VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NÁVRH METOD PRO VIZUALIZACI ŠIFROVANÉHO PROVOZU
DESIGN OF METHODS FOR ENCRYPTED TRAFFIC VISUALIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Pavla Hlučková**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Lukáš Malina, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Studentka: Pavla Hlučková ***ID:*** 203171

Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Návrh metod pro vizualizaci šifrovaného provozu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku monitorování šifrovaného provozu (bez dešifrování), zejména pomocí tzv. rozšířených flow záznamů. Vytvořte datovou sadu sestávající se ze zachyceného provozu různých šifrovaných protokolů v různých situacích a s různými nastaveními (např. přístupy na různé weby z různých OS a prohlížečů). Vyberte vhodné statistiky či jiná meta-data, které lze o šifrovaném provozu změřit, a navrhněte vhodné způsoby jejich vizualizace. Cílem vizualizace by mělo být odhalení jakýchkoliv zajímavých vzorů, podobností, či jiných vlastností odhalujících něco o obsahu či druhu provozu, a to buď za účelem podpory dalších prací zabývajících se detekcí bezpečnostních problémů v síti, či pro didaktické účely. Součástí práce je vytvoření datové sady a počáteční experimenty s jejich analýzou, především implementace výpočtu vhodných statistik popisujících jednotlivá spojení. Cílem bakalářské práce je návrh, implementace a dokumentace konkrétních metod vizualizace.

DOPORUČENÁ LITERATURA:

[1] PFLEEGER, Charles P.; PFLEEGER, Shari Lawrence. Analyzing computer security: a threat/vulnerability/countermeasure approach. Prentice Hall Professional, 2012.

[2] GARCIA-TEODORO, Pedro, et al. Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security, 2009, 28.1-2: 18-28.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: doc. Ing. Lukáš Malina, Ph.D.
Konzultant: Václav Bartoš (CESNET, z. s. p. o.)

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato práce se zabývá návrhem metod pro vizualizaci šifrovaného provozu. Obecně popisuje vybrané protokoly šifrovaného provozu, jejichž vzorky jsou následně zachytávány do datové sady. Práce se podrobněji zaměřuje na problematiku monitoringu síťového provozu za využití IP toků a popisuje způsob, jak lze tento monitoring provádět. Významnou součástí práce jsou vytvořené datové sady ze zmíněných protokolů, dále návrhy vizualizací různých statistik a metadat, která jsou zjistitelná z (rozšířených) IP toků těchto protokolů. Metody vizualizace jsou implementovány pomocí programovacího jazyka Python a technologie Jupyter Notebook.

KLÍČOVÁ SLOVA

Cisco Joy, datová sada, flow monitoring, IP tok, Jupyter Notebook, metadata, metody vizualizace, Python, statistiky, šifrovaný provoz

ABSTRACT

This thesis deals with design of methods for encrypted traffic visualization. It generally describes selected encrypted traffic protocols, whose data samples were collected later on to form a dataset. Furthermore, it focuses on the topic of IP flow monitoring and decribes the means of carrying out such monitoring. An important part of this thesis is the dataset created from the samples of mentioned protocols and the visualizations of different statistics and metadata gatherable from (extended) IP flows of these protocols. The designed methods of visualization are implemented using the Python programming language and the Jupyter Notebook technology.

KEYWORDS

Cisco Joy, dataset, flow monitoring, IP flow, Jupyter Notebook, metadata, methods for visualization, Python, statistics, encrypted traffic

HLUČKOVÁ, Pavla. *Návrh metod pro vizualizaci šifrovaného provozu*. Brno, 2020, 72 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Lukáš Malina, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Návrh metod pro vizualizaci šifrovaného provozu“ jsem vypracovala samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autorky

PODĚKOVÁNÍ

Ráda bych poděkovala vedoucímu bakalářské práce panu doc. Ing. Lukáši Malinovi, Ph.D. a panu konzultantovi Ing. Václavu Bartošovi, Ph.D. ze sdružení CESNET, z. s. p. o. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

Obsah

Úvod a motivace **11**

1 Šifrování **12**
1.1 Hyper Text Transfer Protocol Secure . . . 13
1.1.1 TLS 1.2 Handshake . . . 14
1.1.2 TLS 1.3 Handshake . . . 17
1.2 Secure Shell . . . 19
1.2.1 Bezpečnost . . . 21
1.3 Domain Name System over HTTPS . . . 22
1.4 Simple Mail Transfer Protocol Secure . . . 24
1.4.1 Explicitní TLS a SMTP . . . 24
1.4.2 Implicitní TLS a SMTP . . . 25

2 Monitoring síťového provozu **27**
2.1 Historie flow exportu . . . 28
2.2 Flow monitoring . . . 28
2.2.1 Pozorování paketů . . . 29
2.2.2 Měřicí a exportní procesy . . . 30
2.2.3 Sběr dat . . . 32
2.2.4 Analýza nasbíraných dat . . . 32
2.3 Rozšířené flow záznamy . . . 34
2.3.1 Rozšíření IP toků o data protokolu HTTP . . . 35
2.3.2 Rozšíření IP toků o data protokolu DNS . . . 36
2.3.3 Rozšíření VoIP toků o data protokolu SIP . . . 36
2.3.4 Rozšíření IP toků o data protokolu SSH . . . 37
2.3.5 Rozšíření IP toků o detailnější data protokolu TLS . . . 37
2.4 Nástroj Cisco Joy . . . 39

3 Tvorba datových sad **40**
3.1 Zachycení paketů . . . 40
3.1.1 Datová sada protokolů HTTP/S . . . 40
3.1.2 Datová sada protokolu SSH . . . 41
3.1.3 Datová sada DNS over HTTPS . . . 42
3.1.4 Datová sada SMTPS . . . 42
3.2 Převod na rozšířené flow záznamy . . . 43
3.2.1 Zpracování HTTP vzorků . . . 44
3.2.2 Zpracování HTTPS vzorků . . . 44

3.2.3 Zpracování SSH vzorků 45
3.2.4 Zpracování DNS over HTTPS vzorků 45
3.2.5 Zpracování SMTPS vzorků 46
3.2.6 Sekvence délek paketů/záznamů a časů mezi nimi 46

4 Metody vizualizace šifrovaného provozu 48
4.1 Jupyter Notebook 48
4.1.1 Instalace 49
4.1.2 Užívání vytvořených notebooků 49
4.2 Notebook a knihovna k porovnání toků 50
4.2.1 Ukázky vizualizací 51
4.3 Statistický notebook a knihovna 56
4.3.1 Ukázky vizualizací 56
4.4 Příklad interpretace grafů k detekci zajímavostí 60

Závěr 62

Literatura 63

Seznam symbolů, veličin a zkratek 68

Seznam příloh 70

A Schéma 71

B Obsah přiloženého CD 72

Seznam obrázků

1.1 Schéma symetrické kryptografie . . . 13
1.2 Schéma asymetrické kryptografie . . . 13
1.3 Schéma TLS 1.2 handshake . . . 15
1.4 Wireshark: Přístup na webovou stránku přes HTTPS . . . 15
1.5 Dohodnutí cipher suite ve zprávě Server Hello . . . 17
1.6 Schéma TLS 1.3 handshake . . . 18
1.7 TLS extension key_share . . . 18
1.8 Schéma navázání SSH spojení . . . 19
1.9 Wireshark: Navázání spojení s SSH serverem . . . 20
1.10 Schéma vzniku SSH paketu . . . 22
1.11 Zjednodušené schéma DoH komunikace . . . 23
1.12 Schéma navázání SMTP spojení za využití STARTTLS . . . 25
1.13 Wireshark: Navázání spojení se SMTP serverem za využití STARTTLS 25
2.1 Schéma různých způsobů flow monitoringu . . . 29
4.1 Nastavení základních parametrů k načtení . . . 49
4.2 Nastavení proměnných a zobrazení grafu . . . 50
4.3 Počet bajtů . . . 53
4.4 Počet paketů . . . 53
4.5 Délka trvání toku . . . 53
4.6 Distribuce bajtů souboru . . . 53
4.7 Průměrná distribuce bajtů . . . 53
4.8 Porovnání distribucí bajtů . . . 53
4.9 Heat mapy distribucí bajtů . . . 54
4.10 Užitá cipher suite a další varianty . . . 54
4.11 Sekvence délek paketů/záznamů a mezer mezi nimi . . . 54
4.12 Sekvence délek paketů a mezipaketových mezer . . . 55
4.13 Sekvence délek záznamů a mezizáznamových mezer . . . 55
4.14 Rozložení pravděpodobností přenesených bajtů v daných směrech . . 57
4.15 Průměr, směrodatná odchylka přenesených bajtů v daných směrech . 58
4.16 Rozložení pravděpodobností, průměr a směrodatná odchylka přenesených paketů . . . 58
4.17 Rozložení pravděpodobností délek toků . . . 58
4.18 Průměr, odchylka toků . . . 59
4.19 Distribuce bajtů . . . 59
4.20 Průměrná velikost paketů a mezipaketových mezer . . . 59
4.21 Průměrná velikost záznamů a mezizáznamových mezer . . . 59
4.22 Úspěšná autentizace . . . 60

4.23 Neúspěšná autentizace 60
4.24 Úspěšná a neúspěšná autentizace s upravenou osou y 61
A.1 Blokové schéma průběhu zpracování dat a vizualizace 71

Úvod a motivace

Trend šifrování síťového provozu je zejména z důvodu ochrany soukromí a citlivých informací již několik posledních let na vzestupu. Proto například roste popularita implementování protokolu HTTPS (*Hyper Text Transfer Protocol Secure*), který primárně přináší důvěrnost, na webové stránky. K říjnu 2019 bylo 79 % přístupů na webové stránky přes webový prohlížeč Firefox zabezpečeno pomocí HTTPS [1]. A ke stejnému datu bylo až 94 % veškerých služeb a provozu společnosti Google šifrováno [2].

Stále více aplikací a služeb využívá šifrování jako hlavní metodu zabezpečení. Škála hrozeb (*threat landscape*) je výrazně měněna tímto zrychleným nárůstem šifrovaného provozu. Pozitiva, která šifrování přináší pro běžného uživatele, jsou totiž jen další možností útočníků, jak ukrýt své aktivity před detekcí specializovanými nástroji.

Identifikovat hrozby skryté v šifrovaném síťovém provozu je nesmírně důležité, je to však i velkou výzvou. Síťový provoz je nutné monitorovat kvůli malware i jiným hrozbám, ale pouze takovým způsobem, který nenarušuje původní ideu implementace šifrování – zaručení důvěrnosti. Antivirové, bezpečnostní a middlebox[1] produkty často zachycují a dešifrují TLS (*Transport Layer Security*) či SSL (*Secure Socket Layer*) komunikaci, aby si udržely viditelnost do síťového provozu [4].

Pro většinu organizací je problematické detekovat škodlivý obsah, který je skryt šifrováním. Nemají potřebné zdroje na implementaci řešení, které by nezpomalovalo jejich síť. Tradiční metody hromadného dešifrování, analýzy a opětovného zašifrování často nejsou schůdné, jak z důvodů výkonnostních, tak peněžních. Mnohdy jsou i nevhodné s ohledem na integritu dat a soukromí uživatelů či zaměstnanců.

Z hlediska této práce je na výše zmíněný postup nahlíženo jako na nevhodný. Tato práce je tudíž věnována neinvazivnímu monitoringu síťového provozu bez dešifrování, za udržení původně zamýšlené důvěrnosti dat.

Namísto sledování jednotlivých paketů bude aplikováno sledování celého komunikačního řetězce (jednoho spojení) mezi dvěma danými stranami, tzv. *IP flow* či IP tok. Základnímu šifrovanému toku je možné přiřadit další kontextuální informace získané například z vázaných DNS (*Domain Name System*), TLS či HTTP flow dat. Takový tok se nazývá rozšířený.

Cílem této práce je návrh metod vizualizace, které by mohly pomoci k odhalení jakýchkoliv zajímavostí zjistitelných z toků šifrovaného provozu, zejména HTTPS, SSH (*Secure Shell*), SMTP zabezpečeného pomocí TLS (*Simple Mail Transfer Protocol*) a DNS over HTTPS.

[1] Síťová zařízení plnící jakékoliv jiné funkce než je běžné a standardní chování IP routeru na cestě datagramu od zdrojového hosta po destinaci [3].

1 Šifrování

Tato podkapitola je věnována shrnutí základů kryptografie. Není cílem jít do podrobností, nýbrž všeobecně nastínit problematiku. Skutečnosti důležité k praktické části práce budou probrány do větších detailů v relevantních podkapitolách. Protože sbírané datové sady sestávají z dat protokolů HTTPS, SSH, SMTP s TLS a DNS over HTTPS, budou tyto protokoly probrány podrobněji. Ve všech případech je důraz kladen na navázání spojení, které do určitého bodu není šifrované a odkud je při flow analýze možné získat nejpodstatnější informace o komunikaci.

Kryptologie, jako vědní obor o šifrování a dešifrování, v sobě zahrnuje kryptografii a kryptoanalýzu. Kryptografie se zabývá šifrováním zpráv, zatímco kryptoanalýza se věnuje lámání šifer.

Technologie šifrování je považována za nejspolehlivější metodu zabezpečení informací. V určitých formách existuje již od antiky, kdy vyspělejší národy užívaly podobné metody k udržení svých dokumentů a důležitých zpráv v tajnosti, kdyby přišly do rukou neautorizovanému člověku.

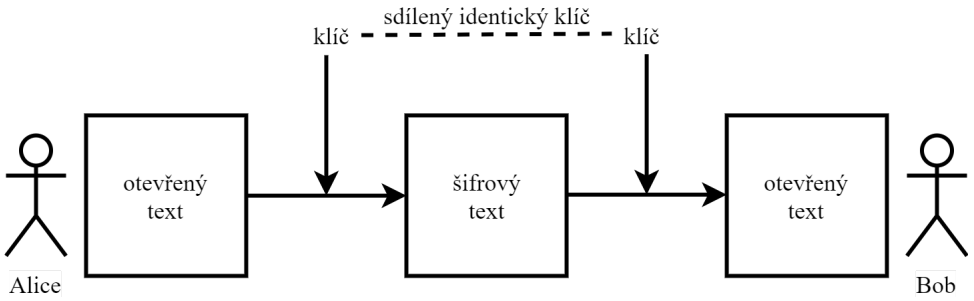
Mezi základní služby, které kryptografie zaručuje, patří:

- **důvěrnost** – utajení obsahu zprávy,
- **autentizace** – ověření totožnosti komunikujících na základě znalosti, vlastnictví či vlastností,
- **integrita** – zamezení neoprávněné modifikace dat (výmaz, úprava, vložení, náhrada), dále schopnost detekce neoprávněné manipulace s daty,
- **nepopiratelnost** – potvrzení původnosti dat.

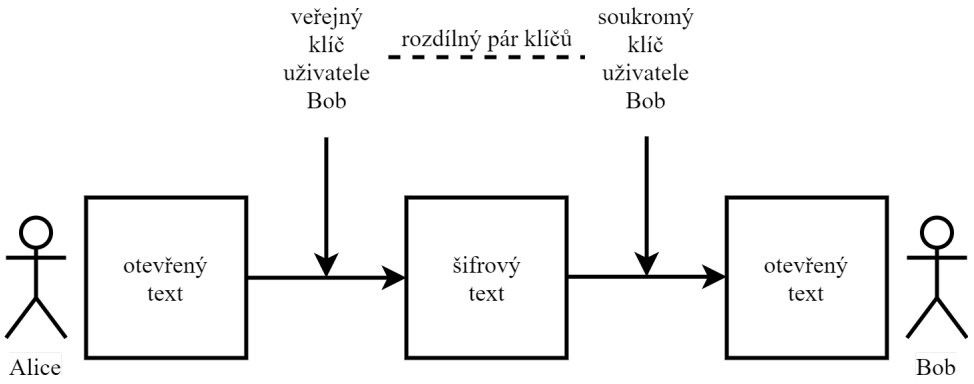
Kryptografické šifry jsou děleny na symetrické, asymetrické a hashovací funkce.

V rámci **symetrické kryptografie** je užíván jeden identický klíč k šifrování i dešifrování (obr. 1.1). Výhodou tohoto přístupu bývá rychlost šifrování, tudíž je lepší při šifrování větších objemů dat. Nevýhoda vyplývá ze sdíleného klíče, který musí být znám oběma stranám komunikace. Jedná se tedy o přenos tohoto utajeného klíče nějakým bezpečným kanálem. Existují dva typy symetrických šifer, a to šifry proudové a blokové. Mezi proudové šifry patří např. RC4, A5/1, ChaCha. Mezi blokové patří např. DES, AES, Blowfish.

V **asymetrické kryptografii** je naopak využíván pár klíčů – klíč soukromý a veřejný. Při šifrování je otevřený text šifrován za použití veřejného klíče adresáta, který jej následně dešifruje svým soukromým klíčem (obr. 1.2). Podepisování pomocí asymetrické kryptografie funguje opačným způsobem, odesílatel podepíše pomocí svého soukromého klíče a adresát ověří podpis veřejným klíčem odesílatele. Mezi asymetrické šifry k šifrování patří např. RSA, ElGamal, k podpisu též RSA, DSA a další.



Obr. 1.1: Schéma symetrické kryptografie.



Obr. 1.2: Schéma asymetrické kryptografie.

Hashovací funkce užívají matematické transformace, aby nevratně a jednosměrně zkomprimovaly informace a poskytly digitální otisk. Neužívají žádný klíč. Primárně se používají k zajišťování integrity zpráv. Mezi zástupce patří např. rodina SHA, MD5.

1.1 Hyper Text Transfer Protocol Secure

Hyper Text Transfer Protocol Secure (HTTPS) je zabezpečenou variantou (variantou využívající šifrování) protokolu HTTP. Ten byl navržen především k přenosu webových stránek. Protokol HTTPS se poprvé objevil v roce 1994 díky společnosti Netscape Communications, která využila SSL (*Secure Socket Layer*) spolu s originálním protokolem HTTP. Odtud vzniklo jméno HTTPS – „HTTP over SSL“ či „HTTP Secure“. V dnešní době se k šifrování užívá nástupce SSL *Transport Layer Security* (TLS). Primární cíl užití TLS protokolu je zaručení soukromé komunikace a integrity dat mezi dvěmi aplikacemi. Nejrozšířenější je verze protokolu 1.2, nejnovější verze je však 1.3.

Protokol TLS je dělen na dvě hlavní části, a to „TLS Record Protocol“ a „TLS Handshaking Protocols“ [5, 6].

TLS Record Protocol pracuje na nejnižší vrstvě hned nad protokolem transportní vrstvy (nejčastěji TCP) zaručujícím spolehlivý přenos dat. Zaručuje privátní spojení, kde je užívána symetrická kryptografie k šifrování dat, ale umožňuje použití i bez šifrování. Generovaný klíč je jednorázový a je založen na společném tajemství stanoveném protokolem *Handshake Protocol*. Pakety obsahují i kontrolu integrity ve formě *Message Authentication Code* (MAC), vypočítanou bezpečnými hashovacími funkcemi [6, 7].

TLS Handshaking Protocols zahrnuje tři subprotokoly, které zprostředkovávají dohodu nad parametry komunikace a autentizace. Dále umožňují reportování chybových stavů oběma stranám komunikace. Subprotokol *Change Cipher Spec Protocol* signalizuje přechod mezi šifrovacími strategiemi, tudíž oznamuje protistraně, že následující konverzace již bude zabezpečena za pomoci dohodnutých parametrů. *Alert Protocol* popisuje upozornění a přiřazuje mu závažnost. Pokud je závažnost fatální, může vést k okamžitému ukončení spojení. *Handshake protocol* je na druhou stranu využíván k vzájemné autentizaci serveru a klienta pomocí asymetrické kryptografie, k vyjednání šifrovacího algoritmu a kryptografických klíčů před přijímáním dat aplikace [6, 7].

Handshake je obecný termín adoptovaný informatiky, který v technickém kontextu označuje automatizované vyjednávání o parametrech spojení ještě předtím, než jsou odesílána reálná data komunikace. Vzhledem k tomu, že se v HTTPS komunikaci jedná o jediná nešifrovaná data, která o IP tocích můžeme získat, budou jim věnovány následující podkapitoly.

1.1.1 TLS 1.2 Handshake

TLS 1.2 Handshake obsahuje tři hlavní kroky. Je to vyjednávání o použité šifře, autentizace a výměna/generování symetrického klíče relace. Na obr. 1.3 je znázorněn průběh celého handshake. Reálný průběh handshake je zachycen v programu Wireshark na obr. 1.4.

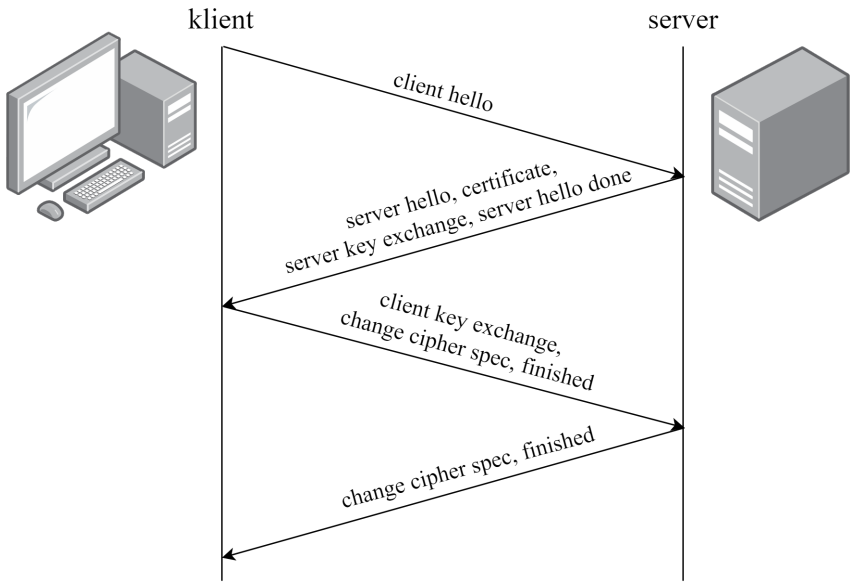
V první zprávě `Client Hello` klient uvádí své schopnosti a nabízí možnosti šifrování (verze TLS, hashovací funkce, podpisové algoritmy, seznam *cipher suites*[1]). Tato zpráva mimo jiné obsahuje i velké náhodné prvočíslo pod názvem `Random`.

Další zpráva pod názvem `Server Hello` přenáší specifické vybrané parametry komunikace, které jsou podporovány jak ze strany klienta, tak ze strany serveru. Server připojí své náhodně vybrané prvočíslo, též pod názvem `Random`. Pokud se

[1]Přednastavená kombinace algoritmu na výměnu klíčů, symetrické blokové šifry na zašifrování proudu dat a hashovací funkce k výpočtu MAC.

klient se serverem neshodnou v žádné z požadovaných vlastností komunikace, je spojení ukončeno. Pod názvem `Certificate` server zasílá svůj SSL certifikát podepsaný certifikační autoritou, u kterého klient ověří legitimitu (digitální podpis, datum expirace, doménové jméno a další data, která mohou potenciálně poukazovat na neplatný certifikát). Zprávy `Server Key Exchange` a `Client Key Exchange` budou probrány níže. Sérii zpráv server zakončuje tzv. `Server Hello Done`, kdy posílá klientovi informaci, že již nebude odesílat v tento moment další zprávy.

Klientova zpráva `Change Cipher Spec` informuje server, že následující zprávy už budou šifrované a zprávou `Finished` ukončuje svoji část handshake. Server nazpět odpoví stejnými zprávami. Následuje výměna šifrovaných zpráv `Application Data` s daty aplikace, jak název napovídá.



Obr. 1.3: Schéma TLS 1.2 handshake.

tls

No.	Time	Source	Destination	Protocol	Length	Info
4	0.027127	10.0.2.15	195.113.172.12	TLSv1.2	571	Client Hello
6	0.057101	195.113.172.12	10.0.2.15	TLSv1.2	2894	Server Hello
8	0.057333	195.113.172.12	10.0.2.15	TLSv1.2	1605	Certificate, Server Key Exchange, Server Hello Done
10	0.061691	10.0.2.15	195.113.172.12	TLSv1.2	180	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
12	0.063125	10.0.2.15	195.113.172.12	TLSv1.2	413	Application Data
14	0.077997	195.113.172.12	10.0.2.15	TLSv1.2	328	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
15	0.080250	195.113.172.12	10.0.2.15	TLSv1.2	2205	Application Data, Application Data

Obr. 1.4: Wireshark: Přístup na webovou stránku přes HTTPS a TLS 1.2.

Co se týká `Server Key Exchange` a `Client Key Exchange`, rozlišujeme v podstatě dva typy TLS handshaků. Jedná se o handshake s RSA autentizací a handshake, kde je na výměnu klíčů použit Diffie–Hellmanův algoritmus. Ten ale sám

o sobě autentizaci nezaručuje, proto je užívaný v kombinaci s nějakým algoritmem digitálního podpisu, většinou ECDSA (DSA na eliptických křivkách), či RSA.

První ze zmíněných handshaků řeší výměnu klíčů následujícím způsobem [8]:

1. výměna zmíněných prvočísel pod názvem `Random`,
2. klient vygeneruje „pre-master secret“ a zašifruje jej veřejným klíčem serveru, tímto zároveň testuje, zda má server opravdu i příslušný privátní klíč k advertizovanému veřejnému klíči,
3. server dešifruje pre-master secret a po sledu operací obě dvě strany komunikace derivují klíč relace „session key“.

Zatímco handshake pomocí Diffie–Hellmanova algoritmu probíhá následně [8]:

1. výměna zmíněných prvočísel pod názvem `Random` – x a y,
2. obě dvě strany si zvolí vlastní pre-master secret (a pro klienta a b pro server) a spočítají operaci $x^{\text{secret}} \bmod (y)$, čísla a a b zůstanou utajena a nikdy neprojdou komunikačním kanálem,
3. strany si navzájem zašlou vypočtená čísla – A a B,
4. klient vypočítá stejný session key z $B^a \bmod (y)$ a server naopak z $A^b \bmod (y)$.

V aktuální nejvyšší verzi TLS 1.3 je RSA handshake výhradně nahrazen handshakem s Diffie–Hellmanovým protokolem. Diffie–Hellmanův protokol obsahuje možnost využití tzv. „*ephemeral keys*“[2] a dosahuje tím dopředné bezpečnosti (*Perfect Forward Secrecy*). Díky této výhodě nemohou být klíče daného spojení nijak ohroženy nebo zneužity ani v případě, že by byl privátní klíč serveru zkompromitován (např. zranitelnost *Heartbleed* v kryptografické knihovně OpenSSL [9]). Další výhodou oproti RSA je to, že při užití RSA handshake existuje nezanedbatelná možnost uhodnutí použitého paddingu, tím pádem zjištění pre-master tajemství a vypočítání klíče [10].

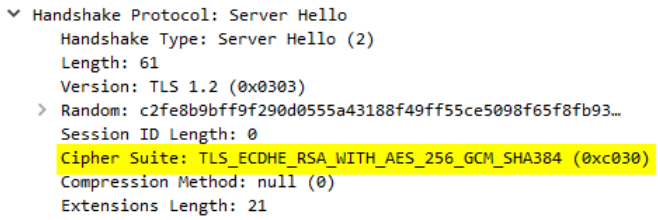
TLS 1.2 podporuje celkem 37 možných variant *cipher suites*. Zástupci jsou například [6]:

- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

Jednotlivé zkratky oddělené podtržítkem po řadě znamenají protokol, algoritmus výměny klíče, autentizační algoritmus, algoritmus symetrického šifrování spolu s délkou klíče v bitech a užitým módem blokové šifry. Posledním parametrem je hashovací algoritmus.

V konkrétním příkladu přístupu na stránku znázorněném na obr. 1.4 bylo využito kombinace zvýrazněné na obr. 1.5. Je zde vidět i zmiňované prvočíslo `Random` zaslané serverem.

[2] Jedná se o jednorázové klíče generované pro každé spojení. Snižují tak jakoukoliv možnost zpětného dešifrování konverzací při vyzrazení či uhádnutí klíče.



Obr. 1.5: Dohodnutí cipher suite ve zprávě Server Hello.

1.1.2 TLS 1.3 Handshake

TLS 1.3 Handshake je kratší než handshake předchozí verze TLS. V podstatě se jedná pouze o délku 1 RTT (*Round Trip Time*[3]). Obsahuje stejné kroky jako handshake verze 1.2, avšak díky podstatnému snížení počtu podporovaných *cipher suites* (z 37 na 5) umožňuje zkrácení délky handshake, tudíž měnší latenci komunikace.

K první zprávě klient kromě `Client Hello` a `Supported Cipher Suites` připojuje i odhad, která kombinace protokolu výměny klíčů a autentizace bude použita. Podle toho rovnou sdílí i další relevantní informace v nově přidané podsekci `key_share` (obr. 1.7) v sekci `TLS extension`.

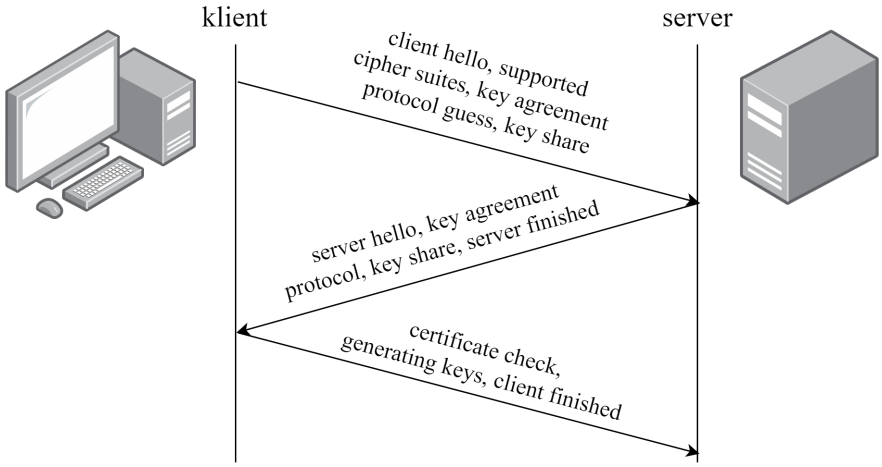
Následuje odpověď od serveru, který zašle svůj certifikát. Pokud klient správně uhodl, jaké kombinace chce server k šifrování použít, server pošle svůj `key_share` a následně jsou všechny ostatní zprávy zašifrované. Server spočítá klíč relace (session key) a zašle `Server Finished`. V tento moment mají obě strany všechno potřebné. Klient si ověří certifikát a vypočítá svůj klíč. Následně oznámí splnění úkonů zprávou `Client Finished`.

Co se týká podporovaných *cipher suites*, existují pouze dvě kritéria k vyjednání, oproti čtyřem ve verzi 1.2. Jedná se o blokovou šifru a tzv. HKDF (*HMAC-based Key Derivation Function*) hash. IETF (*Internet Engineering Task Force*) tímto novým standardem z druhé poloviny roku 2018 zakázala veškeré starší šifry a možnosti jiné výměny klíčů než pomocí algoritmu Diffie–Hellman za použití jednorázových klíčů. Je tak podporována a vynucována dopředná bezpečnost.

TLS 1.3 podporuje celkem 5 možných variant *cipher suites*. Zástupci jsou [7]:

- `TLS_AES_256_GCM_SHA384`
- `TLS_CHACHA20_POLY1305_SHA256`
- `TLS_AES_128_GCM_SHA256`
- `TLS_AES_128_CCM_8_SHA256`
- `TLS_AES_128_CCM_SHA256`

[3]Čas od odeslání paketu, znovu odeslání druhou stranou komunikace, až po jeho opětovné přijmutí.



Obr. 1.6: Schéma TLS 1.3 handshake.

```
▼ Extension: key_share (len=38)
      Type: key_share (51)
      Length: 38
    ▼ Key Share extension
          Client Key Share Length: 36
        ▼ Key Share Entry: Group: x25519, Key Exchange length: 32
              Group: x25519 (29)
              Key Exchange Length: 32
              Key Exchange: 2f350cb6900ab7d5c41b2f60aa567b3f71c8017e86d3b70c...
```

Obr. 1.7: TLS extension key_share.

Význam jednotlivých zkratek je po řadě protokol, AEAD[4] algoritmus (*Authenticated Encryption with Associated Data*) a HKDF hash.

Co se týká autentizace a podpisu v rámci TLS verze 1.3, vyjednávání o jejich metodách bylo odstraněno, aby došlo ke zjednodušení handshake. Metody jsou implementovány na serveru a ten preferovanou metodu pouze sdělí klientovi v příslušné zprávě. Hlavními podpisovými algoritmy jsou RSA, DSA na eliptické křivce (ECDSA) a DSA na Edwardsově křivce (EdDSA).

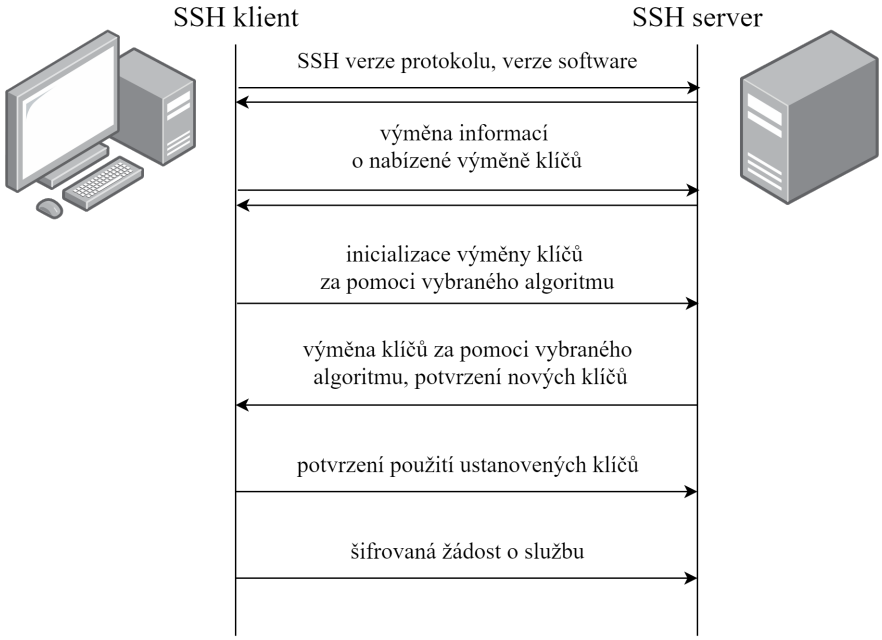
TLS verze 1.3 také přináší jednu další výhodu, jež se týká rychlosti komunikace. Jedná se o tzv. 0-RTT (*Zero Round Trip Time Resumption*) mód, který je volitelný. Je to typ handshake, kdy obě strany komunikace již vlastní předsdílený klíč či tajemství. Tato informace byla získána buď z předchozího plného handshake nebo externě. S první zprávou klient může rovnou zaslat data, která by za normálních podmínek mohl zaslat až po ukončení handshake. Tato data jsou zašifrována pomocí předsdíleného klíče a díky tomu klient zároveň autentizuje server již v první

[4]Asociovaná data jsou nešifrovaná data vázaná k datům šifrovaným (např. hlavička paketu je vázána k šifrovanému payloadu paketu). Tento algoritmus zaručí integritu a autentičnost i asociovaným datům [11].

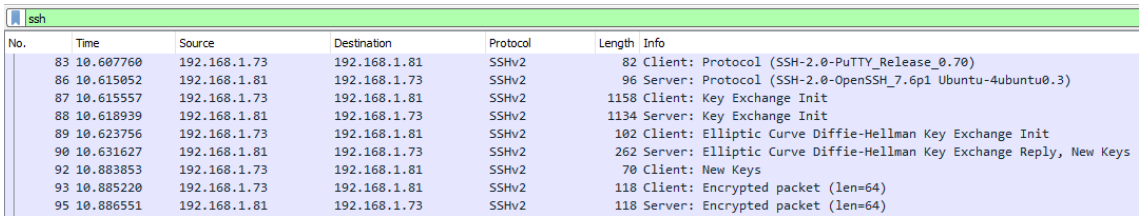
zprávě. Tento typ handshake však přináší kromě výhod v latenci i nějaká negativa. 0-RTT mód totiž nezaručuje dopřednou bezpečnost, protože klíče užité k šifrování jsou derivovány z předsdíleného klíče. Další nevýhodou je pak skutečnost, že je tento protokol zranitelný na útoky typu *replay*. Útočník může zachytit první zprávu klienta a poslat ji znovu serveru. Když nejsou správně implementovány bezpečnostní mechanismy, může ji server vyhodnotit jako validní [7]. Existují ale způsoby, jejichž užitím lze tomuto problému předejít. Jedná se o implementování mechanismů aplikační vrstvy, nebo přístup společnosti Cloudflare, kdy server odpovídá pouze na zprávy 0-RTT typu `GET` bez parametrů, jelikož výsledek těchto žádostí by měl být pokaždé totožný a nezávislý na stavu serveru. Dalším možným mechanismem ochrany je přidání unikátní hlavičky každé 0-RTT zprávě [12].

1.2 Secure Shell

Secure Shell (SSH) je síťový protokol, který umožňuje bezpečné připojení, přihlášení, vzdálené vykonávání terminálových příkazů a další služby skrze nezabezpečenou síť. Nabízí i volitelnou kompresi (`zlib`). SSH či SSH verze 2 částečně nahradilo dřívější protokoly Telnet, Rlogin (*remote login*) a Rsh (*remote shell*), kde byly veškeré informace přenášeny přes síť nešifrovaně. Navázání spojení klienta a SSH serveru je znázorněno na obr. 1.8. Reálné zachycení komunikace z datové sady je zobrazeno na obr. 1.9 v programu Wireshark.



Obr. 1.8: Schéma navázání SSH spojení.



ssh

No.	Time	Source	Destination	Protocol	Length	Info
83	10.607760	192.168.1.73	192.168.1.81	SSHv2	82	Client: Protocol (SSH-2.0-PuTTY_Release_0.70)
86	10.615052	192.168.1.81	192.168.1.73	SSHv2	96	Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3)
87	10.615557	192.168.1.73	192.168.1.81	SSHv2	1158	Client: Key Exchange Init
88	10.618939	192.168.1.81	192.168.1.73	SSHv2	1134	Server: Key Exchange Init
89	10.623756	192.168.1.73	192.168.1.81	SSHv2	102	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
90	10.631627	192.168.1.81	192.168.1.73	SSHv2	262	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
92	10.883853	192.168.1.73	192.168.1.81	SSHv2	70	Client: New Keys
93	10.885220	192.168.1.73	192.168.1.81	SSHv2	118	Client: Encrypted packet (len=64)
95	10.886551	192.168.1.81	192.168.1.73	SSHv2	118	Server: Encrypted packet (len=64)

Obr. 1.9: Wireshark: Navázání spojení s SSH serverem.

Samotný protokol SSH se skládá ze tří základních složek; protokolu transportní vrstvy (*The Transport Layer Protocol*), vrstvy autentizačního protokolu (*The User Authentication Protocol*) a protokolu vrstvy spojení (*The Connection Protocol*) [13].

Transport Layer Protocol zajišťuje a umožňuje autentizaci serveru, důvěrnost a integritu. Pracuje nad protokolem TCP, ale může fungovat i nad dalšími protokoly zajišťujícími spolehlivý přenos dat. Cílem bylo vytvoření jednoduchého flexibilního protokolu, který by umožňoval vyjednávání parametrů a současně by minimalizoval počet RTT. Nejpravděpodobnější počet RTT jsou dva, maximálně se jedná o tři. Dojde při nich ke kompletní výměně klíčů, určení identifikátoru relace (*session id*), autentizaci serveru, požadavku na službu a přijmutí tohoto požadavku [14].

User Authentication Protocol pracuje nad protokolem transportní vrstvy SSH a spoléhá se, že tento protokol pro něj zajišťuje integritu a důvěrnost. Umožňuje autentizaci strany klienta serveru.

Server započne autentizaci tím, že sdělí klientovi, které autentizační metody jsou implementovány a mohou být použity. Server má tudíž kontrolu nad autentizací, ale klient má dostatečné možnosti výběru a může zkoušet všechny nabídnuté možnosti autentizace v jakémkoliv pořadí.

Autentizační metody jsou určené svými názvy. Existuje i metoda „*none*“, která ale nesmí být advertizovaná serverem. Tato metoda však může být požadována ve zprávě odeslané klientem a server ji musí vždy odmítnout, pokud není cílem povolit klientovi přístup bez autentizace. Důvodem odesílání zprávy tohoto tvaru je získání seznamu možností autentizace podporovaných serverem.

Server také mívá určité časové okno pro autentizaci a měl by ukončit spojení, pokud autentizace nebyla provedena v daném okně. Vedle časového okna by měl být limitován i počet neúspěšných přihlášení v rámci jednoho spojení [15].

Žádosti o autentizaci odeslané klientem mohou vyvolat další nadbytečnou výměnu zpráv. Klient může v jakémkoliv čase vyslat zprávu o změně metody autentizace (`SSH_MSG_USERAUTH_REQUEST`), server je nucen zahodit předchozí pokus o autentizaci a pokračovat ve výměně zpráv. Základní metody autentizace, které implementace nabízí, jsou `publickey` (autentizaci pomocí páru asymetrických klíčů),

`password` (autentizaci na základě znalosti hesla), `hostbased` (neinteraktivní forma autentizace, kdy je veřejný klíč hosta uložen na klientově straně) a výše zmíněná metoda `none` [15].

Connection Protocol pracuje nad oběma již zmíněnými vrstvami. Nabízí interaktivní relace. Relace je v tomto kontextu vzdálené spuštění programu. Program může být například shell, aplikace, systémový příkaz nebo nějaký vestavěný podsystém. Každá ze stran komunikace může otevřít tzv. kanál (*channel*). Kanály jsou označeny číslem. Jeden kanál je např. jedna terminálová relace. Vícero kanálů je multiplexováno do hlavního spojení. Díky tomuto protokolu umožňuje SSH i přesměrování TCP portů a X11 spojení[5] [16].

1.2.1 Bezpečnost

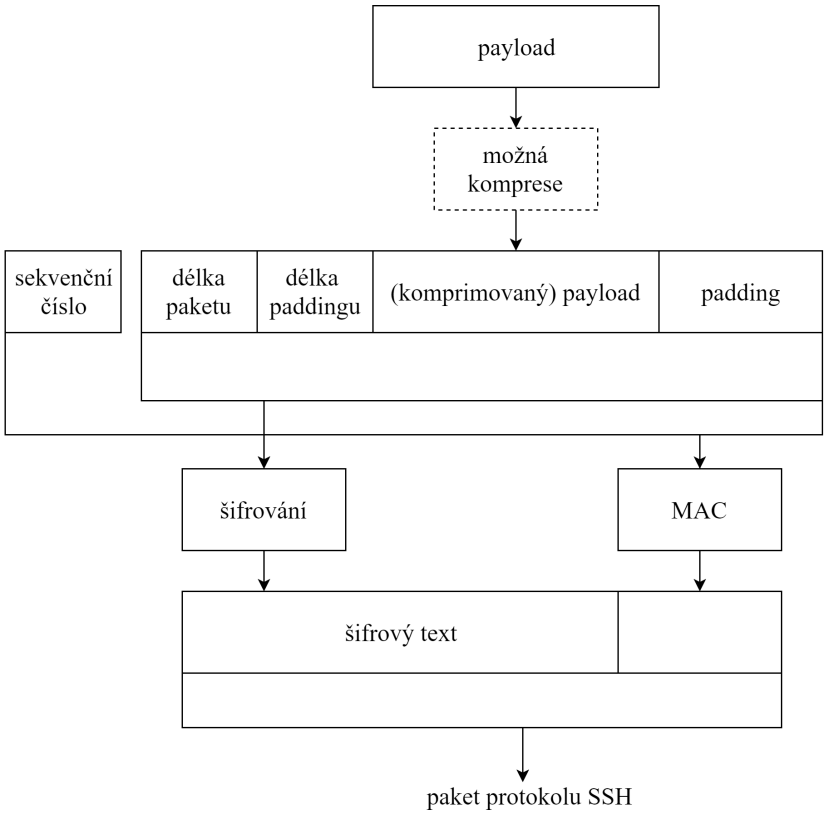
Každý klíč spojení (*session key*) je derivován z hashe, ve kterém jsou zahrnuty i data specifická pouze pro danou relaci. Tento klíč je tedy pevně spojen pouze s danou jedinečnou relací. Implementace protokolu musí zaručovat, že i když nejsou data náhodná, ale pouze pseudonáhodná, musí být kryptograficky dostačující s dostatečným množstvím entropie (např. parametry `Random` užívané Diffie–Hellmanovým protokolem). Na obr. 1.10 je znázorněna tvorba šifrovaného SSH paketu.

Co se týká algoritmů samotných, standard doporučuje pouze několik a důrazně apeluje na implementujícího, aby zkontroloval sílu doporučených algoritmů, popř. aby implementoval algoritmy nové. Ve standardu se doporučují k šifrování blokové šifry (3DES, Blowfish, Twofish, AES) v operačním módu CBC (*Cipher Block Chaining mode*). CBC funguje na principu, že šifrový text z bloku číslo $i - 1$ je užitý jako inicializační vektor v bloku i. Útočník pak může díky poměrně nezanedbatelné míře odhadnutelnosti odhalit prvních 14 či 32 bitů plaintextu. Proto pro zvýšení bezpečnosti je v modernějších implementacích, např. OpenSSH, využito módu CTR (*Counter mode*), který tuto zranitelnost nemá [17].

Dále je protokol SSH náchylný na útoky typu *man in the middle* (MITM) a také útoky odepřením služby (DoS). Protokol jako takový také nevylučuje možnost existence skrytých kanálů, které mohou být použity pro přenos dat způsobem narušujícím bezpečnostní politiku, tzv. *covert channels*.

Pasivní monitoring a analýza provozu SSH může vyzradit útočníkovi určité informace o protokolu, relaci či uživateli, které by jiným způsobem nemohl získat. Pokud není protokol implementován správně, může analýza provozu SSH relace odhalit například délku hesla a další citlivé informace pomocí velikostí paketů, časů mezi pakety aj. Tyto zranitelnosti jsou při správné implementaci znepodstatněny například náhodnými délkami paddingu [13].

[5] Umožňuje spouštění aplikací s grafickým uživatelským rozhraním na vzdáleném serveru.



Obr. 1.10: Schéma vzniku SSH paketu.

1.3 Domain Name System over HTTPS

DNS over HTTPS (DoH) je prokokol, který umožňuje zabezpečení DNS dotazů pomocí šifrování a snižuje tak náchylnost dotazů na MITM útoky, odposlouchávání a zásahy do integrity přenášených dat.

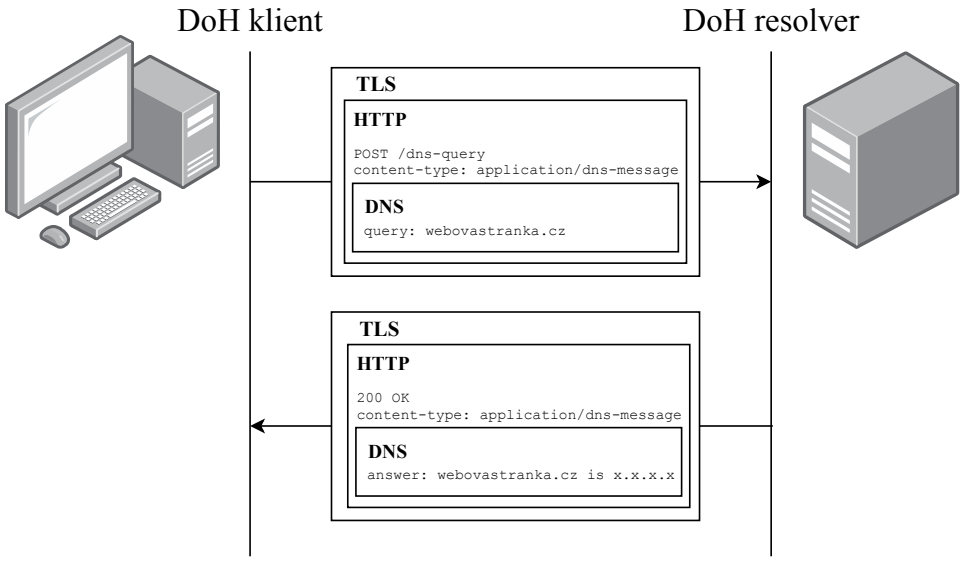
Tento protokol zasílá DNS dotazy a získává DNS odpovědi přes HTTP za využití `https` URI (*Uniform Resource Identifier*), tedy za využití protokolu TLS a jeho výhod ve formě integrity a důvěrnosti přenášených dat (kap. 1.1). DoH klient je nakonfigurován na užití jím specifikovaného URI – může si vybrat konkrétní DoH server k řešení jeho DoH dotazů [18, 19].

DNS dotaz je vložen do HTTP požadavku a zaslán pomocí GET nebo POST metody. Využívá k tomu revizi hypertextového protokolu HTTP/2 či vyšší. DoH klient v požadavku v poli `accept` zasílá informaci, že přijímá typ internetového média `application/dns-message`. Oba způsoby řešení DoH dotazu obsahují pole `method` – GET či POST, `scheme`, které je vždy rovno `https`, `authority` s adresou DoH serveru a `path` s konkrétní specifikací daného DNS dotazu. Metoda POST dále obsahuje pole `content-type` a `content-length`. DoH servery musí implementovat GET i POST metodu řešení HTTP požadavků.

DoH odpověď je vždy typu `application/dns-message`. Obsahuje dvě výše zmíněná pole s informacemi o obsahu. Dále obsahuje pole `status`, kde se vrací stavový kód HTTP, a pole `cache-control` s počtem sekund udávajících dobu, po kterou má být adresa uchována v mezipaměti klienta. Zjištěná IP adresa je přidána na konec záznamu.

Vrácený stavový kód nabývá hodnoty `2xx` (`success`) vždy, když paket obsahuje legitimní a validní DNS odpověď. Týká se to tedy i DNS odpovědí typu `NXDOMAIN`, která značí, že autoritativní server domény nemá v databázi požadované jméno, či `SERVFAIL`, kdy server nemůže z nějakého důvodu na požadavek odpovědět či jej vyřešit. Tudíž i negativní DNS odpověď splňuje podmínky úspěšného stavového kódu HTTP. Jakékoliv ostatní pakety s jinými stavovými kódy HTTP neobsahují žádné DNS odpovědi, ani ty negativní [19].

DoH komunikace je znázorněna ve zjednodušené podobě na obr. 1.11.



Obr. 1.11: Zjednodušené schéma DoH komunikace.

Některé výhody užívání DoH byly zmíněny výše. Existují však i fakta, která se dají považovat za nevýhody. Proti DNS over HTTPS nejčastěji vystupují poskytovatelé internetového připojení (ISP) s argumentem, že bude těžší provádět filtraci a oprávněné zásahy do provozu zejména z právních důvodů, jelikož nebude zachována viditelnost do DNS provozu kvůli šifrování. Bude tedy možné obcházet různé blacklisty, apod. V zemích postižených politickou cenzurou by však tato technologie mohla mít pozitivní dopady. Dalším argumentem proti je fakt, že vyřízení DNS dotazu přes HTTPS může trvat delší dobu než běžné DNS dotazy. ISP totiž většinou mají vlastní DNS servery v síti, tudíž je vzdálenost ke klientovi menší než u DoH, kdy jsou DoH servery ve většině případů umístěny dále od zdroje DNS dotazů. Kvůli

HTTPS protokolu je tu přidána i určitá režie, která není u klasického DNS tak markantní. DNS over HTTPS také oslabuje možnosti monitoringu síťového provozu, kde se užívá dat získaných z DNS (dotaz, odpověď, adresa původu) k zjištění špatných činitelů v síti [20].

Vedle DNS over HTTPS ještě existuje podobný protokol, a to DNS over TLS (DoT). Tento protokol však využívá port 853, zatímco DoH využívá port 443, standard pro šifrovanou webovou komunikaci. Je tedy velmi složité rozlišit DoH toky od běžných webových toků. Tyto toky tudíž nelze jednoduše blokovat bez dotčení veškeré HTTPS komunikace. DoT toky jsou ale jejich specifickým portem prozrazeny, sice nelze zjistit obsah, ale lze je blokovat.

1.4 Simple Mail Transfer Protocol Secure

Simple Mail Transfer Protocol (SMTP) je komunikační protokol aplikační vrstvy, který má za úkol efektivní a spolehlivé přenášení e-mailové komunikace. Primárně funguje nad protokolem TCP, není však vyloučena jeho funkčnost i nad jinými protokoly transportní vrstvy [21].

V kontextu této práce je termínem SMTPS myšlen protokol SMTP zabezpečený šifrováním, ať už pomocí SSL, nebo jeho nástupce TLS. V následujících podkapitolách budou popsány dva hlavní způsoby, které se standardně užívají k zabezpečení SMTP komunikace.

1.4.1 Explicitní TLS a SMTP

Explicitní či oportunistické TLS znamená, že emailová komunikace protokolu SMTP může být povýšena na zabezpečenou komunikaci pomocí protokolu TLS, pokud to server podporuje a pokud nedojde k chybám. Když zmíněné podmínky nejsou splněny, emailová zpráva bude přenesena protokolem SMTP otevřeně, bez šifrování.

Spojení může být povýšeno na šifrované pomocí příkazu `STARTTLS`. Prvně server oznámí klientovi, jaké příkazy a rozšíření podporuje. Podporu `STARTTLS` oznámí pomocí stejně znějícího klíčového slova. Následuje odpověď klienta příkazem `STARTTLS` bez jakýchkoliv parametrů, která značí, že si klient přeje povýšit spojení na zabezpečenou komunikaci. Server odpoví jedním ze tří kódů:

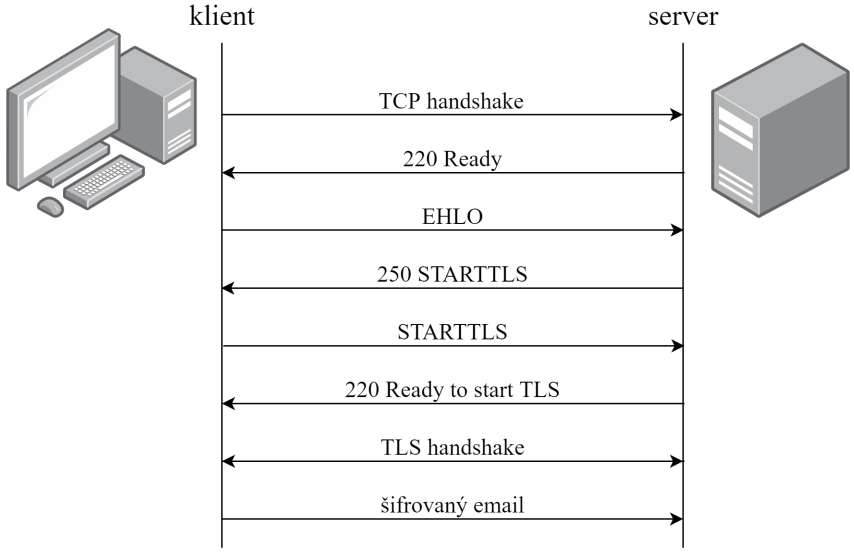
- `220 Ready to start TLS`,
- `501 Syntax error (no parameters allowed)`,
- `454 TLS not available due to temporary reason`.

Pokud je odpověď typu 454, klient se musí rozhodnout, jestli chce emailovou zprávu odeslat i přes to, že spojení nebude šifrované. Po přijmutí odpovědi typu 220 musí klient ihned započnout TLS handshake, před jakýmikoliv dalšími SMTP pakety.

Pokud dojde k nějaké chybě po odsouhlasení užití TLS pomocí `STARTTLS`, spojení musí být ukončeno. Po úspěšném splnění TLS handshake musí klient i server zahodit veškeré informace zjištěné z předchozí nešifrované SMTP komunikace. Klient pokračuje SMTP zprávou s EHLO příkazem, tentokrát již šifrovanou protokolem TLS. Následuje zpráva serveru popisující rozšíření podporovaná serverem, které se mohou lišit od rozšíření advertizovaných před dokončením TLS handshake [22, 23].

Výše popsaný průběh komunikace je znázorněn na obr. 1.12. Reálný průběh komunikace je zachycen na obr. 1.13 v programu Wireshark.

Explicitní TLS k odesílání emailových zpráv nejčastěji využívá port 587 či 2525.



Obr. 1.12: Schéma navázání SMTP spojení za využití STARTTLS.

tcp.port == 587 and (tls or smtp)

No.	Time	Source	Destination	Protocol	Length	Info
4	0.062146	64.233.167.109	10.0.2.15	SMTP	106	S: 220 smtp.gmail.com ESMTP p9sm2416990wrj.29 - gsmtp
6	0.208193	10.0.2.15	64.233.167.109	SMTP	72	C: EHLO [10.0.2.15]
8	0.240199	64.233.167.109	10.0.2.15	SMTP	209	S: 250-smtp.gmail.com at your service, [89.190.46.133] | 250-SIZE 35882577 |
10	0.347400	10.0.2.15	64.233.167.109	SMTP	64	C: STARTTLS
12	0.375662	64.233.167.109	10.0.2.15	SMTP	84	S: 220 2.0.0 Ready to start TLS
14	0.650578	10.0.2.15	64.233.167.109	TLSv1.2	571	Client Hello
16	0.714480	64.233.167.109	10.0.2.15	TLSv1.2	1474	Server Hello
19	0.714682	64.233.167.109	10.0.2.15	TLSv1.2	502	Certificate, Server Key Exchange, Server Hello Done
21	0.771357	10.0.2.15	64.233.167.109	TLSv1.2	147	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
23	0.771955	64.233.167.109	10.0.2.15	TLSv1.2	280	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
24	0.798417	10.0.2.15	64.233.167.109	TLSv1.2	101	Application Data

Obr. 1.13: Wireshark: Navázání spojení se SMTP serverem za využití STARTTLS.

1.4.2 Implicitní TLS a SMTP

Implicitní či vyžadované TLS vynucuje, aby byla e-mailová zpráva přenesena v zašifrované podobě. Proto komunikace ihned po zahájení TCP spojení přechází do TLS,

žádné nešifrované pakety mimo počátku handshake nejsou vyměňovány, na rozdíl od oportunistického TLS a jeho SMTP úvodu. Pokud není server kompatibilní s požadavky nebo dojde k nějaké chybě, nebude emailová zpráva odeslána.

V letech 1997 – 1998 byl port 465 zaregistrován organizací IANA jako `smtps` port, následně byla tato registrace revokována, protože MX (*Mail eXchange*) infrastruktura neumí specifikovat port, tudíž je standardní port 25 užíván automaticky. Registrace dalšího portu byla tedy považována za zbytečné zabírání jednoho ze známých portů a tento port byl přidělen jiné technologii. Port 465 byl ale i po revokaci registrace nadále užíván velkým počtem emailových klientů a software k zabezpečenému odesílání emailových zpráv. I když došlo ke standardizování STARTTLS a portu 587, port 465 byl nadále užíván. Proto organizace IANA přistoupila k výjimce a zaregistrovala roku 2017 TCP port 465 vedle SSM (*Source-Specific Multicast*) alternativně jako `submissions` port. Port 465 nebyl však nikdy oficiálně publikovaný a přijatý organizací IETF. Implicitní TLS k odesílání emailových zpráv i přesto nejčastěji využívá port 465 [22, 24, 25, 26].

2 Monitoring síťového provozu

Návrhy na realizaci monitorování síťového provozu se v uplynulých letech průběžně měnily a rozvíjely. Každý typ monitoringu sloužil k jiným účelům a měl jiné silné a slabé stránky. Obecně by řešení monitoringu síťového provozu mohlo být děleno na aktivní přístup a pasivní přístup.

Monitorování na základě aktivního přístupu znamená generování specifického provozu v síti. Tento provoz slouží k měření různých metrik. Za nástroje aktivního monitoringu lze považovat například `ping` či `traceroute`.

Pasivní přístup k monitoringu naopak žádný provoz negeneruje, pouze sleduje probíhající komunikaci uživatelů. Jedním z nástrojů tohoto typu je např. *(full) packet capture*, kdy jsou zachytávány pakety, dočasně uloženy a analyzovány. V počítačových sítích s vyššími rychlostmi linek je však nutné mít specializované hardwarové komponenty a infrastrukturu, která umožňuje ukládání a následnou analýzu. Toto řešení je také výrazně cenově náročné.

Dalším způsobem pasivního monitoringu je *flow export*. Zde jsou pakety agregovány do „flow“, a to je exportováno k uložení a analýze.

Termín *flow*, často také *IP flow*, *tok* či *IP tok*, je definován jako skupina paketů, které prochází sledovaným bodem v síti během určitého časového intervalu a které mají určitou skupinu stejných vlastností, např. stejné části hlaviček (zdrojová a cílová IP adresa, porty), určité charakteristiky paketů (např. MPLS[1] značky), společnou cestu, jak má být s paketem naloženo (IP adresa dalšího hopu, rozhraní) [28]. V této práci jsou pojmy *flow* a *tok* zaměnitelné a jsou položeny do významu synonym.

Monitorování pomocí flow se vyvinulo od 90. let minulého století ve velmi dobrý nástroj pro kontrolu provozu na sítích, obzvláště těch, které vyžadují vyšší rychlosti linek. Tento typ monitoringu se zaměřuje na datové toky, nikoliv na individuální pakety. Proto je považován za škálovatelnější než tradiční přístupy k monitoringu pomocí analýzy založené na celých paketech. Prvotní práce na toto téma položily základ moderním protokolům jako IPFIX či *NetFlow*.

Mezi další výhody, které flow export přináší, patří to, že je většina síťových prvků již připravena na užívání této formy síťového monitoringu. V roce 2013 to bylo podle průzkumu mezi komerčními a výzkumnými společnostmi až 70 % užívaných zařízení [29].

[1] Metoda směrování síťového provozu ve vysokorychlostních komunikačních sítích pomocí krátkých návěští [27].

2.1 Historie flow exportu

Hlavní idea exportování flow dat se poprvé objevila v publikaci [30] z roku 1991. Organizace IETF se jí sporadicky věnovala v různých pracovních skupinách, ale z této snahy nevzešel žádný standard, hlavně kvůli chybějícímu zájmu výrobců.

Mezitím se společnost Cisco věnovala své technologii exportu flow pod názvem *NetFlow*. Cisco k této technologii mělo blíže hlavně díky jejímu základu v přepínání orientovaném na flow. To funguje způsobem, kde jsou směrovací informace determinovány na základě prvního paketu daného datového toku a uloženy do mezipaměti. Poté jsou všechny zbývající pakety flow podle mezipaměti směrovány stejným způsobem, jako úvodní paket. Když byla objevena hodnota dat uchovávaných v mezipaměti, pro Cisco bylo relativně jednoduché implementovat jejich export a nechalo si *NetFlow* patentovat roku 1996. Jedná se o proprietární protokol kompatibilní s implementacemi třetích stran, jelikož datový formát protokolu je volně dostupný. Všeobecně užívanou a populární se stala až pátá verze protokolu NetFlow, která byla zveřejněna roku 2002. Roku 2004 vyšla verze 9 a roku 2011 techologie *NetFlow-Lite*, která využívá externí zařízení na agregaci paketů, umožňující flow export i na zařízeních, která jej nativně nepodporují.

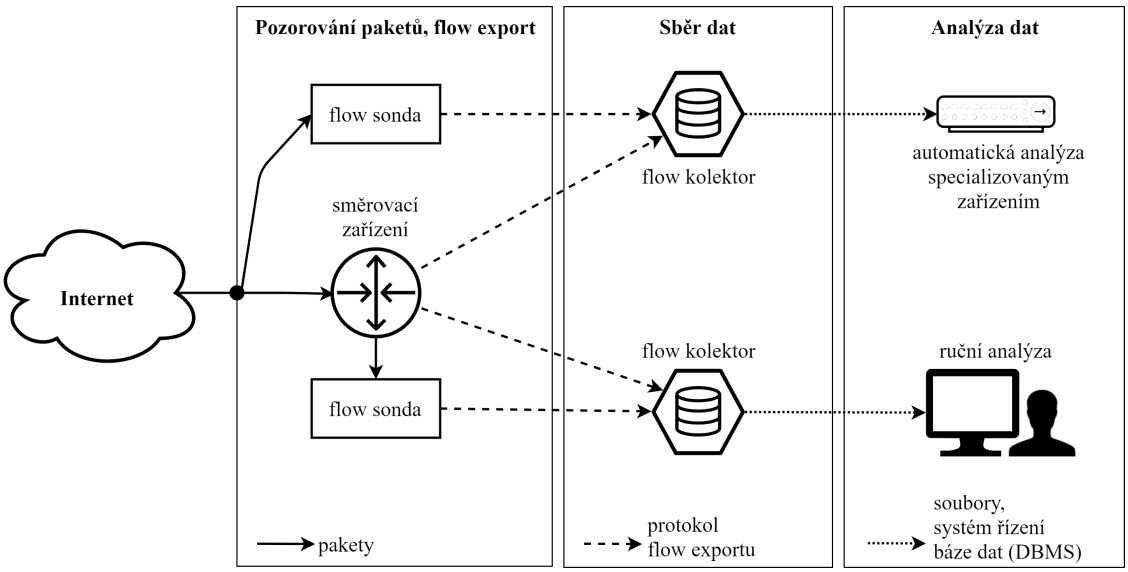
Roku 2004, paralelně s protokolem *NetFlow v9*, se organizace IETF rozhodla standardizovat protokol určený k exportu flow. Byla založena skupina, která sestavila seznam požadavků a zhodnotila nejlepší kandidáty standardu. Standard byl pojmenován *IP Flow Information Export* (IPFIX). Protokol *Netflow v9* byl vybrán, aby položil základy standardu IPFIX. K finalizaci standardu došlo roku 2013 [31, 32].

2.2 Flow monitoring

Monitorování pomocí flow lze obecně rozdělit do následujících procesů:

1. **pozorování paketů**, kdy jsou pakety pozorovány z určitého bodu (např. rozhraní směrujících zařízení) a předzpracovány,
2. **měřicí a exportní procesy**, kdy jsou pakety agregovány do toků a po skončení toku jsou záznamy exportovány datagramy užitého protokolu pro export,
3. **sběr dat**, kdy dochází k příjmu, uložení a předzpracování dat generovaných předchozím stádiem,
4. **analýza nasbíraných dat**, kdy dochází k manuální analýze za cílem odhalení např. určitých nových poznatků ve výzkumu, dohledání informací v případě incidentu či anomálie v běžném provozu, nebo k automatické analýze. Analýza zahrnuje korelaci a agregaci, profilování provozu, klasifikaci a charakterizaci provozu, detekci anomálií a průniku do infrastruktury a v neposlední řadě například i vyhledávání nad archivovanými daty.

Jednotlivé procesy jsou znázorněny na obr. 2.1. Každý z nich je podrobněji popsán v následujících podkapitolách.



Obr. 2.1: Schéma různých způsobů flow monitoringu.

2.2.1 Pozorování paketů

Jedná se o proces, kdy jsou pakety zachytávány a předzpracovávány před dalším použitím. K pozorování dochází z určitého bodu, který může být např. sdílené ethernetové médium, port na routeru nebo skupina rozhraní – fyzických i logických. **Čtení paketů z linky** je obvykle prováděno síťovou kartou. Před uložením do mezipaměti karty musí paket projít sérií kontrol, např. kontrolním součtem.

Dalším krokem je **přidělování přesných časových razítek** paketům (*timestamping*). Tento krok je základem pro správnou finální analýzu provozu, hlavně při dalším zpracování automatizovanými analytickými programy. Razítka se mohou přidělovat buď softwarově, nebo hardwarově. Softwarové orazítkování je běžnější a provádí se prostřednictvím synchronizace za pomoci protokolu *(Simple) Network Time Protocol* ((S)NTP). Softwarové orazítkování má většinou větší zpoždění než jeho hardwarový protiklad. U hardwarové varianty totiž razítkování není zpožděno přeposíláním paketu danému software. Bohužel razítkování založené na hardware ve většině případů vynucuje speciální podporu na síťové kartě, která často není dostupná.

Následuje volitelný krok **vzorkování** paketů. Vzorkování se užívá k optimalizování propustnosti sítě, snížení využití paměti, atd. Je používáno v případech, kdy je očekáváno zahlcení, aby se předešlo ztrátě paketů pro monitoring v co největší míře.

Vzorkování se dělí na systematické a náhodné. Preferovaným přístupem je náhodné vzorkování ke sběru informací o paketech. Cílem je vytvořit co možná nejpřesnější odraz zachyceného provozu pomocí relevantního výběru reprezentativních paketů. Systematické vzorkování totiž může být zkresleno periodickým provozem.

V případě potřeby může následovat stádium **filtrování** paketů, kde jsou pakety deterministicky filtrovány podle vlastností. Existuje filtrování na základě společných vlastností, typicky například podle IP adres nebo aplikací, a filtrování založené na hashích, kdy je hashovací funkce aplikována na určité části paketu a pakety jsou pak přiřazeny vzorku podle specifikovaných hodnot hashů [32].

2.2.2 Měřicí a exportní procesy

Měření a export jsou hlavní části každého systému monitorování síťového provozu pomocí flow záznamů. V tomto stádiu jsou pakety agregovány do flow a flow záznamy jsou exportovány. Starají se o to specializovaná zařízení – *sondy* nebo kompetentní směrovače.

V rámci **měřicího procesu** jsou pakety agregovány na základě tzv. informačních prvků (*information elements*). Mezi typické základní informační prvky patří např. IP adresy, porty, apod. Tento postup vychází ze standardu protokolu IPFIX. *Internet Assigned Numbers Authority* (IANA) v [33] publikuje seznam standardních informačních prvků, organizace si však mohou podle standardu definovat své specifické prvky podle informací, které potřebují. Flow záznam je typicky tvořen množinou (šablonou) informačních prvků. Šablona má ještě navíc přiřazeno své identifikační číslo a délku ve formátu počtu užitých informačních prvků. Za nejmenší vypovídající flow záznam by se dal považovat záznam sestavený z prvků v tab. 2.1. Jedná se většinou o informace ze síťové a transportní vrstvy, trendem posledních několika let je také zakomponovávání informací aplikační vrstvy do flow (tzv. rozšířené flow záznamy), aby bylo možné data podrobněji a přesněji analyzovat (kap. 2.3) [34].

Flow cache obsahuje tabulku se záznamy všech aktuálně probíhajících datových toků v síti. Informační prvky můžou být pro tok buď klíčové nebo neklíčové. Podle klíčových prvků se rozhoduje, zda paket patří k již existujícímu datovému toku, nebo je nutno vytvořit nový datový tok zápisem do tabulky. Běžnými klíčovými informačními prvky, podle kterých se agregují pakety do datových toků, jsou IP adresy obou stran komunikace a čísla portů. Neklíčové jsou pak například počty bajtů a paketů. Tyto informace pak slouží ke sběru charakteristik daného spojení. Jelikož jsou IP adresy stran komunikace dány v klíčových prvcích, flow bývají jednosměrné. Některé modernější implementace flow exportérů však podporují i agregaci obou směrů komunikace do jednoho flow záznamu (tzv. *bidirectional flow*). Za zdrojovou stanici je považován iniciátor komunikace.

Tab. 2.1: Základní šablona flow záznamu.

Záznam IP toku
časové razítko prvního paketu toku
časové razítko posledního paketu toku
IPv4 adresa hosta, který tok započal
zdrojový port
IPv4 adresa hosta, komu je tok adresován
cílový port
identifikační číslo užitého protokolu
počet paketů celého toku
počet bajtů celého toku

Dvousměrné flow jsou zachytávány v datové sadě (kap. 3).

Vzorkování a filtrování flow je dalším krokem před exportováním. Cílem je snížení nároků na proces exportu. Vzorkování funguje stejným způsobem jako u paketů – systematicky (každé n-té flow), nebo náhodně. Filtrování obdobně.

Následují **exportní procesy**. Export vždy probíhá za pomoci specifického protokolu, který je v dané implementaci zvolen pro export.

Zpráva protokolu **IPFIX** má formát sestávající z hlavičky zprávy, kterou následují tzv. *sets*. V textu bude tento pojem užíván ve svém počeštěném tvaru „sety“.

Hlavička IPFIX zprávy sestává z 16 bajtů, kde po dvou bajtech zabírají položky verze protokolu a délka zprávy, zatímco položky čas exportu, sekvenční číslo a *observation domain ID* (identifikační číslo domény, odkud byly pozorovány pakety), mají velikosti po 4 bajtech.

Sety mají tři dané typy a za jednou hlavičkou jich může být připojeno i více než jen jeden. Typy setů jsou:

- **sety šablon**, obsahující šablony, které jsou užity v datových záznamech,
- **sety dat**, ve kterých jsou přenášeny exportované záznamy toků,
- **sety se šablonami možností**, které přenáší informace a meta-data určená kolektorům samotným.

Set má kromě záznamů variabilní délky ještě dvě položky s fixní délkou 2 bajty; identifikační číslo a délku setu [35].

Dalším krokem je takto zkonstruovanou zprávu přenést na daný flow kolektor. Nabízí se tři protokoly transportní vrstvy, a to SCTP (*Stream Control Transmission Protocol*), TCP (*Transmission Control Protocol*) a UDP (*User Datagram Protocol*). Protokol SCTP je nejméně využívaný, jelikož je asi nejnáročnější na implementaci.

Implementace IPFIXu přes TCP je jednodušší, zároveň je úzce spjat s protokolem TLS a tím pádem je nejlepší k exportu datových toků přes Internet. Avšak TCP se nechová požadovaným způsobem při velkém zatížení. Okno TCP přijímače limituje exportní procesy a pokud nejsou dostatečně velké vyrovnávací paměti, může dojít k zahazování paketů na vstupu. Nejvíce užívaným je protokol UDP. Jedná se o nejjednodušeji implementovatelný protokol. Problémem je, že tento protokol nezajišťuje spolehlivý přenos a zároveň postrádá možnost zjištění míry zahlcení sítě nebo mechanismy proti tomuto zahlcení. Přijímač musí být připraven na velké objemy dat (třeba velké množství jednopaketových flow), které mohou být přijímány nárazově např. při určitých typech síťových útoků (útok odepřením služby) [28].

2.2.3 Sběr dat

Exportovaná data z předchozího stádia jsou odesílána specializovanému zařízení s názvem *kolektor*. Toto zařízení má za úkol přijímat exportované datové toky, uložit je a připravit je pro zpracování následnou analýzou dat. Přípravou se rozumí komprese dat, filtrování, anonymizace dat nebo agregace. V kolektoru se o příjem dat starají přijímací procesy a od vybraného modelu sběru se odvíjí výkonnost kolektoru a funkcionalita. Data mohou být automaticky analyzována v reálném čase, pokud jsou uložena v operační paměti, nebo jsou prvně uložena a po určité době opětovně načtena do operační paměti k provedení analýzy.

Podle nároků na vyhledávání, vkládání a místo na disku je vybíráno mezi typy persistentního uložení; prostý datový soubor (binární nebo textový), řádkově orientovaný systém řízení báze dat – DBMS (MySQL, PostgreSQL, ...) či sloupcově orientovaný DBMS (FastBit) [32].

2.2.4 Analýza nasbíraných dat

Analýza dat je posledním krokem monitorovacího modelu. Lze rozlišit mezi čtyřmi základními okruhy podle účelu analýzy.

Sledování využití sítě je asi nejběžnější okruh nabízený veškerým analytickým softwarem. V základu je nabízeno procházení a filtrování exportovaných flow dat. Dále různé statistiky, které je ve většině případů možné si upravit podle potřeby a zájmu (stanice s největším počtem vyslaných a přijatých bajtů, paketů, podle daného protokolu apod.). Statistiky jsou většinou zobrazeny v nějakém přehledném webovém rozhraní na tzv. *dashboardu*. Různé specifické anomálie mohou být následně

analyzovány incident handlery nebo operátory a analytiky SOC[2] (*Security Operations Center*). V neposlední řadě je tu funkce tvoření reportů (*reporting*), kde se vybrané statistiky za dané období například mohou generovat do PDF či zasílat elektronickou poštou.

Monitorování funkce a výkonu služeb na síti (*application performance monitoring*) se zaměřuje na extrahování metrik o spojení z exportovaných flow dat. Typicky se jedná o data jako je latence, RTT, doba odezvy, ztrátovost paketů, využití šířky pásma nebo jitter. Je výhodné tyto metriky získávat z již dostupných dat, která vznikají v důležitých bodech sítě, jelikož pak není nutná instalace specializovaných nástrojů na klientská zařízení. Aplikace využívají tohoto typu monitorování dvěma způsoby. První skupina využívá zaužívaných informačních prvků, které již v daném záznamu datového toku existují z jiných důvodů. Tyto odhady jsou většinou méně přesné. Druhý způsob sestává z užití speciálních rozšíření flow exportérů (sond), aby extrahovaly dodatečné metriky.

Detekce bezpečnostních hrozeb (*threat detection*) přináší analýzu síťové komunikace. Díky velmi specifickým snadno rozpoznatelným signaturám určitých typů škodlivých aktivit umožňuje detekovat útoky typu DDoS (*Distributed Denial of Service*), skenování sítě (např. *port scans*), komunikaci stanic zapojených do botnetu[3] či šíření počítačových červů. K tomuto rozpoznávání se využívá např. identifikace destinace, odkud nebo kam komunikace směřuje. Systémy jsou většinou napojeny na známé databáze IP adres (reputační listy nebo blacklisty), ze kterých byl rozesílán spam, malware, či byly součástmi dalších škodlivých záležitostí, třeba participovali v botnetu. Když jde o komunikaci hosta z vnitřní sítě na IP adresu objevující se v takové databázi, může to znamenat výše zmíněné hrozby, nebo například přítomnost tzv. APT (*Advanced Persistent Threat*). Jsou to moderní způsoby útoků na velké komerční firmy či státem ovládané organizace. Ze strany útočníků je kladen důraz hlavně na utajení přítomnosti této hrozby v rámci organizace, kombinování různých vektorů útoku a dlouhodobé plánování. Proto jsou označeny slovem *persistent*. Tímto dochází k neoprávněnému přístupu útočníků do infrastruktury organizace po dlouhou dobu bez zjištění. Dále se k identifikaci hrozeb může využít definice datového toku jako takového. Jako příklad se nabízí protokol SSH, jelikož je častým cílem slovníkových útoků[4]. I když je SSH komunikace šifrovaná, slovníkový útok má určitá specifika, díky kterým se dá snadno odhalit; například na základě počtu paketů ve flow nebo času mezi datovými toky a mezi pakety. Týmy

[2]Místo, kde jsou podnikové informační systémy (web, aplikace, databáze, datová centra a servery, stanice a další koncové body) monitorovány a bráněny před hrozbami.

[3]Síť počítačů, které jsou infikované specifickým software a jsou řízeny obvykle z jednoho centra. Nejčastěji je síť užívána útočníky k výše zmíněným DDoS útokům nebo rozesílání spamu.

[4]Útok hrubou silou za použití předdefinovaných často užívaných hesel v daném slovníku.

zabývající se monitoringem své sítě jistě využívají funkci upozorňování (*alerting*) na určité anomálie. Systém se je může naučit rozpoznávat pozorováním každodenního provozu a vyhodnocováním odchylek od normálu. Jedná se například o větší než průměrné množství vyměněného provozu, komunikace podezřelým protokolem, na podezřelý port nebo podezřelou aplikací. Cílem je rozpoznat síťové útoky v co možná nejdřívějším stádiu a zabránit jim.

Manuální prohledávání dat je posledním okruhem. Slouží k dohledání konkrétních informací a detailů ohledně incidentů či problémů, které byly již nějakým způsobem detekovány a musí být zpracovány a řešeny. Informace získané tímto způsobem lze dále využít k sestavení určitých specifických behaviorálních vzorů subjektů v monitorované síti a poté k uplatnění takto nadefinovaných vzorů k automatické detekci bezpečnostních hrozeb (eliminace *false-positives*[5]).

2.3 Rozšířené flow záznamy

Tradiční exportované flow záznamy, jak již bylo zmíněno, obsahují data extrahovaná z hlaviček paketů. Jedná se o data ze 3. a 4. vrstvy modelu ISO/OSI; vrstvy síťové a transportní. Konkrétně to jsou IP adresy, porty a další (znázorněno již v tab. 2.1). Dále obsahují i celková data o flow, tedy kolik paketů bylo mezi komunikujícími stranami vyměněno, kolik bajtů bylo transferováno a časová razítka, kdy datový tok začal a kdy byl odpozorován poslední paket tohoto toku.

Rozšířené flow záznamy (*extended flows*) obsahují kromě tradičních informací data o protokolech extrahovaná z aplikační vrstvy modelu ISO/OSI. Speciální sonda totiž dokáže extrahovat a zapracovat i informace z hlaviček daných aplikačních protokolů jako nové informační prvky do flow záznamů. Mezi takové záznamy můžou patřit například doménová jména u protokolu DNS (jsou předmětem méně dynamických změn než IP adresy, umožňují zavedení funkčnějších blacklistů), dále specifické položky z hlaviček HTTP protokolu (pole s `URL`, `User-Agent`, `Content-Type`, `status-code`, `method`, `Server`, přítomnost či absence polí `Cookie` a `Referer`), a nebo např. volaná čísla u protokolu SIP (*Session Initiation Protocol*) [36, 37].

V následujících podkapitolách budou zmíněny publikace, které se zabývají těmito typy rozšířených IP toků. Cílem je nastínit princip a funkcionalitu takového monitoringu a jeho přidanou hodnotu oproti tradičním flow záznamům.

[5]Komunikace, která byla automaticky chybně vyhodnocena jako škodlivá.

2.3.1 Rozšíření IP toků o data protokolu HTTP

Co se týká flow rozšířených o data protokolu HTTP, tomuto tématu se věnuje například vědecká práce Ústavu výpočetní techniky Masarykovy univerzity [38], kde byla potvrzena přidaná hodnota rozšířených IP toků v monitorování. Bylo dokázáno, že klasifikací specifických vzorů chování HTTP provozu je možno detekovat útoky hrubou silou na hesla, spojení s proxy servery, HTTP skenery a *web crawlery*[6]. Díky tomuto přístupu bylo denně detekováno v průměru 16 útoků hrubou silou a 19 skenů sítě, které do té doby nebyly detekované běžnými prostředky. Monitorovaná oblast měla asi 40 tisíc uživatelů a 15 tisíc aktivních IP adres.

V práci autorů ze společnosti Cisco [37] lze také najít kapitolu ohledně tohoto tématu. Bylo zjištěno, že dobrým indikátorem detekce škodlivého toku může být ne/přítomnost určitého pole v HTTP hlavičce. Výsledky jsou výstupem analýzy datové sady o velikosti necelých 3 milionu flow záznamů (1 milion benigních oproti necelým 2 milionům škodlivým). Pro příchozí HTTP toky bylo odhaleno, že přítomnost polí `Server` (nejčastěji s hodnotou `nginx`), `Set-Cookie` a `Location` častěji značí, že tok je škodlivý, oproti polím `Connection`, `Expires`, `Last-Modified`, které byly nejčastěji součástí benigního provozu. Pole `Content-Type` mělo nejčastější hodnotu `image/*` pro benigní provoz, zatímco pro data s malwarovými toky se jednalo o `text/html;charset=UTF-8` a `text/html;charset=utf-8`. Drobné nuance např. v kapitalizaci hrají velkou roli v detekci, a proto by se taková pole neměla normalizovat. Pro odchozí toky bylo zajímavé pole `User-Agent`. V datové sadě obsahovalo tisíce různých řetězců. Pro největší počet malwarových flow mělo hodnotu `Opera/9.50(WindowsNT6.0;U;en)`. Objevovaly se různé varianty kapitalizací písmen v názvu tohoto pole; `User-Agent`, `user-agent`, `User-agent`, `USER-AGENT` a `User-AgEnt`. Pomocí tohoto pole je též možné detekovat nesoulad mezi advertizovaným softwarem v HTTP hlavičce a reálným užívaným software. K tomu se užívá korelace HTTP dat a prohlížeče odhadnutého pomocí TLS metadat a knihoven.

Klasifikace škodlivého provozu v této práci dosáhla vysokých hodnot přesnosti, a to i při minimálním výskytu *false-positives*. Klasifikace pomocí TLS dat samotných měla přesnost (*accuracy*) 96,335 %, resp. 62,857 % pokud je klasifikační mez nastavena tak, aby bylo dosaženo 0 % *false-positive*. Rozšířením TLS flow o data protokolu HTTP byly tyto hodnoty značně zvýšeny na přesnost odhadu 99,955 %, resp. 99,660 %.

[6]Bot, který systematicky prochází veškeré URL adresy na Internetu a indexuje je pro potřeby webových vyhledávačů. Někteří tito boti však nejsou neškodní, mohou spotřebovávat více šířky pásma, než je nutné, krást data a narozdíl od legitimních botů se nevyhýbají např. stránce `robots.txt`, kde webový administrátor definoval adresy, které by legitimní bot neměl indexovat.

2.3.2 Rozšíření IP toků o data protokolu DNS

Zmíněná publikace společnosti Cisco [37] obsahuje i kapitolu zabývající se touto problematikou. Výsledky pochází z datové sady o velikosti 7 milionů škodlivých a 8 milionů benigních DNS dotazů. Autoři využili porovnávání plně specifikovaných doménových jmen (FQDN) oproti jménům generovaným speciálním algoritmem. Tento algoritmus je často zneužíván malwarem. Jednou ze statistik byl počet znaků v doménovém jménu. Pro benigní provoz se jednalo o tvar Gaussovy funkce s nejvyšším počtem doménových jmen o délce $6-7$ znaků. Malwarové toky měly nejčastěji doménová jména o délce 6 znaků, přičemž tento počet značně převýšil všechny ostatní. Další statistiky byly získány z odpovědí na DNS dotazy. Jednalo se například o počet IP adres vrácených v odpovědi. Benigní toky vracely nejčastěji 2 a 8 IP adres, malwarové naopak 4 a 11. Parametr, který se také lišil mezi benigními a škodlivými toky, byla hodnota TTL. Validní odpovědi měly hodnoty 60, 300, 20 a 30. Škodlivé toky nejvíce užívaly hodnotu 100 (okolo 22 % toků), která se v benigních datech vůbec nevyskytovala.

Rozšířením TLS flow o data protokolu DNS dokázala tato publikace zajistit přesnost odhadu 99,883 %, resp. 96,849 % při nulovém počtu *false-positives*. Publikace [37] také kombinací rozšíření IP toků o data obou protokolů HTTP i DNS dosahuje přesnosti odhadu 99,985 %, resp. 99,956 %.

2.3.3 Rozšíření VoIP toků o data protokolu SIP

Běžná flow data nejsou dostačující pro správnou analýzu a detekci hrozeb *Voice over IP* (VoIP) provozu. Rozšíření flow dat o data z hlaviček SIP protokolu však umožňuje některé hrozby snadno detekovat. Práce [39] se zabývá možností detekce zneužití špatně nakonfigurovaných ústředen, kdy je útočník schopen provádět neautorizovaná volání do veřejné telefonní sítě. Úspěšný útok může způsobit značné finanční ztráty majiteli ústředny. Detekce je založena na analýze zpráv `SIP INVITE`, které slouží k navázání spojení. Cílem je zjistit, které IP adresy generují velké množství krátkých zpráv, které se liší pouze prefixem volaného čísla. V publikaci [40] jsou rozebrány další dva typy síťových útoků na SIP server. Jedná se o skenování uživatelských jmen (*extensions*) a hádání hesel. Oba dva útoky se projevují velkým množstvím zpráv, většinou typu `REGISTER`, a odpověďmi serveru `401 Unauthorized`, nebo `404 Not Found`. Detekce je založena na agregování rozšířených toků provozu pomocí IP adresy ústředny, uživatelských jmen a klientských IP adres. Tímto lze odhalit potenciálně škodlivý provoz.

2.3.4 Rozšíření IP toků o data protokolu SSH

V publikaci [41] je rozebrán monitoring pomocí IP toků rozšířených o informace z protokolu SSH. Sonda extrahuje data ze začátku spojení, než se dokončí proces domluvy na šifrování mezi stranami (průběh komunikace byl naznačen v kap. 1.2 a na obr. 1.8). SSH komunikaci lze identifikovat bez ohledu na to, na jakém portu běží. Každý první paket SSH toku totiž obsahuje řetězec `SSH-` na svém začátku. Jakmile je tento paket identifikovaný, sonda extrahuje z každé další komunikace potřebné informační prvky. Je však důležité sledovat i šifrovanou komunikaci, kde dochází k autentizaci, jelikož je nejčastějším terčem útoků. Pro správnou detekci hrozeb je nutné rozpoznat výsledek každého pokusu o autentizaci – jestli byl útok hrubou silou úspěšný, či nikoliv – ale zpráva s touto informací je šifrovaná. Výsledek je však možné rozpoznat podle délky a struktury dané zprávy. Zpráva o úspěšném přihlášení nese pouze identifikační číslo této zprávy. Zpráva o neúspěšném pokusu o přihlášení naopak nese i seznam dalších možných metod autentizace. Z těchto informací se tedy dá pomocí délky zprávy zjistit výsledek pokusu o přihlášení.

2.3.5 Rozšíření IP toků o detailnější data protokolu TLS

Ve výše zmíněných publikacích jde o rozšířená flow data ve smyslu využití dat z aplikační vrstvy. Význam termínu *rozšířené IP toky* lze aplikovat i na toky obohacené o detailnější data ze síťové a transportní vrstvy a o data protokolu TLS, který se nedá jednomyslně zařadit do konkrétní vrstvy modelu ISO/OSI.

V publikaci [42] společnosti Cisco se autoři věnují 18 rodinám malware. Detekce škodlivých toků je tu založena pouze na základě dat získaných z nešifrované části TLS handshaků (kap. 1.1.1), správné klasifikace a strojového učení. Impulsem pro vznik této studie bylo nedostatečné využití potenciálu těchto dat v systémech prevence průniku (*Intrusion Prevention Systems* – IPS) a v jejich pravidlech. V práci je zmíněno, že drtivá většina těchto pravidel je založena pouze na inspekci obsahu paketů. Malé množství pravidel je specifických pro TLS a naprosté minimum je užíváno k detekci malware (pomocí specifických řetězců v certifikátech).

K extrahování informací autoři využili vlastní software – *Cisco Joy* (kap. 2.4). Data, která byla využita k detekci a klasifikaci malware jsou následující:

- **flow metadata** – klasická data zjistitelná o IP tocích; počty bajtů, paketů, zdrojové a cílové porty, trvání flow,
- **sekvence délek paketů a časů** – sekvence, která sestává z délek paketů proložených časy trvání mezery mezi přijímáním příchozích paketů (kap. 3.2.6),
- **distribuce bajtů** – pole o 256 prvcích, kde se udržuje počet výskytů každé hodnoty bajtů obsažených v datovém obsahu paketu,
- **nešifrované části TLS hlaviček**.

Co se týká **TLS klientů**, v rámci benigního provozu byly v téměř 100 % případů nabídnuty dvě varianty *cipher suites*; `TLS_RSA_WITH_AES_128_CBC_SHA` a stejná varianta s `AES_256`. Malwarové toky také v téměř 100 % případů nabízely tři specifické varianty *cipher suites*:

- `TLS_RSA_WITH_3DES_EDE_CBC_SHA`,
- `TLS_RSA_WITH_RC4_128_SHA`,
- `TLS_RSA_WITH_RC4_128_MD5`.

Stojí za zmínku, že *cipher suites* vybírané škodlivým provozem jsou vzhledem ke standardům dnešní doby považovány za slabé.

V rámci zprávy `client hello` jsou advertizovány i rozšíření TLS (`TLS extensions`). U škodlivých toků jde o výrazně menší počet rozšíření – jedno oproti devíti v benigním vzorku provozu. Zmíněné rozšíření má název `signature_algorithms` a jedná se o rozšíření, které je podle standardu nutné implementovat.

Ve zprávě `client key exchange` se u benigního vzorku často objevoval 512 bitový veřejný klíč `ECDHE_RSA`, zatímco škodlivé toky téměř výhradně využívaly 2048 bitového `DHE_RSA` klíče.

Pomocí TLS knihoven došlo k odhadnutí klientského software a bylo zjištěno, že původci škodlivých flow jsou nejčastěji prohlížeče `Opera 12`, `Firefox 46`, `Tor 0.2.`*n* a `Opera 15`. `Firefox 47`, `Chrome 51` či `Safari 9` neměly žádný výskyt v maligním vzorku, ale byly velmi užívané v rámci benigní komunikace.

V rámci **TLS serverů** se zajímavá zjištění týkala např. vybraných variant *cipher suites*, kdy v 90 % případů byly vybírány stejné varianty jako na straně klienta (viz výše). Legitimní benigní komunikace tyto variace téměř nikdy nevybrala.

Co se týká **TLS rozšíření**, škodlivé servery většinou nevybraly žádná, zatímco benigní nejčastěji vybraly rozšíření `renegotiation_info` či `ec_point_formats`.

Byla analyzována i zpráva `certificate`. Benigní provoz v největším počtu směřoval na servery s certifikáty známých stránek (`*.google.com`, `*.facebook.com`). Tyto benigní servery jsou však často užívány i v rámci škodlivých toků; ke kontrole konektivity nebo pro tzv. *command and control*[7]. Malwarové toky nejčastěji obsahovaly certifikáty, jejichž subjekty vykazovaly známky užití algoritmu pro generování domén (např. `33mhwt2j.net`), nebo specificky subjekty `block.io`, `*.wpengine.com` a další. Zajímavými prvky certifikátů jsou položky vypršení validity certifikátu (pro malware nejčastěji 375, 7305 či 6358 dní) a počet alternativních jmen subjektu (pro malware téměř výhradně 3).

V této studii bylo dosaženo za pomoci výše zmíněných dat exportovaných programem *Cisco Joy* přesnosti klasifikace flow a přiřazení k rodině malware 99,6 %.

[7] Řízení skupiny infikovaných stanic v síti – botnetu.

2.4 Nástroj Cisco Joy

Cisco Joy je opensource nástroj pod licencí BSD. Vznikal jako výzkumný projekt. Pod svým názvem skrývá jak knihovnu, tak API. Jedná se o nástroj, který je určen k extrahování specifických informací o datech přímo na určitém rozhraní, buď z probíhajícího provozu, nebo ze souborů se zachycenými pakety (PCAP). Tyto soubory následně převádí do formátu JSON, který je flexibilní a vhodný pro spoustu nástrojů analýzy dat a pro většinu programovacích jazyků. *Joy* podporuje důležité protokoly, jako jsou HTTP, TLS, DNS a DHCP (*Dynamic Host Configuration Protocol*) a zakomponovává rozšířené informace o nich v rámci datového toku do výsledného JSON souboru. Časem se ukázalo, že *Joy* poskytuje velkou přidanou hodnotu zachyceným datům a proto se jej Cisco rozhodlo přepsat do kompatibilnějšího formátu a zaručit podporu IPFIX exportu pro kolektory tohoto typu. Dále podporuje velké množství možných nastavení záchytu a zpracování, proto se jedná o poměrně všestranný nástroj. Podporuje i anonymizaci dat. Možnosti nastavení jsou popsány v dokumentaci na [43].

Výstup programu je ve výchozím nastavení zobrazen v tab. 2.2. Tento objekt je možné rozšiřovat pomocí dalších volitelných parametrů.

Tab. 2.2: Struktura základního JSON objektu.

název prvku	obsah prvku
`sa`	zdrojová IP adresa
`da`	cílová IP adresa
`pr`	číslo protokolu podle standardu organizace IANA
`sp`	zdrojový port
`dp`	cílový port
`bytes_out`	počet přenesených bajtů ve směru od zdroje k cíli
`bytes_in`	počet přenesených bajtů ve směru od cíle ke zdroji, pouze pro dvousměrné toky
`num_pkts_out`	počet přenesených paketů ve směru od zdroje k cíli
`num_pkts_in`	počet přenesených paketů ve směru od cíle ke zdroji, pouze pro dvousměrné toky
`time_start`	časový údaj pozorování prvního paketu daného toku
`time_end`	časový údaj pozorování posledního paketu daného toku
`packets`	pole obsahující údaje o sekvenci délek paketů, směrů a mezipaketových časů ve výchozím nastavení pro prvních 50 paketů toku

3 Tvorba datových sad

Aby bylo možné navrhovat metody vizualizace šifrovaného provozu, prvně je třeba vytvořit různé vzorky provozu a přesně je zdokumentovat. Nad těmito vzorky budou stavěny návrhy vizualizací. Jelikož je nutné přesně vědět, co se v daném vzorku šifrovaného provozu dělo, není možné brát data z reálné sítě nebo data automaticky generovaná. Veškeré použité vzorky byly tudíž ručně generovány a zachytávány, aby byla zajištěna kontrola nad zachycenými daty. V této práci budou zachytávány vzorky provozu protokolu HTTP/S, SSH, SMTPS a DoH.

3.1 Zachycení paketů

3.1.1 Datová sada protokolů HTTP/S

V této datové sadě šlo o porovnání provozu protokolu HTTP k tomu stejnému provozu protokolu HTTPS. Z tohoto důvodu bylo nutné najít webové servery, které nepřesměrovávají z protokolu HTTP na jeho zabezpečenou verzi. Cílem bylo zachytit vzorky reálného provozu, proto nebyl zaváděn vlastní web server a ani nebyl zprovozněn *SSL strip*[1].

Jako cílová destinace byla vybrána stránka `var.liberouter.org`, na kterou se lze připojit jak přes HTTP, tak přes HTTPS. Jelikož se jedná o poměrně statickou stránku, bylo anticipováno, že přenášená data budou malá. Jako druhá webová stránka byl tedy vybrán server `web.mit.edu`, který obsahuje velké množství dat, obrázků, skriptů, apod. Tímto byly k porovnání získány statistiky závisející na obsahu stahovaných dat.

Za klientské stanice byly vybrány dva stroje virtualizované programem `Oracle VM VirtualBox`. Prvním strojem je `Linux Ubuntu v18.04.3 LTS` a druhým stroj `Microsoft Windows 10 v1809`. Pro kontrolu dat z virtuálního stroje Windows byla stejná data zachycena i z fyzického počítače `Microsoft Windows 10 v1903`. Alokované prostředky jsou zobrazeny v tab. 3.1.

Klientský software byl třech typů. Jednalo se o prohlížeče `Google Chrome 78`, `Mozilla Firefox 71` a `Opera 64`, kde verze prohlížečů byly shodné napříč platformami. Vždy byl zachycen přístup na danou webovou stránku z jednoho prohlížeče za použití HTTP a za použití HTTPS, tudíž jeden vzorek (soubor typu `.pcap`) obsahoval právě jeden *bidirectional flow* (kap. 2.2.2). Mezi těmito přístupy byla vždy

[1] V běžném významu způsob MITM útoku na HTTP žádost zaslanou klientem na server a zachycení odpovědi serveru se zprávou `redirect`. Komunikace mezi klientem a útočníkem je nešifrovaná, naopak mezi útočníkem a serverem je šifrovaná. V rámci získávání dat by SSL strip byl využit k získání stejné komunikace v obou formách, šifrované i nešifrované.

vymazána mezipaměť daného prohlížeče. Mezipaměť DNS čištěna nebyla, jelikož nebylo cílem rozšiřovat zachycené IP toky o tato data.

U klientské stanice Ubuntu byl záchyt prováděn programem `tcpdump` z terminálu na ethernetovém rozhraní. V rámci systémů Windows byl záchyt prováděn programem `Wireshark` stejným způsobem. Oba typy záchytů byly následně zpracovány také v programu `Wireshark` a vyčištěny od irelevantního provozu filtrováním paketů. Zbylé pakety byly ponechány a soubory byly uloženy pod specifickými názvy podle tab. 3.2.

Pro zajištění určité možnosti srovnání a průměrování výsledků byl stejný druh komunikace zachycen vždy pětkrát.

Tab. 3.1: Alokované prostředky strojů.

Prostředek	**VM Linux**	**VM Win 10**	**Win 10**
CPU [jádro]	1	1	4
RAM [GB]	6	6	16

Tab. 3.2: Názvy souborů pro i = pořadové číslo vzorku.

Prohlížeč	**HTTP**	**HTTPS**
Google Chrome	c*i*.pcap	cs*i*.pcap
Mozilla Firefox	f*i*.pcap	fs*i*.pcap
Opera	o*i*.pcap	os*i*.pcap

3.1.2 Datová sada protokolu SSH

Cílem tvorby této datové sady bylo nasbírat vzorky obsahující různorodý provoz, jehož statistiky by mohly vykázat něco zajímavého. Protože u protokolu SSH nelze zajistit totožná nešifrovaná data k porovnání, bylo nutno nasbírat data zachycující různé úkony a tyto pak porovnat proti sobě. Jako zmíněné úkony byly vybrány následující aktivity:

- přihlášení na SSH server pomocí hesla, procházení a výpis adresáře,
- přihlášení na SSH server pomocí hesla a ukončení spojení,
- přihlášení na SSH server pomocí klíče a ukončení spojení,
- pokus o autentizaci se zadáním špatného hesla třikrát,
- vzdálené vykonání příkazu (výpis adresáře) pomocí `ssh hostname command`, autentizace pomocí hesla.

Na virtuální stroj `Linux Ubuntu v18.04.3 LTS` byl nainstalován SSH server `OpenSSH 7.6`. Do autorizovaných klíčů byl vložen veřejný klíč fyzického počítače `Microsoft Windows 10 v1903` vygenerovaný programem `PuTTY`. V programu `PuTTY` byla také prováděna autentizace za pomoci dvojice asymetrických klíčů a odtud byl zachycen vzorek. Ostatní vzorky provozu pochází ze zachycení vykonání patřičných příkazů na příkazovém řádku počítače Windows. Každý vzorek obsahuje jedno spojení a byl zachycen programem `Wireshark`. Následně byl ze souboru vyfiltrován provoz, který nesouvisel s datovou sadou a ponechán pouze provoz protokolu SSH (jeho verze 2.0). Poté byl vzorek uložen pod názvem pořadového čísla do patřičné složky podle svého druhu.

Stejně jako u datové sady HTTP/S, i tady je zachycen stejný druh toku pětkrát.

3.1.3 Datová sada DNS over HTTPS

Vzorek *DNS over HTTPS* (DoH) byl poskytnut sdružením CESNET k analýze. Jednalo se o soubor typu `.pcap`, který obsahoval toky směrované na Raspberry Pi v pozici DNS over HTTPS klienta. K získání vzorku bylo využito opensourcového řešení pomocí DoH proxy `cloudflared`. Ta směruje DNS over HTTPS dotazy na veřejný DNS resolver společnosti Cloudflare podporující DoH.

3.1.4 Datová sada SMTPS

K zachycení vzorků odchozí emailové komunikace byl na virtuální stroj `Microsoft Windows 10 v1809` nainstalován emailový klient `Mozilla Thunderbird`.

Odchozí server `smtp.gmail.com` byl nastaven dvěma způsoby. Prvně bylo využito implicitního TLS na portu 465, později byl server přenastaven na port 587 a bylo užito explicitní TLS příkazem `STARTTLS`.

Datové sady byly vytvořeny ze zachycení následujících emailových zpráv. Všechny varianty byly zachyceny vždy pětkrát pro obě nastavení odchozího SMTP serveru a pro jednoho příjemce:

- krátký textový obsah:
  - předmět 4 znaky,
  - obsah 4 znaky,
- dlouhý textový obsah:
  - předmět 4 znaky,
  - obsah 2269 znaků,
- žádný textový obsah, soubor `.jpg` v příloze:
  - předmět 4 znaky,
  - soubor 100 kB,

- dlouhý textový obsah, soubor `.jpg` v příloze:
  - předmět 4 znaky,
  - obsah 2269 znaků,
  - soubor 100 kB,
- žádný textový obsah, soubor `.pdf` v příloze:
  - předmět 4 znaky,
  - soubor 100 kB,
- dlouhý textový obsah, soubor `.pdf` v příloze:
  - předmět 4 znaky,
  - obsah 2269 znaků,
  - soubor 100 kB.

Následující vzorky byly zaslány vždy pětkrát pro obě nastavení odchozího SMTP serveru a deset příjemců:

- krátký textový obsah:
  - předmět 4 znaky,
  - obsah 4 znaky,
- dlouhý textový obsah, soubor `.jpg` v příloze:
  - předmět 4 znaky,
  - obsah 2269 znaků,
  - soubor 100 kB.

Dlouhý text zprávy byl pseudočeský, vygenerovaný pomocí generátoru `Blábot`.

Zachycení a filtrace komunikace probíhala v programu `Wireshark`. Vzorky dat jsou uloženy podle svých typů v příslušných složkách.

3.2 Převod na rozšířené flow záznamy

Statistiky o šifrovaném provozu budou získávány především prostřednictví flow monitorování s rozšířenými flow záznamy. K výpočtu statistik byl zvolen opensourcový nástroj a exportér *Cisco Joy* (kap. 2.4), vzhledem k výhodám, které nabízí. Umožňuje totiž exportovat velké množství rozšiřujících dat nad rámec běžných flow záznamů. Užitečnost právě těchto dat byla popsána v kap. 2.3.

Vzorky provozu protokolů HTTP/S (kap. 3.1.1), SSH (kap. 3.1.2), DNS over HTTPS (kap. 3.1.3) a SMTPS (kap. 3.1.4) byly automatizovaně pomocí `bash` skriptu zpracovávány programem *Joy* a výsledky byly ukládány v komprimovaném formátu `gz`. Průběh zpracování je popsán v následujících podkapitolách.

3.2.1 Zpracování HTTP vzorků

Joy v základu vytvoří JSON objekt takový, jaký je popsaný v kap. 2.4. Další parametry slouží k rozšíření tohoto objektu o údaje, které jsou vybrány. Vybrané parametry jsou rozepsány níže. *Joy* byl spuštěn nad nasbíranými vzorky **HTTP** následně:

```
bin/joy bidir=1 http=1 dist=1 entropy=1 retrans=1 num_pkts=200
output=jméno.gz cesta_ke_vstupnímu_souboru
```

Parametr typu boolean `bidir` znamená, že *Joy* bude agregovat jednosměrné toky, které se v čase prolínají, do jednoho dvousměrného spojení.

Parametr `http` značí, že Joy bude extrahovat data HTTP hlaviček a přidávat je do finálního objektu JSON. Pro příchozí hlavičky jsou to například data typu `version`, `code`, `server`, `Last-Modified`, `Content-Length` a další. Pro odchozí hlavičky se jedná o data `method`, `uri`, `host`, `User-Agent`, `Referer` aj.

Další parametr pod názvem `dist` zahrnuje distribuci bajtů do výsledného JSON objektu jako pole. Toto pole obsahuje počty výskytu jednotlivých bajtů v datových položkách daného toku. Jsou seřazeny od hodnoty bajtu nula (`00` v hexadecimální soustavě) až po 255 (`FF`).

Dalším vybraným parametrem je `entropy`. Do objektu jsou přidány dvě hodnoty. Hodnota `entropy` znamená entropii v bitech na jeden bajt. Jedná se entropii spočítanou z rozdělení pravděpodobnosti nad bajty a nabývá hodnot od nuly do osmi. Druhá hodnota `total_entropy` značí celkovou hodnotu entropie v bajtech přes všechny bajty daného IP toku. Pro jednosměrný tok je k výpočtu použit počet `bytes_out`, pro dvousměrný tok součet `bytes_out` a `bytes_in`. Nabývá hodnot od nuly do osminásobku zmíněných hodnot počtu bajtů.

Parametr `retrans` značí, že do pole `packets` budou zahrnovány i retransmise paketů TCP. Tento parametr byl pro datové sady HTTP/S důležitý, jelikož bez jeho použití byly statistiky v rámci tohoto pole nesprávné.

Poslední je parametr s názvem `num_pkts`. Může nabývat hodnot od nuly do 200. Daný počet paketů na začátku toku bude zaznamenán s podrobnějšími informacemi v poli `packets` daného JSON objektu. Když není specifikováno jinak, s výchozím nastavením je zpracováváno prvních 50 paketů IP toku.

Následuje pouze uvedení jména výstupního souboru a cesta ke vstupu, který má být programem zpracován.

3.2.2 Zpracování HTTPS vzorků

Program *Cisco Joy* byl nad nasbíranými daty protokolu **HTTPS** spuštěn s následujícími parametry:

```
bin/joy bidir=1 tls=1 dist=1 entropy=1 retrans=1 num_pkts=200
output=jméno.gz cesta_ke_vstupnímu_souboru
```

Parametry jsou stejné jako u HTTP s výjimkou `http`, který byl nahrazen parametrem s názvem `tls`. Tato data jsou tedy rozšířena o informace extrahované z protokolu TLS (kap. 1.1.1). Jedná se o velký počet polí, např. prvočísla `random` serveru a klienta, délka klíče klienta (`c_key_length`), jméno serveru, na který se klient chce připojit (`sni`), kód vybrané kombinace šifrovacích a podpisových algoritmů *cipher suite* (`scs`), ale i seznam všech podporovaných variant (`cs`). Pole `s_cert` obsahuje veškerá data o certifikátu (podpis, podpisový algoritmus, délku klíče, informace o vystaviteli a subjektu, alternativní jména subjektu, platnost certifikátu a další). Objekt dále obsahuje seznam TLS rozšíření klienta i serveru a množinu dalších informací.

3.2.3 Zpracování SSH vzorků

Nad nasbíranými vzorky protokolu **SSH** byl *Joy* spuštěn s těmito parametry:

```
bin/joy bidir=1 dist=1 ssh=1 entropy=1 retrans=1 num_pkts=200
output=jméno.gz cesta_ke_vstupnímu_souboru
```

Parametry se opět liší pouze ve specifikaci daného protokolu, o jehož data je nutno rozšířit základní IP tok. Namísto `http` či `tls` je tedy využito parametru pod názvem `ssh`. Spuštění programu *Joy* s tímto parametrem obohatí základní objekt o informace o SSH klientovi, např. verze protokolu (`protocol`), podporované algoritmy výměny klíče (`kex_algos`), podporované algoritmy šifrování (`c_encryption_algos`), algoritmy výpočtu MAC (`c_mac_algos`) a komprese (`c_comp_algos`), ale i o stejné informace o straně SSH serveru. Jsou tu zachyceny i veřejné klíče obou stran komunikace s vybraným algoritmem výměny klíčů.

3.2.4 Zpracování DNS over HTTPS vzorků

Tok **DNS over HTTPS** je dlouhý tok, ve kterém se přenese malé množství bajtů. Z tohoto důvodu bylo složité získat záznamy, které by byly rozšířené o data protokolu TLS. *Joy* totiž dokáže exportovat tok o maximální délce 30 sekund. Pokud tok přesáhne tuto hranici, následné pakety jsou odstřihnuty do dalšího toku a již nejsou rozšířeny o TLS data. Taková data byla vyfiltrována a bylo ponecháno 30 vzorků rozšířených o data protokolu TLS.

Joy byl nad datovou sadou spuštěn stejným příkazem, jako tomu bylo u zpracování dat HTTPS.

3.2.5 Zpracování SMTPS vzorků

Jelikož **SMTPS** je protokol SMTP „zabalený“ v protokolu TLS, veškeré informace zjistitelné bez dešifrování komunikace pochází právě z vnějšího protokolu. Proto byl program *Cisco Joy* spuštěn nad těmito zachycenými vzorky stejným příkazem, jaký byl užit u vzorků protokolu HTTPS.

Při převodu vzorků směrovaných na port 587 *Joy* navzdory očekáváním nerozšířil toky o data protokolu TLS. Pravděpodobným problémem je užití `STARTTLS`, kdy je tok na počátku složen ze SMTP paketů, později je však díky příkazu `STARTTLS` transformován na komunikaci skrytou v protokolu TLS. Toky směrované na port 465, kde je TLS využíváno implicitně, jsou však o tato data rozšířeny.

3.2.6 Sekvence délek paketů/záznamů a časů mezi nimi

Cisco Joy umí exportovat informace o sekvenci velikostí paketů, mezipaketových časů a směrů až u prvních dvou set paketů toku. Tato položka se nazývá *Sequence of Packet Lengths and Times* (SPLT). Ve výsledném JSON objektu je uložena pod názvem `packets`. Ukázka z datové sady je zachycena na výpisu 3.1.

Výpis 3.1: Ukázka SPLT jako JSON výstup programu Cisco Joy.

```
"packets": [
{
  "b": 348,            # délka paketu [B]
  "dir": ">",          # směr paketu [< in, > out]
  "ipt": 469           # čas uplynulý od posledního
                       # paketu [ms]
{
  "b": 1460,
  "dir": "<",
  "ipt": 15
},
{
  "b": 1460,
  "dir": "<",
  "ipt": 0
}, ...
```

Pokud tok obsahuje TLS handshake, *Joy* extrahuje TLS údaje a výsledný JSON o ně rozšíří. Mimo jiné jsou takové toky rozšířeny o informace o sekvenci velikostí

záznamů (*records*), časů mezi nimi a směrů těchto záznamů. Tyto informace se souhrnně nazývají *Sequence of Record Lengths and Times* (SRLT). Ve výsledném JSONu jsou zaznamenány v objektu `tls` pod názvem `srlt`. Ukázka z datové sady je zachycena na výpisu 3.2.

V rámci jednoho paketu může být i více záznamů, například data aplikací malé velikosti, nebo jeden záznam může být rozdělen na více paketů (záznam s přenosem certifikátu by byl rozdělen na více paketů podle maximální délky segmentu TCP).

Výpis 3.2: Ukázka SRLT jako JSON výstup programu Cisco Joy.

```
"srlt": [
  {
    "b": 512,       # velikost záznamu [B]
    "dir": ">",     # směr záznamu [< in, > out]
    "ipt": 0,       # mezizáznamový čas [ms]
    "tp": 22,       # typ obsahu podle IANA
    "hs_types": [
      1             # typ handshake podle IANA
    ],
    "hs_lens": [
      508           # délka handshake v záznamu [B]
    ]
  },
  {
    "b": 1,
    "dir": "<",
    "ipt": 0,
    "tp": 20
  }, ...
```

4 Metody vizualizace šifrovaného provozu

Pro návrh metod vizualizace toků šifrovaného provozu byla vybrána technologie *Jupyter Notebook*, jelikož nabízí příhodné prostředí pro výstup této práce. Technologie je blíže popsána v kap. 4.1.

Pro zpracování vzorků z datových sad a vykreslení grafů byly v programovacím jazyce Python sepsány dvě knihovny. První je určena k porovnání ručně vybraných vzorků a druhá k vizualizaci statistik většího množství vzorků stejného typu. Každá knihovna byla využita ve vlastním souboru Jupyter Notebook.

Notebooky pracují se soubory typu JSON, které obsahují jeden tok jako výstup z programu *Cisco Joy*. Tato data jsou předána knihovně, která následně vrátí vizualizace dat. Zpracované vzorky datových sad byly využívány k testování navržených vizualizací.

4.1 Jupyter Notebook

Jupyter Notebook (dále jen „JN“) je opensourcový výpočetní notebook. Jedná se o nástroj, který umožňuje kombinovat vysvětlující text, kód a výpočetní výstup do jednoho souboru. Funguje interaktivně a ve webovém rozhraní. V poslední době narůstá JN popularita zejména v oblasti výzkumu a datové vědy [44], podporuje totiž mnoho programovacích jazyků a usnadňuje přístup ke vzdáleným datům a výpočty nad nimi.

JN je rozdělen do buněk. Existují buňky různých typů, v této práci jsou však využity jen dva typy – `Markdown` pro dokumentaci a text a `Code`, který obsahuje spustitelný kód programovacího jazyka podporovaného vybraným kernelem. Buňky se spouští postupně podle pořadí.

V této bakalářské práci byly k návrhu metod vizualizace toků šifrovaného provozu využity JN s Python 3 kernelem. Tento formát byl vybrán, jelikož notebooky umožňují jednoduchou dokumentaci a vysvětlení kódu, interaktivní výměnu dat a přizpůsobení obsahu kódu. Notebooky jsou navrženy tak, aby sloužily k rychlému načtení dat a umožnily následný průzkum zajímavostí načtených dat. V `Markdown` buňkách je dokumentována struktura notebooku a vysvětlení k proměnným a funkcím. V `Code` buňkách je spustitelný kód, kde je možné upravovat proměnné a měnit například názvy grafů, os, barvy grafů atd. Po spuštění notebooku jsou v daných buňkách volány funkce k vykreslování grafů z knihoven.

V dalších podkapitolách jsou tyto notebooky popsány podrobněji. Popis je také ilustrován pomocí grafů.

4.1.1 Instalace

JN lze jednoduše nainstalovat se všemi prerekvizitami v rámci programu `Anaconda` dostupného pro Windows i Linux. Nabízí i opensourcovou variantu *Individual Edition*. Je to nejpopulárnější distribuční platforma Pythonu pro datovou vědu. Obsahuje repozitáře k instalaci tisíců balíčků použitelných ve datové vědě, strojovém učení, vizualizacích dat, neuronových sítích aj. [45].

Po instalaci je možné na příkazovém řádku `Anacondy` spustit prostředí pomocí příkazu `jupyter notebook`. V prohlížeči na adrese localhost se otevře souborový systém, kterým lze navigovat do složky se soubory Jupyter Notebooku s koncovkou `.ipynb`. Notebooky byly vytvářeny v distribuci `Anaconda3` 2019.10 s Pythonem verze 3.7.4.

JN je možné také nainstalovat pomocí manažeru `pip` za splnění prerekvizity nainstalovaného Pythonu 3.3 a výše, či Pythonu 2.7.

4.1.2 Užívání vytvořených notebooků

V první buňce s kódem notebooku dochází k nastavení základních parametrů. Načítání se liší u obou notebooků. V notebooku pro porovnání toků je nutné specifikovat názvy souborů a jejich popisy do `dictionary` objektu. Ve statistickém notebooku je naopak nutné specifikovat počet souborů, jelikož tento notebook předpokládá načítání souborů pod jménem `1.json` až n`.json`, kde n je celkový počet souborů, které mají být vizualizovány. První varianta buňky je zobrazena na obr. 4.1. Pro statistickou variantu se mění proměnná `file_names` na proměnnou `n`. Dále je nutné specifikovat název složky, která je umístěna na stejném místě, jako spuštěné notebooky. V této složce musí být umístěna data načítaná v rámci objektu. Následně je možné specifikovat barvy, které budou užity v grafech.

In [1]:
```
import os
import lib.flow_analysis as flow
%matplotlib inline

dir_name = 'složka'
file_names = {
    "ukázka_1.json" : "popisek 1",
    "ukázka_2.json" : "popisek 2",
    "ukázka_3.json" : "popisek 3"
}
colors = ['primární_barva','sekundární_barva']
```

Obr. 4.1: Nastavení základních parametrů k načtení.

Poté je zjištěna celá cesta k souborům a do proměnné `data` jsou načteny veškeré potřebné informace. U každého typu grafu je dedikovaná buňka s parametry, které se dají podle potřeby nastavit. Argumenty jsou naparsovány a předány funkci vedle načtených dat. Na obr. 4.2 je ilustrován tento popis.

Ve statistickém notebooku je vedle základních parametrů ještě možnost nastavení parametru `bins` u histogramů. Bližší dokumentace, popis proměnných a užívání je uveden v notebooku. Funkce knihoven jsou dokumentovány pomocí komentářů v kódu.

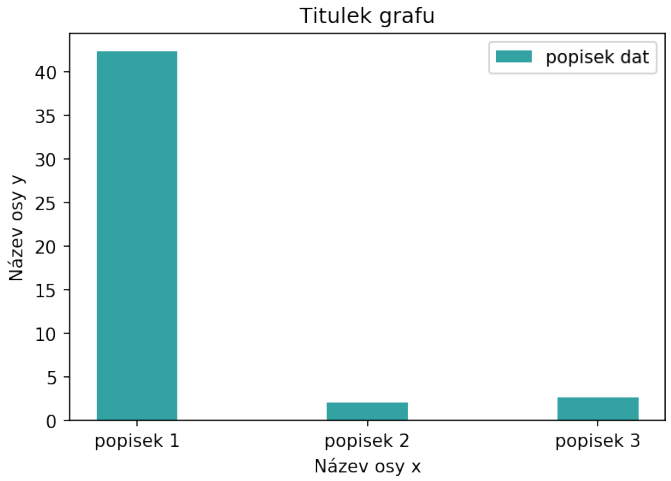
Grafy vizualizují vybraná metadata a statistiky, které mají umožnit jednoduché nahlédnutí do zachycených dat s potenciálem odhalit zajímavé inkonzistentní toky.

Pro lepší představu je blokové schéma vizualizace a sběru dat zobrazeno na obr. A.1.

```
In [5]:
x = 'Název osy x'
y = 'Název osy y'
title = 'Titulek grafu'
label = 'popisek dat'

graph_params = flow.set_graph_params(x,y,title,label,colors)

flow.flow_duration(data,graph_params,file_names)
```



Obr. 4.2: Nastavení proměnných a zobrazení grafu.

4.2 Notebook a knihovna k porovnání toků

Notebook slouží k ručnímu zkoumání a porovnávání vícero toků, ať už stejného, či rozdílného typu. Tyto toky nejsou nijak průměrovány, jsou tedy zobrazeny v původní podobě, aby bylo možné jednoduše najít odchylky a rozdíly. Grafy notebooku se dynamicky škálují podle počtu zadaných a načtených toků.

V notebooku byla využita naprogramovaná knihovna `flow_analysis`, která umožní následující vizualizace:

- počet příchozích/odchozích bajtů (`bytes_in_out(...)`),
- počet příchozích/odchozích paketů (`packets_in_out(...)`),
- délka trvání toku (`flow_duration(...)`),

- rozložení distribuce bajtů ve formě histogramu
  - jednoho konkrétního toku (`byte_dist_show(...)`),
  - všech vybraných toků (`byte_dist_avg(...)`),
  - porovnání těchto distribucí (`byte_dist_compare(...)`),
- rozložení distribuce bajtů ve formě *heat mapy* (`heatmap_dist(...)`),
- užitá *cipher suite* jednoho toku a další varianty (`used_ciphersuite(...)`),
- sekvence délek paketů a mezipaketových časů (SPLT)
  - všech vybraných toků (`splt_chart(...)`),
  - jednoho konkrétního toku (`splt_chart_one(...)`),
- sekvence délek záznamů a mezizáznamových časů (SRLT)
  - všech vybraných toků rozšířených o TLS (`srlt_chart(...)`),
  - jednoho konkrétního toku rozšířeného o TLS (`srlt_chart_one(...)`).

Načtení dat specifikovaných v notebooku provádí funkce `load_data(...)`. Vrátí objekt typu `dict`. Parametry grafu jako název os, barvy, titulek aj. jsou z notebooku sesbírány pomocí funkce `set_graph_params(...)`, která přijímá variabilní počet argumentů. Podle něj argumenty parsuje do dalšího `dict` objektu. Objekt je následně spolu s objektem s načtenými daty předán vizualizujícím funkcím.

Knihovna mimo výše zmíněných funkcí obsahuje ještě další funkce potřebné ke správnému zpracování dat (`normalize(...)`, `protocol_determination(...)` aj.). Tyto funkce jsou volány v rámci vizualizujících funkcí.

V knihovně je využíváno externích knihoven `matplotlib`, `seaborn`, `pandas` a `numpy`.

4.2.1 Ukázky vizualizací

K vizualizaci vyměněného **počtu bajtů** v komunikaci byl vybrán sloupcový graf. Graf sleduje počet bajtů v odchozím a příchozím směru pro každý zadaný tok a jejich jednoduché vizuální porovnání. Ukázková data na obr. 4.3 pochází z datové sady HTTP/S protokolu – přístup na webovou stránku `var.liberouter.org` z VM Linux.

Počet paketů v toku je vizualizován sloupcovým grafem stejným způsobem jako výše zmíněné bajty. Pro ukázku na obr. 4.4 byla vybrána sada úkonů z datové sady protokolu SSH.

Délka trvání toku je také zobrazena za využití sloupcového grafu. Data na obr. 4.5 pochází z datové sady SMTPS na portu 587 (STARTTLS).

Následujících pět vizualizací se zabývá distribucemi bajtů toků. Vizualizace **distribuce bajtů vybraného souboru** je provedena pomocí histogramu hodnot bajtů obsažených v paketech toku. Ukázkový soubor pochází z datové sady HTTP/S protokolu a VM Windows. Jedná se o soubor `o1.json` (tedy přístup

na webovou stránku `web.mit.edu` přes HTTP v prohlížeči Opera). Ukázka je na obr. 4.6.

Další vizualizací je **průměrná distribuce bajtů všech souborů**. Distribuce bajtů načtených souborů jsou průměrovány a vizualizovány stejným způsobem, jako tomu je u histogramu výše. Načtené soubory pochází z datové sady HTTP/S, přístup na `web.mit.edu` přes HTTPS v prohlížeči Opera. Jsou průměrovány soubory `os1-5.json`. Ukázka je na obr. 4.7.

Vizualizace **porovnání distribucí bajtů** vybraného souboru vůči průměru všech souborů shrnuje předešlé dva grafy do jednoho. Funkce, jíž výstupem je tento graf, normalizuje hodnoty převodem na procenta tak, aby porovnání nebylo zkresleno počtem paketů toku či délkou toku. Porovnání je zobrazeno na obr. 4.8.

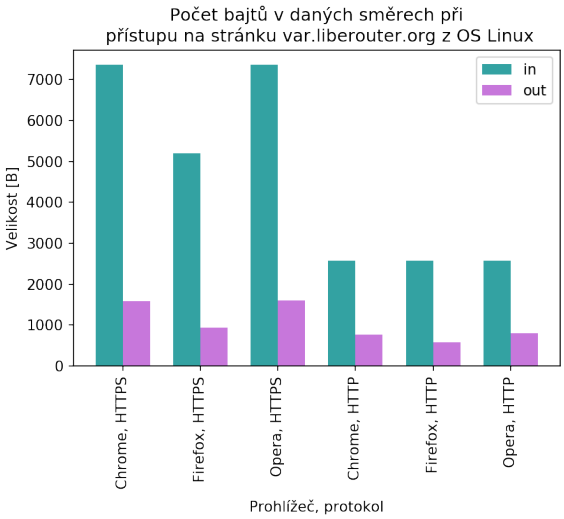
Vizualizace na obr. 4.9 ukazují jiný pohled na uvedené distribuce bajtů. Jedná se o ***heat mapy***, kde je počet výskytů namísto sloupce znázorněn odstínem barev. Čím tmavší je modrá, tím větší počet bajtů mělo danou hodnotu. Čtverec vlevo nahoře má hodnotu 0 (`00`) a čtverec vpravo dole 255 (`FF`).

Na ukázce 4.10 je zobrazen výstup funkce na **zjištění užité cipher suite** a všech variant, které byly nabízeny k vyjednávání. Tento výstup je zobrazen pouze tehdy, pokud je tok rozšířený o data protokolu TLS. Potřebné informace jsou zapsány v souboru `ciphersuites.txt`, uloženém na stejném místě jako vytvořené knihovny. Tento soubor obsahuje seznam nejčastěji užívaných cipher suites, ke zkratkám jsou přiřazeny celé názvy a síla cipher suite. Tyto informace jsou pak zobrazeny ve výpisu a názvy ciphersuites v tabulce.

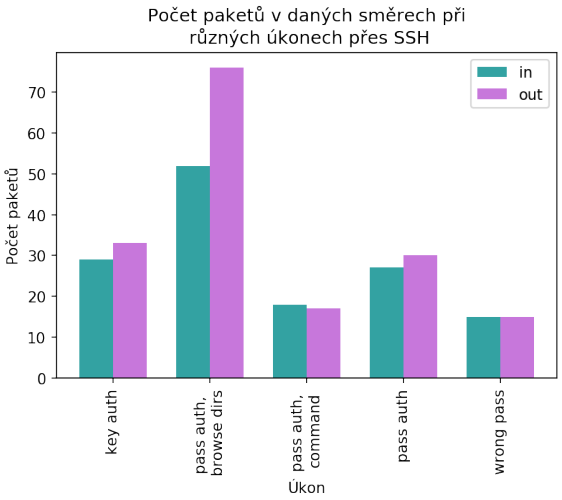
Vizualizace **sekvence délek paketů a časů mezi nimi** (SPLT, kap. 3.2.6) je zobrazena na ukázce 4.12, data pochází ze sady protokolu SSH. Vertikální linky odpovídají velikosti přenesených paketů, barvy směru paketů a osa x času od počátku toku, kdy byl paket zaznamenán. Velikosti vertikálních čar jsou škálovány poměrně k největšímu přenesenému paketu všech toků. Jelikož z této vizualizace není možné zjistit přesné velikosti paketů, byl navržen druhý graf k vizualizaci SPLT pouze jednoho vybraného toku. Na ose y je dána velikost paketu v bajtech. Ukázka tohoto grafu je na obr. 4.11 vlevo, podrobněji je zde vidět tok s názvem „key auth“ z obr. 4.12.

Vizualizace **sekvence délek záznamů a časů mezi nimi** (SRLT, kap. 3.2.6) je ukázána na obr. 4.13 v rámci datové sady HTTPS OS Linux a `var.liberouter.org`. SRLT je možné vizualizovat pouze u toků, které jsou rozšířeny o data protokolu TLS exportovaná *Joyem*. Vizualizace SRLT funguje stejným způsobem jako u SPLT. SRLT jednoho toku z obr. 4.13, specificky toku s popisem „Chrome 1“, je podrobněji na obr. 4.11 vpravo.

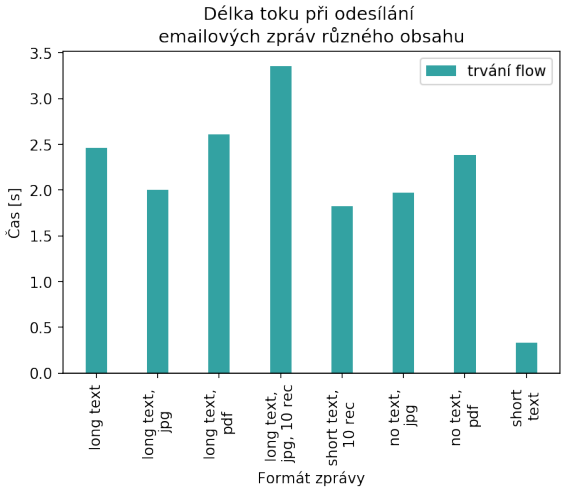
Obr. 4.3: Počet bajtů.



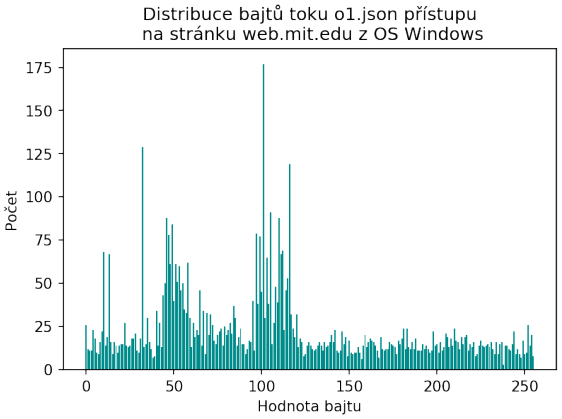
Obr. 4.4: Počet paketů.



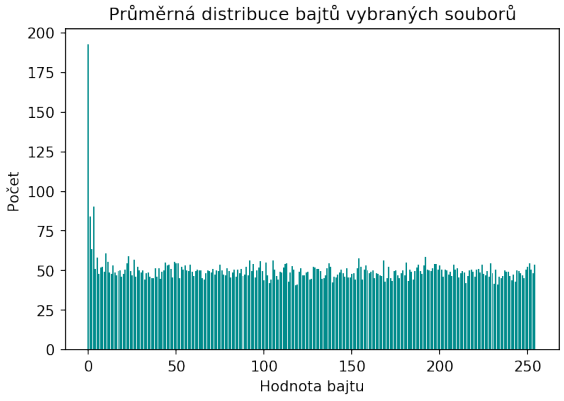
Obr. 4.5: Délka trvání toku.



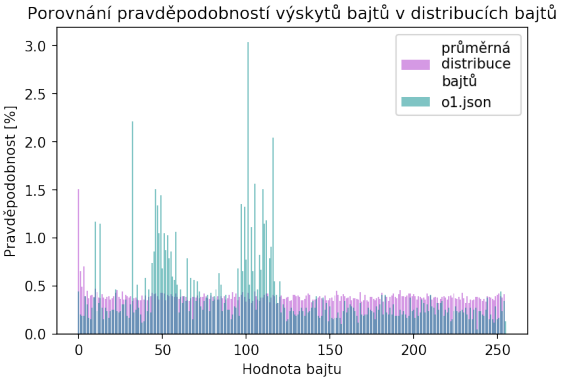
Obr. 4.6: Distribuce bajtů toku.



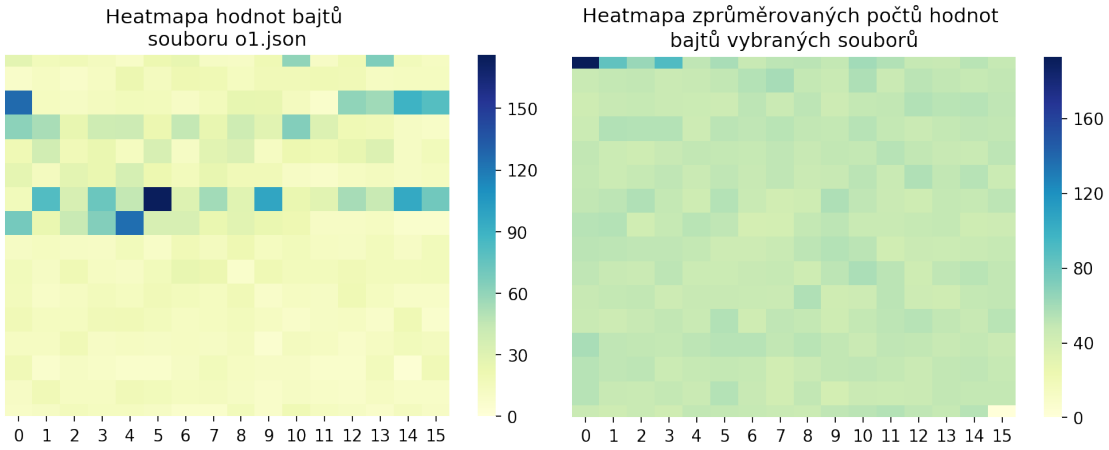
Obr. 4.7: Průměrná distribuce bajtů.



Obr. 4.8: Porovnání distribucí bajtů.



Obr. 4.9: Heat mapy distribucí bajtů.

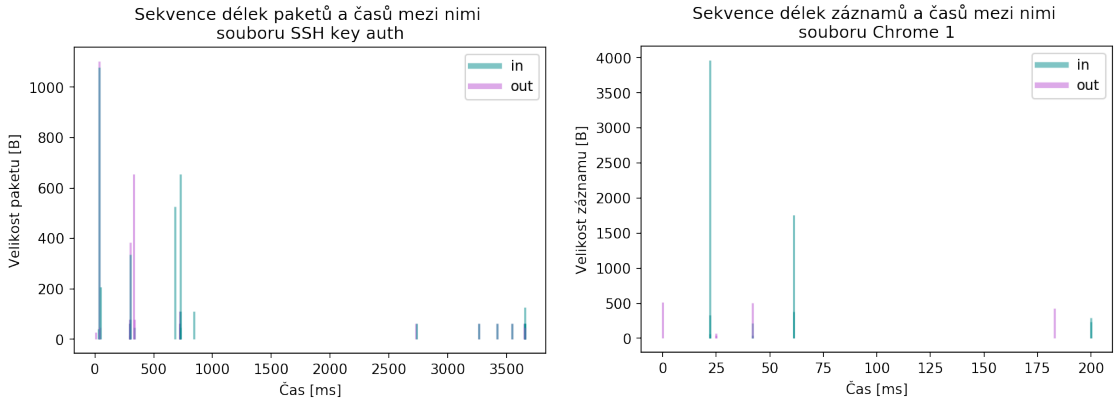


Obr. 4.10: Užitá cipher suite a další varianty.

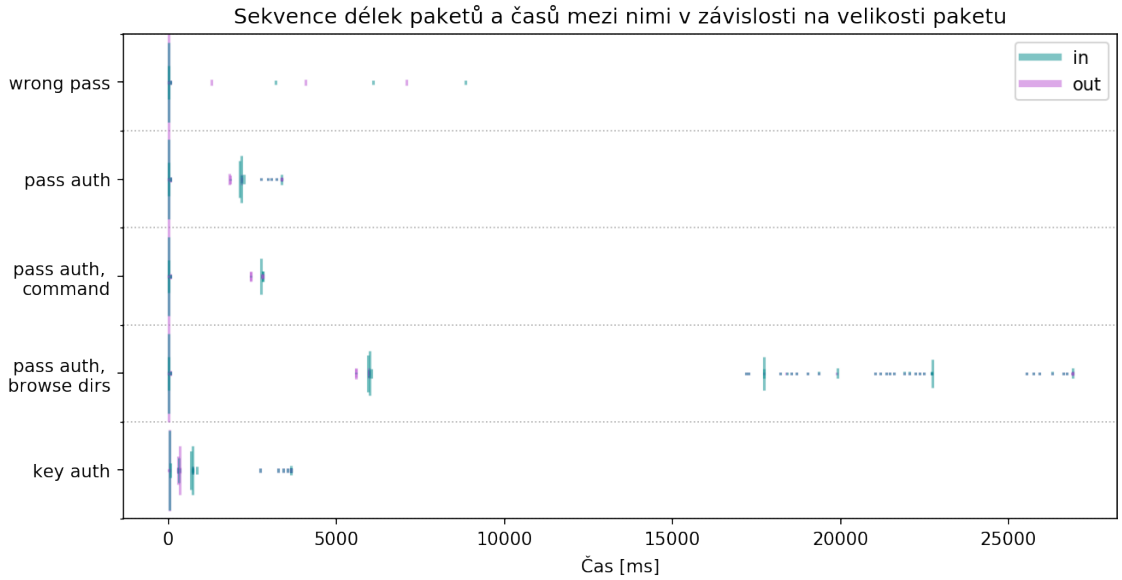
Je užita silná cipher suite ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256.

Cipher suites
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA

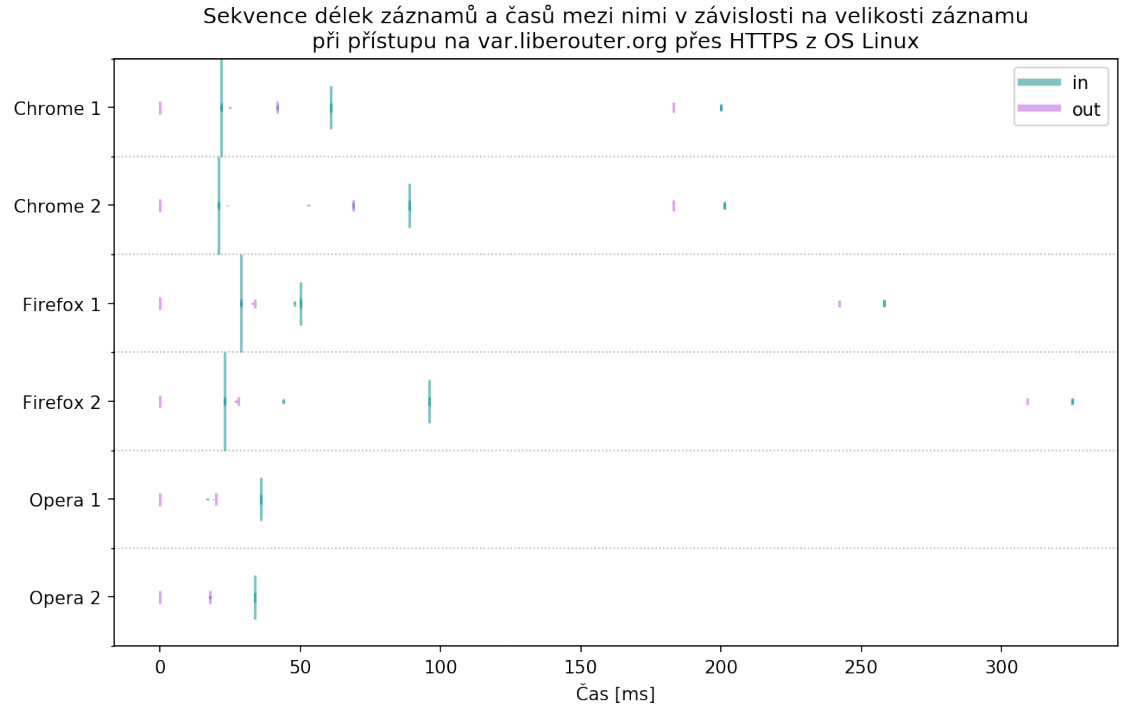
Obr. 4.11: Sekvence délek paketů/záznamů a mezer mezi nimi.



Obr. 4.12: Sekvence délek paketů a mezipaketových mezer.



Obr. 4.13: Sekvence délek záznamů a mezizáznamových mezer.



4.3 Statistický notebook a knihovna

Tento notebook byl vytvořen primárně k porovnávání většího množství toků stejného druhu. Byla využita naprogramovaná knihovna `flow_stat_analysis`, která umožní následující vizualizace:

- počet příchozích/odchozích bajtů
  - rozložení pravděpodobností (`bytes_in_out(...)`),
  - průměr, směrodatná odchylka (`bytes_stats(...)`),
- počet příchozích/odchozích paketů
  - rozložení pravděpodobností (`packets_in_out(...)`),
  - průměr, směrodatná odchylka (`packets_stats(...)`),
- délka trvání toku
  - rozložení pravděpodobností (`flow_duration(...)`),
  - průměr, směrodatná odchylka (`flow_stats(...)`),
- průměrná distribuce bajtů souborů ve formě histogramu (`byte_dist_avg(...)`),
- velikost prvních n paketů toku (`splt_packets(...)`),
- velikost prvních n mezipaketových mezer toku (`splt_ipt(...)`),
- velikost prvních n záznamů toku (`srlt_packets(...)`),
- velikost prvních n mezizáznamových mezer toku (`srlt_ipt(...)`).

Všechny tyto funkce na vstupu požadují dva objekty – objekt načtených dat z funkce `load_data(...)` a objekt s parametry grafu (výstup funkce na parsování argumentů notebooku – `set_graph_params(...)`).

Knihovna dále obsahuje statistické funkce k výpočtům či k normalizaci dat potřebných u vizualizací.

V knihovně je využíváno externích knihoven `matplotlib`, `numpy`, `pandas`, `math`, `seaborn` a `scipy`.

4.3.1 Ukázky vizualizací

Všechny následující ukázky vizualizací pochází z datové sady protokolu HTTPS, přístup na `var.liberouter.org` přes HTTPS z OS Linux i Windows. Jedná se o 30 vzorků.

Ukázka **rozložení pravděpodobností počtů příchozích/odchozích bajtů** je vizualizována na obr. 4.14. Vlevo je graf rozložení pravděpodobností v základním nastavení – s hodnotou proměnné `bins` rovno `None`. Každá hodnota mezi maximem a minimem přenesených bajtů odpovídá jednomu binu. Na pravé straně je graf s `bins = range(0, 8000, 500)`; při upravení hodnoty `bins` je graf transformován na graf hustoty pravděpodobnosti v závislosti na šířce `bins`. Graf je taktéž proložen

křivkou odhadu KDE (*Kernel Density Estimation*). Na obr. 4.15 je přes vizualizované toky naznačená plná čára s průměrem počtu bajtů v daných směrech, přerušovaná čára značí pásmo směrodatné odchylky. Specifické hodnoty jsou v notebooku vždy uvedeny pod grafem v tabulce; ukázka na obr. 4.15 vpravo.

Vizualizace **rozložení pravděpodobností počtů příchozích/odchozích paketů** je ve stejném formátu zobrazena na obr. 4.16. Hodnota `bins` nebyla upravována.

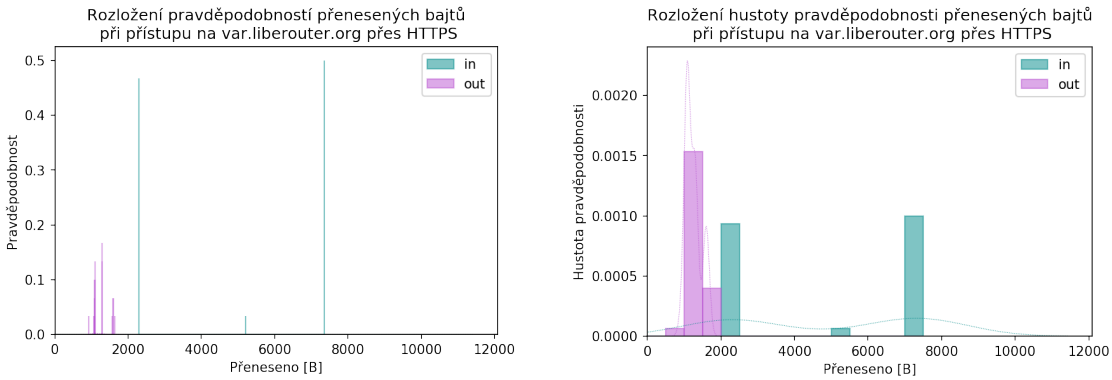
Rozložení pravděpodobností délek toků je zobrazeno na grafech v obr. 4.17. Vlevo hodnota `bins` nebyla upravována, vpravo byla upravena na `bins = range(0, 1050, 50)`. Následný graf 4.18 ukazuje průměr a směrodatnou odchylku délky toků.

Na obr. 4.19 je ukázka vizualizace procentuální **průměrné distribuce bajtů**, která byla normalizována.

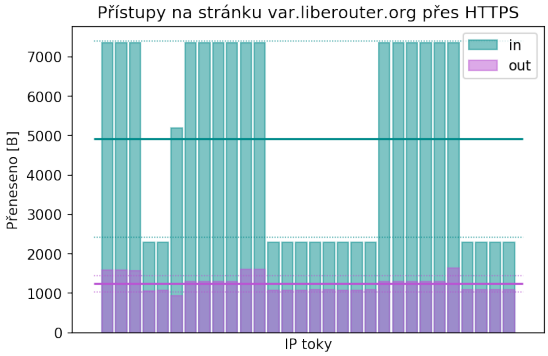
Pro vizualizace **SPLT** a **SRLT** byl vybrán tzv. „boxplot“. Hodnoty jsou tu rozděleny na kvartily; boxplot samotný je ohraničen prvním a třetím kvartilem. Průměr sledovaných hodnot je označen přerušovanou čarou a medián plnou čarou. „Vousy“ (*whiskers*) ukazují variabilitu dat nad třetím kvartilem a pod prvním kvartilem. Sahají až do minima a maxima. Mimo těchto dat se v grafu může objevit ještě tzv. „outlier“. Jedná se o odlehlou hodnotu datového souboru, která je považována za výjimku či extrém.

Na obr. 4.20 jsou zobrazeny průměry velikostí paketů a mezipaketových mezer ve formě boxplotu. Počet zobrazených paketů lze nastavit hodnotou proměnné v notebooku. Barvy mediánu a průměru jsou určeny hlavními barvami notebooku, výplň boxplotů lze nastavit v buňce u grafu. Sloupcový graf v pozadí, jemuž náleží druhá osa y s šedým popisem, udává počet vzorků, ze kterých byl boxplot pro daný paket/mezeru vytvořen. Tento graf napomáhá lepší interpretaci dat v boxplotech. Pro toky rozšířené o data protokolu TLS byla vytvořena varianta vizualizace i pro SRLT. Ukázka je na obr. 4.21.

Obr. 4.14: Rozložení pravděpodobností přenesených bajtů v daných směrech.

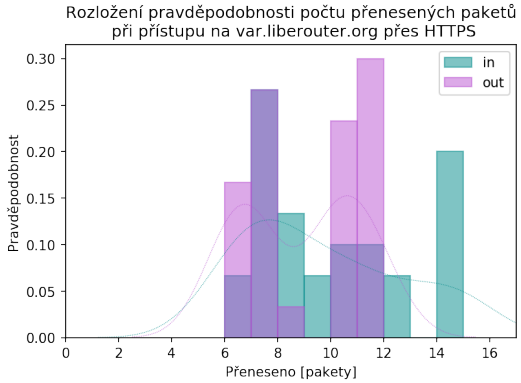


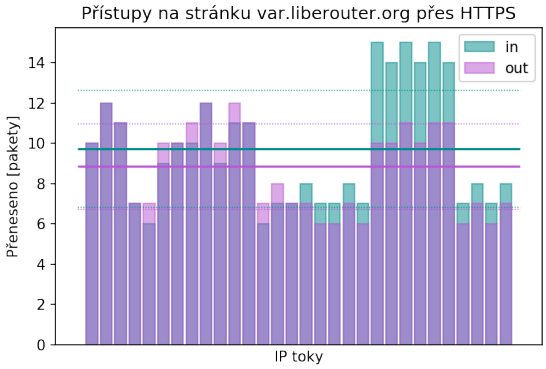
Obr. 4.15: Průměr, směrodatná odchylka přenesených bajtů v daných směrech.



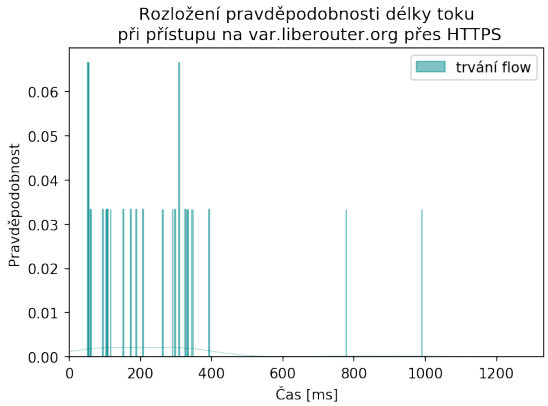
data\směr	in	out
průměr	4918.2	1246.2
rozptyl	6197661.16	41157.16
odchylka	2489.51	202.87

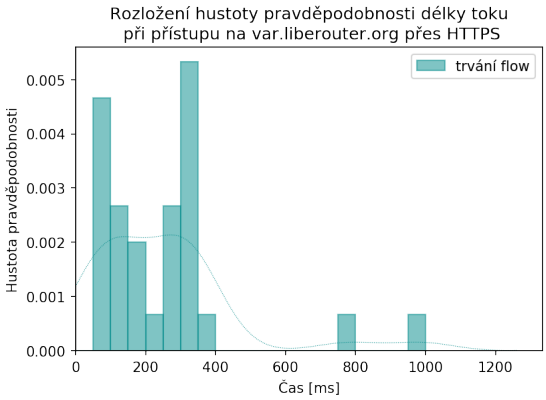
Obr. 4.16: Rozložení pravděpodobností, průměr a směrodatná odchylka paketů.



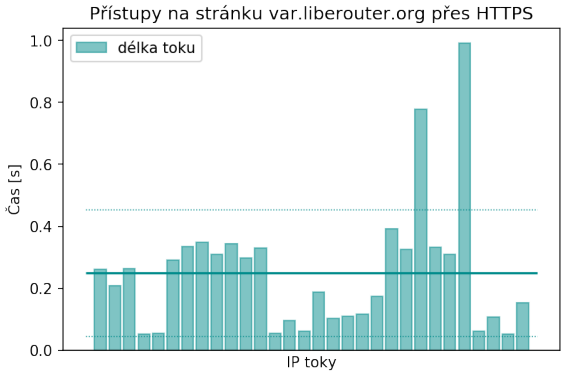


Obr. 4.17: Rozložení pravděpodobností délek toků.

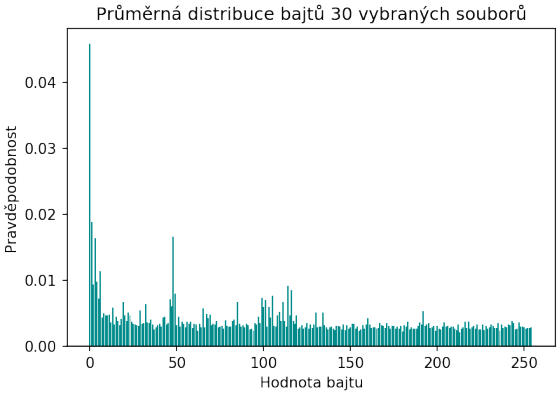




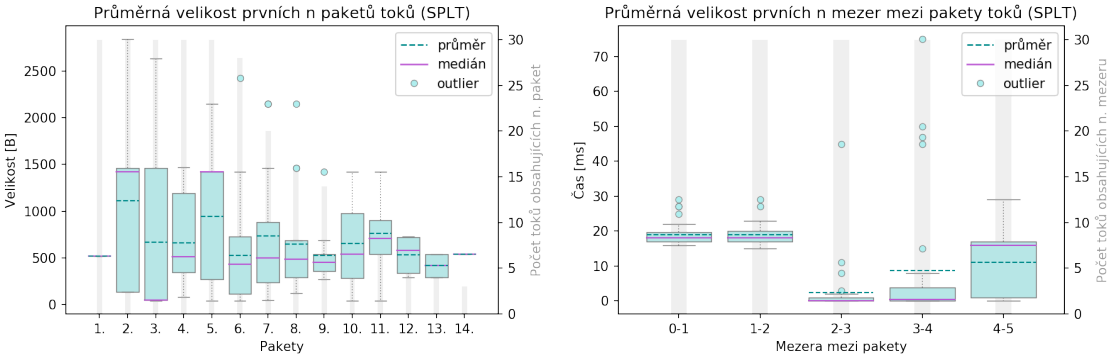
Obr. 4.18: Průměr, odchylka toků.



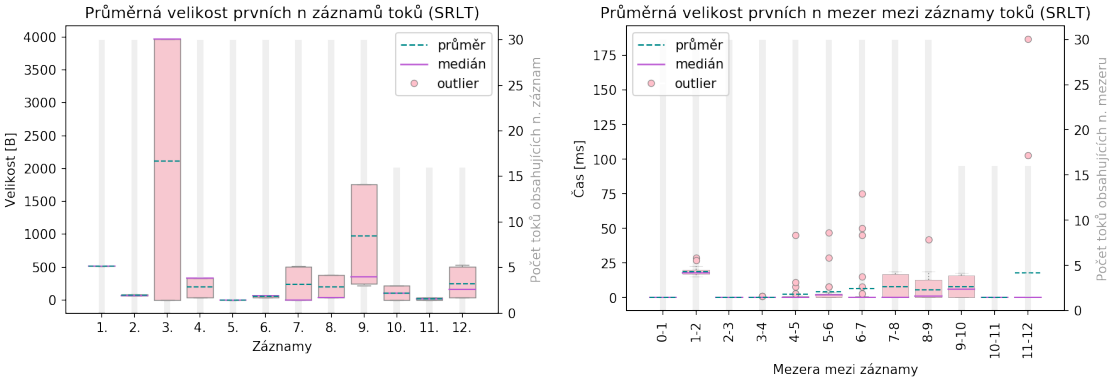
Obr. 4.19: Distribuce bajtů.



Obr. 4.20: Průměrná velikost paketů a mezipaketových mezer.



Obr. 4.21: Průměrná velikost záznamů a mezizáznamových mezer.



4.4 Příklad interpretace grafů k detekci zajímavostí

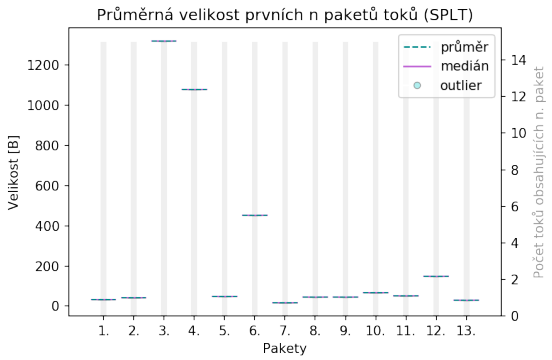
Tato kapitola na praktickém příkladě představuje, jak mohou být navržené vizualizace využity pro vývoj nových metod detekce nežádoucího provozu.

Na vizualizacích níže jsou zobrazeny vzorky toků protokolu SSH. Toky byly seskupeny podle výsledku autentizace – obr. 4.22 obsahuje toky s úspěšnou autentizací, naopak obr. 4.23 toky, kde autentizace nebyla úspěšná.

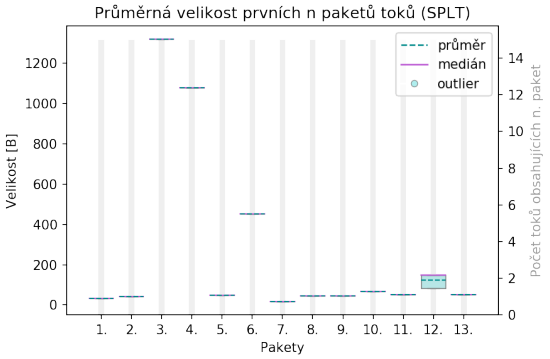
Vzorky toků pro neúspěšnou autentizaci byly ručně vytvořeny s cílem napodobit slovníkový útok na SSH. V každém z 15 toků bylo použito jiné nesprávné heslo. K porovnání tu tedy vzniká vzorek legitimního provozu oproti simulaci útoku hrubou silou na autentizaci protokolu SSH.

Již z prvního pohledu na obr. 4.22 je zřejmé, že všechny toky s úspěšnou autentizací byly konzistentní napříč pokusy. Boxplot je v tomto případě reprezentován pouze horizontální úsečkou, která znázorňuje jednu konkrétní hodnotu velikosti paketu. Na obr. 4.23 s neúspěšnou autentizací však můžeme vidět paket, který se statisticky liší od ostatních. Jedná se o 12. paket toku a na grafu je zobrazen klasickým boxplotem s kvartily. Toto zobrazení znamená větší variaci ve velikostech 12. paketů toku. Vzorky těchto toků se liší pouze jedním parametrem – užitým heslem. Z tohoto tvrzení a povahy vzorků vyplývá, že se uživatelské heslo přenáší ve 12. paketu od počátku SSH toku.

Obr. 4.22: Úspěšná autentizace.



Obr. 4.23: Neúspěšná autentizace.

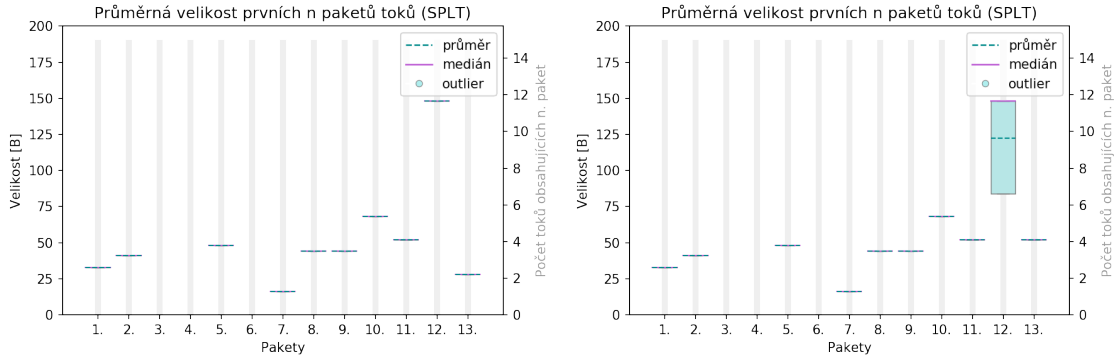


V následnosti na fakta výše lze tvrdit, že se ve 13. paketu přenáší odpověď serveru na uživatelské heslo – či bylo přijato, nebo odmítnuto. Výsledek autentizace by měl být rozeznatelný podle velikosti tohoto paketu. V grafu na obr. 4.24 je upravena maximální hodnota osy y na hodnotu 200 B. Tato úprava zapříčinila posunutí hodnot 3., 4. a 6. paketu mimo graf, nicméně rozdíly 12. a 13. paketů jsou nyní markantnější.

Velikost 12. paketu se u neúspěšných přihlášení pohybuje mezi 85 B a 150 B. Z tohoto faktu mimo jiné vyplývá, že implementace paddingu není optimální.

Pro 13. paket toku je z grafu zřejmé, že pro neúspěšnou autentizaci má tento paket větší velikost. Po ručním prozkoumání těchto paketů bylo zjištěno, že odpověď na úspěšnou autentizaci měla velikost 28 B ve všech případech, pro neúspěšnou autentizaci se jednalo o 52 B. Toto odpovídá teorii o protokolu SSH, kdy jsou v odpovědi na neúspěšnou autentizaci přenášeny informace o selhání autentizace a o dalších možných metodách autentizace. V odpovědi na úspěšnou autentizaci naopak není třeba předávat dodatečné informace, proto je tento paket menší.

Obr. 4.24: Úspěšná a neúspěšná autentizace s upravenou osou y.



Výše uvedený příklad byl inspirován nedávno vzniklou prací [46]. Tato práce využívá mj. velikosti konkrétních paketů toku k detekci útoků hrubou silou na protokol SSH. K objevení užitečnosti velikostí konkrétních paketů vedl v této práci poměrně složitý proces. Využití navržených vizualizací by však mohlo do budoucna napomáhat jednoduššímu zjištění podobných, na první pohled skrytých informací o protokolech. Po odhalení by tyto informace mohly položit základ dalším systémům pro detekci potenciálních hrozeb v síti.

Závěr

Pro uvedení kontextu práce byly v teoretické části popsány vybrané protokoly šifrovaného provozu. Jejich vzorky byly následně zachytávány a užívány v praktické části práce. Jednalo se o protokoly TLS, SSH, DNS over HTTPS a protokol SMTP zabezpečený protokolem TLS. Následně byla nastudována problematika monitorování šifrovaného provozu bez dešifrování, zejména pomocí rozšířených flow záznamů a jejich exportu. Tato technika monitoringu byla podrobně popsána v druhé kapitole této práce. Byly tu shrnuty i současné vědecké práce na téma rozšířených flow záznamů a demonstrovány velmi dobré výsledky, kterých tyto práce dosáhly.

Cílem praktické části bylo vytvořit různé datové sady šifrovaných protokolů a tato data využít k návrhu metod vizualizace šifrovaného provozu.

Byly nasbírány datové sady uvedených protokolů. U dvojice protokolů HTTP/S se jednalo o přístup na různé webové stránky z různých prohlížečů na různých operačních systémech. V datové sadě protokolu SSH jsou nasbírány vzorky různých úkonů prováděných vzdáleně přes tento protokol. V sadě DNS over HTTPS jsou zachyceny DNS dotazy a odpovědi šifrované protokolem TLS. Vzorky SMTP jsou nasbírány pro dvě nastavení odchozího serveru, a to port 465 a implicitní TLS a port 587 a STARTTLS. Vzorky obsahují provoz pro různé typy odeslaných zpráv s různými přílohami pro různý počet příjemců. Nasbírané vzorky toků byly exportérem Cisco Joy převedeny na rozšířené flow záznamy. Takto zpracované vzorky byly využity jako data k testování navržených vizualizací.

Byly vytvořeny dva notebooky technologie Jupyter Notebook, jako platforma pro jednoduché interaktivní vizualizování dat, která mají formát JSON a jsou výstupem exportéru Cisco Joy. Oba notebooky k vizualizacím využívají vlastní naprogramované knihovny. První notebook se soustředí na porovnávání vybraných toků proti sobě, druhý notebook je dedikován statistickému porovnání vícero toků stejného typu. Grafy v obou noteboocích jsou přizpůsobitelné, lze jim nastavit popisy, barvy atd. K vizualizaci byla vybrána zejména data ohledně přenesených bajtů, paketů, distribuce bajtů toku, délky toku, cipher suite, sekvence paketů/záznamů a směrů v čase a sekvence mezer mezi nimi. Výběr těchto metadat k vizualizaci byl inspirován vědeckými pracemi na téma monitoringu pomocí rozšířených flow záznamů.

Vytvořené notebooky a knihovny jsou využitelné zejména pro další výzkum či didaktické účely. Umožňují člověku obeznámenému s principy programování jednoduše získat vizualizace pro své nasbírané toky. Notebooky jsou interaktivní a modulární, můžou být dynamicky měněny a rozšiřovány podle potřeb analytika či studenta.

Literatura

[1] *Let's Encrypt Stats*. Let's Encrypt. [online]. 2019, poslední aktualizace 21. 10. 2019, [cit. 29. 10. 2019]. Dostupné z URL: <https://letsencrypt.org/stats/>.

[2] *HTTPS encryption on the web*. Google Transparency Report. [online]. 2019, poslední aktualizace 26. 10. 2019, [cit. 29. 10. 2019]. Dostupné z URL: <https://transparencyreport.google.com/https/overview?hl=en>.

[3] CARPENTER B. *Middleboxes: Taxonomy and Issues*. RFC 3234. 2002.

[4] DURUMERIC, Z., MA, Z., SPRINGALL, D., BARNES, R., SULLIVAN, N., BURSZTEIN, E., BAILEY, M., HALDERMAN, J., PAXSON, V. (2017). *The Security Impact of HTTPS Interception*. doi: 10.14722/ndss.2017.23456

[5] BOBBY. *How Does HTTPS Work? RSA Encryption Explained*. Top Security. [online]. 2017, poslední aktualizace 10. 7. 2017, [cit. 20. 11. 2019]. Dostupné z URL: <https://tiptopsecurity.com/how-does-https-work-rsa-encryption-explained/>.

[6] DIERKS, T., RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. 2008.

[7] RESCORLA, E. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. 2018.

[8] NOHE, P. *Taking a Closer Look at the SSL/TLS Handshake*. HashedOut. [online]. 2017, poslední aktualizace 30. 4. 2019, [cit. 24. 11. 2019]. Dostupné z URL: <https://www.thesslstore.com/blog/explaining-ssl-handshake>.

[9] LARSON, S. *After Heartbleed, "Forward Secrecy" Is More Important Than Ever*. ReadWrite. [online]. 2014, poslední aktualizace 15. 4. 2014, [cit. 24. 11. 2019]. Dostupné z URL: <https://readwrite.com/2014/04/15/heartbleed-perfect-forward-secrecy-security-encryption/>.

[10] NOHE, P. *Stop using RSA key exchange*. HashedOut. [online]. 2018, poslední aktualizace 4. 12. 2018, [cit. 24. 11. 2019]. Dostupné z URL: <https://www.thesslstore.com/blog/bleichenbachers-cat-rsa-key-exchange/>.

[11] Wikipedie a přispěvatelé. *Autentizované šifrování*. [online]. 2019, poslední aktualizace 23. 6. 2019, [cit. 27. 11. 2019]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Autentizovan%C3%A9_%C5%A1ifrov%C3%A1n%C3%AD>.

[12] SULLIVAN, N. *Introducing Zero Round Trip Time Resumption (0-RTT)*. Cloudflare. [online]. 2017, poslední aktualizace 15. 3. 2017, [cit. 4. 12. 2019]. Dostupné z URL: <`https://blog.cloudflare.com/introducing-0-rtt/`>.

[13] YLONEN, T., LONVICK, C. *The Secure Shell (SSH) Protocol Architecture*. RFC 4251. 2006.

[14] YLONEN, T., LONVICK, C. *The Secure Shell (SSH) Transport Layer Protocol*. RFC 4253. 2006.

[15] YLONEN, T., LONVICK, C. *The Secure Shell (SSH) Authentication Protocol*. RFC 4252. 2006.

[16] YLONEN, T., LONVICK, C. *The Secure Shell (SSH) Connection Protocol*. RFC 4254. 2006.

[17] TASCHNER, CH. *SSH CBC vulnerability*. Carnegie Mellon University CERT. [online]. 2008, poslední aktualizace 12. 1. 2009, [cit. 30. 11. 2019]. Dostupné z URL: <`https://www.kb.cert.org/vuls/id/958563/`>.

[18] RESCORLA, E. *HTTP Over TLS*. RFC 2818. 2000.

[19] HOFFMAN, P., MCMANUS, P., ICANN, MOZILLA. *DNS Queries over HTTPS (DoH)*. RFC 8484. 2018.

[20] *DNS-over-HTTPS performance*. SamKnows. [online]. 2019, poslední aktualizace 9. 8. 2019, [cit. 10. 5. 2020]. Dostupné z URL: <`https://www.samknows.com/blog/dns-over-https-performance`>.

[21] KLENSIN, J. *Simple Mail Transfer Protocol*. RFC 5321. 2008.

[22] MALEK, P. *STARTTLS vs SSL vs TLS*. Mailtrap. [online]. 2019, poslední aktualizace 8. 10. 2019, [cit. 10. 5. 2020]. Dostupné z URL: <`https://blog.mailtrap.io/starttls-ssl-tls/`>.

[23] HOFFMAN, P. *SMTP Service Extension for Secure SMTP over Transport Layer Security*. RFC 3207. 2002.

[24] MOORE, K., WINDROCK, Inc., NEWMAN, C., ORACLE. *Cleartext Considered Obsolete: Use of Transport Layer Security (TLS) for Email Submission and Access*. RFC 8314. 2018.

[25] SAHOO, D. *SMTP Port 25, 465, 587 or 2525 – How to Choose The Right SMTP Port?*. Pepipost. [online]. 2020, poslední aktualizace

15. 5. 2020, [cit. 18. 5. 2020]. Dostupné z URL: <https://pepipost.com/blog/25-465-587-2525-choose-the-right-smtp-port/>.

[26] *Service Name and Transport Protocol Port Number Registry*. IANA. [online]. Poslední aktualizace 13. 5. 2020, [cit. 15. 5. 2020]. Dostupné z URL: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

[27] WEINBERG, N., JOHNSON, J. *MPLS explained*. Network World. [online]. 2003, [cit. 3. 11. 2019]. Dostupné z URL: <https://www.networkworld.com/article/2297171/network-security-mpls-explained.html>.

[28] QUITTEK, J., ZSEBY, T., CLAISE, B., ZANDER, S. *Requirements for IP Flow Information Export (IPFIX)*. RFC 3917. 2004.

[29] STEINBERGER, J., SCHEHLMANN, L., ABT, S., BAIER, H. (2013). *Anomaly Detection and Mitigation at Internet Scale: A Survey*. 49-60. doi: 10.1007/978-3-642-38998-6_7

[30] MILLS, C., HIRSH, D., RUTH, G. *INTERNET ACCOUNTING: BACKGROUND*. RFC 1272. 1991.

[31] HOGG, S. *History of flow analysis protocols and how we use them*. Network World. [online]. 2014, poslední aktualizace 14. 3. 2014, [cit. 5. 12. 2019]. Dostupné z URL: <https://www.networkworld.com/article/2226538/your-father-s-flow-export-protocol--part-1-.html>.

[32] HOFSTEDE, R., ČELEDA, P., TRAMMELL, B., DRAGO, I., SADRE, R., SPEROTTO, A., PRAS, A. *Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX*. IEEE Communications Surveys and Tutorials, IEEE, 2014, roč. 16, č. 4, s. 2037-2064. ISSN 1553-877X. doi: 10.1109/COMST.2014.2321898.

[33] *IP Flow Information Export (IPFIX) Entities*. IANA. [online]. 2005, poslední aktualizace 25. 7. 2019, [cit. 3. 12. 2019]. Dostupné z URL: <https://www.iana.org/assignments/ipfix/ipfix.xml>.

[34] CLAISE, B., TRAMMELL, B. *Information Model for IP Flow Information Export (IPFIX)*. RFC 7012. 2013.

[35] *IPFIX message format*. IBM Knowledge Center. [online]. 2017, [cit. 5. 12. 2019]. Dostupné z URL: <https://www.ibm.com/support/knowledgecenter/en/SSCVHB_1.2.1/collector/cnpi_ipfix_message_format.html>.

[36] BARTOŠ, V. *Heartbleed Detection at CESNET using Extended Flow Monitoring.* Proceedings of 8th International Scientific Conference on Security and Protection of Information, 2015.

[37] ANDERSON, B., MCGREW, D. *Identifying Encrypted Malware Traffic with Contextual Flow Data.* AISec '16. 2016. New York, NY, USA: ACM, 2016, pp. 35–46. doi: 10.1145/2996758.2996768.

[38] HUSÁK, M., VELAN, P., VYKOPAL, J. *Security Monitoring of HTTP Traffic Using Extended Flows.* 2015 10th International Conference on Availability, Reliability and Security, Toulouse, 2015, pp. 258-265. doi: 10.1109/ARES.2015.42.

[39] ČEJKA, T., BARTOŠ, V., TRUXA, L., KUBÁTOVÁ, H. *Using Application-Aware Flow Monitoring for SIP Fraud Detection.* IFIP International Federation for Information Processing 2015 S. Latré et al. (Eds.): AIMS 2015, LNCS 9122, pp. 87–99, 2015. doi: 10.1007/978-3-319-20034-7_10.

[40] JANSKY, T., ČEJKA, T., BARTOŠ, V. *Hunting SIP Authentication Attacks Efficiently.* D. Tuncer et al. (Eds.): AIMS 2017, LNCS 10356, pp. 125–130, 2017. doi: 10.1007/978-3-319-60774-0 9.

[41] ČELEDA, P., VELAN, P., KRÁL, B., KOZÁK, O. *Enabling SSH Protocol Visibility in Flow Monitoring.* 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 2019, pp. 569-574.

[42] ANDERSON, B., SUBHARTHI, P., MCGREW, D. *Deciphering Malware's use of TLS (without Decryption).* Journal of Computer Virology and Hacking Techniques. 2016. doi: 10.1007/s11416-017-0306-6.

[43] MCGREW, D., ANDERSON, B., PERRICONE, P., HUDSON, B. *Cisco Joy* [online]. GitHub repozitář. 2018, poslední aktualizace 8. 11. 2019, [cit. 9. 12. 2019]. Dostupné z URL: <`https://github.com/cisco/joy`>.

[44] YEGULALP, S. *What is Jupyter Notebook? Data analysis made easier.* InfoWorld. [online]. 2019, poslední aktualizace 22. 4. 2019, [cit. 2. 5. 2020]. Dostupné z URL: <`https://www.infoworld.com/article/3347406/what-is-jupyter-notebook-data-analysis-made-easier.html`>.

[45] *Anaconda, Individual Edition.* [online]. 2020, [cit. 7. 5. 2020]. Dostupné z URL: <`https://www.anaconda.com/products/individual`>.

[46] HYNEK, K., BENEŠ, T., ČEJKA, T., KUBÁTOVÁ, H. *Refined detection of SSH brute-force attackers using machine learning*. In Proceedings of the 35th International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2020), 21-23 September 2020, Maribor, Slovenia (bude publikováno).

Seznam symbolů, veličin a zkratek

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
API	Application Programming Interface
APT	Advanced Persistent Threat
CBC	Cipher Block Chaining mode
CTR	Counter mode
DBMS	Database Management System
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DNS	Domain Name System
DoH	DNS over HTTPS
DoS	Denial of Service
DoT	DNS over TLS
DSA	Digital Signature Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
FQDN	Fully Qualified Domain Name
HKDF	HMAC-based Key Derivation Function
HMAC	Hash-based Message Authentication Code
HTTPS	Hyper Text Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IPFIX	IP Flow Information Export
IPS	Intrusion Prevention System
ISO/OSI	International Organization for Standardization / Open Systems Interconnection
ISP	Internet Service Provider
JSON	JavaScript Object Notation
KDE	Kernel Density Estimation
MAC	Message Authentication Code
MD5	Message-Digest Algorithm
MITM	Man in the Middle
MPLS	Multiprotocol Label Switching
MX	Mail Exchange
NTP	Network Time Protocol

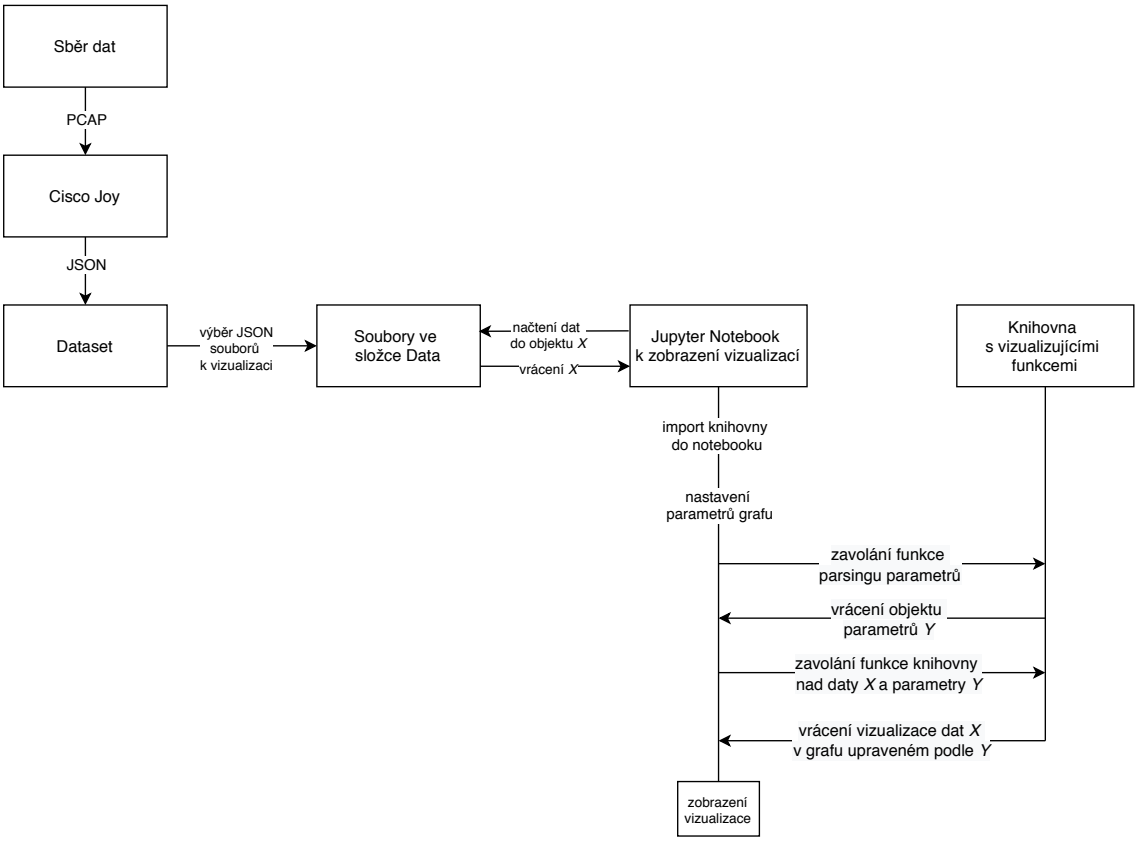
ping	Packet Internet Groper
RC4	Arcfour
RSA	Rivest, Shamir, Adleman algorithm
RTT	Round Trip Time
SCTP	Stream Control Transmission Protocol
SHA	Secure Hash Algorithm
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNI	Server Name Indication
SNMP	Simple Network Management Protocol
SOC	Security Operations Center
SPLT	Sequence of Packet Lengths and Times
SRLT	Sequence of Record Lengths and Times
SSH	Secure Shell
SSL	Secure Socket Layer
SSM	Source-Specific Multicast
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTL	Time To Live
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoIP	Voice over IP
3DES	Triple Data Encryption Standard

Seznam příloh

A Schéma **71**

B Obsah přiloženého CD **72**

A Schéma



Obr. A.1: Blokové schéma průběhu zpracování dat a vizualizace.

B Obsah přiloženého CD

Notebooky byly vytvářeny v programu Jupyter Notebook obsaženém v distribuci Anaconda3 2019.10 s Pythonem verze 3.7.4.

- `/` ..... kořenový adresář přiloženého CD
  - `Data` ..... vybraná data k načtení přednastavenými notebooky
  - `Dataset` ..... datové sady
    - `JSON` ..... exportované rozšířené flow záznamy ve formátu .json
      - `DoH` ..... 30 vzorků DNS over HTTPS
      - `HTTP+HTTPS` ..... 150 vzorků HTTP/S
      - `SMTPS`
        - `STARTTLS 587` ..... 40 vzorků SMTP na port 587
        - `TLS 465` ..... 40 vzorků SMTP na port 465
      - `SSH` ..... 25 vzorků SSH
    - `PCAP` ..... původní vzorky dat ve formátu .pcap
      - `DoH`
      - `HTTP+HTTPS`
      - `SMTPS`
      - `SSH`
  - `lib` ..... knihovny a potřebné soubory
    - `ciphersuites.txt` ..... zkratky cipher suites namapované k názvům a síle
    - `flow_analysis.py` ..... knihovna pro porovnávací notebook
    - `flow_stat_analysis.py` ..... knihovna pro statistický notebook
  - `BP_Hluckova-Pavla_xhluck01.pdf` ..... text bakalářské práce
  - `Vizualizace IP toků a porovnání.ipynb` ..... porovnávací notebook
  - `Vizualizace IP toků a statistiky.ipynb` ..... statistický notebook