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.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	29. července 2022
Datum schválení:	29. června 2022

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 elabortes the problem of correlation of the network flow records. It tries to find solution, which would allow to automatically correlate flows from 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
27. července 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

1 Úvod **2**

2 Použití proxy serverů **4**
2.1 Proxying 4
2.2 Fungování proxy serverů 5
2.3 Rozdělení proxy serverů 7

3 Síťový provoz a jeho monitorování **12**
3.1 Proces sledování síťového provozu 13
3.2 NetFlow v9 15
3.3 IP Flow Information Export (IPFIX) 18
3.4 Implementace v systémech Flowmon 22

4 Datová sada **23**
4.1 Prostředí pro vytvoření datové sady 23
4.2 Popis datové sady 26
4.3 Vytvoření záznamů o síťovém toku 29

5 Navržené řešení a algoritmus **32**
5.1 Korelace v této práci 32
5.2 Současné korelační metody 32
5.3 Analýza datové sady 34
5.4 Navrhovaný způsob řešení problému korelace 38
5.5 Vytvořený algoritmus 42

6 Implementace **45**
6.1 Vytvořená aplikace 45

7 Testování a vyhodnocení **52**
7.1 Testovací sada 52
7.2 Způsob testování 53
7.3 Zhodnocení 57

8 Závěr **58**

Literatura **60**

A Definice typů polí v NetFlow v9 **63**

Kapitola 1

Úvod

Proxy servery jsou jednou z možností, jak skrýt a chránit komunikaci ve velkých sítích. 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. Také mohou být využity jako mezistupeň pro rozložení zátěže na koncové servery, které nezvládnou obsloužit větší množství konkurenčních připojení. V moderní síťové struktuře jsou zařízení fungující jako proxy často uzlem, přes který prochází síťová komunikace. Na trhu existuje velké množství řešení, které nabízí software fungující jako proxy. Buďto funguje jen jako jednoduchý, případně s určitým řízením přístupu anebo kombinovaný s firewallem. Proxy servery mohou pro různé účely využívat domácnosti, firmy, vzdělávací instituce a také poskytovatelé internetových služeb a aplikací. Je možné je využít jako jediný bod přístupu k internetu z vnitřní sítě. Může se však také stát, že jsou proxy využity pro ilegální činnost a provádění útoků z anonymity, kterou proxy servery poskytují.

Při monitorování síťového provozu bývají k dispozici pouze informace obsažené v komunikaci mezi jednotlivými stranami, přičemž zpracovávány bývají záznamy získané z různých míst sítě. V případě směrování komunikace skrz proxy, je komunikace, která logicky probíhá mezi dvěma koncovými body v síti, rozdělena na více menších částí. Mohou se tedy vyskytnout důvody, kvůli kterým by bylo užitečné umět k sobě spojit tyto části a tím získat znalost o celé komunikaci mezi koncovými body v síti. To však může představovat značně komplikovaný problém, protože komunikace u proxy končí a začíná, čímž je z jedné komunikace generováno více záznamů, které nemají přímo viditelnou logickou návaznost.

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. Zkoumán a řešen je tedy problém nalezení kombinace záznamů toků, které je možné uspořádat a tak nalézt spojitou logickou komunikaci mezi dvěma body v síti. Řešeny jsou i případy, kdy komunikace prochází přes více proxy serverů.

V této práci jsou záznamy síťové komunikace vytvořeny z odchycené síťové komunikace a exportovány dle standardu IPFIX. Ty jsou dále zkoumány s cílem zjistit, zda je možné pouze na základě informací v nich obsažených k sobě přiřadit jednotlivé záznamy toků a poté z nich vytvořit záznam odpovídající celé komunikaci mezi koncovými stanicemi. Řešení problému korelace je hledáno za účelem jeho následného použití v praxi. V nejlepším případě by tedy řešení mělo vyžadovat co nejmenší uživatelskou intervenci. 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.

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 [10], 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

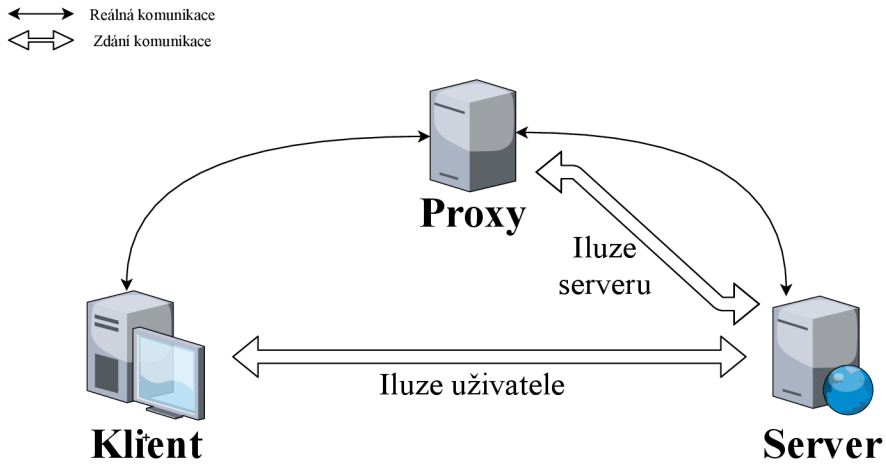
Proxying je způsob, jak ve velké síti, například síti velké korporace, umožnit přístup pouze k jednomu nebo malé skupině strojů z podsítě, přičemž je zachováno zdání, že je umožněn přístup ke všem strojům dané podsítě [8]. *Proxy server* je specializovaná aplikace, která zajišťuje komunikaci mezi vnitřní sítí a Internetem [21].

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 [19].

Proxy server přistupuje jménem klienta či uživatele k určité síťové službě a fakticky tím zakrývá přímé spojení mezi oběma partnery [21]. Ze své podstaty, kdy jsou proxy servery využívány jako firewally, jsou proxy servery schopné monitorovat a filtrovat komunikaci, která přes ně prochází. Protože proxy servery vznikly především jako nástroj používaný pro komunikaci v Internetu, je proxy servery nejčastěji zpracovávaný protokol HTTP [19]. Vznikly ovšem i další protokoly, které poskytují zapouzdření a větší zabezpečení při komunikaci skrz proxy server (například protokol SOCKS) [17].

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 [8]. Základní schéma jednoduchého proxy serveru je možné vidět na obrázku 2.1. Jednotlivé typy proxy serverů jsou více rozebrány v dalším textu.



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. Aby byla komunikace směřována skrz proxy server, lze na straně klientské stanice provést nastavení na několika úrovních.

1. Na úrovní *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.

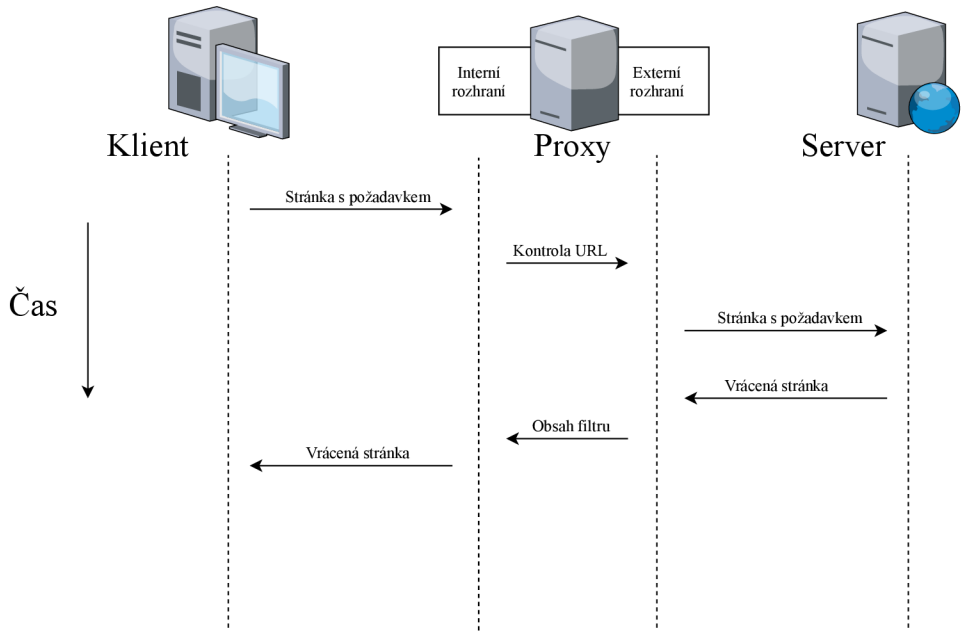
2. Na úrovní *operačního systému*, jehož nastavení umožňuje využívat proxy server. Operační systém je upraven, aby na základě kontroly IP adres rozhodl, zda nemá být spojení uskutečněno přes proxy server.

3. Na úrovni *uživatele* používajícího systém. Uživatel využívá software bez funkce využití proxy serveru. Sám zajišťuje připojení k proxy serveru a určení cílového serveru pro komunikaci.

4. Na úrovni *routeru*, který odchytává procházející provoz, který odesílá na proxy server, anebo přímo funguje jako proxy server. Přímo na klientské stanici přitom nedochází k žádným úpravám [8].

Příklad komunikace s využitím jednoduchého proxy serveru je následující. Nejdříve klient naváže potřebné spojení s proxy serverem a vyžádá si určitou internetovou službu (za využití aplikačních protokolů HTTP, FTP, ...). 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 [8, 21]. Příklad jednoduché komunikace je na obrázku 2.2.



Obrázek 2.2: Fungování proxy serveru, stavový diagram

Proxy server většinou funguje na takzvaném opevněném hostiteli se 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, směrování a přeposílání IP paketů je zakázané, což umožňuje softwaru běžícím na tomto hostiteli úplnou kontrolu nad procházejícím provozem [21].

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 zařízení v sítích na obou stranách proxy serveru nemusí mít o sobě vzájemně žádné znalosti. 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 [19].

Další funkcí, důležitou při popisu proxy serveru, je omezení přenosového pásma při komunikaci. 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 [19].

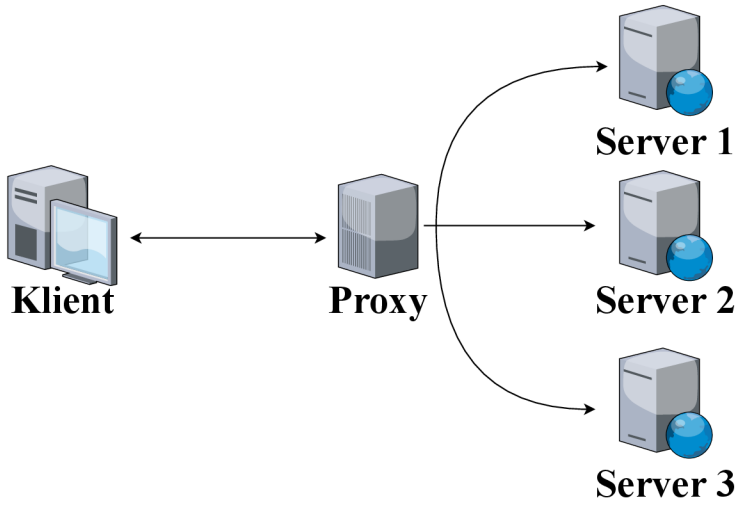
2.3 Rozdělení proxy serverů

Proxy servery lze rozdělit na mnoho typů dle různých charakteristik. 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 [21].

Oproti tomu *reverzní proxy* funguje na trochu jiném principu. 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. Kromě již dříve zmíněného ukládáni do vyrovnávací paměti, může být proxy server také využit pro směrování klientských 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. Také je možnost před server umístit více proxy serverů, mezi kterými může být zátěž dále vyrovnávána [21, 19]. 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 [21].

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 předávání HTTP komunikace 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. V principu je možné servery nastavit pro obsluhu skoro jakéhokoliv protokolu. U některých 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 [21, 8].

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 tento protokol. Server zprávy většinou pouze přeposílá, ovšem může je i upravovat, anebo úplně měnit [20, 12].

V případě, že je při HTTP komunikaci využito proxy, můžou být do hlavičky přidány informace o klientovi, které jsou normálně při použitá proxy ztraceny. Pro toto je určena standardizovaná hlavička `Forwarded`. Ta může obsahovat čtyři direktivy:

1. `By` - identifikátor rozhraní proxy serveru, na kterém byla přijata zpráva (napřklad IP adresa).
2. `For` - identifikátor klienta, který zaslal požadavek, případně všech dalších předchozích proxy serverů. Může obsahovat stejné informace jako direktiva By.
3. `Host` - obsah hlavičky Host přijaté proxy serverem v rámci zprávy.
4. `Proto` - indikuje využitý protokol. Typicky http nebo https.

Stejné informace mohou obsahovat i hlavičky `X-Forwarded-For`, `X-Forwarded-Host` a `X-Forwarded-Proto`, které ale nejsou standardizované a nemusí být vždy podporovány. Všechny tyto hlavičky jsou většinou využívány pro účely ladění funkcionality proxy, protože v reálném provozu je jejich použitím ztracena výhoda anonymizace klienta.

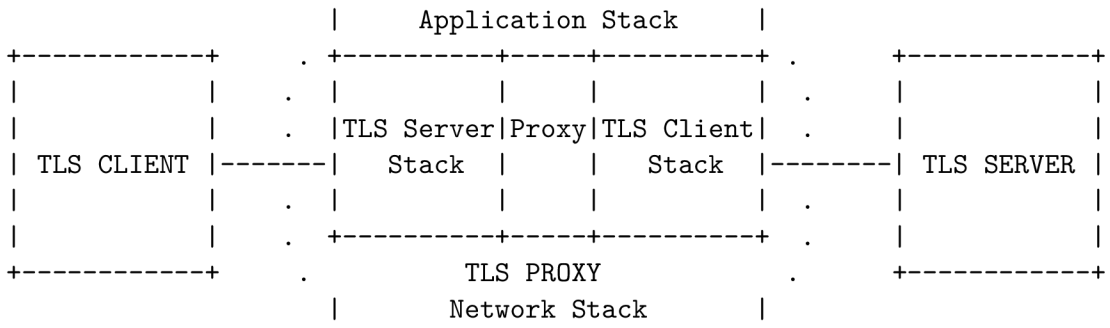
Proxy server může připojit hlavičku `Via`, pomocí které přidává do zprávy informaci o sobě samotném [3]. Příklad hlavičky je možné vidět na obrázku 2.4.

```
Hypertext Transfer Protocol
    CONNECT www.facebook.com:443 HTTP/1.1\r\n
    User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) \
        Gecko/20100101 Firefox/75.0\r\n
    Proxy-Connection: keep-alive\r\n
    Connection: keep-alive\r\n
    Forwarded: by=10.0.0.20;for=157.240.30.35;proto=https
    Host: www.facebook.com:443\r\n
```

Obrázek 2.4: Příklad HTTP hlavičky při využití proxy

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 [26]. Schéma můžeme vidět na obrázku 2.5.

Pro zahájení šifrované HTTP komunikace skrze proxy server je definována specifická metoda. Tato metoda má název `CONNECT` a je používána stejným způsobem jako například metoda `GET`. Dle definice metoda žádá vytvoření bezpečného tunelu od příjemce zprávy k finálnímu cíli a následné přeposílání zpráv v obou směrech [13]. Klient tedy zahájí komunikaci zasláním zprávy s touto metodou proxy serveru. V reakci na ni proxy server naváže TCP spojení s cílovým serverem. Jakmile je spojení úspěšně navázáno, zasílá zpět klientovi HTTP potvrzení, po kterém může následovat ustanovení TLS spojení. Nicméně je nutné podotknout, že ne všechny proxy servery typ zprávy CONNECT podporují, případně jej limitují pouze na port 443 [3].

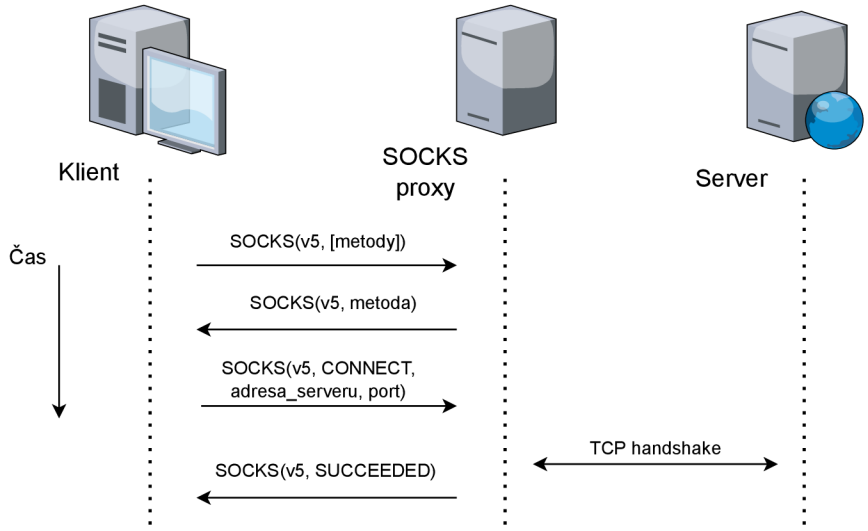


Obrázek 2.5: Schéma TLS proxy serveru [26]

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. Pro upravení je nutné všechna volání standardních síťových funkcí nahradit voláním SOCKS verzí těchto funkcí. 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 navíc od svého předchůdce podporu pro protokol UDP a ICMP a možnost autentizace [21, 8].



Obrázek 2.6: Vizualizace zasílání zpráv SOCKS

Procedura připojení klientů založených na TCP je následující. Klient se připojí k SOCKS serveru, přičemž obvyklý port přiřazený službě je 1080. Klient zašle zprávu s verzí protokolu a výběrem metod pro autentizaci. Server odpoví s jednou vybranou autentizační metodou, případně v odpovědi stanoví, že žádná z metod inzerovaných klientem není akceptovatelná. Následuje vyjednávání o specifické metodě. Ve chvíli kdy je dokončeno, zasílá klient požadavek. Ten má následující strukturu.

- Verze.
- Příkaz - `CONNECT`, `BIND` (např. pro FTP), `UDP ASSOCIATE` (specifické pro UDP)
- Type adresy - IPv4, IPv6, doménové jméno.
- Cílová adresa.
- Cílový port.

Odpověď SOCKS serveru má odpovídající strukturu, pouze pole s kódem příkazu nahradí kódem odpovědi [17]. Na obrázku 2.6 je možné vidět vizualizaci výše popsaného mechanismu.

Dle [8] je stanoveno, že 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. Nikde nebyla nalezena informace, která by toto tvrzení vyvracela, takže lze předpokládat, že v protokolu podpora pro logování opravdu není a je nutné ji přidat při implementaci [21, 8].

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 jsou využívány pro sledování existujícího provozu, tedy pro sledování provozu vytvářeného 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 [15].

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 se vyskytnou v určitém bodě 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:

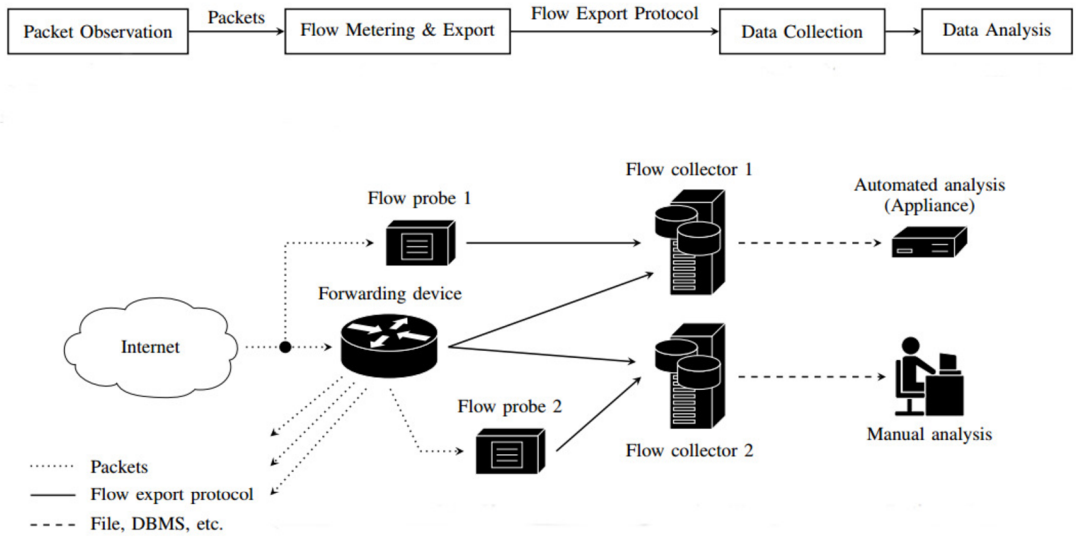
1. jedna nebo více položek hlaviček paketů, položek hlaviček transportních protokolů nebo položek hlaviček aplikačních protokolů,
2. jedna nebo více charakteristik samotného paketu,
3. jedno nebo více polí odvozených ze zpracování paketu.

Paket náleží toku pouze, pokud splňuje všechny definované vlastnosti. Položky, dle kterých bývají pakety obvykle sdružovány do toků, jsou IP adresy, čísla portů, specifické hodnoty aplikačních protokolů anebo specifické hodnoty směrovacích protokolů. Také můžou být využity informace jako next-hop IP adresa paketu, případně rozhraní, kterým paket opouští zařízení [6].

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 rekurzivně 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 [6].

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

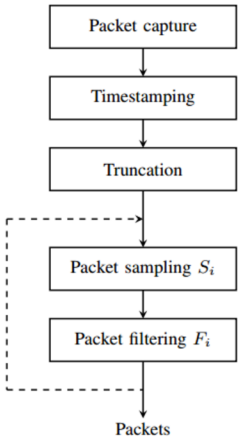


Obrázek 3.1: Proces sledování a exportu síťového toku[15]

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 kontrolou hodnoty `checksum`. Druhý krokem je označení časovou známkou (timestamp). Tento krok je velmi důležitý pro následující zpracovávající funkce a analytické aplikace. Podle časové známky jsou v pozdějších fázích 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

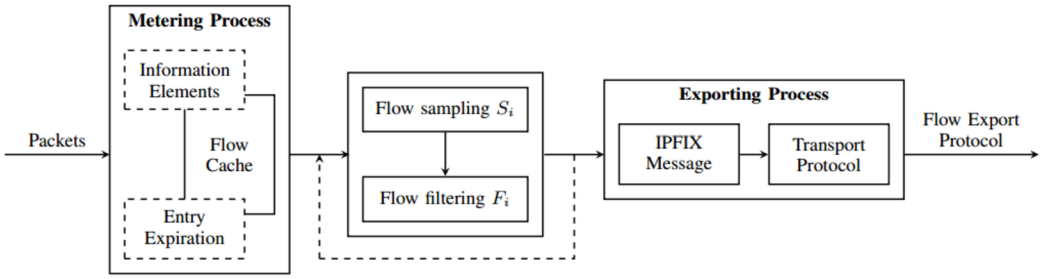
zajímavé. Všechny kroky kromě zachycení paketu a označení časovou značkou jsou nepovinné. Celý proces této fáze je znázorněn na obrázku 3.2 [15].



Obrázek 3.2: Proces zachytávání síťových paketů

- **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 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. Většinou se jako klíčová hodnota využívá zdrojová a cílová IP adresa anebo čísla portů. Informace jsou ve flow cache drženy do chvíle, kdy je tok považován za ukončený. Tok je považován za ukončený buď na základě pravidel o překročení časového limitu (*timeout*) anebo ve chvíli kdy se objeví specifická událost. Timeout může být například aktivní, kdy je záznam exportován po předem určeném časovém intervalu, anebo nečinný (idle), kdy se po nějaký časový interval již neobjevil žádný paket náležící do toku. Dalšími důvody jsou například přirozené skončení toku, kdy je pozorován TCP paket s FIN anebo RST příznakem, nouzová expirace, kdy jsou toky uměle označeny za skončené z důvodu nedostatku paměti, anebo vymazání paměti (cache flush). Následně jsou záznamy o tocích vzorkovány a filtrovány. Rozdíl oproti stejnému kroku v předchozím fázi je, že v tuto chvíli se již pracuje se záznamy o tocích, což znamená že buď jsou zaznamenány všechny pakety náležící do toku, anebo žádný. Záznamy jsou 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 [6, 15].

- **Sběr a analýza dat**. Sběr dat je prováděn kolektory, které přijímají, ukládají a provádí předzpracování dat od jednoho nebo více exportérů v síti. Typickými úkony



Obrázek 3.3: Proces měření a exportu síťového toku [15]

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. Pro tuto práci je důležitá oblast Analýzy toku. Typicky jsou v do této této oblasti zahrnovány následující funkce: procházení a filtrování dat o tocích, přehled statistik, určování zařízení, které generují největší provoz, hlášení, například o množství provedené komunikace mezi zařízeními a varování, například o využití nepovolených protokolů nebo v případě nezvyklého počtu spojení generovaného zařízeními [15].

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 datových položek s korespondujícím popisem struktury a sémantiky. To poskytuje flexibilní a rozšířitelný přístup ke sledování síťových toků. Je totiž možné přidat nové datová položky, aniž by bylo nutné měnit strukturu exportovaného formátu. Tyto nové položky obsahují vlastní popis, takže i kolektory, které jim nerozumí, mohou záznam o toku interpretovat [9, 23, 16].

Záznam NetFlow v9 je složen 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 položek, které jsou obsaženy v datových FlowSets. Tyto data se mohou objevit ve stejném paketu, anebo v dalších příchozích paketech [4, 9]. Ilustraci poskytuje obrázek 3.4.

Obrázek 3.4: NetFlow v9 exportní paket [4]

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 [4, 9].

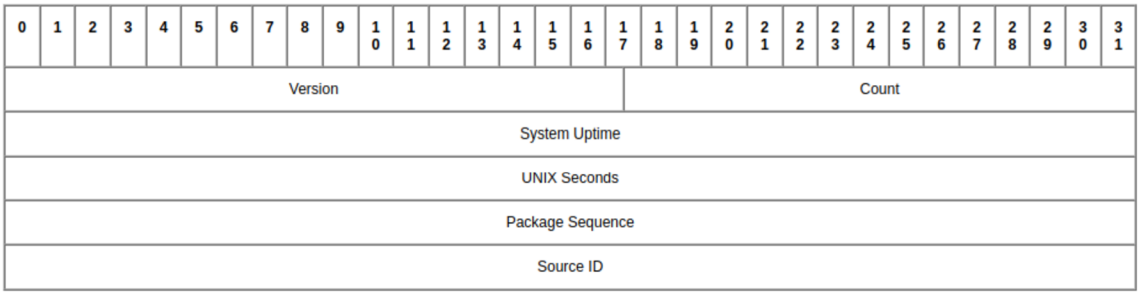
Možné kombinace šablon a dat, které se mohou objevit v exportním paketu, jsou vypsány dále.

- 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. Nicméně platnost šablon po určité době vyprší a je nutné je opět zaslat. Pakety obsahující pouze šablonové FlowSets nejsou obvyklé, ale mohou se vyskytnout [4, 9].

Formát hlavičky

Formát hlavičky je stejný jako v předchozích verzích NetFlow [4]. Hlavička je ilustrována na obrázku 3.5.



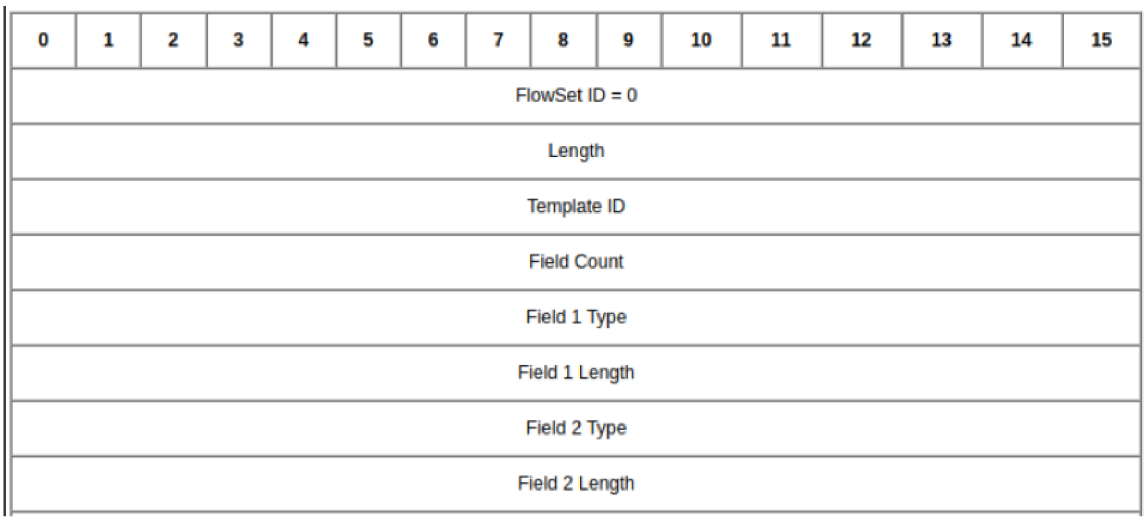
Obrázek 3.5: Hlavička exportního paketu NetFlow v9 [4]

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

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 položek v rámci datových záznamů, se kterými mají shodující se ID [4]. Ilustrace je na obrázku 3.6.

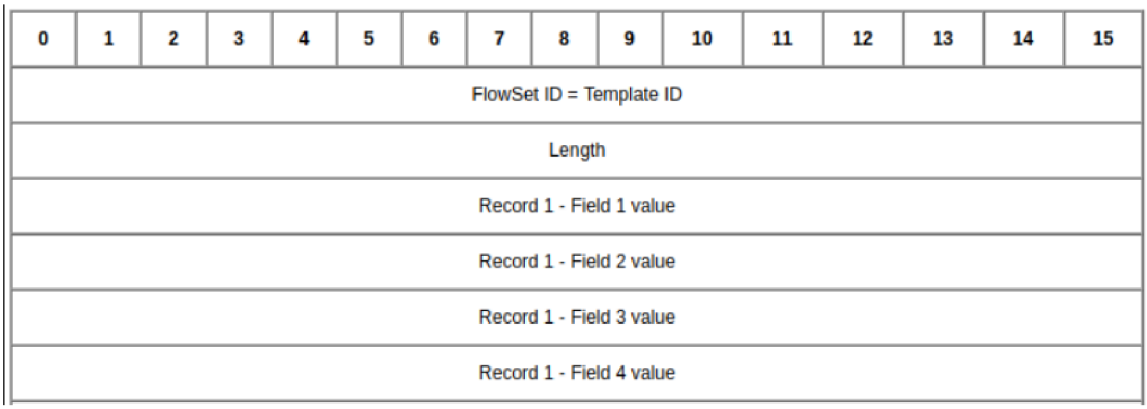


Obrázek 3.6: Formát šablonového FlowSetu [4]

- *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 položku 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 položek (Field count)* je počet položek v šabloně záznamu. Protože jeden FlowSet může obsahovat více záznamů, může tato hodnota sloužit k rozlišení začátku a konce dvou různých záznamů.
- *Typ položky (Field type)* numerická hodnota, která určuje typ položky. Výpis všech typů položek je v příloze A.
- *Délka položky (Field legth)* je délka položky daného typu v bytech [9, 4].

Formát datového FlowSet

Ilustrace datového FlowSet je na obrázku 3.7.



Obrázek 3.7: Formát datového FlowSetu [4]

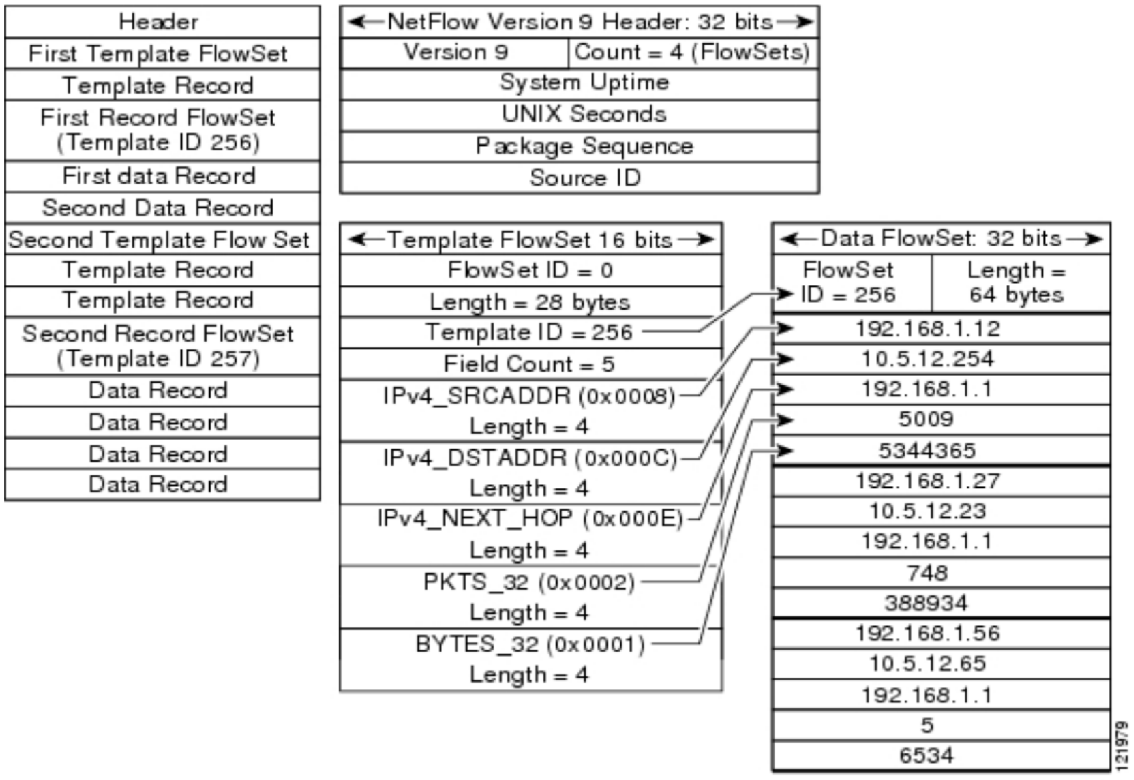
- *FlowSet ID* je ID hodnota datového FlowSet. Mapuje se na nějaké ID šablony. Pomocí tohoto ID kolektor nalezne šablonu záznamu, podle které dekóduje záznamy o toku z FlowSet dat.
- *Délka (Length)* je celková délka FlowSet. Je to suma délky FlowSet ID, hodnoty délky samotné, všech záznamů o tocích a doplňujících bitl, pokud nějaké jsou.
- *Záznam - Položka (Record - Field)* je kolekce dat záznamu o toku, kde každý záznam obsahuje skupinu datových položek. Typ a délka každé položky 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 [9, 4].

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 [4]. Na obrázku 3.8 je možné vidět příklad exportního paketu paketu.

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 bezsměrový, 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 [6, 25].

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) [6].

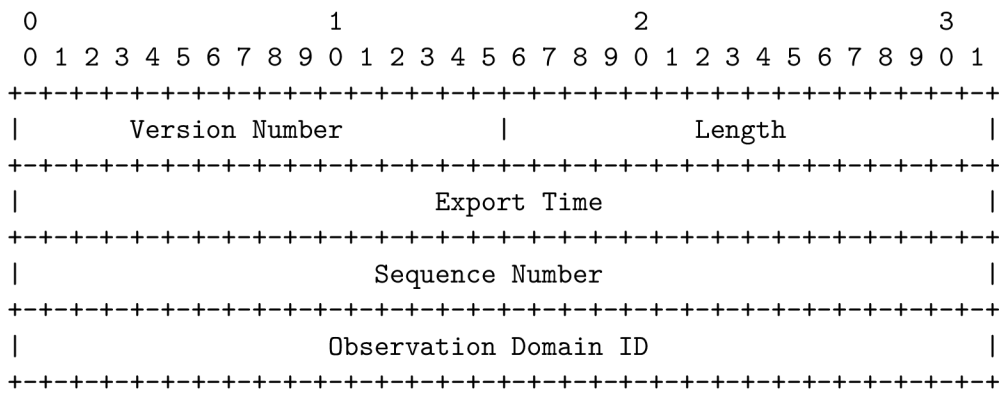


Obrázek 3.8: Příklad NetFlow paketu [14]

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 definuje, 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 [6, 15].

Formát hlavičky zprávy

- *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



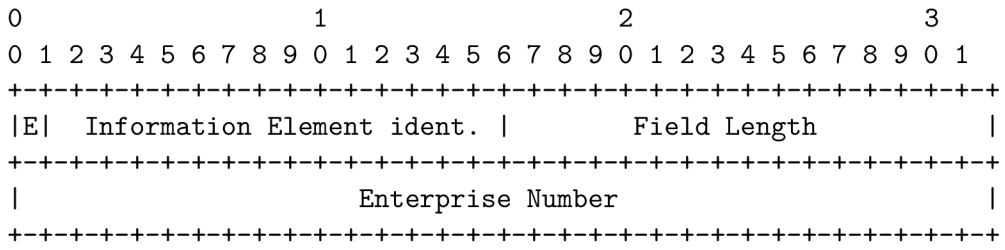
Obrázek 3.9: Formát hlavičky IPFIX zprávy [6]

síťový tok. V případě, že tok nemůže být přiřazen k žádné doméně, měla by být hodnota 0 [6].

Formát setu

Set je generický pojem pro kolekci záznamů, které mají podobnou strukturu. Pro popis je nutné nejdříve definovat pojem specifikátor položky.

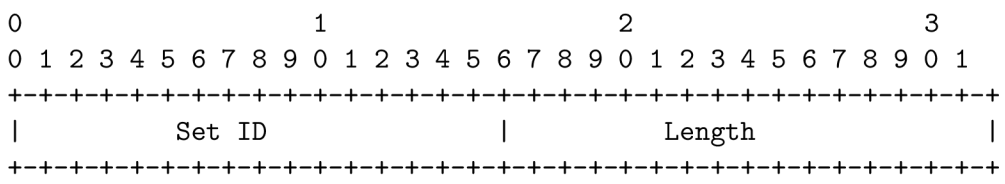
Specifikátor položky je způsob, jakým jsou v IPFIX záznamech popsány informační elementy. Jeho strukturu je možné vidět na obrázku 3.10. Informační elementy jsou v něm identifikovány pomocí *identifikátoru*. Pokud je enterprise bit (označení E) nastaven na 0, jde o informační element, který definuje IANA. Lze jej tedy najít v databázi této organizace. Pokud ne, jde o specifický informační element nějaké firmy, jejíž identifikátor musí být dále přítomen (jinak je položka nepovinná). Identifikátor firmy se vyskytuje v položce *Enterprise Number*. Specifikátor obsahuje také délku korespondujícího informačního elementu.



Obrázek 3.10: Formát specifikátoru položky [6]

Každý typ setu se skládá z hlavičky a jednoho nebo více záznamů. **Hlavička setu** obsahuje dvě hodnoty. Ilustraci je možné vidět na obrázku 3.11.

- *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.

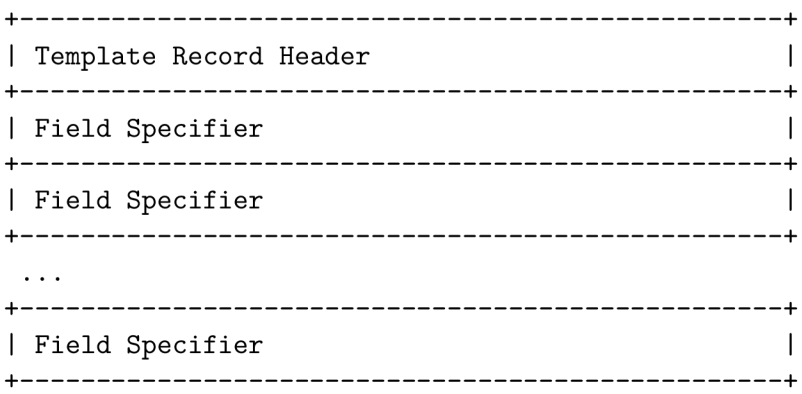


Obrázek 3.11: Formát hlavičky setu [6]

- *Délka (Length)* je celková délka setu v oktetech.

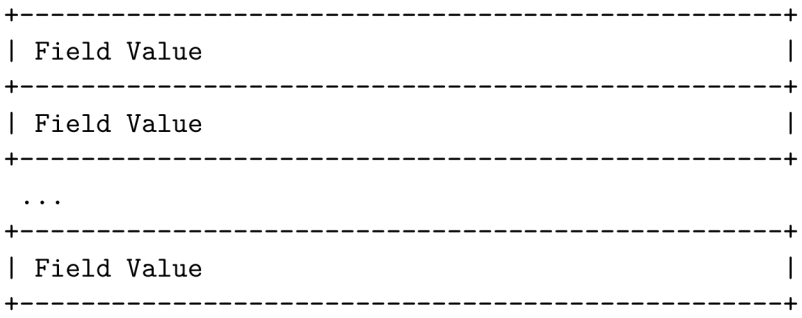
Záznam má 3 typy. Těmi jsou Šablonový záznam, Datový záznam a Záznam šablony možností. Dále jsou tyto typy popsány.

- *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 položek pro informační elementy, které definuje IANA, nebo informační elementy definované firmou. Na obrázku 3.12 je možné vidět strukturu záznamu. Hlavička obsahuje ID, dle kterého jsou přiřazovány datové záznamy, a celkovou délku záznamu.



Obrázek 3.12: Formát šablonového záznamu [6]

- *Datový záznam.* Strukturu záznamu je možno vidět na obrázku 3.13. Skládá se pouze z jednoho nebo více datových položek. 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.
- *Záznam šablony možností.* Tento typ záznamu předává kolektoru dodatečné informace, které nelze předat jen pomocí záznamů o toku. Popisuje ostatní datové položky, která nenesou pouze informace o toku [6, 25].



Obrázek 3.13: Formát datového záznamu [6]

3.4 Implementace v systémech Flowmon

Implementace exportu záznamů ve formátu NetFlow a IPFIX v systémech firmy Flowmon majoritně odpovídá popsaným doporučením dle standardů. Firma si navíc definuje vlastní položky dle pravidel, které byly popsány. Čtyři z těchto položek, která souvisí s protokoly důležitými pro tuto práci jsou zobrazena na obrázku 3.14. V prvním sloupci je identifikátor položky, dále označení této položky a nakonec její význam. Číselný identifikátor firmy (Enterprise Number) využívaný v těchto položkách je *39499*.

1	IPFIX_INVEA_HOST	HTTP hostname
2	IPFIX_INVEA_URL	HTTP URL
4	IPFIX_INVEA_METHOD_ID	HTTP method
12	IPFIX_INVEA_STATUS_CODE	HTTP result code

Obrázek 3.14: Exportované pole v systémech Flowmon

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 struktura sítě jako takové a také role přidělené strojům v této síti. Následně je popsána vytvořená datová sada. Na konci je popsán nástroj pro export záznamů o síťovém toku a proces exportu jako takový.

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.

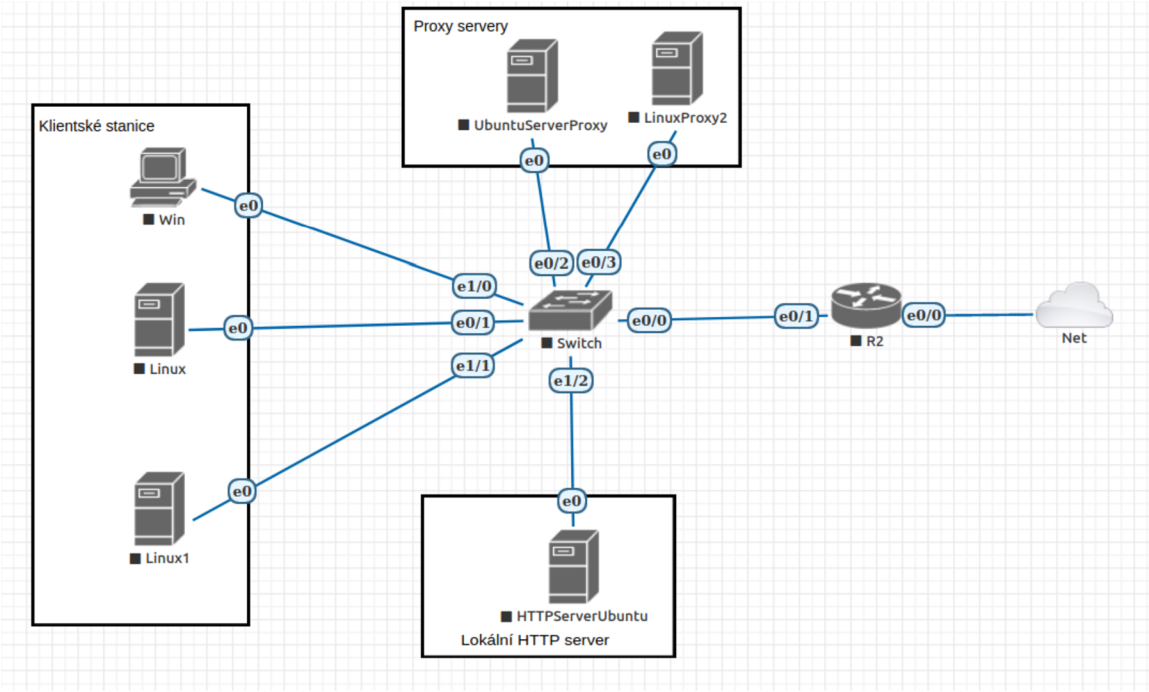
Nástroj je dostupný ve formě OVF obrazu připraveného pro virtualizaci anebo ISO obrazu připraveného pro instalaci. Pro účely práce byla vybrána verze určená pro virtualizaci. Obraz softwaru byl spouštěn ve virtualizačním softwaru VMWare Workstation Player. Po spuštění virtuálního EVE-NG softwaru a konfiguraci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

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

žícímu softwaru 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 z prostředí aplikace. Jde o otevřený projekt se síťovou strukturou. Pro vytváření síťových struktur je zde přehledné grafické prostředí. Pomocí funkcí aplikace je možné přidávat a spravovat jednotlivé uzly sítě a provádět akce související s uzly v struktuře (např. zapnout uzly, vypnout uzly nebo resetovat 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 Switch - L2-ADVENTERPRISE-M-15.1-20140814,
- L3 Router - L3-ADVENTERPRISE9-15.5.2T,
- OS Ubuntu - linux-ubuntu-18.04-server,
- OS Windows - Win10_21H2_Czech_x64.



Obrázek 4.1: Screenshot prohlížečové aplikace EVE-NG

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. V 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. Také byly zapojeny dva proxy servery do zřetězení, kdy se jeden dotazoval druhého a až ten na cílovou adresu. 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. Router měl nastavený NAT překlad adres. Výpis strojů je uveden dále.

Pro případ zřetězení proxy serverů za sebe byla síťová struktura poněkud změněna oproti obrázku 4.1. Router byl ze sítě odpojen a přístup k Internetu byl veden přímo na server LinuxProxy2. IP adresa na připojeném rozhraní byla získána skrz DHCP protokol (rozshas 192.168.0.0/24). Také bylo přidáno jedno spojení mezi druhým proxy serverem a switchem, aby mohlo být simulováno reálné zapojení proxy serverů.

Klientská stanice Linux

Jde o stroje s operačním systémem Ubuntu Server 20.04. Pro tyto stroje 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. Počet těchto strojů byl v síťové struktuře roven dvěma z důvodu spojování síťových toků z více zařízení.

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.

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, který sloužil jako jednoduchý HTTP server. Podpora pro HTTPS byla také definována. K tomu byla využita OpenSSL aplikace dostupná v základní výbavě operačního systému.

Proxy servery Linux

Stroje s operačním systémem Ubuntu Server 20.04, které sloužily jako proxy servery. Oproti popisu v kapitole 2 bylo ve fázi vytváření datové sady stroji přiděleno pouze jedno rozhraní. Na jeden ze strojů 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.

Na druhý ze strojů byla nainstalována pouze aplikace Squid, která podporuje jednoduché řetězení proxy serverů. Tento stroj byl využíván právě při zkoumání zřetězených proxy.

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.

```
.
├── 01_prvni_cast
│   ├── dopredny_HTTP_privoxy
│   ├── dopredny_HTTP_SOCKS
│   ├── dopredny_HTTP_squid
│   ├── dopredny_HTTP_tinyproxy
│   └── reverzni_HTTP_haproxy
├── 02_druha_cast
│   ├── dopredny_HTTP_squid
│   └── dopredny_SOCKS
└── 03_treti_cast
    ├── facebook
    ├── fit.vut
    ├── lokalni_server
    ├── microsoft
    ├── mit.edu
    └── vice_domen

```

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 rámci testování bylo poté nutné vytvořit další záznamy, které v rámci popisu a započítání celkové velikosti datové sady zahrnuty nejsou. Jejich popis je v kapitole 7.

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 problému 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.

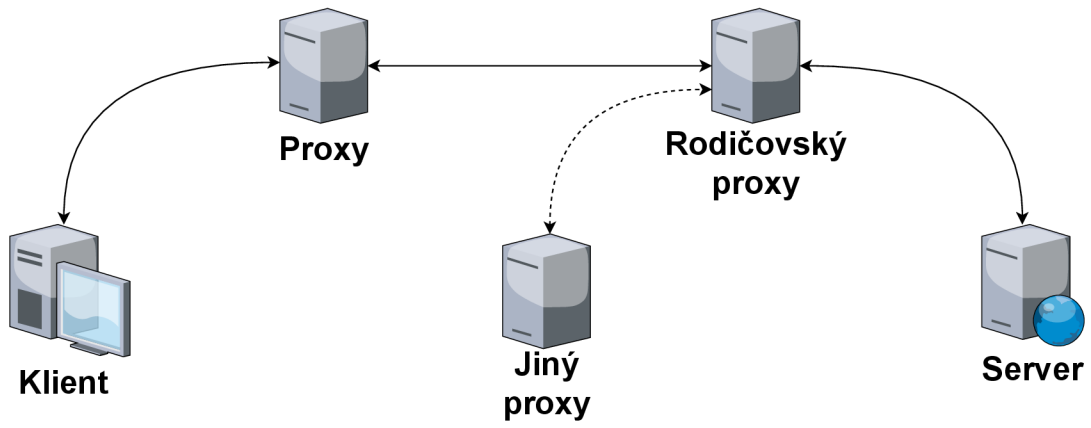
Cílová doména	Průměrná velikost souboru (MB)	Počet souborů	Průměrný počet TCP toků do proxy	Průměrný počet TCP toků z proxy
Lokální server	0,03	12	4	14
facebook.com	1	12	8	16
google.com	0,5	12	7	15
seznam.cz	11	12	55	117
wikipedia.org	0,18	12	2	6
microsoft.com	1,8	12	13	63
mit.edu	4,4	12	14	30
fit.vut.cz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

Tabulka 4.1: Statistické udaje první části datové sady

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.

Zřetězení více proxy serverů za sebou.



Obrázek 4.3: Vizualizace zřetězených proxy serverů

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. Rodičovský server může přijímat požadavky od více proxy serverů a také může odesílat požadavky na jiný další proxy server. Tímto je mezi proxy servery vytvořena stromová struktura. Tato struktura může mimo jiné dále napomáhat k rozložení zátěže v síti. Vizualizaci takto zřetězených proxy serverů je možné vidět na obrázku 4.3.

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.

Vytvořená datová sada by měla být dostatečná pro analýzu a vytvoření řešení. Obsahuje malé soubory pouze s jednoduchým spojením skrze jeden proxy, podle kterých mohlo být prozkoumáno chování proxy serverů. Bylo vyzkoušeno více proxy softwarů, takže mohlo být kontrolováno chování jiných implementací. Dále obsahuje soubory s odchycenou komunikací, která by mohla být označena jako reálná komunikace v moderních sítích. Ty mohly být využity pro testování a hledání chyb v řešení. Sada také obsahuje komunikaci pro prozkoumání zřetězených proxy a reverzního typu proxy. Sada také obsahuje dostatečné množství záznamů pro analýzu protokolů HTTP, HTTPS i SOCKS.

4.3 Vytvoření záznamů o síťovém toku

V této sekci je popsán nejdříve software využívaný pro export záznamů o síťových tocích. Využívaným softwarem byla Flowmon sonda, která byla poskytnuta firmou Flowmon. Ná-

sledně je popsán samotný proces exportu záznamů o síťových tocích, jak byly prováděny v rámci této práce.

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 [2]. 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. Exportér nevyužívá časovou informaci z dodaného souboru, ale přidává k záznamům vlastní časovou značku, odpovídající reálnému času exportu. Pro zachování časových souvislostí záznamů je tedy nutné exportovat s nastaveným příznakem speed na hodnotu 1.

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 v síťové komunikaci, ze které byly exportovány záznamy, 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 bylo zajištěno exportování dalších atributů souvisejících s těmito protokoly. Blíže jsou záznamy rozebrány v kapitole 5.

Záznamy exportované Flowmon sondou jsou ve formátu IPFIX. Nicméně pro účely této práce byly ukládány do souborů ve formátu csv. Exportované záznamy jsou 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 je nejdříve popsán pojem korelace, jak je chápán pro potřeby této práce. Následně jsou 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 vstup. Následně poté je popsán s možnými rozšířeními, které zpřesňují jeho výsledky.

5.1 Korelace v této práci

V této sekci je blíže rozebrán pojem korelace síťových toků. Nejdříve je představen popis problému vzájemného přiřazování síťových toků, jak byl nalezen v odborné literatuře. Následně je na základě tohoto popisu definován pojem korelace síťových toků. Popsaný pojem je přímo konkretizován pro proxy servery a pro použití v této práci.

Ve článku [27] je matematicky definován problém *trasování síťových spojení*. Nejdříve definuje tok c_i jako spojení mezi dvěma body v síti H_i a H_{i+1}. Následně definuje řetězec toků jako sérii navazujících spojení $(c_1, c_2, ..., c_n)$ mezi body v síti $H_1, H_2, ..., H_{n+1}$. Samotný problém je pak se znalostí toku c_n určit právě všechny ostatní toky $(c_1, c_2, ..., c_{n-1})$. Toto lze označit jako *problém korelace* [27].

V této práci je řešení problému popsaného výše omezeno pouze na ty části komunikace, které prochází skrz proxy server. V případě sledování jednoho proxy serveru, je tedy hledána dvojice toků, které se vyskytují na opačných stranách proxy serveru. Pro tyto toky musí platit, že patří do stejné série navazujících spojení. Pokud se vezme v potaz více zřetězených proxy serverů, je pro n proxy serverů hledáno $n + 1$ toků, které patří do stejné série navazujících spojení. Obecně je tedy hledána série vzájemně logicky navazujících záznamů síťových toků.

5.2 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.

Definice požadovaných vlastností metody

V již zmiňovaném článku *Inter-Packet Delay Based Correlation for Tracing Encrypted Connections Through Stepping Stones* je definován model korelační metody pro určení závislosti datových toků. Definována je *ideální korelační funkce*, která může být následně upravena pro problém řešený v této práci. Nechť je $Proxy_{in}$ označení množiny všech datových toků na jedné straně proxy serveru, $Proxy_{out}$ označení množiny všech datových toků na druhé straně proxy serveru. Pak je ideální korelační funkce definována jako $CF : Proxy_{in} \times Proxy_{out} \rightarrow 0, 1$, kde:

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

Korelace síťových toků je často založena na charakteristice těchto toků, jako je obsah paketů, informace obsažené v hlavičkách a časy začátků a konců toků. Charakteristiku datového toku lze modelovat metrickou funkcí

$$M : C \times P \rightarrow Z$$

C je soubor datových toků určených ke korelaci, P je doména parametrů datových toků a Z je doména korelační metriky. Na základě metriky lze postavit ohodnocenou korelační funkci

$$CVF : Z \times Z \rightarrow \langle 0, 1 \rangle$$

Výsledkem funkce je reálné číslo mezi 0 a 1. Pro aproximaci funkce CF pomocí CVF lze zavést prahovou hodnotu $0 \leq \delta \leq 1$. Poté platí, že toky c_i a c_j jsou v korelaci, pokud platí

$$CVF = \left(M\left(c_i, p\right), M\left(c_j, p\right)\right) \geq \delta. \tag{5.1}$$

Za předpokladu, že jsou tedy c_i a c_j ve stejném řetězci spojení, stačí pro určení korelace nalézt anebo sestrojit M, p, CVF a δ takové, že

$$\forall c_i, c_j \in C : CVF\left(M\left(c_i, p\right), M\left(c_j, p\right)\right) \geq \delta \tag{5.2}$$

V případě proxy serverů je tedy cílem nalézt unikátní charakteristiky toků, které se nemění při cestě skrz jednotlivé servery a nejsou ovlivněny šifrováním. Pokud jsou tyto charakteristiky dostatečné pro rozlišení a vyloučení nesouvisejících toků, může z nich být vytvořena spolehlivá metoda pro korelaci [27].

Výpis existujících metod

V následujícím textu jsou stručně vypsány některé metody, které mohou být využity pro korelaci síťových toků.

Počítání paketů

Ve článku [24] uvedena a analyzována možnost korelace pomocí počítání paketů. Tento přístup lze uplatnit ve chvíli, kdy do nějakého uzlu v síti skrz vstupní linku přichází a skrz výstupní linku odchází osamělý datový tok (jediný tok vyskytující se na dané lince). Počítání probíhá v rámci jednoho časového intervalu, kdy je předpoklad, že vnitřní zpoždění uzlu je v porovnání s intervalem malé. Pokud je počet paketů vstupního toku téměř stejný jako výstupního, lze mezi nimi předpokládat závislost [24].

Sledování zpráv v síti s rozdílným zpožděním

V článku [11] George Danezis navrhuje způsob jak v sítích s rozdílným zpožděním sledovat vstupní síťový tok až k výstupu. Princip metody založen na pozorování vstupního síťového toku. Tento tok může být reprezentován jako funkce objemu dat v čase. Pomocí konvoluce této funkce s exponenciální funkcí zpoždění sítě je vytvořena šablona, která určuje předpoklad jak bude tok v síti vypadat. Všechny toky v síti jsou s touto šablonou porovnány a je určena míra podobnosti daných toků s touto šablonou. Dle toho lze určit cestu daného spojení skrz síť [11].

Korelace dle vzájemné informace a frekvenční analýzy

ve článku [28] je definována *vzdálenostní funkce*, na základě které může být určena korelace mezi dvěma toky. Pro vytvoření této funkce jsou v tomto článku navrhovány dvě třídy přístupů. První je založen na statistické informaci o míře výskytu počtu paketů v čase. Druhý využívá frekvence počtu výskytu paketů v rámci časového intervalu [28].

5.3 Analýza datové sady

V této sekci je popsána analýza, která byla provedena nad vytvořenou datovou sadou. Nejdříve je popsán záznam jako samotný a jeho jednotlivé komponenty a atributy. Dále jsou diskutovány možnosti, jakými by mohlo být možné vytvořit řešení problému korelace.

Analýza

Záznam síťového toku exportovaný pomocí Flowmon sondy lze pomyslně rozdělit na několik částí. Popis záznamů jak je zde uveden je stručný a popisuje pouze části důležité pro tuto práci. Hned na začátku je jednoznačný identifikátor v rámci jednoho záznamu s názvem EXPORT COUNTER. Dále jsou v záznamu obsaženy časové informace o začátcích a koncích jednotlivých toků. Jak již bylo zmíněno tyto informace jsou softwarem exportovány dle vnitřního času sondy a ne dle časových značek obsažených v pcap souborech. Tato skutečnost je poněkud nepohodlná ve chvíli, kdy jsou záznamy exportovány po menších částech, protože jsou tím ztraceny časové souvislosti mezi jednotlivými toky. Pokud je však záznam exportován v reálném čase anebo jsou správně upraveny vstupní pcap soubory (více souborů z různých míst je spojeno), nemá tato skutečnost na analýzu žádný vliv.

V záznamech se následně vyskytují informace o velikosti toku v bajtech, o počtu paketů v toku a následně informace o L3 a L4 vrstvě. Je zde označen L3 protokol se zdrojovou a cílovou adresou a také L4 protokol se zdrojovým a cílovým portem. Dále již následují informace o aplikačních protokolech. Seznam informací těchto protokolů, 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

Ne vždy byly všechny informace kompletní. Někdy v záznamech chyběla hodnota (v záznamech se vyskytuje speciální hodnota NIL) ať už proto že se ji nepodařilo vyexportovat anebo v odchycené komunikaci vůbec nebyla přítomná.

Velké množství informací, se kterými bylo možné pracovat, poskytl protokol TLS. Tento protokol posílá v rámci handshake velké množství dat, které jsou teoreticky 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 5.2 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 není počítání paketů v našem případě spolehlivě použitelné. Při pohledu na exportované záznamy toků je vidět, že počty paketů v tocích nejsou tak podobné v souvisejících tocích a zároveň tak diverzní mezi jednotlivými nesouvisejícími toky, aby mohla být dle nich jednoznačně určena korelace.

Pro návrh řešení lze však vycházet z obecné metody korelace. Jednoduchý proxy server předává mnohem více informací v nezměněné podobě, než sítě s důrazem na anonymizaci, pro které jsou korelační metody převážně určeny, a tedy lze navrhnout metodu korelace na základě charakteristik protokolů v tocích. Například lze využít velké množství informací exportovaných z protokolu TLS a 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 5.2.

Techniky využití TLS protokolu

Velké množství položek 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 tech-

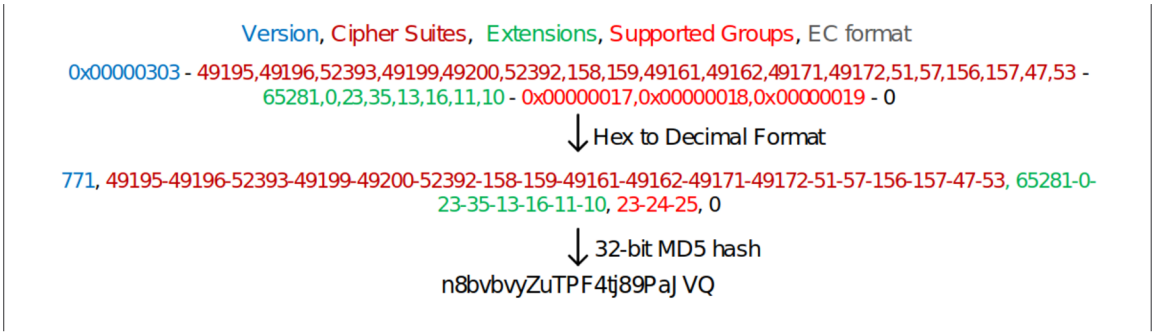
nik, která takto využívá atributy exportované z toků obsahujících TLS protokol je metoda určování 'otisku prstu' aplikace pomocí *JA3hashe*.

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 [7].

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 [7]. Příklad vytvoření JA3 hashe je na obrázku 5.1.



Obrázek 5.1: Příklad vypočítání JA3 hashe [18]

Z exportovaných atributů v záznamech o síťovém toku není možné JA3 hash vypočítat. Nicméně naštěstí samotný software `flowmonexp5` export takového atributu povoluje. Je 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 je získán a vyexportován JA3 hash v těch záznamech o tocích, ze kterých to je možné.

Analýza 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 vyjmuty vždy dva TCP toky, u kterých byla jistota vzájemné souvislosti. Exportované záznamy z těchto souborů sloužily jako základ pro první analýzu TLS spojení skrz proxy server.

Při analýze jednotlivých exportovaných toků bylo zjištěno, že testované 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. Lze předpokládat, že 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 [5] 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 [22]. 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 exportované hodnoty v záznamu je *Request Host*. Dále také byla přítomna hodnota s informací o metodě použité v komunikaci (GET, POST, apod.) a to ve formě číselné hodnoty s názvem *Method Mask*. Nejvíce ale byl rozdíl mezi toky vidět v časových souvislostech. Tím, že HTTP komunikace byla minorita, byly jednotlivé HTTP dotazy od sebe více vzdáleny. V tabulce 5.3 lze vidět příklad exportovaných toků s popsanými hodnotami.

Výše popsané atributy byly následně využity v řešení.

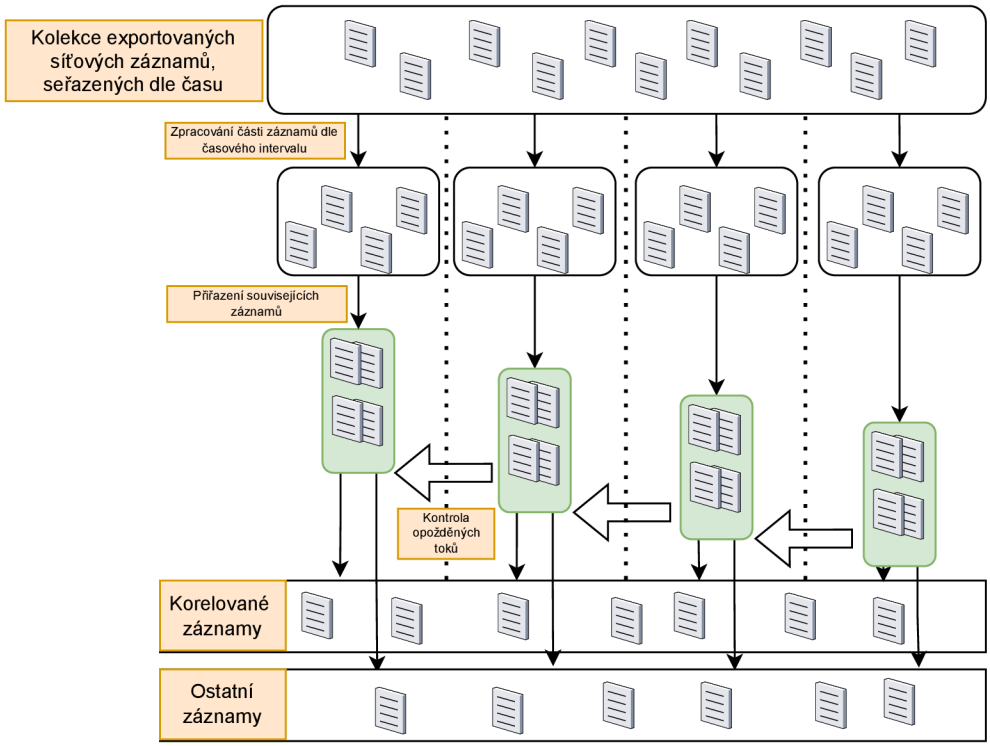
IPV4_SRC	IPV4_DEST	HTTP_METHOD_MASK	HTTP_REQUEST_HOST
10.0.0.20	142.251.36.99	NIL	NIL
10.0.0.20	142.251.36.132	512	www.google.com
157.240.30.3	10.0.0.20	NIL	NIL
10.0.0.20	8.8.8.8	NIL	NIL
10.0.0.20	157.240.20.19	512	static.xx.fbcdn.net
10.0.0.20	8.8.8.8	NIL	NIL
10.0.0.50	10.0.0.20	33024	static.xx.fbcdn.net:443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

5.4 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.



Obrázek 5.2: Vizualizace navrženého řešení

Navržený způsob korelace HTTPS toků

V sekci 5.3 byly již předeslány charakteristiky protokolu, které byly využity při návrhu řešení problému korelace. Těmito charakteristikami byly hodnoty JA3 hash, Server Name Indication a Server a Client Random. V následujícím textu je popsán způsob, jak jsou pomocí těchto hodnot toky korelovány.

Je předpoklad, že kombinace čtyř zmíněných hodnot bude dostatečná unikátní a charakteristická pro každou sérii toků, aby dle ní mohly být správně určeny všechny související toky a nebyly chybně určeny žádné nesouvisející. Teoreticky by mohla pro jednoznačné určení stačit i pouze dvojice hodnot Server Random a Client Random. Vzhledem k jejich velikosti a jedinečnosti pro každé spojení by mohlo jít o unikátní charakteristiku toků, nicméně využití více hodnot může jen zvýšit spolehlivost řešení a eliminovat nečekané chyby. Proto je využita i hodnota JA3 hashe a Server Name Indication.

Záznamy jsou zpracovány po částech. V rámci této práce je vždy určena část záznamů, která se bude zpracovávat. V reálném provozu by se vždy zpracovávala část, která by byla zrovna k dispozici. Každý záznam toku je tedy v řešení charakterizován touto čtveřicí, případně trojicí za předpokladu že hodnota SNI není dostupná (více rozebráno dále). V celém souboru záznamů jsou dle této čtveřice vzájemně přiřazovány toky. Přiřazování probíhá tak, že jsou jako související určeny vždy ty záznamy, které mají všechny porovnávané hodnoty stejné.

Korelace je určena ve chvíli, kdy jsou takto přiřazeny všechny záznamy o tocích, u kterých je to možné, a je zkontrolováno, že žádné dva toky v přiřazených záznamech nebyly mezi stejnými body v síti (nejsou k sobě přiřazeny toky ze stejné strany proxy serveru). Na obrázku 5.2 je vizualizace popsaného řešení.

Záznamy o tocích, kterým nebyl přiřazen žádný související tok, jsou označeny jako toky, u kterých nebylo možné určit korelaci. Neznamená to, že korelace v tu chvíli neexistuje, pouze že nebyla nalezena. V tu chvíli je například pravděpodobné, že záznam o toku se kterým by mohla být korelace určena ještě nebyl zpracován. Takové záznamy jsou tedy zpracovány znovu s další částí. Poté již zpracovávány nejsou.

Problémem tohoto řešení by mohla být náhodně generovaná hodnota Server a Client Random. Pokud by byl zpracováván dostatečně velký celek dat naráz, může se objevit obdoba takzvaného 'narozeninového problému'. To znamená, že se vygeneruje stejná hodnota 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. Takováto chyba je velice nepravděpodobná, vzhledem k velikosti těchto hodnot a také díky využití dalších charakteristik toků.

Lze předpokládat, že záznamy o tocích, které lze korelovat dle jejich chování, lze korelovat i dle času jejich výskytu. U proxy serveru lze předpokládat, 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 koncovým 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.

Řešení je tedy ještě rozšířeno a před tím, než je u záznamů toků určena korelace, jsou zkontrolovány časy jejich začátků. Pokud se tyto časy výrazně liší, korelace mezi těmito toky není určena.

Navržený způsob korelace HTTP toků bez TLS protokolu

Pro korelaci záznamů toků, které neobsahují pouze HTTP protokol, byl zvolen podobný způsob, jaký byl popsán u záznamu toků s přítomností TLS protokolu. Pro vzájemné přiřazení těchto záznamů byla jen použita kombinace jiných charakteristik. Těmi byly typ HTTP metody (Method Mask), doménové jméno dotazovaného cíle (Request Host) a typ obsahu.

Bohužel pouze tyto charakteristiky nestačí pro jednoznačné a správné určení korelace. Kombinace typu metody, doménového jména a typy obsahu není jedinečná pro každé spojení, protože může existovat více opakovaných dotazů v čase. Nicméně další unikátní atributy již bylo velmi těžké v záznamech najít. Využít se však dala skutečnost, že nezabezpečené HTTP komunikace, tedy komunikace bez TLS protokolu probíhá oproti té zabezpečené velmi malé množství. Jednotlivé toky, které nemají žádnou souvislost, tedy budou od sebe časově velmi vzdáleny. Využita zde může být tedy časová korelace.

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 dříve kontrolováno, že se cílový port a cílová IP adresa v záznamech o 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.

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.

Navržené řešení pro záznamy toků s TLS protokolem tedy počítá s možností naprosto ignorovat hodnotu SNI a určovat korelaci jen podle JA3 hashe a Server a Client Random hodnoty. Tyto tři hodnoty by samotné měly být dostatečné pro správné určení korelace. Spolu s kontrolou časových údajů a hodnot IP adres a portů by měly být eliminovány i nečekané chyby.

Více proxy serverů za sebou

Komunikace procházející přes více proxy serverů obsahuje charakteristiky popsané v tomto textu dříve. Zpracování záznamu takové komunikace a nalezení toků, u kterých je možné určit korelaci, se tedy může řídit stejnými pravidly. Pokud by byly známé IP adresy rozhraní proxy serverů, může být kontrolována i návaznost jednotlivých vzájemně přiřazených toků. Tím může být zajištěno, že žádný záznam o toku není nikde opomenut a korelace je určena správně. Jinak lze provést pouze kontrolu počtu vzájemně přiřazených toků. Ten musí být logicky odpovídající počtu proxy serverů v řetězci. V reálném světě ovšem nelze předem určit kolik záznamů bude přesně spojeno, 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.

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. Možnost korelace za pomocí TLS protokolu je v tuto chvíli nevyužitelná, 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.

5.5 Vytvořený algoritmus

V této 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 5.4, a tedy řeší výhradně HTTP a HTTPS síťovou komunikaci.

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.

1. Pokud není záznam o toku seřazen dle času začátku toku, proběhne jeho seřazení dle toho údaje.

2. Je vybrána část vyexportovaných záznamů o tocích 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. Označme ji jako C.

3. Ze zpracovávaná části jsou vyfiltrovány záznamy, které neobsahují TCP komunikaci. Následně je část rozdělena na tři části. Jedna z těchto částí obsahuje záznamy komunikace obsahující protokol TLS (označme C_TLS), druhá pouze HTTP bez zabezpečení (označme C_HTTP) a třetí všechny ostatní záznamy (označme C_OTHER).

4. Proběhne zpracování záznamů C_TLS.

    4.1. Pro každý záznam je vytvořena uspořádaná čtveřice K (JA3 hash, Server Name Indicator, Server Random, Client Random). Pokud je hodnota SNI nedostupná anebo je předem určeno, že nebude využívána, je vytvořena pouze trojice.

    4.2. Záznamy jsou rozděleny dle hodnot čtveřice K. Přiřazeny jsou k sobě vždy ty záznamy, které se shodují v celé čtveřici (trojici). Rozdělené záznamy označíme jako C_TLS_CORR.

4.3. Pokud již existují rozdělené záznamy C_TLS_CORR ze zpracování předchozí části, jsou porovnány se záznamy C_TLS_CORR aktuálně zpracovávané části. Pokud jsou v aktuální C_TLS_CORR záznamy, které by mohly být přiřazeny k jednomu nebo více záznamům z předchozí C_TLS_CORR, je tak provedeno. Takové záznamy jsou nadále již vyjmuty z aktuální C_TLS_CORR. Ostatní záznamy zůstávají nezměněny.

4.4. Záznamy z předchozí C_TLS_CORR, které mají vzájemně přiřazený alespoň jeden další záznam jsou kontrolovány. Jsou kontrolovány časové značky, přičemž začátky jednotlivých toků od sebe nesmí lišit o více než 300 ms. Dále je kontrolován privátní rozsah IP adres, pokud je to povoleno. Záznamy, které projdou kontrolami, jsou uloženy jako záznamy CORR, u kterých byla úspěšně určena korelace. Pokud záznamy kontrolou neprojdou anebo pokud není k záznamu přiřazen žádný jiný, jsou uloženy jako záznamy REST, u kterých korelace určena nebyla.

4.5. Zbylé záznamy z C_TLS_CORR jsou uloženy pro další zpracování.

5. Dále proběhne zpracování záznamů C_HTTP.

5.1. Pro každý záznam je vytvořena uspořádaná trojice K (Method Mask, Request Host, Content Type).

5.2. Záznamy jsou rozděleny dle K. Přiřazeny jsou k sobě vždy ty záznamy, které se shodují v celé trojici. Rozdělené záznamy označíme jako C_HTTP_CORR.

5.3. Pokud již existují rozdělené záznamy C_HTTP_CORR ze zpracování předchozí části, jsou porovnány se záznamy aktuální C_HTTP_CORR. Pokud jsou v aktuální C_HTTP_CORR záznamy, které by mohly být přiřazeny k jednomu nebo více záznamům z předchozí C_HTTP_CORR, je tak provedeno. Takové záznamy jsou nadále již vyjmuty z aktuální C_HTTP_CORR. Ostatní záznamy zůstávají nezměněny.

5.4. Záznamy z předchozí C_HTTP_CORR, které mají vzájemně přiřazený alespoň jeden další záznam jsou následně přiřazovány dle časových údajů. Pro to aby mohly být záznamy toků vzájemně přiřazeny, musí být jejich vzdálenost v čase menší jak 100 ms. Dále jsou kontrolována čísla cílových portů daných toků a také jejich cílové IP adresy, které musí být rozdílené. Záznamy, které jsou vzájemně přiřazeny a projdou kontrolami, jsou uloženy jako záznamy CORR, u kterých byla určena korelace. Pokud záznamy kontrolou neprojdou anebo pokud není k záznamu přiřazen žádný jiný, jsou uloženy jako záznamy REST, u kterých korelace určena nebyla.

5.5. Zbylé záznamy z aktuální C_HTTP_CORR jsou uloženy pro další zpracování.

6. Každá uložená sada záznamů CORR a REST je vyexportována na výstup. 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. Všechny zbylé záznamy z aktuálně zpracovávané části C_TLS_CORR a C_HTTP_CORR jsou uloženy pro další iteraci jako záznamy z předchozí části.

8. Je vybrána další část exportovaného záznamu C, stejně jako v bodě 2. Pokud již nejsou žádné záznamy na zpracování, tak algoritmus stejně pokračuje. Algoritmus končí, jakmile už nejsou pro zpracování žádné záznamy ani z předchozí části.

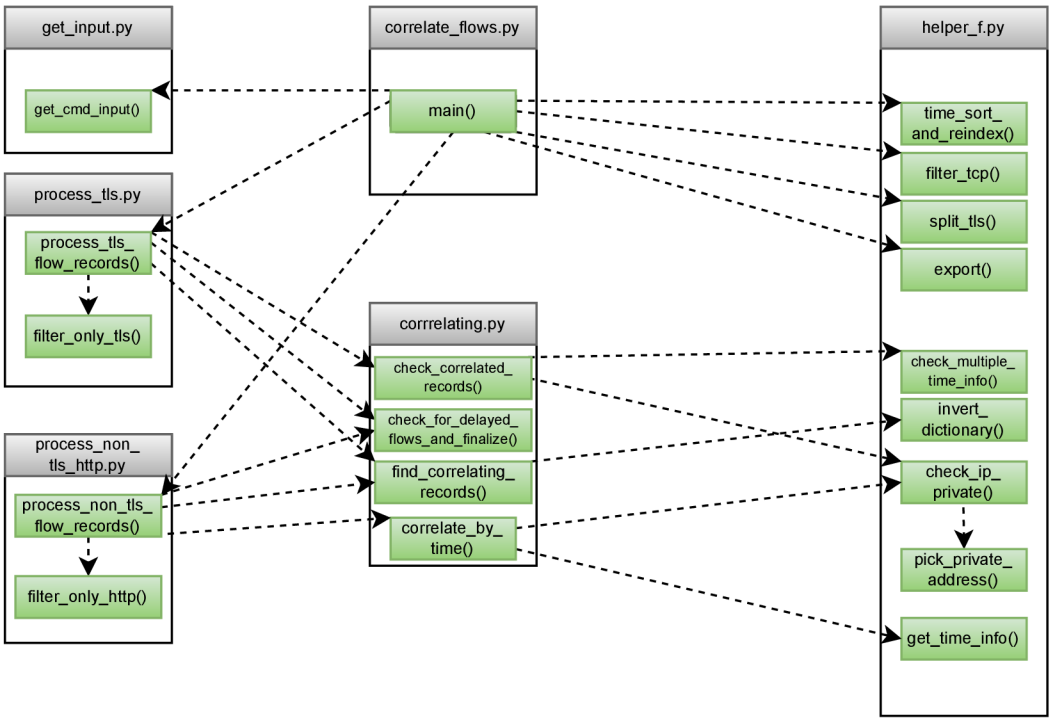
9. Algoritmus se vrací zpět k bodu 3.

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 očekávaný způsob spuštění programu a formát uživatelských vstupů pro program. Při spuštění může uživatel buď zadat parametr `help`, který způsobí vypsání nápovědy a konec programu. Dále může uživatel při spuštění vynutit využívání hodnoty SNI při korelaci argumentem `sni_check`. Stejně může uživatel vynutit kontrolu privátních rozsahů argumentem `private_check`. Uživatel může změnit velikost intervalu zpracovaných záznamů argumentem `process-window`, který je implicitně nastaven na 30 sekund. Dále je možné upravit informaci o počtu zřetězených proxy serverů v záznamech argumentem `proxy-chain`, což je potřeba při zpracování záznamů se zřetězenými proxy servery. Lze také změnit hodnotu pro kontrolu časových souvislostí argumentem `time-int`. Ta je implicitně nastavena na 0.3 sekund.

```
1 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
```

Výpis 6.1: Spuštění 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 seřazena 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. Funkci je možné vidět ve výpisu z kódu 6.2. Po seřazení je soubor záznamů přeindexován pomocí funkce `pandas.reset_index()`, aby indexy odpovídaly pořadí.

```
1 def time_sort(df : pd.DataFrame) -> pd.DataFrame:
2     df["START_SEC"] = pd.to_datetime(df["START_SEC"])
```

```
    df = df.sort_values(by=["START_SEC"])
    return df
```

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. To nastane ve chvíli, kdy velikost slovníků `tls_flows_values` a `http_flows_values`, které reprezentují záznamy zpracované v předchozí iteraci, je nulová. V každém cyklu jsou nejdříve vyfiltrovány záznamy co neobsahují TCP a následně jsou data rozdělena na data obsahující TLS záznamy, obsahující pouze HTTP záznamy a ostatní data. Filtrování probíhá podle hodnot určitých charakteristik záznamů. TLS záznamy jsou určeny podle přítomnosti vyexportované hodnoty JA3 hashe. Ze zbývajících záznamů jsou určeny HTTP záznamy podle hodnoty typu zprávy. Příklad funkce je na výpisu z kódu 6.3. První dvě tyto části dat jsou dále zpracovávány zvlášť, třetí číst již dále není zpracovávána.

```
def split_tls(df: pd.DataFrame) -> tuple[pd.DataFrame, pd.DataFrame]:
    non_tls = df[df['TLS_JA3_FINGERPRINT'] == 'NIL']
    non_tls_http = non_tls[non_tls['HTTP_METHOD_MASK'] != 'NIL']
    tls = df[df['TLS_JA3_FINGERPRINT'] != 'NIL']
    return non_tls, non_tls_http, 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`.

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

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. V těchto dílčích slovnících jsou záznamy toků vždy identifikovány dle indexů hodnot. Pro každý záznam je nejdříve vytvořen slovník obsahující dvojici

hodnot `(SERVER_RANDOM, CLIENT_RANDOM)`. Následně je pro každý záznam vytvořena n-tice s touto dvojicí, hodnotou JA3 hashe a hodnotou SNI, pokud je povolena. Tyto n-tice jsou opět uloženy do slovníku, kde klíčem je index identifikující záznam. Funkce, která je využita pro rozdělení, má název `find_correlating_records()` a je umístěna v souboru `correlating.py`. Její předpis a tělo je na výpisu z kódu 6.5. Vstupem funkce je výše popsaný slovník s n-ticemi. Výstupem funkce je zase slovník, ovšem nyní již invertovaný a s agregovanými hodnotami. Tím se jednoduše najdou v celém souboru záznamů ty, které mají stejné hodnoty n-tic.

```
1 def find_correlating_records(flows : dict) -> dict:
2     corr_f = HF.invert_dictionary(flows)
3
4     return corr_f
```

Výpis 6.5: Funkce pro rozdělení záznamů na menší části

Pro invertování slovníku je vytvořena funkce `invert_dictionary()`, která vymění ve slovníku jednotlivé klíče a hodnoty a v novém slovníku agreguje všechny hodnoty se stejným klíčem do jednoho záznamu. Zpracování hodnot ve slovníku probíhá cyklicky, časová složitost je lineární. Funkci je možné vidět na výpisu z kódu 6.6.

```
1 def invert_dictionary(dictionary : dict) -> dict:
2     new_dict = defaultdict(list)
3     for key in dictionary:
4         new_dict[dictionary[key]].append(key)
5     return new_dict
```

Výpis 6.6: Funkce pro invertování slovníku

Struktura invertovaného slovníku je tedy následující.

```
{ (JA3, [SNI], (SERVER_RANDOM, CLIENT_RANDOM) : [INDEX1, INDEX2, ...] }
```

Další dvě důležité funkce jsou `check_for_delayed_flows_and_finalize()` a `check_correlated_records()`. Ty jsou volány pouze pokud již jsou k dispozici zpracované záznamy z předchozí iterace. První z těchto funkcí přijímá invertované slovníky z předchozí a aktuálně zpracovávané části záznamů. Cyklicky pro každou položku slovníku z aktuální části zkontroluje, zda ji nelze přiřadit k položce z části předchozí. Pokud je taková položka nalezena, je taková položka ze slovníku s aktuálně zpracovávanými záznamy odebrána a přidána k záznamům předchozím. Ze slovníku předchozích záznamů je následně vytvořeno pole obsahující n-tice s indexy záznamů. Všechny indexy v jedné n-tici reprezentují vzájemně přiřazené toky. Funkce vrací pole takto přiřazených záznamů a slovník s ostatními záznamy z aktuálně zpracovávané části. Předpis funkce je možné vidět na výpisu z kódu 6.7.

```
1 def check_for_delayed_flows_and_finalize(prev_corr_flows : dict, flows :
      dict) -> tuple[list, dict]:
```

Výpis 6.7: Funkce pro kontrolu zpožděných záznamů

Další funkce provádí kontrolu vzájemně přiřazených záznamů. Vstupem funkce je seznam přiřazených záznamů, slovník s časovými značkami záznamů a s cílovými IP adresami. Dále je předáván maximální počet záznamů, který může být v řetězci, hodnota pro kontrolu časových souvislostí a příznak, zda bude prováděna kontrola privátního rozsahu. Pro každou n-tici je provedena kontrola, zda obsahuje povolené množství prvků (to znamená více jak

jedna a méně jak maximální počet). Dále je volána funkce pro kontrolu rozsahu IP adres a pro kontrolu časových souvislostí. Pokud jsou kontroly v pořádku, je n-tice vrácena v poli korelovaných záznamů. Pokud ne, jsou jednotlivé indexy vráceny v poli záznamů, u kterých korelace nebyla nalezena. Předpis funkce je možné vidět na výpisu z kódu 6.8.

```
def check_correlated_records(correlated : list, time : dict, ipv4 : dict
    , CHECK_IP_PRIVATE : bool, REVERSE_PROX : bool, flow_count : int,
    time_int : float) -> tuple[list, list]:
```

Výpis 6.8: Funkce pro kontrolu přiřazených záznamů

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. Funkce přijímá argument `check` typu bool. Pokud je nastaven na False, kontrola se neprovádí a funkce zrovna vrací True. Jinak je kontrola provedena a vrácená hodnota záleží na předané IP adrese. Funkci je možné vidět na výpisu z kódu 6.9.

```
def check_ip_private(check : bool, ip_addr : str):
    if not check:
        return True
    return ip.ip_address(ip_addr).is_private
```

Výpis 6.9: Funkce pro kontrolu privátního rozsahu

Funkce pro kontrolu časových souvislostí má název `check_multiple_time_info()`. Vstupem funkce je seznam přiřazených indexů záznamů, slovník s časovými značkami záznamů a hodnota pro kontrolu, která je implicitně nastavena na 0.3. Protože záznamy jsou seřazeny a jsou vždy zpracovávány popořadě, je jisté, že indexy jsou seřazeny dle časové posloupnosti. Kontrolovány jsou tedy vždy dvě časové značky odpovídající dvěma po sobě jdoucím indexům. Pokud se liší o větší hodnotu než byla předána funkci, je vráceno False. Jinak je vráceno True. Předpis funkce je možné vidět na výpisu z kódu

```
def check_multiple_time_info(val : list, time : dict, secs : float =
    0.3) -> bool:
```

Výpis 6.10: Předpis funkce pro kontrolu časových souvislostí

Na konci funkce `process_tls_flow_records` je vrácena dvojice seznamů, slovník a hodnota bool. 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. Slovník reprezentuje rozdělené záznamy, které budou dále zpracovány v další iteraci. Hodnota bool reprezentuje zpětnou vazbu ohledně hodnoty SNI. Pokud byl při zpracovávání záznamů objeven takový, který má hodnotu SNI rovnu `NIL`, je v této hodnotě vráceno False. Jinak je vrácena hodnota předaná při volání funkce.

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`.

```
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,
list]
```

Výpis 6.11: 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 vytvořen slovník s charakteristikami záznamů toků a je využita funkce `find_correlating_records()` a `check_for_delayed_flows_and_finalize()`. Pouze v tomto případě jsou zpožděné toky hledány pouze v té části záznamů, které mají maximální rozdíl od předchozích jednu sekundu. Pro to byla vytvořena funkce `get_potentionally_delayed()`. 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 indexů záznamů. Informace o časových značkách jsou předány pomocí slovníku s těmito hodnotami. Vždy dva po sobě jdoucí záznamy jsou porovnány, je zkontrolována jejich cílová IP adresa a cílový port a rozdíl v jejich časech. Vzhledem k možnosti zřetězení proxy serverů 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.

```
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]:
```

Výpis 6.12: Předpis funkce pro korelaci dle času

Výstupem hlavní funkce jsou dva seznamy a jeden slovník. Jejich význam je stejný jako v předchozí části. První seznam reprezentuje korelované záznamy toků a druhý záznamy, u kterých nebylo možné určit korelaci. Slovník reprezentuje rozdělené záznamy, které budou dále zpracovány v další iteraci.

Export a pokračování

Výstupní seznamy s korelovanými a ostatními záznamy z obou funkcí jsou spojeny. Oba seznamy jsou exportovány 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()`). Pro každý exportovaný tok je vždy exportována sedmice hodnot. Těmi jsou identifikátor záznamu přidělený Flowmon sondou (EXPORT COUNTER), časová značka začátku toku, zdrojová a cílová IP adresa, zdrojový a cílový port a protokol L4 vrstvy. Záznamy toků u kterých byla určena korelace jsou exportovány do souboru s názvem `correlation_output.csv`. Na jednom řádku v tomto souboru jsou vždy záznamy, u kterých byla určena korelace. Ostatní záznamy jsou exportovány do souboru s názvem `rest_of_flows.csv`, vždy jeden záznam na řádek.

Následně jsou uloženy aktuálně zpracovávané záznamy do proměnné pro zpracování v další iteraci. Jsou načteny další záznamy, které opět odpovídají předem definovanému časovému intervalu. Soubor zpracovávaných dat je opět seřazen dle časové informace a program pokračuje další iterací.

Příklad použití

Níže je uveden příklad použití aplikace. Aplikace je spuštěna se předaným vstupním souborem `linux+linux.csv`. Pomocí argumentu je povolena kontrola privátního rozsahu toků IP adres a velikost intervalu zpracovávané části záznamů je nastavena na 60 sekund. Hodnota pro kontrolu časových souvislostí je nastavena na jednu sekundu.

```
python3 correlate_flows.py -i linux+linux.csv -p -w 60 -t 1
```

Na obrázcích 6.2 a 6.2 jsou možné vidět snímky obrazovky z souboru se záznamy korelovaných toků. Kvůli velkému počtu záznamů jsou snímky rozděleny na dvě části. Na prvním obrázku je první část souboru, na druhém druhá část. Jednotlivé řádky reprezentují korelované toky.

	A	B	C	D	E	F	G
1	FLOW1_EXPORT_COUNTE	FLOW1_START_SEC	FLOW1_L3_IPV4_SRC	FLOW1_L3_IPV4_DST	FLOW1_L4_PORT_SR	FLOW1_L4_PORT_DS	FLOW1_L4_PROTO
2	1	2022-07-22 17:25:02.33079296	10.0.0.10	10.0.0.60	40268	3128	6
3	2	2022-07-22 17:25:02.56077694	10.0.0.10	10.0.0.60	40272	3128	6
4	23	2022-07-22 17:25:03.55095358	10.0.0.10	10.0.0.60	40274	3128	6
5	4	2022-07-22 17:25:03.55387069	10.0.0.10	10.0.0.60	40276	3128	6
6	15	2022-07-22 17:25:03.55585446	10.0.0.10	10.0.0.60	40278	3128	6
7	11	2022-07-22 17:25:03.55782149	10.0.0.10	10.0.0.60	40280	3128	6
8	10	2022-07-22 17:25:03.55935546	10.0.0.10	10.0.0.60	40282	3128	6
9	16	2022-07-22 17:25:03.56690364	10.0.0.10	10.0.0.60	40284	3128	6
10	30	2022-07-22 17:25:03.79561556	10.0.0.10	10.0.0.60	40286	3128	6
11	32	2022-07-22 17:25:08.98971575	10.0.0.70	10.0.0.60	58300	3128	6
12	36	2022-07-22 17:25:09.00872674	10.0.0.70	10.0.0.60	58302	3128	6
13	48	2022-07-22 17:25:09.01950988	10.0.0.70	10.0.0.60	58304	3128	6
14	38	2022-07-22 17:25:09.02933997	10.0.0.70	10.0.0.60	58306	3128	6
15	40	2022-07-22 17:25:09.23702317	10.0.0.70	10.0.0.60	58312	3128	6
16	82	2022-07-22 17:25:20.79328756	10.0.0.10	10.0.0.60	40290	3128	6
17	79	2022-07-22 17:25:21.28938537	10.0.0.10	10.0.0.60	40292	3128	6
18	65	2022-07-22 17:25:21.29675695	10.0.0.10	10.0.0.60	40294	3128	6
19	68	2022-07-22 17:25:21.29973408	10.0.0.10	10.0.0.60	40296	3128	6
20	76	2022-07-22 17:25:21.33594372	10.0.0.10	10.0.0.60	40298	3128	6
21	117	2022-07-22 17:25:22.45254475	10.0.0.10	10.0.0.60	40300	3128	6

Obrázek 6.2: Screenshot souboru s korelovanými toky

H	I	J	K	L	M	N
FLOW2_EXPORT_COUNTE	FLOW2_START_SEC	FLOW2_L3_IPV4_SRC	FLOW2_L3_IPV4_DST	FLOW2_L4_PORT_SR	FLOW2_L4_PORT_DS	FLOW2_L4_PROTO
0	2022-07-22 17:25:02.33281784	10.0.0.60	142.251.36.100	36782	443	6
5	2022-07-22 17:25:02.56182845	10.0.0.60	142.251.36.100	36786	443	6
27	2022-07-22 17:25:03.55252505	10.0.0.60	157.240.30.27	34304	443	6
20	2022-07-22 17:25:03.55477569	10.0.0.60	157.240.30.27	34306	443	6
6	2022-07-22 17:25:03.55690253	10.0.0.60	157.240.30.27	34308	443	6
13	2022-07-22 17:25:03.55896205	10.0.0.60	157.240.30.27	34310	443	6
12	2022-07-22 17:25:03.56175715	10.0.0.60	157.240.30.27	34312	443	6
7	2022-07-22 17:25:03.58156883	10.0.0.60	157.240.30.27	34314	443	6
18	2022-07-22 17:25:03.79730967	10.0.0.60	157.240.30.27	34316	443	6
47	2022-07-22 17:25:08.99622673	10.0.0.60	142.251.36.99	37386	443	6
53	2022-07-22 17:25:09.02987062	10.0.0.60	142.251.36.131	35494	443	6
50	2022-07-22 17:25:09.02987246	10.0.0.60	142.251.36.131	35496	443	6
37	2022-07-22 17:25:09.03135529	10.0.0.60	142.251.36.131	35498	443	6
46	2022-07-22 17:25:09.27908732	10.0.0.60	142.251.37.110	44480	443	6
86	2022-07-22 17:25:20.79871124	10.0.0.60	104.244.42.65	59212	443	6
77	2022-07-22 17:25:21.32348922	10.0.0.60	152.199.21.141	57446	443	6
66	2022-07-22 17:25:21.32361462	10.0.0.60	152.199.21.141	57450	443	6
64	2022-07-22 17:25:21.32349343	10.0.0.60	152.199.21.141	57448	443	6
72	2022-07-22 17:25:21.33776024	10.0.0.60	152.199.21.141	57452	443	6
111	2022-07-22 17:25:22.45449032	10.0.0.60	152.199.21.141	57454	443	6

Obrázek 6.3: Screenshot souboru s korelovanými toky

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.

Pro automatizované testování v tomto případě nastává následující problém. 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. Výhodou pro kontrolování záznamů byla možnost přistoupit k souborům se záznamy proxy serverů. Například software Squid ukládá do svých vnitřních záznamů přijímané žádosti, přičemž ukládá jak adresu klienta, tak cílovou adresu požadavku.

Testovací datová sada má podobu kolekce exportovaných záznamů síťových toků uložených v souborech formátu csv. Tyto záznamy byly získány z části z datové sady určené pro analýzu a z části ze souborů s komunikací, které byly odchytávané přímo pro účely testování. Některé tyto soubory byly vytvářeny dodatečně, protože nim bylo nutné získat i soubor se záznamy proxy serveru, které u souborů v původní datové sadě chyběly. Takto bylo vytvořeno dalších přibližně 400 MB záznamů.

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 pro jednotlivé domény vyfiltrovány vždy tři dvojice toků, u kterých byla jistá souvislost. Bylo důležité zachovat toky v časových souvislostech, aby záznamy po exportu mohly být využity v testování správně. U komunikace odchycené z více zřetězených proxy serverů, jenž byla již ze základu

rozložena ve více souborech, byly tyto soubory také spojeny. Nicméně z takto spojených souborů byly sondou Flowmon exportovány prázdné záznamy. Nebyl nalezen důvod, proč se tak dělo, předpoklad je buď chyba v programu Wireshark, který byl pro spojování souborů s komunikací využit, anebo v sondě Flowmon. Záznamy této komunikace jsou tedy vyexportovány zvlášť a až poté byly spojeny a jejich časy ručně upraveny, aby byly zachovány časové souvislosti.

Další část testovací sady tvoří velké soubory s odchycenou síťovou komunikací, které obsahují reálný síťový provoz. Soubory 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 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. K exportovaným záznamům je vždy přiložena část záznamů proxy serveru, která odpovídá časovému intervalu, ve kterém byla komunikace získána.

Vzhledem k tomu, že 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 popsané záznamy obsahující malé množství toků, především s protokolem HTTP bez protokolu TLS.

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 část sady obsahuje soubor s exportovaným záznamem komunikace a následně soubor s výsledky korelace. Části, pro které bylo potřeba dále získat komunikaci nad rámec datové sady určené pro analýzu obsahuje soubory s touto odchycenou komunikací. Složky s jednoduchými záznamy obsahují soubory s vyfiltrovanou komunikací obsahující pouze popsané toky.

7.2 Způsob testování

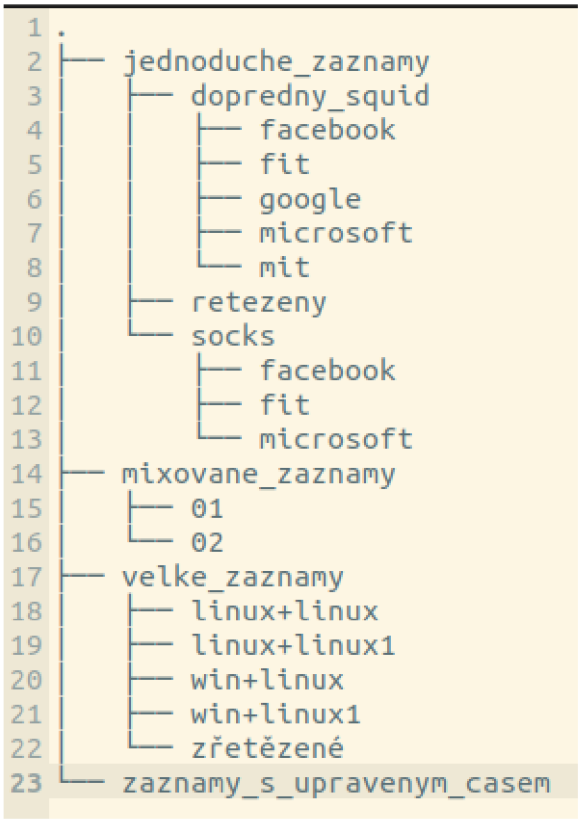
V této sekci je popsán způsob, jakým byly testy prováděny. Je popsán účel testování a také jaká část testovací sady byla pro které testy využita.

Testování správné funkčnosti

První testy, které byly provedeny, cílily na ověření správného fungování algoritmu a implementace. Tyto testy byly prováděny s malou datovou sadou, u které 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 správného určení korelace 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. Jediná dvojice záznamů u které nebyla nalezena korelace byly dva záznamy komunikace se



Obrázek 7.1: Souborová struktura testovací sady

serverem Google. U těchto záznamů chyběla totiž hodnota Random protokolu TLS a tudíž nebylo možné najít spojitost za využití tohoto protokolu. Při kontrole komunikace bylo zjištěno, že hodnoty v komunikaci byly obsaženy, ale nebyly z nějakého důvodu exportovány, předpoklad je že šlo o chybu Flowmon sondy.

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 270 toků, z nichž přibližně 230 bylo TCP. Z toho se úspěšně povedlo určit korelaci u 220 toků (110 dvojic), což znamená určení korelace u přibližně 95 % toků.

Testování správnosti řešení

Pro testování správnosti a použitelnosti řešení v reálném provozu byla využita část testovací datové sady s velkými záznamy. Tato část obsahuje neupravený provoz, jaký se vyskytuje v reálné komunikaci v Internetu. Tyto testy cílily na ověření navrženého řešení a jeho použitelnosti v praxi. Využity byly soubory se záznamy o síťovém toku s variantní velikostí od 1,1 do 1,8 MB. Soubory obsahovaly i některé neplatné, špatně vyexportované záznamy, případně záznamy se špatnou reprezentací dat.

Pro kontrolu výsledků byly využity záznamy proxy serverů. Struktura souboru se záznamy proxy serveru Squid, který byl pro testování využíván, je následující.

```
time elapsed remotehost code/status bytes method URL peerstatus/peerhosttype
```

Typ proxy	Doména	Počet HTTP toků	Nalezené korelace (počet toků)	Procentuální úspěšnost (%)
Dopředný HTTP	facebook.com	6	6	100
Dopředný HTTP	fit.vut.cz	6	6	100
Dopředný HTTP	google.com	6	4	66,6
Dopředný HTTP	microsoft.com	6	6	100
Dopředný HTTP	mit.edu	6	6	100
Řetězený HTTP	facebook.com	3	3	100
Dopředný SOCKS	facebook.com	2	2	100
Dopředný SOCKS	fit.vut.cz	2	2	100
Dopředný SOCKS	microsoft.com	2	2	100

Tabulka 7.1: Testování jednoduchých záznamů

Soubor	Počet TCP toků	Nalezené korelace (toky)	Procentuální úspěšnost (%)
01	172	162	94,18
02	60	58	96,66
Celkem	232	220	94.82

Tabulka 7.2: Testování mixovaných záznamů

Pro potřeby této práce jsou využitelné následující položky:

- *time*, reálný čas zaznamenání dotazu,
- *elapsed*, doba po kterou server zpracovával dotaz v milisekundách,
- *remotehost*, identifikátor klienta,
- *method*, HTTP metoda
- *peerhost*, identifikátor cíle požadavku[1].

Podle času zaznamenání bylo možné přiřadit záznamy proxy serverů k exportovaným záznamům síťových toků. Protože proxy server Squid zaznamenává požadavek až po jeho zpracování, byl užitečná i hodnota `elapsed`. Dle ní šlo dopočítat, kdy mohl požadavek na server reálně přijít. Následné položky byly využity pro kontrolu určených korelací.

Pro kontrolu korelací byl vytvořen skript `evaluate.py`. Ten očekává na vstupu adresář, který obsahuje soubor s výstupem aplikace pro určení korelace a soubor se záznamy proxy serveru. Samotné testování probíhá následovně. Záznamy proxy serveru jsou implicitně seřazeny dle času. jsou tedy postupně procházeny a dle dvojice zdrojová a cílová IP adresa jsou k nim hledány korelované záznamy. Korelované záznamy jsou sice také seřazeny dle času, nicméně se v nich střídají záznamy s a bez TLS protokolu. Záznamy jsou tedy dle čísla portu (443 a 80) rozděleny na dvě části a z těch je vybíráno na základě zaznamenané metody. Tyto záznamy jsou taktéž procházeny postupně, přičemž pro každý záznam proxy serveru je vyzkoušeno až 10 korelovaných záznamů, než je vyzkoušen další. Výsledky jsou uloženy vždy do souboru `evaluation.txt` a jsou sumarizovány v tabulce 7.3. Procentuální úspěšnost byla počítána jako procento správně korelovaných záznamů ze všech záznamů proxy. Je to kvůli tomu, že v záznamech se někdy vyskytuje dvakrát korelovaný záznam,

který však je v záznamu proxy serveru obsažen pouze jednou. Lepší je tedy hodnotit, zda byla nalezena korelace toků odpovídající záznamu proxy serveru.

Soubor	Počet TCP záznamů	Nalezené korelace (korelované záznamy)	Správně korelované záznamy	Počet záznamů proxy	Procentuální úspěšnost (%)
linux+linux	744	215	168	198	84,84
linux+linux1	658	166	83	93	89,24
win+linux	493	97	44	197	22,33
win+linux1	671	115	179	373	47,98
zřetězené (2 soubory)	1282	211	77	241	23,23

Tabulka 7.3: Testování velkých záznamů

Velké soubory byly využity i pro testování času běhu aplikace. Byly měřeny každý soubor zvlášť a poté byly spojeny do jednoho velkého souboru. Pro měření byla využita konzolová aplikace `time` a měřeno bylo na procesoru měřeno na procesoru Intel Core i5, 10. generace. Bylo provedeno 5 běhů programu a výsledky byly zprůměrovány. Výsledky je možné vidět v tabulce 7.4.

Soubor	Počet záznamů	Velikost (MB)	Průměrný čas běhu (ms)
linux+linux	916	1,3	660
linux+linux1	729	1,1	630
win+linux	1033	1,5	650
win+linux1	1097	1,7	680
Spojený	3396	4,9	1000

Tabulka 7.4: Testování časové náročnosti

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 jeden soubor vytvořený ze souboru `win+linux.csv`, ze kterého byla vyjmuta podmnožina toků. Takto byl vytvořen soubor, který obsahuje 100 záznamů. Těmto záznamům byly upraveny hodnoty časových značek, aby se lišily maximálně v hodnotách stovek milisekund a takto byla hledána korelace mezi záznamy. Výsledky je možné vidět v tabulce 7.5

Počet toků	Nalezené korelace (toky)	Procentuální úspěšnost (%)
100	20	20

Tabulka 7.5: Testování časových korelací

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 se již míra úspěšnosti snižovala, 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. Výsledky u testování záznamů zřetězených proxy serverů nejsou nejlepší. Při kontrole souboru s výsledky korelace ale nebyl nalezen žádný viditelný problém, je tedy možné, že byly pouze špatně nastaveny testy.

Pro záznamy bez protokolu TLS je řešení pouze velmi obtížně využitelné. Je očividné, že spoléhat se na časovou korelaci není správné řešení a bude nutné nejspíše upravit export záznamů bez tohoto protokolu, aby mohlo být využito více položek z komunikace. Zlepšení korelace záznamů bez TLS protokolu tedy zůstává na dalším řešení. Problém je také navržení způsobu korelace záznamů reverzních proxy serverů, které nebylo vůbec vytvořeno.

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.

Rychlost zpracování záznamů aplikace není nijak zvláštní. Je vidět, že si aplikace drží konstantní běh pro různý počet záznamů, při ztrojnásobení počtu záznamů se čas běhu zvýšil pouze 1,5 krát. Ukázalo se tedy, 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. Byl definován pojem korelace síťových toků jako takový. Dále 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. Testy probíhaly za prvé pomocí dříve vytvořené datové sady a za druhé pomocí dalších dat získaných pro účely testování. Vyhodnocování proběhlo za využití záznamů provozu proxy serverů.

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 nedostatečné a bude potřeba ho dále vyvíjet.

V porovnání s jinými řešeními podobných problémů korelací síťových toků (Coufal [10]) 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. Rozhodně lze pokračovat ve vývoji řešení pro HTTP komunikaci bez TLS protokolu anebo pro reverzní proxy servery. Vytvořená aplikace je lehce rozšířitelná a proto je také možné ji dále vyvíjet a zpřesňovat metody pro určování korelací.

Literatura

[1] *Feature: Customizable Log Formats*. [cit. 26.7.2022]. Dostupné z: https://wiki.squid-cache.org/Features/LogFormat.

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

[3] *Proxy servers and tunneling*. [cit. 19.7.2022]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Proxy_servers_and_tunneling.

[4] *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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

[12] 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.

[13] Fielding, R. T. a Reschke, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [RFC 7231]. RFC Editor, červen 2014. DOI: 10.17487/RFC7231. Dostupné z: https://rfc-editor.org/rfc/rfc7231.txt.

[14] Gopularam, B., Dara, S. a Nalini, N. Experiments in Encrypted and Searchable Network Audit Logs. In:. únor 2015. DOI: 10.1109/EITES.2015.13.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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

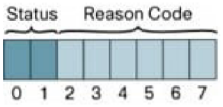
HM			used for sampled NetFlow: 0x01 Deterministic Sampling ,0x02 Random Sampling
FLOW_ACTIVE_TIMEOUT	36	2	Timeout value (in seconds) for active flow entries in the NetFlow cache
FLOW_INACTIVE_TIMEOUT	37	2	Timeout value (in seconds) for inactive flow entries in the NetFlow cache
ENGINE_TYPE	38	1	Type 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**