamech o síťovém toku nebylo možné JA3 hash vypočítat. Nicméně naštěstí samotný software `flowmonexp5` export takového atributu povoloval. Bylo nutné ještě rozšířit příkaz pro export a parametr způsobující export TLS atributů doplnit následovně:

```
-X /usr/local/lib/flowmonexp/plugin-tls.so \
-P tls:fields=MAIN#CLIENT#CERT#JA3
```

Díky tomu byl získán a vyexportován JA3 hash v těch záznamech o tocích, ze kterých to bylo možné.

Analýza fungování a atributů TLS protokolu

Analýza probíhala nejdříve nad záznamy exportovanými z jednotlivých pcap souborů obsahující pouze komunikace s jedním webovým serverem. Využívány byly záznamy komunikace, které byly vždy rozděleny podle komunikace s jednou doménou, jak bylo popsáno v kapitole 4. Z těchto záznamů byly za pomocí programu Wireshark vytaženy vždy dva TCP toky, u kterých byla jistota, že je možné korelovat skrz proxy server. Exportované záznamy z těchto souborů sloužily jako základ pro první analýzu TLS spojení skrz proxy server.

Při analýze jednotlivých exportovaných toků bylo zjištěno, že zkoušené proxy servery využívají stejnou strukturu Client Hello zprávy, jakou přijaly od klienta, pro iniciování komunikace s koncovým serverem. To znamená, že atributy TLS komunikace jsou pro oba toky totožné a mimo jiné také, že totožné byly také JA3 hashe.

IPV4_SRC	IPV4_DEST	TLS_JA3_FINGERPRINT
10.0.0.20	142.251.36.99	NIL
10.0.0.20	142.251.36.132	B20B44B18B853EF29AB773E921B03422
157.240.30.3	10.0.0.20	NIL
10.0.0.20	8.8.8.8	NIL
10.0.0.20	157.240.20.19	B20B44B18B853EF29AB773E921B03422
10.0.0.20	8.8.8.8	NIL
10.0.0.50	10.0.0.20	B20B44B18B853EF29AB773E921B03422
10.0.0.50	10.0.0.20	B20B44B18B853EF29AB773E921B03422
10.0.0.20	157.240.20.19	B20B44B18B853EF29AB773E921B03422

Tabulka 5.2: Příklad exportovaných záznamů s JA3 hashem

Ve chvíli, kdy však byly porovnány vyexportované JA3 hashe napříč různými záznamy o síťových tocích, bylo jasné, že je pouze za použití této metody nebude možné od sebe rozlišit na tolik, aby mohla být určena jejich korelace. JA3 hashe byly sice totožné pro toky u kterých byla korelace jistá, nicméně stejný JA3 hash se vyskytoval u vícero takových toků. Nejspíše to bylo způsobeno použitím internetového prohlížeče, který určoval složení TLS Client Hello zprávy. Tyto hashe byly často stejné i pro dva různé stroje se stejnou verzí operačního systému a prohlížeče. Nicméně pro různé operační systémy se hodnoty JA3 hashe vždy lišily. Výsledkem tohoto zkoumání tedy bylo, že bude minimálně možné pomocí metody JA3 hashe minimálně toky rozdělit na více menších skupin, které bude potřeba dále zkoumat. V tabulce 5.2 je možné vidět příklad exportovaných toků s hodnotou JA3 hashe.

Přestože nebylo možné využít pro korelaci záznamů toků výhradně metodu JA3 hashe, záznam obsahoval řadu dalších atributů protokolu TLS, které nebyly touto metodou využívány, anebo u nich byla předpokládána unikátnost. Jeden z takových atributů, který se jevil jako spolehlivý byla hodnota *Server Name Indication (SNI)*. Tato hodnota, přestože je to dle RFC 6066[3] volitelné rozšíření, se vyskytovala ve všech exportovaných tocích ve kterých byly přítomny TLS záznamy a jednoznačně určovala doménové jméno cílového serveru.

Další atributy, které byly jednoznačně unikátní vždy pro danou komunikaci, byly *Server Random* a *Client Random*. Tyto hodnoty jsou z definice náhodně generované pro každé

spojení, protože z nich jsou vypočítávány klíče pro šifrovanou komunikaci.[18] Nicméně opět zde platilo, že stejnou hodnotu jakou proxy server přijal od klienta použil při komunikaci s koncovým serverem.

Výše popsané atributy TLS protokolu byly následně využity v řešení.

Analýza ostatní komunikace

Je zřejmé, že TLS protokol nebyl přítomen ve všech zaznamenaných síťových komunikacích. Například byla přítomna DNS komunikace, která probíhala mezi proxy serverem a nastaveným DNS serverem v reakci na HTTP CONNECT zprávu od klientského stroje. Nicméně DNS protokol nebyl v této práci řešen. Dalo by se sice říct, že dotaz na server databáze doménových jmen je způsoben žádostí o spojení z klientské stanice, je však těžké u tohoto vymezit jasné hranice.

Opomenout však nejde čistou HTTP komunikaci. Ta se vyskytla ve shromážděné datové sadě a lze předpokládat že se stále vyskytuje i v reálném světě, i když už nyní majorita Internetu využívá šifrovaná spojení.

Očividně pro nešifrovanou HTTP komunikaci bylo nutné vytvořit jiný způsob korelace. Jasná nevýhoda byla absence zpráv s jasnou a pokaždé stejnou strukturou na začátcích komunikace, jak tomu bylo u TLS. Co bylo ale výhodou byla přítomnost doménového jména na které komunikace cílila v čisté podobě. Název exportovaného atributu v záznamu je *Request Host*. Dále také byla přítomen atribut s informací o metodě použité v komunikaci (GET, POST, apod.) a to ve formě číselné hodnoty s názvem *Method Mask*. V tabulce 5.3 lze vidět příklad exportovaných toků s popsanými hodnotami.

Výše popsané atributy byly následně využity v řešení.

IPV4_SRC	IPV4_DEST	HTTP_METHOD_MASK	HTTP_REQUEST_HOST
10.0.0.20	142.251.36.99	NIL	NIL
10.0.0.20	142.251.36.132	512	www.google.com
157.240.30.3	10.0.0.20	NIL	NIL
10.0.0.20	8.8.8.8	NIL	NIL
10.0.0.20	157.240.20.19	512	static.xx.fbcdn.net
10.0.0.20	8.8.8.8	NIL	NIL
10.0.0.50	10.0.0.20	33024	static.xx.fbcdn.net:443
10.0.0.50	10.0.0.20	33024	static.xx.fbcdn.net:443
10.0.0.20	157.240.20.19	512	static.xx.fbcdn.net

Tabulka 5.3: Příklad exportovaných záznamů s JA3 hashem

5.3 Navrhovaný způsob řešení problému korelace

V této sekci je popsán navržený způsob, jak řešit problém korelace záznamů toků proxy serverů. Jsou stručně shrnuty poznatky z analýzy datové sady, které jsou pak aplikovány na daný problém. Způsob řešení je nejdříve popsán ve svém základu a poté s možnými navrhovanými rozšířeními.

Vzhledem k tomu, že největší majorita případů, kdy jsou využívány proxy servery, je při HTTP a HTTPS komunikaci, což potvrzuje i vytvořená datová sada, tak navržené řešení se soustředí výhradně na tento typ komunikace. Pro ostatní komunikaci by sice nejspíše bylo možné podobné řešení vytvořit, nicméně zpracovávat větší množství protokolů by bylo již nad možnosti této práce, proto byl vybrán nejčastěji zastoupený protokol v komunikaci.

Jak již bylo zmíněno, protokol SOCKS poskytuje pouze zapouzdření zpráv a do vzhledu exportovaných záznamů vůbec nezasahuje.

Navržený způsob korelace HTTP toků s TLS protokolem

V sekci 5.2 byly již předeslány atributy, které byly využity při návrhu řešení problému korelace. Těmito atributy byly JA3 hash, Server Name Indication a Server a Client Random. V následujícím textu je popsán způsob, jak jsou pomocí těchto atributů toky korelovány.

Způsob, který je zvolen pro hledání záznamů toků, u kterých lze určit korelaci, je následující. Záznamy toků jsou dle jednotlivých atributů rozdělovány do podmnožin, u kterých platilo, že záznamy náležící do dané podmnožiny mají stejnou hodnotu daného atributu jako všechny další v této podmnožině, ale různou hodnotu daného atributu od všech ostatních zpracovávaných záznamů. Ve chvíli, kdy jsou rozděleny do podmnožin, opakuje se stejný postup s dalším atributem, ale pro každou podmnožinu zvlášť.

Pořadí atributů, jak jsou využívány pro rozdělování toků do podmnožin, je následující. Nejdříve je pro rozdělení využit JA3 hash. Díky jeho hodnotě je možné jednoduše rozlišit od sebe různá zařízení s různými aplikacemi. Úspěšně jsou díky němu tedy korelovány toky, které jsou svým způsobem osamocené v celé komunikaci (například související s aplikací, která využívá unikátní nastavení navázání TLS spojení). Ostatní toky jsou díky této hodnotě alespoň rozděleny do podmnožin dle druhu systému anebo aplikace, což napomáhá dalšímu rozdělení.

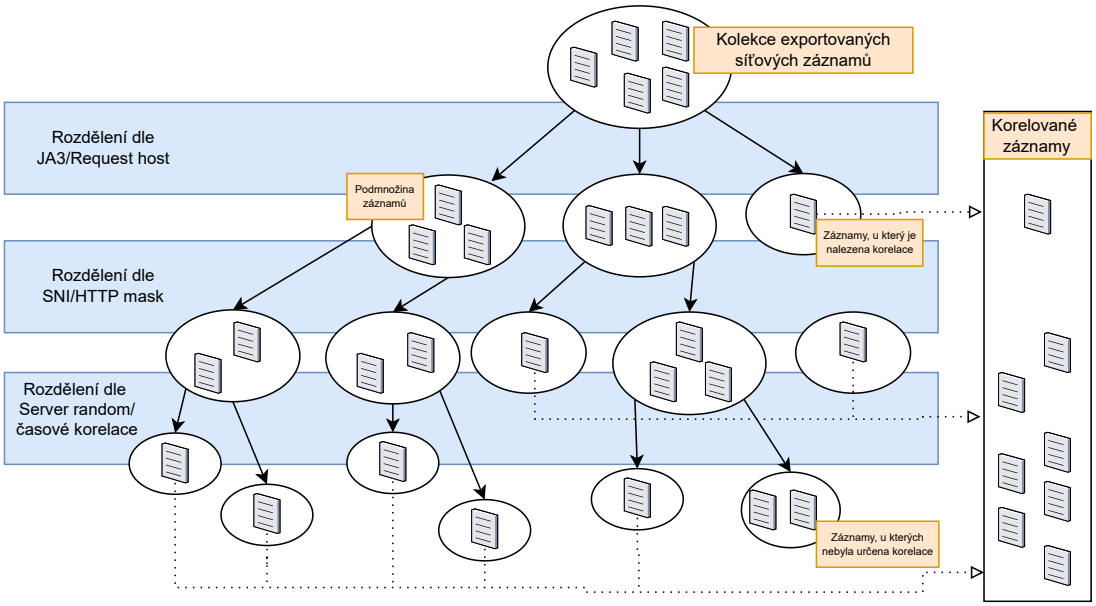
Další využívaný atribut je Server Name Indication (SNI). Jeho použitím vzniká ještě přesnější rozdělení, neboť kombinace identifikátoru zařízení a doménového jména již poskytuje dost velký potenciál pro unikátnost.

Jako poslední využívaný atribut byl zvolen Server Random, který byl zvolen byl spíše nežli atribut Client Random. To bylo z důvodu, že ačkoliv oba atributy jsou dle definice náhodně voleny, nebyla nalezeno přesné chování TLS protokolu ve chvíli, kdy se klient snaží obnovit předchozí TLS spojení se serverem, přičemž zasílá stejné Session ID jako v předchozí komunikaci. I když Session ID s navrženým řešením nijak nesouvisí, indikuje toto chování TLS protokolu znovupoužití stejných hodnot při různých komunikacích ze strany klienta, což by v některých implementacích mohlo znamenat znovupoužití hodnoty Random, a proto byla vybrána hodnota Server Random. Na obrázku 5.2 je vizualizace popsaného řešení.

V základní verzi řešení je podmínkou určení korelace dvou záznamů to, že v jedné podmnožině zůstanou pouze dva záznamy o tocích. V té chvíli jsou tyto záznamy vyjmuty z dalšího zpracovávání a označeny jako související. Toky, které ani po zpracování pomocí všech zvolených atributů nesplňují podmínku, jsou označeny jako toky, u kterých nebylo možné určit korelaci. Neznamená to, že korelace v tu chvíli neexistuje, pouze že nebyla nalezena.

Problémem, který je zřejmě viditelný v tomto návrhu řešení, je že pokud by byl zpracováván dostatečně velký celek dat naráz, může se objevit obdoba takzvaného 'narozeninového problému'. To znamená, že se vygeneruje stejná hodnota Server Random pro dvě různé komunikace. To by způsobilo, že by mezi dvěma komunikacemi mohla být chybně určena korelace, anebo že by korelace nebyla určena vůbec. Takovéto chování je znatelná chyba v navrženém řešení, kterou lze ovšem jednoduše opravit.

Lze předpokládat, že záznamy o tocích, které lze korelovat dle jejich chování, lze korelovat i dle času jejich výskytu. U proxy server lze předpokládat chování, že se snaží přijaté zprávy přeposílat v co nejkratším čase. Tudíž zpoždění mezi časem začátku komunikace mezi klientem a proxy serverem a časem začátku komunikace mezi proxy serverem a kon-



Obrázek 5.2: Vizualizace navrženého řešení

covým serverem se bude blížit časové hodnotě, která je potřebná pro navázání spojení mezi klientem a proxy serverem, předání a zpracování požadavku. Tato hodnota byla dle dat získaných v této práci a popsaných v kapitole 4 vždy menší jak 300 ms. Tato hodnota byla určena dle dostupných dat a nemusí být naprosto přesná, nicméně pro přesnější určení by bylo potřeba více diverzních dat.

Před tím, než je tedy u záznamů toků určena jasná korelace, jsou porovnány časy jejich začátku. Pokud se tyto časy výrazně liší, znamená to, že není mezi toky nalezena jasná korelace.

Navržený způsob korelace HTTP toků bez TLS protokolu

Pro korelaci záznamů toků, které neobsahují TLS protokol, byl zvolen totožný způsob, jaký byl popsán u záznamu toků s přítomností TLS protokolu. Pro rozdělení těchto záznamů byly jen použity jiné atributy. Tyto atributy byly popořadě Request Host a Method Mask.

Podle Request Host, neboli doménového jména cílového serveru, jsou záznamy komunikace rozděleny do podmnožin dle cílového serveru. Tímto způsobem jsou opět snadno určeny korelace u osamělých záznamů toků (obsahují komunikaci s doménovým jménem unikátním pro všechny záznamy) a ostatní záznamy alespoň rozdělí. Další rozdělení probíhá podle Method Mask neboli číselné hodnoty určující metodu komunikace.

Další unikátní atributy již bylo velmi těžké v záznamech najít, a proto je další určování korelace prováděno dle časových údajů. Ohledně určování souvislostí záznamů toků zde platí stejný předpoklad jako v případě záznamů toků s přítomností TLS protokolu. Nicméně protože díky absenci navazování TLS spojení může komunikace probíhat rychleji, byla hodnota hranice určení korelace zvolena jako 100 ms. Protože se může stát, že jsou v rychlém sledu po sobě na proxy server odeslány dva klientské požadavky se stejným cílem a stejnou metodou, je navíc kontrolováno, že se cílový port a cílová IP adresa v tocích liší, než je u nich určena korelace.

Problémy a rozšíření

Přestože je navržené řešení univerzálně použitelné, nepokrývá spolehlivě všechny případy, se kterými je možné se v záznamech o síťových tocích setkat. Výhodou je, že řešení bylo navrženo tak, aby ve svém základu nepotřebovalo žádné předchozí informace o sítích, ze kterých zpracovává záznamy. Může být tedy jen vylepšeno tak, že mu některé takové informace budou poskytnuty.

Zpracování části dat

Navrhované řešení předpokládá zpracování souboru dat naráz. Toto však v praxi není příliš použitelné, neboť záznam o síťovém toku se většinou zpracovává hned, jak je vytvořen, protože není možné ukládat tak obrovské množství dat, jaké generuje síťová komunikace. Řešením je zpracovávat záznamy vždy po určitých časových intervalech, kdy by byla zpracována ta část záznamu, která byla za daný interval získána a byly by určeny korelované záznamy. Zbytek záznamů, které nebyly korelovány, by byl zpracován navíc i s dalším intervalem, v případě, že by daný interval skončil například v půlce komunikace a byl tudíž v daném intervalu vyexportován pouze jeden tok a ne oba, případně všechny.

Kontrola IP adres a portů

Lze předpokládat, že navrhované řešení bude využito v praxi, kdy proxy servery, jejichž záznamy toku zpracovává, stojí na hranicích privátních sítí a internetu. Dále jde také předpokládat, že síťový administrátor využívající řešení pro zpracování dat má informace o proxy serveru, jehož záznamy zpracovává. Těmito informacemi jsou myšleny například čísla využívaných portů, jaké rozhraní, a tedy jaká IP adresa je na straně vnitřní sítě a jaké na straně vnější. Všechny tyto informace mohou být v řešení využity, a tím může být zlepšena jeho přesnost.

První vytvořené rozšíření návrhu řešení je tedy takové, že může být u jednoho z korelovaných toků kontrolován privátní rozsah IP adres obsažených v záznamu komunikace. To, u jakého z dvou toků tento rozsah bude kontrolován, lze určit vstupem, zda se jedná o dopřední nebo reverzní proxy server. Další rozšíření může být kontrola čísla portů obsažených v záznamu komunikace. Pokud ani v jednom z korelovaných záznamů toků není využito číslo portu, na kterém poslouchá, případně skrz který komunikuje proxy server, nemůže být určena korelace, neboť se očividně nejedná o komunikaci skrz proxy server. Opět lze určit, který ze záznamů toků má být kontrolován dle toho, zda jde o dopředný nebo reverzní proxy server.

Problémy se SNI

Při konzultaci navrhovaného řešení s vývojářem firmy Flowmon bylo nadneseno, že hodnota SNI začíná být v TLS komunikaci nedostupná, například kvůli zašifrování hodnoty nebo její úplné absenci, což by mohlo způsobovat chybu při zpracování a korelaci záznamů toků.

V navrženém řešení by absence SNI hodnoty u některých záznamů mohla způsobit špatné určení jejich korelace, případně chybné vyhodnocení, že určení korelace není možné. V závislosti na implementaci by také mohla způsobit vnitřní chybu programu a následně pád celé aplikace pro zpracovávání a korelaci. Proto další rozšíření přidané do řešení je možnost rozhodnout se naprosto ignorovat hodnotu SNI a určovat korelaci jen podle JA3 hashe a Server Random hodnoty. Spolu s kontrolou časových údajů a hodnot IP adres a portů by i tak měly být určeny korelace toků správně.

Více proxy serverů za sebou

Komunikace procházející přes více proxy serverů si držela rysy popsané v tomto textu dříve. Zpracování záznamu takové komunikace a nalezení toků, u kterých je možné určit korelaci, se tedy může řídit stejnými pravidly. Jediné co je potřeba upravit, je ukončující podmínka, kdy je rozhodnuto, že je nalezen finální počet toků, u kterých je určena vzájemná korelace. Nejlehčí by bylo vždy předem vědět, kolik proxy serverů se v původní síti nacházelo a kolik záznamů je tedy potřeba spojit dohromady. To ovšem v reálném světě není možné, protože mohou být nastaveny pravidla kdy komunikace s určitou doménou bude procházet skrze několik proxy serverů v síti a s jinou třeba pouze přes jeden. Může ale být určen maximální počet záznamů toků, které je možné spojit dohromady, a v tu chvíli jsou záznamy toků, jejichž počet v dané podmnožině je menší než maximální počet určených toků, určeny v korelaci.

Reverzní proxy servery

Navržené řešení bohužel nelze aplikovat na reverzní proxy servery. Obecně pro tento typ nebylo nalezeno žádné řešení. Reverzní proxy server naváže spojení se serverem, který stojí za ním, hned při svém spuštění a již nenavazuje nové v případě požadavku klientského stroje, nicméně využívá předchozí. Naprosto tedy kvůli tomuto odpadá možnost korelace za pomocí TLS protokolu, protože exportované atributy jsou úplně diverzní ve všech záznamech, i když byla komunikace odchytávána i v době spuštění aplikace proxy serveru. Možná není ani korelace dle velikosti toků případně počtu paketů, protože reverzní proxy servery si nejspíše v rámci komunikace s koncovým severem předávají i nějaké režijní informace navíc. Jediné co zůstává by mohl být způsob korelace dle času, ale ani zde nebyl nalezen žádný vzorec.

5.4 Vytvořený algoritmus

V následující sekci je popsán samotný algoritmus, který by měl být obecně využitelný pro korelaci záznamu síťových toků proxy serverů. Tento algoritmus zakládá na navrženém řešení popsaném v sekci 5.3, a tedy řeší výhradně HTTP a HTTPS síťovou komunikaci. Algoritmus je popsán jak formou slovní, tak formou pseudokódu.

Algoritmus pro svůj vstup předpokládá již vyexportovaný záznam o síťovém toku, který obsahuje informace o HTTP a TLS protokolech a jeho výstupem je vždy sada záznamů, mezi kterými byla určena korelace. Počet záznamů v sadě je přímo závislý na počtu proxy serverů, přes které komunikace procházela.

Popis algoritmu

Dále následuje slovní popis algoritmu v bodech. Hned poté je algoritmus popsán též pomocí pseudokódu, přičemž na pseudokód jsou ve slovním popisu odkazy.

1. Pokud není záznam o toku seřazen dle času začátku toku, proběhne jeho seřazení dle toho údaje (blok 5.1, řádek 3-4).

2. Je vybrána část vyexportovaného záznamu o toku od začátku až po záznam, jehož hodnota časového údaje o začátku toku je o T větší než hodnota stejného údaje prvního záznamu, kde T je předem specifikovaná hodnota v sekundách. Tato část je dále zpracovávána samostatně (blok 5.1, řádek 7).

3. Ze zpracovávaná části jsou vyfiltrovány záznamy, které neobsahují TCP komunikaci. Následně je část rozdělena na dvě menší části. Jedna z těchto částí obsahuje záznamy komunikace obsahující protokol TLS a druhá ostatní záznamy - neobsahující protokol TLS (blok 5.1, řádky 11-13, 15).

4. Proběhne zpracování záznamů, které obsahují TLS komunikaci (blok 5.2).

   4.1. Záznamy jsou rozděleny na menší části dle hodnoty JA3 hash. Pro každou část platí, že všechny záznamy v této části mají stejnou hodnotu JA3 hash. Pokud jsou v tomto kroku vytvořeny podmnožiny, kde jsou přítomny přesně dva záznamy, je předpoklad že jde o záznamy, u kterých by již mohlo být možné určit korelaci (řádek 5). Je na nich provedena kontrola časové korelace (začátku záznamů navazujících toků nesmí být od sebe dále v čase než 300 ms), dále je na nich provedena kontrola privátního rozsahu IP adres jednoho z toků, pokud je povolena (řádek 7-9). Pokud jsou kontroly v pořádku, jsou dané záznamy vyjmuty z dalšího zpracování a uloženy jako sada. Záznamy u kterých kontroly neprojdou a ostatní záznamy jsou předány k dalšímu zpracování (řádek 10-11).

   4.2. Pokud je povoleno rozdělení dle hodnoty SNI a pokud je tato hodnota ve všech zbylých záznamech, je každá část zbylých záznamů dále rozdělena dle této hodnoty. Pro záznamy, u nichž by již bylo možné určit korelaci, jsou provedeny stejné kontroly jako v předchozím bodě a je s nimi nakládáno stejně. Pokud rozdělení dle SNI povoleno není anebo pokud hodnota chybí v záznamech, pokračuje se hned následujícím bodem.

   4.3. Každá část zbylých záznamů je dále rozdělena dle hodnoty Server Random. Opět jsou pro záznamy u nichž by mohla být určena korelace provedeny kontroly a je s nimi nakládáno stejně jako v předchozích bodech. V tuto chvíli je však porovnán počet záznamů v každé vytvořené podmnožině s hodnotou maximálního počtu záznamů, u kterých je možné určit vzájemnou korelaci. Pokud je počet záznamů v podmnožině menší, jsou toky určeny v korelaci přes více proxy serverů.

   4.4. Zbylé záznamy jsou označeny jako záznamy, u kterých nebyla nalezena korelace s žádným jiným tokem a jsou uloženy zvlášť.

5. Dále proběhne zpracování ostatních záznamů (blok 5.3).

   5.1. Záznamy jsou rozděleny dle hodnoty Request Host. Pro každou část platí, že všechny záznamy v této části mají stejnou hodnotu Request Host. Pravidla při nalezení záznamů, u kterých by mohla být určena korelace jsou stejné jako v bodě 5. Ostatní záznamy jsou předány k dalšímu zpracování.

   5.2. Každá část zbylých záznamů je dále rozdělena dle hodnoty Method Mask. Pravidla při nalezení záznamů, u kterých by mohla být určena korelace jsou stejné jako v předchozím bodě. Ostatní záznamy jsou předány k dalšímu zpracování.

   5.3. Každá číst zbylých záznamů je dále rozdělena dle časové korelace. Pro to aby mohly být u záznamů toků určena korelace v této fázi, musí být jejich vzdálenost v čase menší jak 100 ms. Dále musí být rozdílené číslo cílových portů daných toků a také jejich cílová IP adresa. Pro záznamy, u nichž je určena korelace jsou opět uloženy jako sada a vyjmuty z dalšího zpracování.

   5.4. Zbylé záznamy jsou označeny jako záznamy, u kterých nebyla nalezena korelace s žádným jiným tokem a jsou uloženy zvlášť.

6. Každá uložená sada toků, u kterých byla určena korelace, je vyexportována na výstup (blok 5.1, řádek 17). Pro každý tok je exportována šestice atributů. Tyto atributy jsou: pořadové číslo v rámci všech exportovaných záznamu, čas začátku toku, zdrojová IP adresa, cílová IP adresa, zdrojový port, cílový port.

7. Je vybrána další část exportovaného záznamu, stejně jako v bodě 2. Pokud již nejsou žádné záznamy na zpracování, tak algoritmus skončí (blok 5.1, řádek 21).

8. K vybrané části jsou přidána ty záznamy z předchozí části, u kterých nebyla nalezena korelace. Vybraná část je potom znovu seřazena dle času začátku toku (blok 5.1, řádek 24).

9. Algoritmus se vrací zpět k bodu 3.

Pseudokód

Dále následuje zápis algoritmu v pseudokódu.

```
1  begin
2      Read (flow_records)
3      If Not Sorted(flow_records)
4          Than Sort(flow_records)
5      End If
6      initial_time <- flow_record[0]->time
7      record_part <- k In flow_records Where {
8                  k->time < (initial_time + time_window)
9              }
10     While Not Empty(record_part)
11         FilterNonTCP(record_part)
12         tls <- GetRecordWithTLS(record_part)
13         (correlated,not_correlated) <- ProcessTLS(tls)
14         http <- GetRecordWithoutTLS(record_part)
15         (correlated, not_correlated) <- ProcessHTTP(http)
16         Export(correlated)
17
18         flow_record <- flow_record / record_part
19         initial_time <- flow_record[0]->time
20         record_part <- k In flow_records Where {
21                 k->time < (initial_time + time_window)
22             }
23         AppendAndSort(record_part, not_correlated)
24     End While
```

Výpis 5.1: Zápis hlavní části algoritmu v pseudokódu

```
1  function ProcessTLS(tls)
2      parts <- tls
3      attributes <- ['JA3hash', 'SNI', 'Server Random']
4      For i In Range(0,3)
5          parts <- SplitUpByValue(parts, attributes[i])
6          For p In parts
```

```
            If CorrelateCondition(p)
               And TimeCorrelation(p)
               And CheckIPRange(p)
                correlated <- p
                parts / p
            EndIf
        EndFor
    EndFor
    Return correlated, parts
```

Výpis 5.2: Zápis funkce pro zpracování záznamů s TLS v pseudokódu

```
function ProcessHTTP(http)
    parts <- http
    attributes <- ['Request Host', 'Method mask',]
    For i In Range(0,2)
        parts <- SplitUpByValue(parts, attributes[i])
        For p In parts
            If CorrelateCondition(p)
               And TimeCorrelation(p)
               And CheckIPRange(p)
                correlated <- p
                parts / p
            EndIf
        EndFor
    EndFor
    For p In parts
        If TimeCorrelation(p)
           And DifferentDestinationIP(p)
           And DifferentDestinationPort(p)
            correlated <- p
            parts / p
        EndIf
    EndFor
    Return correlated, parts
```

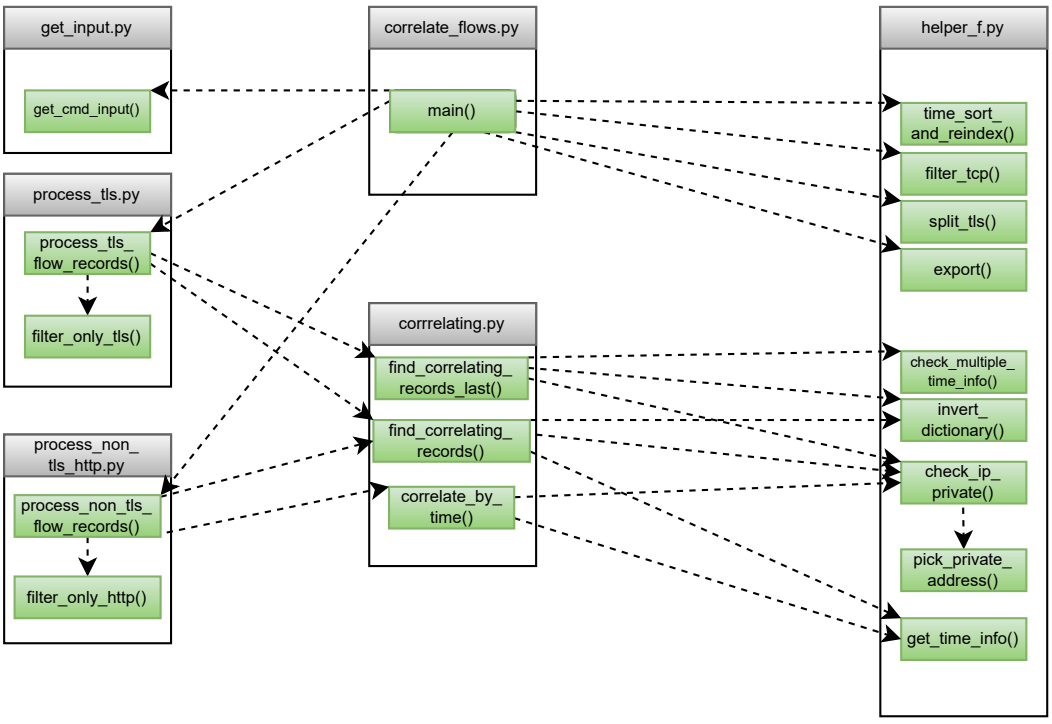
Výpis 5.3: Zápis funkce pro zpracování záznamů bez TLS v pseudokódu

Kapitola 6

Implementace

V této kapitole je popsána vytvořená aplikace. Je popsán způsob implementace, struktura programu a jeho využitelnost. Rozebrány jsou důležité a zajímavé funkce a jsou popsány i další funkce, které ve výsledné aplikaci zahrnuty nejsou, nicméně pro vytvoření řešení a výsledného programu byly nezbytné.

6.1 Vytvořená aplikace



Obrázek 6.1: Programová struktura - vizualizace

Aplikace je založena na algoritmu, prezentovaném v kapitole 5 a je možné díky ní řešit problém korelace záznamů síťových toků v praxi.

Jako implementační jazyk byl zvolen Python. Vybrán byl kvůli jeho charakteristicky jednoduchému zpracování textových řetězců, jednoduchému zpracování dvojic klíč:hodnota díky slovníkům a vestavěným knihovnám pro jednoduchou práci s formátem `csv`.

Vytvořena byla konzolová aplikace, která se spouští v příkazovém řádku. Vzhledem k tomu, že výsledky této práce by měly být využity firmou Flowmon, nebyla potřeba vytvářet žádný frontend pro aplikaci. Pokud firma řešení využije, bude aplikaci napojovat na svůj již postavený software, tudíž by se frontendová část vytvořená v této práci nikdy nevyužila.

Struktura aplikace se strukturou souborů je na obrázku 6.1, přičemž jednotlivé soubory jsou rozebrány dále.

Hlavní skript

Hlavní skript je v souboru `correlate_flows.py`. Obsahuje funkci `main()`, která je volána pouze, pokud je skript spouštěn jako hlavní. Nejdříve je ve skriptu načten uživatelský vstup, který je předáván skrz argumenty při volání v konzoli. Na výpisu z kódu 6.1 je možné vidět zprávu, která se zobrazí v případě špatného uživatelského vstupu anebo při zadání argumentu `-help`. Na výpisu je dobře vidět, jaké uživatelské vstupy program očekává a jaká je jejich výchozí hodnota.

```
1  def print_help(): # Prints help message
2      print("""
3      This is script for correlating network flow records\n
4      Command line usage:\n
5      python3 correlate_flows.py [-h / --help] [-i / --input=] [-s / --
       sni_check=] [-r / --reverse] [-p / --private_check] [-w / --process-
       window=] [-n / --proxy-chain=] [-t / --time-int=]\n
6      Where:
7      [-h / --help]            Prints this message
8      [-i / --input=]          Sets the input csv file (default output.csv)
9      [-s / --sni_check=]      Whether to check for missing SNI (possible
       for future usage, default turned on)
10     [-r / --reverse]         Sets the reverse proxy flag (presently not
       supported, by default application process forward proxy)
11     [-p / --private_check] Whether to check the private address range in
        flows positiones as originals (default false)
12     [-w / --process-window=]Length of the processed interval in sec (
       default 30)
13     [-n / --proxy-chain=]    Max number of proxy servers put in chain
       communication (default 1)
14     [-t / --time-int=]       Set time interval in seconds for time
       correlation check (default 300 ms)
15     """)
```

Výpis 6.1: Nápověda k programu

Následně je v hlavní funkci otevřen vstupní soubor. Momentálně je očekáván soubor ve formátu csv, který na prvním řádku obsahuje hlavičky. Pro načtení a práci s daty je využívána knihovna `pandas`. Kvůli některým polím v exportovaných tocích, kdy byl někdy jako hodnota přítomen řetězec, který obsahoval stejný znak, jaký byl určen jako oddělovač

v csv souboru, obsahovaly některé řádky exportovaných záznamů špatný počet hodnot. Tyto řádky jsou označeny jako 'bad lines' a v programu je nastaveno, že při jejich výskytu má být vypsáno varování a řádky mají být zahozeny.

Protože nelze předpokládat, že jsou vstupní data implicitně seřazeny dle časové posloupnosti, byla implementována funkce pro seřazení a nové označení vstupních dat. V té jsou nejdříve hodnoty v polích "START_SEC"(čas začátku toku) převedeny na datový typ `pandas.datetime`, u kterého je možná komparace. Následně je využita funkce `pandas.sort_values()` pro uspořádání záznamů dle této hodnoty. Jako řadící algoritmus je ponechán výchozí, jímž je Quicksort. Následně je datům přidělen nový index dle momentálního pořadí. Funkci je možné vidět ve výpisu z kódu **??**.

```
1 def time_sort_and_reindex(df : pd.DataFrame) -> pd.DataFrame: # Sort the
      DataFrame by time value and give the records new indexing
2     df["START_SEC"] = pd.to_datetime(df["START_SEC"])
3     df = df.sort_values(by=["START_SEC"])
4     return df.reset_index(drop=True)
```

Výpis 6.2: Funkce pro řazení dle času začátku toku

Pro implementaci zpracování dat po částech odpovídajících časovému oknu byly opět využity funkce knihovny pandas. Hodnota, která určuje velikost zpracovávaného okna, je převedena na datový typ `pandas.Timedelta`, který lze jednoduše přičíst k časové hodnotě `datetime`. Tato hodnota je tedy přičtena k hodnotě START_SEC prvního záznamu a data jsou dle ní rozděleny na dvě části - část s hodnotou času začátku toku menší než porovnávaná hodnota a část s hodnotou začátku toku větší. Obě tyto části jsou uloženy pro další použití.

Následně jsou cyklicky za využití `while` cyklu zpracovávána data. Cyklus skončí ve chvíli, kdy již nelze žádná data zpracovávat. V každém cyklu jsou nejdříve vyfiltrovány záznamy co neobsahují TCP a následně jsou data rozdělena na data obsahující TLS záznamy a ostatní data. Příklad funkce je na výpisu z kódu **??**. Tyto dvě části dat jsou dále zpracovávány zvlášť.

```
1 def split_tls(df: pd.DataFrame) -> tuple[pd.DataFrame, pd.DataFrame]: #
    Split records on those with and without TLS protocol
2     non_tls = df[df['TLS_JA3_FINGERPRINT'] == 'NIL']
3     tls = df[df['TLS_JA3_FINGERPRINT'] != 'NIL']
4     return non_tls, tls
```

Výpis 6.3: Funkce pro rozdělení záznamů toků s TLS a bez

Zpracování záznamů s TLS protokolem

Hlavní volanou funkcí pro zpracování záznamů s TLS protokolem je funkce `process_tls_flow_records()`. Tato funkce je umístěna v souboru `process_tls.py`.

```
1 def process_tls_flow_records(records : pd.DataFrame, SNI_CHECK : bool =
    False, CHECK_IP_PRIVATE : bool = True, REVERSE_PROX : bool = False,
    flow_cnt : int = 1, time_int : float = 0.3) -> tuple[list, list]
```

Výpis 6.4: Předpis funkce pro zpracování záznamů s TLS

Na začátku funkce jsou pro lepší zpracování vyfiltrovány všechny záznamy, které nebudou potřebné ve zpracování a datová struktura je převedena na slovník. Slovník v tuto chvíli má následující strukturu:

```
{ NÁZEV POLE : { 1 : HODNOTA, 2 : HODNOTA, ... }, ... }
```

Z tohoto velkého slovníku jsou vybrány jednotlivé slovníky odpovídající hodnotám, podle kterých probíhá rozdělení do menších částí. Další zpracování probíhá za využití těchto dílčích slovníků, přičemž záznamy toků jsou identifikovány dle indexů hodnot. Funkce, která je využita pro rozdělení má název `find_correlating_records()` a je umístěna v soubory `correlating.py`. Její předpis a tělo je na výpisu z kódu 6.5. Vstupem funkce je momentální seznam podmnožin (na začátku tedy seznam obsahující jednu množinu záznamů) a slovník s hodnotami dalšího zpracovávaného atributu. Výstupem funkce jsou dva seznamy. První je seznam nalezených toků, který je předávám jako seznam n-tic, kde jedna n-tice reprezentuje indexy korelovaných toků. Druhý je seznam slovníků, kde seznam reprezentuje toky, které nebyly zatím korelovány, každý slovník reprezentuje menší množinu záznamů po rozdělení a každý záznam slovníku je hodnota dalšího atributu záznamu toku, který byl do menší množiny záznamů přidělen.

```
def find_correlating_records(current_crit_dict_list : list,
    next_crit_dict : dict, records : dict, CHECK_IP_PRIVATE : bool,
    REVERSE_PROX : bool, time_int : float) -> tuple[list, list]: # Find
    correlating records
    flows = []
    rest = []
    time = records["START_SEC"]
    ipv4 = records["L3_IPV4_DST"]

    for current_crit_dict in current_crit_dict_list:
        inverted = HF.invert_dictionary(current_crit_dict)
        for _, val in inverted.items():
            if len(val) == 2 and HF.get_time_info(time[val[0]], time[val
    [1]]) < datetime.timedelta(seconds=time_int) and HF.check_ip_private
    (CHECK_IP_PRIVATE, HF.pick_private_address(val,ipv4,REVERSE_PROX)):
                flows.append(tuple(val))
            else:
                d = {}
                for k in val:
                    d[k] = next_crit_dict[k]
                rest.append(d)
    return flows, rest
```

Výpis 6.5: Funkce pro rozdělení záznamů na menší části

Za zmínění stojí funkce `invert_dictionary()`, která vymění ve slovníku jednotlivé klíče a hodnoty a v novém slovníku agreguje všechny hodnoty se stejným klíčem do jednoho záznamu. Je možné ji vidět na výpisu z kódu 6.6.

```
def invert_dictionary(dictionary : dict) -> dict: # Aggregate keys by
    the same value
    new_dict = defaultdict(list)
    for key in dictionary:
        new_dict[dictionary[key]].append(key)
    return new_dict
```

Výpis 6.6: Funkce pro invertování slovníku

Dále je potřeba zmínit funkci `check_ip_private()`, která kontroluje, zda se zadaná IP adresa nachází v privátním rozsahu. Využita je pro to knihovna `ipaddress` a příznak poskytovaný touto knihovnou `ipaddress.IPv4Address.is_private`, který vrací `bool` hodnotu v závislosti na tom, zda je daná adresa v privátním rozsahu nebo ne. Dle nastaveného příznaku se kontroluje buď první nebo poslední předaná adresa, v závislosti na tom zda je zpracováván proxy server dopředný nebo reverzní.

```
1 def check_ip_private(check : bool, ip_addr : str): # Check whether the
      IP address is in the private range
2     if not check:
3         return True
4     return ip.ip_address(ip_addr).is_private
```

Výpis 6.7: Funkce pro kontrolu provátního rozsahu

Postup rozdělení na menší části je opakován třikrát, jak bylo popsáno v kapitole 5. Pokud je uživatelsky vypnuto zpracování SNI, je pouze vytvořen nový seznam slovníků s hodnotami Server Random místo SNI. Při posledním opakování je volána funkce, která jako zbytkovou část vrátí pouze seznam indexů, ne seznam slovníků.

Na konci je vrácena dvojice seznamů. První seznam je seznam n-tic indexů záznamů toků, který reprezentuje korelované toky. Druhý je seznam indexů záznamů toků, u kterých nebylo možné nalézt korelaci.

Zpracování záznamů bez TLS protokolu

Hlavní volanou funkcí pro zpracování záznamů s TLS protokolem je funkce `process_non_tls_flow_records()`. Tato funkce je umístěna v souboru `process_non_tls_http.py`.

```
1 def process_non_tls_flow_records(records : pd.DataFrame,
      CHECK_IP_PRIVATE : bool = True,  REVERSE_PROX : bool = False,
      flow_cnt : int = 1, time_int : float = 0.3) -> tuple[list, list]
```

Výpis 6.8: Předpis funkce pro zpracování záznamů bez TLS

Průběh funkce pro zpracování záznamů bez TLS protokolu je velice podobný jako funkce pro zpracování záznamů s TLS protokolem. Opět je využita funkce `find_correlating_records()`. Za zmínku stojí funkce, která je volána pro korelaci pomocí časových údajů. Název této funkce je `correlate_by_time()` a nachází se ve stejném souboru jako ostatní funkce pro korelaci.

Ve funkci jsou postupně načítány záznamy, které jsou předané formou seznamu slovníků, kde hodnoty ve slovnících jsou časové údaje začátků toků. Vždy dva po sobě jdoucí záznamy jsou porovnány, je zkontrolována jejich cílová IP adresa a cílový port a rozdíl v jejich časech. Protože jde o poslední fázi zpracování, je předpoklad, že může být nalezeno více toků, u kterých lze určit vzájemnou korelaci. Určené záznamy jsou tedy ukládány, dokud není uložen maximální počet záznamů dle příznaku nebo dokud nelze přidat další záznam. Funkci je možné vidět ve výpisu z kódu 6.9.

Výstupem hlavní funkce jsou dva seznamy, jeden reprezentuje korelované záznamy toků a druhý záznamy, u kterých nebylo možné určit korelaci.

```
1 def correlate_by_time(times_dict_list : list,  filt_rec_dict : dict,
      CHECK_IP_PRIVATE : bool, REVERSE_PROX : bool, flow_cnt : int) ->
      tuple[list, list]: # Find correlating records by time values
```

```
    flows = []
    rest = []
    corr_tuple = None
    corr_ipv4_dest = []
    ipv4_dest = filt_rec_dict["L3_IPV4_DST"]

    for times_dict in times_dict_list:
        prev_k = None
        for key, time in times_dict.items():
            if prev_k == None:
                prev_k = key
            else:
                if corr_tuple != None and len(corr_tuple) < flow_cnt and
 ipv4_dest[prev_k] not in corr_ipv4_dest and ipv4_dest[prev_k] !=
ipv4_dest[key] and HF.get_time_info(times_dict[prev_k], time) <
datetime.timedelta(seconds=0.1) and HF.check_ip_private(
CHECK_IP_PRIVATE,HF.pick_private_address([prev_k, key], ipv4_dest,
REVERSE_PROX)):
                    if corr_tuple == None:
                        corr_tuple = (prev_k, key)
                        corr_ipv4_dest += ipv4_dest[prev_k]
                    else:
                        corr_tuple = corr_tuple + (key)
                    prev_k = key
                else:
                    if corr_tuple == None:
                        rest.append(prev_k)
                        prev_k = key
                        corr_ipv4_dest = []
                    else:
                        flows.append(corr_tuple)
                        corr_tuple = None
                        prev_k = None
        if prev_k != None:
            rest.append(prev_k)
    return flows, rest
```

Výpis 6.9: Funkce pro korelaci dle času

Export a pokračování

Výstupní seznamy z obou funkcí jsou spojeny. Seznam, reprezentující korelované toky je exportován ve formátu csv. K tomu je využita funkce `export()`, ve které probíhá vytvoření struktury dat pro export (pandas.DataFrame) a následně výpis do csv souboru (pandas.DataFrame.to_csv()).

Následně jsou načteny další záznamy, které opět odpovídají předem definovanému časovému intervalu. K těmto záznamům jsou přidány i záznamy, u kterých nebyla v předchozím zpracování určena korelace a soubor zpracovávaných dat je opět seřazen dle časové informace.

Kapitola 7

Testování a vyhodnocení

V následující kapitole je rozebrána fáze testování. Je popsána datová sada na které byla vytvořená aplikace testována, je rozebrán způsob, jakým bylo testování prováděno, jak byly připraveny vstupy a jaké k nim byly očekávány výstupy. Na konci kapitoly jsou navržené řešení a aplikace diskutovány z pohledu úspěšnosti v testech a reálné nasaditelnosti do provozu.

7.1 Testovací sada

V této sekci je popsána datová sada, která byla využita pro testovací účely. Je popsán způsob její přípravy a také myšlenky, které vytváření testovací datové sady sledovalo.

Automatizované testování bohužel v tomto případě nepřicházelo v úvahu. Pokud by měly být vytvořeny automatizované testy, musel by být pro ně vytvořen další algoritmus pro kontrolu určené korelace, čímž by vlastně vznikl problém testování vytvořeného testovacího algoritmu. Byl tedy zvolen přístup, kdy byla datová sada vytvořena ručně a kladl se spíš důraz na kvalitu a pestrost testovaných případů. Testovací sada je tedy docela malá, nicméně výpovědní hodnotu o kvalitě vytvořeného řešení by měla mít dostatečnou.

Testovací datová sada má podobu kolekce exportovaných záznamů síťových toků uložených v souborech formátu csv. K tomu byl využit skript, který není popsán v kapitole 6, ale je součástí odevzdaného řešení. Dále, pokud bylo potřeba, byly tyto záznamy různě kombinovány a spojovány. K tomu byl také využit skript dříve nezmíněný ale v řešení obsáhnutý, případě bylo takto činěno ručně.

Testovací sada vychází z datové sady popsané v kapitole 4. Testovací sada má několik částí, z nichž každá obsahuje data, která byla používána pro testování jiných vlastností aplikace. Sada obsahuje nejdříve malé části odchycené komunikace, oddělené jednotlivé síťové toky, u kterých je zřetelně jasná kauzalita a tudíž i korelace. Tyto malé části komunikace byly exportovány z dat získaných pro potřeby analýzy. Vytváření proběhlo za využití programu Wireshark, který dovoluje pomocí filtrů jednoduše zobrazit jednotlivé toky v odchycené síťové komunikaci. Takto byly zvlášť do souborů exportovány jednotlivé záznamy toků, kdy v jednom souboru byl vždy jeden záznam toku, a soubory se záznamy u kterých byla jistá korelace byly náležitě označeny. Pro účely testu byly tyto záznamy spojeny do jednoho souboru. U komunikace odchycené z více zřetězených proxy serverů, jenž byla již ze základu rozložena ve více souborech, byly i jednotlivé toky uloženy do více souborů. Aby byla jistá jejich soudržnost, byly tyto jednotlivé záznamy umístěny do jedné složky. Pro účely testu byly ale také spojeny do jednoho souboru.

Další část testovací sady tvoří již popsané velké soubory s odchycenou síťovou komunikací, které obsahují reálný síťový provoz. Tím je myšlena komunikace následující zadání URI určité domény do adresního řádku prohlížeče. Soubory tedy obsahují komunikaci se servery, které poskytují webovou aplikaci, s reklamními servery, statistickými a jinými servery. Navíc je v souborech kombinována komunikace od více klientů se stejnými i různými operačními systémy. Co se týče typů proxy serverů, jsou v testovacích datech zahrnuty komunikace využívající dopředné HTTP a SOCKS proxy servery. Pro reverzní proxy server nebylo nalezeno řešení a tudíž nebyly záznamy komunikace z něj využívány v testech. I v této části testovací datové sady jsou obsaženy záznamy z komunikace přes zřetězené proxy servery.

Vzhledem k tomu, že velká část navrženého řešení spoléhá na časovou korelaci, další část sady obsahuje záznamy, které byly upraveny, aby byly časové hodnoty začátků toků co nejblíže k sobě. Takto upraveny byly pouze již diskutované záznamy obsahující malé množství toků.

Poslední část sady se skládá ze záznamů, které obsahují smíchanou komunikaci více typů proxy serverů (HTTP a SOCKS). Sada opět vychází z dat získaných pro potřeby analýzy. Na obrázku 7.1 je možné vidět souborovou strukturu datové sady. Každá složka obsahuje soubor s exportovaným záznamem komunikace a následně soubor s výsledky korelace, složky s jednoduchými záznamy navíc obsahují i soubory s jednotlivými exportovanými toky.

```
.
├── jednoduche_zaznamy
│   ├── dopredny_squid
│   │   ├── facebook
│   │   ├── fit
│   │   ├── google
│   │   ├── microsoft
│   │   └── mit
│   ├── retezeny
│   │   ├── fit
│   │   ├── lokalni
│   │   └── mit
│   └── socks
│       ├── facebook
│       ├── fit
│       └── microsoft
├── mixovane_zaznamy
│   ├── 01
│   └── 02
├── velke_zaznamy
└── zaznamy_s_upravenym_casem
    ├── facebook
    ├── fit
    └── microsoft
```

Obrázek 7.1: Souborová struktura testovací sady

7.2 Způsob testování

V této sekci je popsán způsob, jakým byly testy prováděny. Je popsán účel, za jakým byly testy prováděny a také jaká část testovací sady byla pro tyto testy využita.

Testování správné korelace

První testy, které byly provedeny cílily na ověření správnosti algoritmu a implementace. Jak již bylo zmíněno, nebylo možné vytvořit automatizované řešení a proto musely být tyto testy prováděny s malou datovou sadou, u kterých se daly jednoduše a přehledně kontrolovat výsledky.

Byla provedena série testů, pro které byly využity jednoduché záznamy síťových toků. Aplikace byla postupně spouštěna se všemi soubory na vstupu a výsledky korelace byly ručně kontrolovány. Pro každý testovaný případ byl zpracováván soubor obsahující maximálně šest záznamů, takže bylo jednoduché správně určit, zda byly po zpracování aplikací vráceny požadované výsledky.

Tato série testů dopadla velmi dobře, úspěšnost určení korelace správně byla prakticky stoprocentní. Nicméně záznamy v této sadě obsahovaly pouze síťové toky s přítomností TLS, tudíž nebyl otestován postup související s toky obsahujícími pouze HTTP komunikaci. Navíc z důvodu, že byly jednotlivé toky exportovány zvlášť a poté teprve skládány do jednoho souboru, byly ztraceny časové souvislosti mezi záznamy, takže funkce pro kontrolu časové korelace bylo nutné obejít.

Další testy se stejným cílem byly provedeny s využitím namixované datové sady. Namixované záznamy byly využity za účelem ozkoušení chování aplikace v případě, kdy dostane podobná data, u kterých ale očividně nemá být určena korelace. Tato část sady obsahovala dva soubory, které dohromady obsahovaly celkově asi 300 toků, z nichž přibližně 230 bylo TCP. Z toho se úspěšně povedlo určit korelaci u asi 80 toků, což znamená určení korelace u přibližně 34,8 % toků. Bohužel není známá úspěšnost, protože není jisté kolik toků u kterých by korelace šla nalézt soubor dat obsahoval.

Testování robustnosti aplikace

Z pohledu testování správnosti určování jsou nám velké soubory dat k ničemu, nicméně můžou sloužit k otestování chování aplikace při zpracovávání velkého objemu dat. Aplikace pracuje na cyklickém rozdělování dat do menších struktur a tak je důležité zkusit, zda je toto řešení efektivní a zda aplikaci u většího množství dat netrvá analýza příliš dlouho. Při nasazení v reálném provozu by totiž poté byla nepoužitelná. Využity byly soubory se záznamy o síťovém toku s variantní velikostí od 0,4 do 1,7 MB. Soubory obsahovaly i nějaké neplatné, špatně vyexportované záznamy. Protože šlo o záznamy exportované z jednoho záznamu komunikace, byly tyto záznamy jako jediné exportovány v reálném čase a tudíž se správnými časovými souvislostmi.

Získané výsledky korelace byly zběžně zkontrolovány, především zda jsou na správných místech očekávaná čísla IP adres a portů. Otestován mohl být také příznak pro kontrolu privátního rozsahu. Bylo zjištěno, že pokud se privátní rozsah nekontroluje, objevuje se ve výsledcích korelace dost chyb (například korelovány toky se stejnou počáteční IP adresou u toků bez TLS protokolu), ale při jeho použití se míra chyb rapidně snížila.

Průměrná časová náročnost aplikace při zpracování záznamu o velikosti 1,3 MB byla 57 ms reálného času (měřeno na procesoru Intel Core i5, 10. generace, bylo provedeno 5 běhů programu).

Bohužel bylo zjištěno že způsobem jakým byly zpracovávány data nelze zpracovat záznamy toků ze zřetězených proxy serverů. Data byly získávány odchytáváním komunikace na rozhraní stroje, čímž pádem ale síťová komunikace mezi proxy servery byla zdvojená (nicméně aplikace správně určila korelaci všech toků, v tomto případě 4).

Testování časových korelací

Pro testování časových korelací byla vytvořena speciální datová sada. Byly v ní ručně upraveny časové hodnoty, aby byly co nejblíže u sebe anebo se překrývaly. Datová sada obsahovala tři soubory z první datové sady jednoduchých záznamů. Časová hodnota byla upravena, aby se lišila maximálně v hodnotách stovek milisekund a takto byla hledána korelace mezi záznamy.

Očividně je časová korelace nastavena dobře, protože výsledky byly stejně dobré jako v předchozích případech s těmito daty, jen nebylo nutné ručně obcházet funkce pro kontrolu času.

7.3 Zhodnocení

Celkově se řešení navržené v této práci jeví jako slibné. Vytvořená aplikace vracela dobré a správné výsledky při testech na malých datových sadách, kdy bylo přesně jasné jaké mají být výsledky. U větších datových sad s nefiltrovanou reálnou komunikací již šla míra úspěšnosti dolů, nicméně je obtížné určit kolik ze záznamů u nichž nebyla určena korelace bylo takto zpracováno špatně a u kolika korelace určit nejde. Řešení očividně funguje kvalitně alespoň pro záznamy síťových toků dopředných proxy s přítomností protokolu TLS, kterých je naštěstí momentálně v Internetu většina, což je dobrá zpráva. Bylo zjištěno že navrhovaná rozšíření jsou funkčním vylepšením řešení, jak ukázalo například použití kontroly privátních rozsahů IP adres. Nicméně zlepšení korelace záznamů bez TLS protokolu anebo navržení způsobu korelace záznamů reverzních proxy serverů zůstává na dalším řešení.

Během testování bylo odkryto několik problémů, které zůstaly během analýzy i návrhu skryty, jako například zpracování záznamů více zřetězených proxy serverů anebo problémy s časovou korelací. Na těchto poznatcích, ale také na problémech diskutovaných již dříve v této práci může být navázáno při dalším vývoji a rozšiřování programu. Aplikace jako taková je připravena na doplnění neimplementované funkcionality a je možné ji jednoduše integrovat do dalšího softwaru. Ukázalo se, že způsob přístupu k řešení a implementace není nijak limitující co se týče časové náročnosti. Pokud by měl být kladen větší důraz na rychlost zpracování záznamů, aby mohla být aplikace využita v reálném provozu, je možné ji například vytvořit v jiném programovacím jazyce, nejlépe kompilovaném (například C++).

Kapitola 8

Závěr

Cílem této práce bylo vyřešit problém korelace záznamů síťových toků proxy serverů. V průběhu řešení byla nejdříve nastudována problematika proxy serverů. Byla vysvětlena jejich historie, vývoj v čase a moderní využití. Podrobně bylo popsáno jejich fungování, různé typy, které se vyskytují a protokoly, které jsou s technologií proxy serverů blízce spjaty. Dále byl definován síťový provoz a pojmy, které jsou pro jeho popis důležité. Byl popsán proces monitorování síťového toku a exportu záznamů o něm. Blíže byly přiblíženy standardy, které se všeobecně využívají v moderním světě pro srozumitelný záznam informací o zachycených síťových tocích.

Pro řešení problému korelace bylo potřeba nejdříve vytvořit datovou sadu, dle které mělo být možné analyzovat a pochopit chování proxy serverů v praxi. Dále také měla datová sada sloužit pro následné testování navrženého řešení. V rámci práce bylo vytvořeno a popsáno prostředí, ve kterém probíhalo odchytávání síťové komunikace, ze které byla datová sada následně vytvořena. Byly nastaveny a využity různé druhy proxy serverů a byla odchytávána jak uměle vytvořená komunikace se serverem v místní síti, tak komunikace se servery v Internetu.

Dále byly diskutovány způsoby korelace síťových toků. Byly prezentovány již existující metody a předchozí práce, které se zabývaly podobným problémem. Na to navázala vlastní analýza datové sady a fungování proxy serverů. Z té vzešel návrh řešení, který byl popsán, algoritmizován a implementován. Implementované řešení bylo následně testováno pomocí dříve vytvořené datové sady.

Vytvořené řešení může být označeno jako efektivní pouze pro část síťového provozu, který přes proxy servery může procházet, ovšem lze říct, že pro tu největší. Algoritmus pro dopředné proxy servery celkem jistě určí korelace a má minimální chybovost co se týče špatného určování korelací. Pro reverzní proxy servery algoritmus nefunguje, protože mají jiné chování protokolu TLS. Pro ostatní komunikaci je fungování dostatečné, přičemž lze ještě vylepšit správným nastavením parametrů.

V porovnání s jinými řešeními podobných problémů korelací síťových toků (Coufal [8]) procházela tato práce podobným postupem. Bylo zjištěno, že žádná z vytvořených metod nemůže být přímo spolehlivě využita, byla vytvořena vlastní metoda, která se soustředila na specifické atributy záznamů síťových toků a bylo vytvořeno řešení fungující dobře pro velký okruh komunikací.

Další pokračování práce je rozhodně možné v další analýze provozu proxy serverů, v bližším prozkoumání fungování protokolů například na reálném provozu a v případném rozšíření řešení o další protokoly. Vytvořená aplikace je lehce rozšiřitelná a proto je také možné ji dále vyvíjet a zpřesňovat metody pro určování korelací.

Literatura

[1] *Flowmon Sond a exportér NetFlow a Ipfix*. [cit. 5.5.2022]. Dostupné z: https://www.flowmon.com/cs/products/appliances/probe.

[2] *NetFlow Version 9 Flow-Record Format*. Internet-Draft. Cisco, květen 2011 [cit. 7.1.2022]. Dostupné z: https://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.html.

[3] 3RD, D. E. E. *Transport Layer Security (TLS) Extensions: Extension Definitions* [RFC 6066]. RFC Editor, leden 2011. DOI: 10.17487/RFC6066. Dostupné z: https://www.rfc-editor.org/info/rfc6066.

[4] AITKEN, P., CLAISE, B. a TRAMMELL, B. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information* [RFC 7011]. RFC Editor, září 2013. DOI: 10.17487/RFC7011. Dostupné z: https://rfc-editor.org/rfc/rfc7011.txt.

[5] ALTHOUSE, J. *TLS Fingerprinting with JA3 and JA3S* [online]. Salesforce Engineering [cit. 9.5.2022]. Dostupné z: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967.

[6] CHAPMAN, E. D. Z. . S. C. D. B. *Firewalls building internet*. 1. vyd. Cambridge : O´Reilly, 2000. ISBN 1-56592-871-7.

[7] CLAISE, B. *Cisco Systems NetFlow Services Export Version 9* [RFC 3954]. RFC Editor, říjen 2004. DOI: 10.17487/RFC3954. Dostupné z: https://rfc-editor.org/rfc/rfc3954.txt.

[8] COUFAL, Z. *Korelace dat na vstupu a výstupu sítě Tor*. Brno, 2014. Diplomová práce. Vysoké učení technické, Fakulta Informačních Technologií.

[9] DANEZIS, G. The Traffic Analysis of Continuous-Time Mixes. In:. Květen 2004, sv. 3424, s. 35–50. DOI: 10.1007/11423409_3. ISBN 978-3-540-26203-9.

[10] FIELDING, R. T. a RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing* [RFC 7230]. RFC Editor, červen 2014. DOI: 10.17487/RFC7230. Dostupné z: https://rfc-editor.org/rfc/rfc7230.txt.

[11] HOFSTEDE, R., ČELEDA, P., TRAMMELL, B., DRAGO, I., SADRE, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS* [online]. 2014, sv. 16, č. 4. DOI: http://dx.doi.org/10.1109/COMST.2014.2321898. ISSN 1553-877X. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6814316.

[12] Lee, C.-Y., Kim, H.-K., Ko, K.-H., Kim, J.-W. a Jeong, H. A VoiP Traffic Monitoring System based on NetFlow v9. *International Journal of Advanced Science and Technology*. Duben 2009, sv. 4.

[13] Leech, M. D. *SOCKS Protocol Version 5* [RFC 1928]. RFC Editor, březen 1996. DOI: 10.17487/RFC1928. Dostupné z: https://rfc-editor.org/rfc/rfc1928.txt.

[14] Matoušek, P., Burgetová, I., Ryšavý, O. a Victor, M. On Reliability of JA3 Hashes for Fingerprinting Mobile Applications. In: Goel, S., Gladyshev, P., Johnson, D., Pourzandi, M. a Majumdar, S., ed. *Digital Forensics and Cyber Crime*. Cham: Springer International Publishing, 2021, s. 1–22. ISBN 978-3-030-68734-2.

[15] Matthew Strebe, C. P. *Firewally a proxy-servery : praktický průvodce*. 1. vyd. Brno : Computer Press, 2003. ISBN 8072269836.

[16] Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J. et al. *Hypertext Transfer Protocol – HTTP/1.1* [RFC 2616]. RFC Editor, červen 1999. DOI: 10.17487/RFC2616. Dostupné z: https://rfc-editor.org/rfc/rfc2616.txt.

[17] Northcutt, S. *Bezpečnost sítí : velká kniha*. 1. vyd. Brno : CP Books, 2005. ISBN 80-251-0697-7.

[18] Rescorla, E. a Dierks, T. *The Transport Layer Security (TLS) Protocol Version 1.2* [RFC 5246]. RFC Editor, srpen 2008. DOI: 10.17487/RFC5246. Dostupné z: https://www.rfc-editor.org/info/rfc5246.

[19] Rohmad, M., Farok, A., Manaf, M. a Ab Manan, J.-L. Enhanced Netflow version 9 (e-Netflow v9) for network mediation: Structure, experiment and analysis. Září 2008, s. 1 – 6. DOI: 10.1109/ITSIM.2008.4632080.

[20] Serjantov, A. a Sewell, P. Passive Attack Analysis for Connection-Based Anonymity Systems. In:. Březen 2004, sv. 4, s. 116–131. DOI: 10.1007/978-3-540-39650-5_7. ISBN 978-3-540-20300-1.

[21] Trammell, B. a Boschi, E. An introduction to IP flow information export (IPFIX). *IEEE Communications Magazine*. 2011, sv. 49, č. 4. DOI: 10.1109/MCOM.2011.5741152.

[22] Wang, E., Ossipov, A. a DuToit, R. *TLS Proxy Best Practice*. Internet-Draft draft-wang-opsec-tls-proxy-bp-00. Internet Engineering Task Force, červen 2020 [cit. 3.1.2022]. Work in Progress. Dostupné z: https://datatracker.ietf.org/doc/html/draft-wang-opsec-tls-proxy-bp-00.

[23] Wang, X. a Wu, S. Inter-Packet Delay Based Correlation for Tracing Encrypted Connections through Stepping Stones. In:. říjen 2002, sv. 2502, s. 244–263. DOI: 10.1007/3-540-45853-0_15. ISBN 978-3-540-44345-2.

[24] Zhu, Y., Fu, X., Graham, B., Bettati, R. a Zhao, W. Correlation-Based Traffic Analysis Attacks on Anonymity Networks. *IEEE Trans. Parallel Distrib. Syst.* Červenec 2010, sv. 21, s. 954–967. DOI: 10.1109/TPDS.2009.146.

Příloha A

Definice typů polí v NetFlow v9

Field Type	Value	Length (bytes)	Description
IN_BYTES	1	N (default is 4)	Incoming counter with length N x 8 bits for number of bytes associated with an IP Flow.
IN_PKTS	2	N (default is 4)	Incoming counter with length N x 8 bits for the number of packets associated with an IP Flow
FLOWS	3	N	Number of flows that were aggregated; default for N is 4
PROTOCOL	4	1	IP protocol byte
SRC_TOS	5	1	Type of Service byte setting when entering incoming interface
TCP_FLAGS	6	1	Cumulative of all the TCP flags seen for this flow
L4_SRC_PORT	7	2	TCP/UDP source port number i.e.: FTP, Telnet, or equivalent
IPV4_SRC_ADDR	8	4	IPv4 source address
SRC_MASK	9	1	The number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation
INPUT_SNMP	10	N	Input interface index; default for N is 2 but higher values could be used
L4_DST_PORT	11	2	TCP/UDP destination port number i.e.: FTP, Telnet, or equivalent

		is 4)	number of bytes associated with an IP Flow
OUT_PKTS	24	N (default is 4)	Outgoing counter with length N x 8 bits for the number of packets associated with an IP Flow.
MIN_PKT_LNGTH	25	2	Minimum IP packet length on incoming packets of the flow
MAX_PKT_LNGTH	26	2	Maximum IP packet length on incoming packets of the flow
IPV6_SRC_ADDR	27	16	IPv6 Source Address
IPV6_DST_ADDR	28	16	IPv6 Destination Address
IPV6_SRC_MASK	29	1	Length of the IPv6 source mask in contiguous bits
IPV6_DST_MASK	30	1	Length of the IPv6 destination mask in contiguous bits
IPV6_FLOW_LABEL	31	3	IPv6 flow label as per RFC 2460 definition
ICMP_TYPE	32	2	Internet Control Message Protocol (ICMP) packet type; reported as ((ICMP Type*256) + ICMP code)
MUL_IGMP_TYPE	33	1	Internet Group Management Protocol (IGMP) packet type
SAMPLING_INTERVAL	34	4	When using sampled NetFlow, the rate at which packets are sampled i.e.: a value of 100 indicates that one of every 100 packets is sampled
SAMPLING_ALGORIT	35	1	The type of algorithm

HM			used for sampled NetFlow: 0x01 Deterministic Sampling ,0x02 Random Sampling
FLOW_ACTIVE_TIMEOUT	36	2	Timeout value (in seconds) for active flow entries in the NetFlow cache
FLOW_INACTIVE_TIMEOUT	37	2	Timeout value (in seconds) for inactive flow entries in the NetFlow cache
ENGINE_TYPE	38	1	Type of flow switching engine: RP = 0, VIP/Linecard = 1
ENGINE_ID	39	1	ID number of the flow switching engine
TOTAL_BYTES_EXP	40	N (default is 4)	Counter with length N x 8 bits for bytes for the number of bytes exported by the Observation Domain
TOTAL_PKTS_EXP	41	N (default is 4)	Counter with length N x 8 bits for bytes for the number of packets exported by the Observation Domain
TOTAL_FLOWS_EXP	42	N (default is 4)	Counter with length N x 8 bits for bytes for the number of flows exported by the Observation Domain
Vendor Proprietary	43		
IPV4_SRC_PREFIX	44	4	IPv4 source address prefix (specific for Catalyst architecture)
IPV4_DST_PREFIX	45	4	IPv4 destination address prefix (specific for Catalyst architecture)

MPLS_TOP_LABEL_TYPE	46	1	MPLS Top Label Type: 0x00 UNKNOWN 0x01 TE-MIDPT 0x02 ATOM 0x03 VPN 0x04 BGP 0x05 LDP
MPLS_TOP_LABEL_IP_ADDR	47	4	Forwarding Equivalent Class corresponding to the MPLS Top Label
FLOW_SAMPLER_ID	48	1	Identifier shown in "show flow-sampler"
FLOW_SAMPLER_MODE	49	1	The type of algorithm used for sampling data: 0x02 random sampling. Use in connection with FLOW_SAMPLER_MODE
FLOW_SAMPLER_RANDOM_INTERVAL	50	4	Packet interval at which to sample. Use in connection with FLOW_SAMPLER_MODE
Vendor Proprietary	51		
MIN_TTL	52	1	Minimum TTL on incoming packets of the flow
MAX_TTL	53	1	Maximum TTL on incoming packets of the flow
IPV4_IDENT	54	2	The IP v4 identification field
DST_TOS	55	1	Type of Service byte setting when exiting outgoing interface
IN_SRC_MAC	56	6	Incoming source MAC address
OUT_DST_MAC	57	6	Outgoing destination MAC address
SRC_VLAN	58	2	Virtual LAN identifier

			associated with ingress interface
DST_VLAN	59	2	Virtual LAN identifier associated with egress interface
IP_PROTOCOL_VERSION	60	1	Internet Protocol Version Set to 4 for IPv4, set to 6 for IPv6. If not present in the template, then version 4 is assumed.
DIRECTION	61	1	Flow direction: 0 - ingress flow, 1 - egress flow
IPV6_NEXT_HOP	62	16	IPv6 address of the next-hop router
BPG_IPV6_NEXT_HOP	63	16	Next-hop router in the BGP domain
IPV6_OPTION_HEADERS	64	4	Bit-encoded field identifying IPv6 option headers found in the flow
Vendor Proprietary	65		
Vendor Proprietary	66		
Vendor Proprietary	67		
Vendor Proprietary	68		
Vendor Proprietary	69		
MPLS_LABEL_1	70	3	MPLS label at position 1 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_2	71	3	MPLS label at position 2 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.

MPLS_LABEL_3	72	3	MPLS label at position 3 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_4	73	3	MPLS label at position 4 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_5	74	3	MPLS label at position 5 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_6	75	3	MPLS label at position 6 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_7	76	3	MPLS label at position 7 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_8	77	3	MPLS label at position 8 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_9	78	3	MPLS label at position 9 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_10	79	3	MPLS label at position 10 in the stack. This

			comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
IN_DST_MAC	80	6	Incoming destination MAC address
OUT_SRC_MAC	81	6	Outgoing source MAC address
IF_NAME	82	N (default specified in template)	Shortened interface name i.e.: "FE1/0"
IF_DESC	83	N (default specified in template)	Full interface name i.e.: '"FastEthernet 1/0"
SAMPLER_NAME	84	N (default specified in template)	Name of the flow sampler
IN_ PERMANENT _BYTES	85	N (default is 4)	Running byte counter for a permanent flow
IN_ PERMANENT _PKTS	86	N (default is 4)	Running packet counter for a permanent flow
* Vendor Proprietary*	87		
FRAGMENT_OFFSET	88	2	The fragment-offset value from fragmented IP packets
FORWARDING STATUS	89	1	Forwarding status is encoded on 1 byte with the 2 left bits giving the status and the 6

			remaining bits giving the reason code.  Status is either unknown (00), Forwarded (10), Dropped (10) or Consumed (11). Below is the list of forwarding status values with their means. Unknown • 0 Forwarded • Unknown 64 • Forwarded Fragmented 65 • Forwarded not Fragmented 66 Dropped • Unknown 128, • Drop ACL Deny 129, • Drop ACL drop 130, • Drop Unroutable 131, • Drop Adjacency 132, • Drop Fragmentation & DF set 133, • Drop Bad header checksum 134, • Drop Bad total Length 135, • Drop Bad Header Length 136, • Drop bad TTL 137, • Drop Policer 138, • Drop WRED 139, • Drop RPF 140, • Drop For us 141,

			• Drop Bad output interface 142, • Drop Hardware 143, Consumed • Unknown 192, • Terminate Punt Adjacency 193, • Terminate Incomplete Adjacency 194, • Terminate For us 195
MPLS PAL RD	90	8 (array)	MPLS PAL Route Distinguisher.
MPLS PREFIX LEN	91	1	Number of consecutive bits in the MPLS prefix length.
SRC TRAFFIC INDEX	92	4	BGP Policy Accounting Source Traffic Index
DST TRAFFIC INDEX	93	4	BGP Policy Accounting Destination Traffic Index
APPLICATION DESCRIPTION	94	N	Application description.
APPLICATION TAG	95	1+n	8 bits of engine ID, followed by n bits of classification.
APPLICATION NAME	96	N	Name associated with a classification.
postipDiffServCodePoint	98	1	The value of a Differentiated Services Code Point (DSCP) encoded in the Differentiated Services Field, after modification.
replication factor	99	4	Multicast replication factor.
DEPRECATED	100	N	DEPRECATED
layer2packetSectionOffset	102		Layer 2 packet section offset. Potentially a

			generic offset.
layer2packetSectionSize	103		Layer 2 packet section size. Potentially a generic size.
layer2packetSectionData	104		Layer 2 packet section data.
	105 to 127		**Reserved for future use by cisco**