VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

IDENTIFIKACE APLIKACÍ V SÍŤOVÉM PROVOZU

APPLICATION IDENTIFICATION IN NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAKUB POMSÁR**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. IVANA BURGETOVÁ, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

164305

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Pomsár Jakub**
Program:	Informační technologie
Název:	**Identifikace aplikací v síťovém provozu**
Kategorie:	Data mining
Akademický rok:	2024/25

Zadání:

1. Seznamte se se způsoby síťové komunikace mobilních i desktopových aplikací a podrobně prostudujte protokol TLS.
2. Seznamte se s dostupnými anotovanými datovými sadami obsahujícími TLS komunikaci aplikací a prostudujte možné způsoby identifikace aplikací na základě této komunikace.
3. Navrhněte nový způsob identifikace aplikací v síťovém provozu založený na TLS datech, zaměřte se na kontextový přístup.
4. Po dohodě s vedoucí práce navrhněte vhodný způsob hodnocení úspěšnosti identifikace aplikací.
5. Implementujte navržený způsob identifikace aplikací a otestujte ho na vhodném vzorku dat.
6. Zhodnoťte dosažené výsledky.

Literatura:

- AGGARWAL, Charu C a HAN, Jiawei. *Frequent pattern mining*. *9783319078212*. 2014. Cham: Springer, 2014. ISBN 9783319078205. Dostupné z: https://doi.org/10.1007/978-3-319-07821-2.
- MATOUŠEK Petr, BURGETOVÁ Ivana a VICTOR Malombe. Mobile Device Fingerprinting. FIT-TR-2020-05, Brno, 2020.
- Matoušek, Petr: Síťové aplikace a jejich architektura, Brno: VUTIUM, 2014, 396 s., ISBN 978-80-214-3766-1.

Při obhajobě semestrální části projektu je požadováno:

- Body 1. - 4.

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

Vedoucí práce:	**Burgetová Ivana, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	22.10.2024

Abstrakt

Cílem této práce je úspěšně zúžit výslednou množinu kandidátních aplikací při použití metod pro klasifikaci TLS spojení za účelem identifikace aplikací. Za tímto účelem jsou vytěženy frekventované vzory obsahující otisky okolních spojení, které jsou využity k určení nejpravděpodobnější aplikace. Práce těží frekventované vzory pomocí algoritmu *Apriori*; jako vstupní data jsou použity otisky *JA3/JA4* a jejich kombinace. Podařilo se dosáhnout téměř totožné úspěšnosti jako při použití samotných otisků, ale s výrazně zúženou kandidátní množinou a přijatelnou časovou náročností. Výsledky této práce umožňují efektivnější klasifikaci šifrovaného provozu generovaného aplikacemi.

Abstract

The goal of this thesis is to successfully reduce the resulting set of candidate applications when using methods for TLS connection classification for the purpose of application identification. To achieve this, frequent patterns containing fingerprints of surrounding connections are extracted and used to determine the most probable application. The work mines frequent patterns using the Apriori algorithm; JA3/JA4 fingerprints and their combinations are used as input data. It was possible to achieve nearly the same accuracy as when using the fingerprints alone, but with a significantly reduced candidate set and acceptable computational cost. The results of this work enable more effective classification of encrypted traffic generated by applications.

Klíčová slova

Identifikace, kandidátní množina, otisky *JA3*, otisky *JA4*, algoritmus *Apriori*, protokol *TLS*, aplikace, šifrovaný provoz, navázání spojení

Keywords

Identification, candidate set, *JA3* fingerprints, *JA4* fingerprints, *Apriori* algorithm, *TLS* protocol, applications, encrypted traffic, handshake

Citace

POMSÁR, Jakub. *Identifikace aplikací v síťovém provozu*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Identifikace aplikací v síťovém provozu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením paní Ing. Ivany Burgetové, PhD. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jakub Pomsár
13. května 2025

Poděkování

Rád bych poděkoval své vedoucí práce, paní Ing. Ivaně Burgetové, Ph.D., za odborné vedení, cenné rady a podporu, kterou mi během zpracování této práce věnovala. Její zpětná vazba a vedení mi velmi pomohly. Dále děkuji své rodině a blízkým za podporu a pochopení během celého studia.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době téměř každá aplikace, která komunikuje s okolním světem, zajišťuje bezpečnost a soukromí svých uživatelů šifrováním své komunikace. V momentě, kdy určitá aplikace generuje nežádoucí provoz, nebo je třeba identifikovat aplikace, které představují zranitelné místo v síti, je nutné tyto aplikace správně rozpoznat. K tomu se využívá právě ono šifrované spojení, pro které existují metody klasifikace, jež umožňují extrahovat tzv. otisk – jednoduše, rychle a s poměrně dobrou přesností.

Výsledky těchto metod však zpravidla nejsou jednoznačné. Ve většině případů mohou být výsledky až matoucí, zejména kvůli velkému počtu aplikací, které vystupují jako kandidáti pro dané spojení.

Cílem této práce je tedy pokusit se zúžit výslednou skupinu kandidátů na aplikace za využití uvedených metod a frekventovaných otisků z okolních spojení. Pro každé spojení bude použita metoda pro identifikaci a následně se určí nejpravděpodobnější aplikace na základě vytěžených frekventovaných vzorců. Je zároveň žádoucí, aby byla zachována přesnost a aby využití okolních spojení a dalších informací nebylo příliš časově náročné.

V následující kapitole jsou uvedeny potřebné znalosti síťových protokolů. V navazující části jsou popsány současné metody pro identifikaci šifrované komunikace. Teoretické základy pro těžbu frekventovaných vzorů se nacházejí ve čtvrté kapitole. Návrh a implementace řešení jsou uvedeny v páté kapitole. Předposlední kapitola se věnuje analýze a testování chování systému, aby bylo ověřeno splnění stanovených požadavků. Závěr tvoří sedmá kapitola této práce.

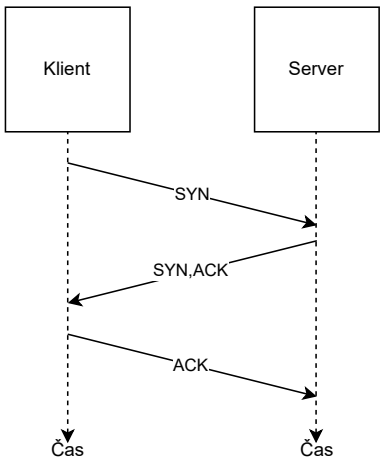
Kapitola 2

Potřebné znalosti sítových protokolů

Tato kapitola se zaměřuje na nezbytné znalosti potřebné k pochopení síťové problematiky. Představuje základní protokoly, jako je *Transmission Control Protocol* (dále označovaný jako *TCP*), jehož role je klíčová při navazování spojení a komunikaci. Podrobně vysvětluje *Transport Layer Security* (dále označovaný jako *TLS*), což je protokol pro standardní zajištění bezpečné komunikace, zejména fázi navázání spojení, poskytující bezpečnostní vlastnosti, strukturu zpráv a rozdíly verzí *v1.2* a *v1.3*.

2.1 Transmission Control Protocol – TCP

TCP je nejpoužívanějším protokolem *transportní vrstvy* v sadě protokolů *TCP/IP*, poskytuje spolehlivou službu přenosu dat po bytech pro aplikace. Aplikační přenos dat po bytech je přenášen přes síť prostřednictvím TCP segmentů, přičemž každý TCP segment je odeslán jako datagram protokolu IP (Internet Protocol). Spolehlivost TCP spočívá v detekci ztráty segmentů (pomocí pořadových čísel) a chyb (pomocí kontrolních součtů pro každý segment) a také v jejich opravě prostřednictvím opětovného přenosu [9]. Předtím, než jeden proces může začít posílat data druhému, musí se nejdříve oba vzájemně „domluvit“, tedy provedou takzvaný *handshake* [14]. To znamená, že si musí vyměnit několik úvodních segmentů, aby stanovily parametry následného přenosu dat. Handshake obsahuje přesně 3 segmenty. První dva nenesou žádnou zátěž, tedy žádná data aplikační vrstvy, třetí segment ji nést může. Z počtu segmentů při ustanovení spojení také vyplývá název „three-way handshake“ („třífázové navázáni spojení“ – nadále bude využívána anglická forma, jelikož je to běžné i mezi odborníky v oboru), jak je znázorněno v diagramu 2.1. Při komunikaci se pro ověření správného pořadí segmentů a případné vyžádání opětovného zaslání ztraceného segmentu používají *sekvenční* (sequence) a *potvrzovací* (acknowledgment) čísla. Tyto čísla se rovněž používají ve fázi *three-way handshake*.



Obrázek 2.1: Handshake TCP

Navázání spojení probíhá následovně:

1. Klient odešle na server datagram s příznakem *SYN*[1], náhodně vygenerovaným pořadovým číslem X a potvrzovacím číslem 0.

2. Server odešle klientovi datagram s příznaky *SYN* a *ACK*[2], potvrzovacím číslem $X+1$ a náhodně vygenerovaným pořadovým číslem Y.

3. Klient dokončuje navázání datagramem s příznakem *ACK*, pořadovým číslem $X+1$ a potvrzovacím číslem odpovědi $Y+1$.

Jak již bylo zmíněno, TCP je jedním z nejpoužívanějších spolehlivých protokolů v dnešní době. Zajišťuje spolehlivé doručování dat mezi dvěma koncovými body, avšak má také své neduhy. Jedním z nejvýraznějších je absence jakéhokoli šifrování, což znamená, že veškerá data přenášená pomocí TCP jsou odesílána v otevřené, nešifrované podobě. Tato slabina může být využita k odposlechu nebo manipulaci s daty, což představuje bezpečnostní riziko.

Na řadu tedy přichází *TLS* protokol, který poskytuje *šifrování* a *autenticitu* na transportní vrstvě[3], a tím umožňuje bezpečný přenos aplikačních dat. V následující podkapitole 2.2 jsou blíže popsány principy fungování a využití *TLS* v kombinaci s *TCP*.

2.2 Transport Layer Security – TLS

Šifrování, autentizace a integrita jsou tři zásadní bezpečnostní funkce, které *TCP* postrádá a *TLS* naopak poskytuje. Tento protokol je nástupcem *SSL*[4], který byl nahrazen kvůli závažným bezpečnostním chybám a nedostatečné odolnosti vůči útokům. *TLS* nabízí modernější šifrovací algoritmy a lepší zabezpečení, čímž překonává nedostatky *SSL*. V dnešní době je *SSL* prakticky historií, a proto se tato práce zaměřuje výhradně na komunikaci pomocí *TLS*. Vzhledem k tomu, že v poskytnutých datových sadách, jak je zmíněno v sekci 5.1, se převážně objevuje *TLS* **verze 1.2**, bude pozornost této sekce věnována právě této verzi.

Protokol se skládá ze dvou vrstev: *TLS Record Protocol* a *TLS Handshake Protocol*. *Record Protokol* zapouzdřuje různé protokoly vyšších vrstev. Jeden z těchto zapouzdřených protokolů je *TLS Handshake*, který umožňuje serveru a klientovi vzájemnou autentizaci a vyjednání šifrovacích algoritmů a kryptografických klíčů před zahájením komunikace aplikačních protokolů.

TLS Record Protocol následně přebírá data aplikační vrstvy, která mají být přenesena. Operuje nad těmito daty, která jsou nejprve fragmentována, případně komprimována, je k nim přidán kód MAC (Message Authentication Code – kód autentizace zprávy), poté jsou zašifrována a přidána hlavička protokolu. Takto vytvořená datová jednotka je poté předána do TCP segmentu pro přenos [26].

Jednou z výhod je skutečnost, že *TLS* je nezávislý na aplikačním protokolu a datech aplikační úrovně, které šifruje. Protokoly vyšší úrovně mohou na něj navazovat **transparentně**. Kromě snad nejpoužívanějšího aplikačního protokolu HTTP se TLS využívá také pro protokoly jako například *FTP*, *SMTP*, *NNTP* a další. TLS není omezeno pouze na funkčnost nad TCP, ale dokáže zabezpečit komunikaci stejně přes *UDP* či *DCCP*. Díky již zmíněné

[1] Synchronize sequence number (česky synchronizace pořadového čísla)
[2] Acknowledgment (česky potvrzení)
[3] TLS nelze přesně zařadit k jedné vrstvě ISO/OSI modelu
[4] Secure Sockets Layer (vrstva bezpečných schránek)

transparentnosti vůči přenášenému aplikačnímu protokolu může také poskytovat zabezpečení při VPN spojení, typicky v případě použití *OpenConnect* nebo *OpenVPN* [10]. Další možností využití je při e-mailové komunikaci pomocí příkazu *STARTTLS* [16].

Standard však nespecifikuje, jak protokoly implementují zabezpečení pomocí *TLS*. Rozhodnutí o tom, jak zahájit *handshake* a jak interpretovat autentizační certifikáty, které jsou vyměněny, jsou ponechána na uvážení návrhářů a vývojářů protokolů operujících nad *TLS* [24].

Dle *ISO/OSI* modelu nelze protokol *TLS* správně klasifikovat, jelikož operuje především nad *TCP*, které se nachází ve 4. vrstvě, a samotné *TLS* poskytuje bezpečné navázání a udržení relace, při fázi *handshake*, což je funkce 5. vrstvy (*relační*). Zároveň šifruje aplikační data, což je funkcí prezentační vrstvy. Z pohledu šifrování, které je hlavní funkcí, lze *TLS* nejlépe zařadit do 6. vrstvy (*prezentační*). Nicméně klasifikace *TLS* pouze do jedné vrstvy není přesná, protože plní funkce napříč několika vrstvami modelu *ISO/OSI*.

2.2.1 Základní bezpečnostní funkce

Tato podkapitola, převzata a upravena z [24], se zaměřuje na základní bezpečnostní funkce protokolu TLS, konkrétně na symetrickou kryptografii, integritu dat a autentizaci, které společně zajišťují bezpečnost komunikace.

Důvěrnost

Symetrická kryptografie je základní metodou pro šifrování přenášených dat. Pro šifrování se používají různé algoritmy, jako je *AES* (Advanced Encryption Standard), který je v dnešní době považován za velmi bezpečný a široce používaný. Starší algoritmy, jako *RC4*, byly dříve běžně nasazovány, ale kvůli nalezeným zranitelnostem a slabinám jsou již považovány za zastaralé a jejich použití se nedoporučuje [22].

Důvěrnost je zajištěna tím, že klíče používané pro šifrování jsou unikátně generovány pro každou relaci (tzv. „session“). Tyto klíče se odvozují z tajemství, které je bezpečně vyjednáno během *handshake* fáze. Nejběžněji používanými metodami jsou výměny klíčů na bázi algoritmů *Diffie-Hellman* a *RSA*, který se používá pro asymetrické šifrování. Protokol *TLS* rovněž umožňuje komunikaci bez šifrování, což však odporuje hlavnímu účelu protokolu.

Integrita dat

Kontrola integrity je klíčová pro zajištění, že data nebyla během přenosu pozměněna třetí stranou. Přenos zpráv zahrnuje kontrolu integrity zpráv pomocí hodnoty *MAC*. Pro výpočet MAC se používají bezpečné hašovací algoritmy, jako je *SHA-256*, který se stal standardem pro moderní aplikace. Starší algoritmy, jako je *MD5* a *SHA-1*, byly dříve široce používány, ale kvůli objevení kolizí a zranitelností se již nedoporučují [27].

Autentizace

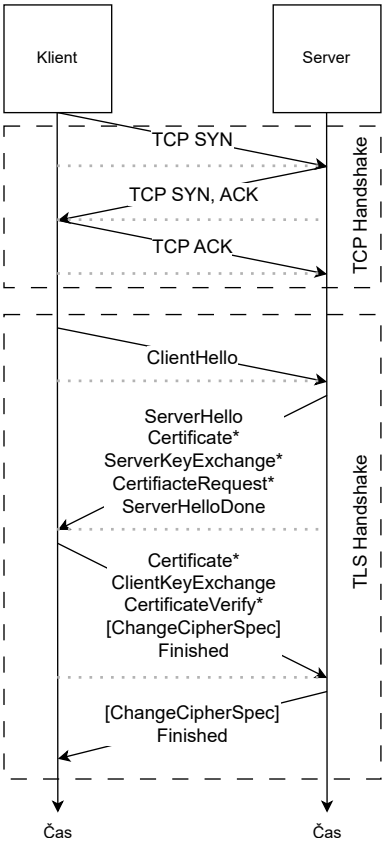
Autentizace se nejčastěji provádí pomocí asymetrické kryptografie. Servery se autentizují vůči klientům pomocí digitálních certifikátů, které obsahují veřejný klíč serveru a jsou podepsány důvěryhodnou certifikační autoritou (CA). Autentizace klientů je volitelná, ale v některých případech, například v případě přístupu k citlivým systémům, může být požadována. Nejčastěji se používají algoritmy jako *RSA* nebo *ECDSA*.

Tyto základní bezpečnostní funkce symetrické kryptografie, integrity dat a autenticity společně zajišťují důvěryhodnost a bezpečnost moderní internetové komunikace, čímž umožňují uživatelům důvěřovat přenášeným informacím.

2.2.2 Navázání spojení

Počátkem každé komunikace je navázání spojení mezi účastníky. U TLS tomu není jinak a, jak již bylo zmíněno, tato fáze se nazývá „TLS handshake“. Po navázání a ustanovení spojení protokolem *TCP*, k čemuž slouží *three-way handshake*, jenž je podrobně rozepsán v 2.1, se *TLS* pokusí navázat bezpečné spojení a vyjednat detaily budoucí komunikace. Pro lepší představu je znázorněno diagramem 2.2.

Postup ustanovení spojení mezi klientem a serverem probíhá následovně:



Obrázek 2.2: Handshake TLS

1. Klient zašle zprávu *ClientHello*, obsahující seznam kryptografických algoritmů, které podporuje, spolu s náhodně vygenerovaným číslem (označovaným jako „nonce“[5]).

2. Ze seznamu je serverem vybrán algoritmus pro **symetrickou** kryptografii, algoritmus pro **asymetrickou** kryptografii a typ ***HMAC*** (Hash-based Message Authentication Code, česky autentizační kód zprávy). Dále je zvolena **hašovací funkce**, např. *MD5*, *SHA-2*,... Vybrané algoritmy jsou zaslány zpět klientovi spolu s certifikátem serveru a náhodně vygenerovaným číslem (nonce), jako součást zprávy *Server Hello*.

3. Klient ověří certifikát, získá veřejný klíč serveru, vygeneruje *Pre-Master Secret* (dále označovaný jako „PMS“), zašifruje *PMS* veřejným klíčem serveru a odešle šifrovaný *PMS* serveru.

4. Pomocí stejné funkce pro odvození klíče, jak je stanoveno standardem *TLS* [24], klient a server nezávisle vypočítají *Master Secret* (dále označovaný jako „MS“) z *PMS* a nonce. *MS* se poté rozdělí na dva šifrovací klíče a dva *HMAC* klíče. Dále, pokud vybraný symetrický šifrovací algoritmus používá režim *CBC* (Cipher Block Chaining, česky „Řetězení šifrovacích bloků“, a dále označován zkratkou *CBC*), například *3DES* nebo *AES*, jsou z *MS* získány také dva inicializační vektory (*IV*) –jeden pro každou stranu spojení. Následně jsou všechny zprávy mezi klientem a serverem šifrovány a autentizovány (pomocí *HMAC*). Obě strany si vzájemně zašlou *HMAC* vypočítaný na základě všech předchozích zpráv, aby ověřily integritu celé fáze navazování a vyjednávání spojení.

[5]Kryptografická nonce je označení pro náhodné číslo, které lze použít pouze jednou. Jeho přítomnost zvyšuje obtížnost podvržení zprávy.

Proces navazování spojení, odehrávající se v 5.vrstvě (*relační*) *ISO/OSI* modelu, pomocí protokolu *TLS Handshake* je klíčovým prvkem k zajištění bezpečné komunikace mezi dvěma stranami. Správný výběr algoritmů a hašovacích funkcí, ověřování HMAC a použití nonce jsou zásadní pro ochranu dat a zajištění důvěrnosti. V další podsekci 2.2.3 je podrobně popsán obsah těchto zpráv, které nejsou šifrovány a využívají se k tvorbě TLS otisku. Bližší popis TLS otisku je v kapitole 3.

2.2.3 Obsah zprávy Client Hello

V této podsekci je popsán obsah zpráv *Client Hello*. Následující struktura *Client Hello* zpráv je převzata z **RFC 5246** [24].

Struktura *Client Hello* je definována následovně:

```
struct
{
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present)
    {
        case false:
          struct {};
        case true:
          Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

Kde:

- **client_version** značí verzi *TLS* protokolu, kterou klient podporuje (např. 771 – pro *TLS v1.2*),
- **random** je klientská nonce,
- **session_id** označuje identifikátor relace,
- **cipher_suites** reprezentuje seznam šifrovacích sad podporovaných klientem,
- **compression_methods** značí seznam metod komprese podporovaných klientem,
- **extensions_present** indikuje přítomnost rozšíření,
- **extensions**, je kolekce rozšíření poskytujících dodatečné zabezpečení spojení. Běžná rozšíření jsou například *supported_groups*, které identifikuje eliptické křivky podporované klientem, a *ec_point_formats*, které specifikuje množinu bodů používaných pro reprezentaci eliptických křivek [20]. Více v 2.2.5.

Zpráva *Client Hello* je zásadním „stavebním kamenem“ *TLS* komunikace, protože propaguje možnosti a vlastnosti klienta, jako jsou podporované verze protokolu, šifrovací sady a rozšíření. Hodnoty vybraných atributů zprávy mohou být použity pro identifikaci pomocí *JA3/JA4* otisků, více viz 3.

2.2.4 Obsah zprávy Server Hello

Následující struktura *Server Hello* zprávy je převzata z **RFC 5246** [24]. Struktura zprávy *Server Hello* je následující:

```
struct
{
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    select (extensions_present)
    {
        case false:
          struct {};
        case true:
          Extension extensions<0..2^16-1>;
    };
} ServerHello;
```

Kde:

- **server_version** značí verzi *TLS* protokolu, kterou server vybral na základě *client_version*,
- **random** je nonce serveru,
- **session_id** označuje identifikátor relace,
- **cipher_suite** reprezentuje jednu šifrovací sadu vybranou z *cipher_suites*,
- **compression_method** značí jednu metodu komprese zvolenou serverem,
- **extensions_present** indikuje přítomnost rozšíření,
- **extensions**, stejně jako u *Client Hello*, určuje jaká rozšíření vybraná serverem jsou podporována. Více v 2.2.5.

Výběr z výše zmíněných elementů zpráv je využit pro tvorbu a následnou identifikaci pomocí *JA3/JA4* otisků, jak je detailněji popsáno dále v kapitole 3. V následující podsekci 2.2.6 jsou rozepsány podrobnosti implementace *TLS* verze *1.3* a také jsou zmíněny změny *Hello* zpráv oproti verzi *1.2*.

2.2.5 Možná rozšíření

Detailní popis všech možných atributů a funkcí těchto zpráv je nad rámec této technické zprávy. Proto jsou uvedeny pouze potřebné informace pro pochopení a následné vysvětlení v kontextu s *JA3* a *JA4* otisky. Z tohoto důvodu jsou zde uvedena pouze nejčastější a informačně nejvýznamnější rozšíření, která jsou používána k tvorbě otisků nebo ke zlepšení přesnosti identifikace.

- **Server Name Indicator** (*Indikátor jména serveru*, dále jen „SNI“) umožňuje klientům poskytnout název serveru, se kterým se spojují. Tato funkce je žádoucí pro usnadnění zabezpečených připojení k serverům, které provozují více „virtuálních“ serverů na jedné adrese [8].
- **Application-Layer Protocol Negotiation** (*Vyjednání aplikačního protokolu*, dále jen „ALPN“) V situaci, kdy je na jednom portu serveru, například portu 443, podporováno více aplikačních protokolů, klient a server musí vyjednat aplikační protokol, který bude použit pro každé připojení. S *ALPN* klient odesílá seznam podporovaných aplikačních protokolů jako součást zprávy *TLS Client Hello*. Server si vybere protokol a odešle vybraný protokol jako součást zprávy *TLS Server Hello* [11]. Vyjednávání aplikačního protokolu tak může být provedeno v rámci protokolu *handshake*, aniž by se přidávaly další síťové výměny, a umožňuje serveru přiřadit k jednotlivým aplikačním protokolům různé certifikáty.

2.2.6 TLS verze 1.3

TLS verze 1.3 je **nejnovější** verzí protokolu *TLS*. Mezi nejvýznamnější rozdíly posledních dvou verzí patří především změny v seznamu povolených šifer u symetrické kryptografie, kde v novější verzi zůstávají pouze **AEAD** (*Authenticated Encryption with Associated Data* – česky autentizované šifrování s připojenými daty) algoritmy. *Handshake* protokol je zrychlen pomocí techniky **0-RTT** (*Zero Round-Trip Time* – česky nulová doba navázání spojení), která šetří jednu cestu tam a zpátky na úkor některých bezpečnostních vlastností. Nová verze také **šifruje** všechny zprávy protokolu *Handshake* po zprávě *Server Hello* [23]. Plánuje se také začít šifrovat *Hello* zprávy, čímž se znemožní identifikaci pomocí pouhých *JA3*/*JA4* otisků [25].

Také byly představeny takzvané **GREASE** (*Generate Random Extensions And Sustain Extensibility*) hodnoty, které jsou přidávány do atributů zpráv handshake protokolu, k ověření správné implementace protější strany. Při správném zpracování musí být tyto hodnoty ignorovány druhou stranou. V případě, že *GREASE* hodnoty nejsou tolerovány, je vyhlášena chyba, která značí nevalidní implementaci [5].

TLS je typicky implementován nad *TCP* protokolem, ovšem verze 1.3 dokáže komunikovat také přes *UDP* (*User Datagram Protocol*) v případě, že je použit protokolem **QUIC** (*Quick UDP Internet Connections*). *QUIC* využívá *UDP* jako transportní protokol a zároveň na něm staví pokročilé funkce, jako je správa spojení a zajištění spolehlivosti [13]. Má zabudované zabezpečení pomocí *TLS v1.3*, díky čemuž poskytuje všechny bezpečnostní výhody *TLS* [28].

Kapitola 3

Otisky JA3, JA4 a jejich použití

Tato kapitola se zabývá pasivními metodami pro identifikaci šifrované komunikace a jejich vývojem od *JA3* k *JA4+*. S narůstajícím trendem šifrování v síťovém provozu bylo třeba vyvinout způsob klasifikace či identifikace šifrovaných dat kvůli správě a analýze komunikace. V roce 2015 byla představena, dnes již populární až možná zastaralou, pasivní metodu tvorby otisků TLS spojení označovanou jako *JA3* otisky[1]. Tvůrci *John B. Althouse*, *Jeff Atkinson* a *Josh Atkins* v rámci společnosti *Salesforce* ve své metodě využívají atributy *Hello* zpráv *TLS* spojení. S příchodem nové verze *TLS v1.3*, která šifruje všechny zprávy protokolu *Handshake* po odeslání *Server Hello* a umožňuje využívat jako transportní protokol *QUIC*, stejně jako další změny, které ztěžují efektivní identifikaci, již nebylo možné aplikovat *JA3* otisky pro tato spojení.

Pro pasivní metody identifikace zabezpečené komunikace se tak objevuje další výzva. V roce 2023 přichází na řadu sada metod *JA4+*[2], od stejných tvůrců, v zastoupení firmy *FoxIO*. Příklady využití těchto otisků zahrnují skenování pro odhalení hrozeb, detekci malwaru, prevenci únosu relace, automatizaci dodržování předpisů, sledování polohy, detekci *DDoS* útoků, seskupování hrozeb, detekci reverzních shellů a mnoho dalších [3].

V následující sekci 3.1 je představen koncept *JA3* otisků a jejich využití. Na ni navazuje sekce 3.2, která seznamuje se sadou metod *JA4+* pro identifikaci různých vlastností šifrovaných spojení, jejichž součástí jsou také *JA4* otisky. Porovnání *JA3* a *JA4* otisků lze nalézt v podsekci 3.3.

3.1 JA3 otisky

JA3 (pojmenování vzniklo podle jmen tří hlavních tvůrců: **J**ohn B. **A**lthouse, **J**eff **A**tkinson a **J**osh **A**tkins) je metoda tvorby otisku *TLS* spojení, která se zaměřuje na různé aspekty a atributy *TLS handshake*. Používá hodnoty ze zpráv *Client Hello* či *Server Hello*, které obsahují několik detailů, jež dokážou jedinečně charakterizovat jak klienta, tak server. Po extrakci těchto informací jsou jednotlivé hodnoty zřetězeny a je proveden výpočet haše typu *MD5*. Výstup hashovací funkce představuje zmíněný *JA3* otisk, který poskytuje konzistentní a identifikovatelný podpis.

Metoda *JA3* využívá následující atributy *Client Hello*: `client_version`, `cipher_suite`, `extensions`, `supported_groups`[3] a `ec_point_formats`[4]. Pro podrobnější popis těchto polí

[1] Více na https://github.com/salesforce/ja3
[2] Více na https://github.com/FoxIO-LLC/ja4
[3] Dříve definováno jako `elliptic_curves`
[4] Dříve definováno jako `elliptic_curve_point_formats`

zprávy viz. sekce 2.2.3. Tyto hodnoty jsou pak spojeny dohromady v uvedeném pořadí, přičemž jednotlivá pole jsou oddělena čárkou (‘,‘) a hodnoty v rámci každého pole jsou odděleny pomlčkou (‘–‘). Musí být brány v potaz také hodnoty *GREASE*; bližší popis lze nalézt v sekci 2.2.6. Při extrakci hodnot z polí jsou buďto ignorovány, nebo nahrazeny konstantou, aby byla zachována konzistence otisku.

Pro lepší porozumění je předvedena názorná ukázka, převzata z [4]. Řetězec po zpracování všech výše zmíněných polí vypadá následovně. Tento řetězec je poté použit jako vstup do hashovací funkce *MD5*, která vytvoří daný otisk.

Řetězec	`769,47-53-5-10-49161-49162-49171-49172-50-56-19-4,0-10-11,23-24,0`
	↓
Otisk	`ada70206e40642a3e4461f35503241d5`

V případě, že nejsou žádná rozšíření dostupná, pole jsou ponechána prázdná:

Řetězec	`769,4-5-10-9-100-98-3-6-19-18-99,,,`
	↓
Otisk	`de350869b8c85de67a350c8d186f11e6`

Tato metoda se dá také aplikovat na zprávu *Server Hello* pro identifikaci serveru. Jedná se o takzvaný *JA3S* otisk. Využívá hodnoty polí `server_version`, `cipher_suite` a `extensions`, jak je podrobněji popsáno v sekci 2.2.4. Postup je poté totožný s tvorbou otisku pro klienta, a to tedy zřetězením a použitím jako vstupu pro hashovací funkci *MD5*.

Ukázka tvorby *JA3S* otisku, také převzata z [4]:

Řetězec	`769,47,65281-0-11-35-5-16`
	↓
Otisk	`4835b19f14997673071435cb321f5445`

V průběhu času se však ukázalo, že *JA3* otisky již nejsou tak přesné, jak bylo potřeba, a to z následujících důvodů:

- **Náhodné uspořádání TLS rozšíření**: Některé prohlížeče začaly náhodně měnit pořadí rozšíření za účelem zmaření snah o identifikaci. *JA3* otisky se spoléhají na sekvenční pořadí, které se náhodným řazením mění, což mělo za důsledek **nespolehlivost** při identifikaci jedinečných klientů.
- **Nekonzistence databází**: Implementace tvorby se lišily pro různé programy a databáze, což vedlo **k různým výsledkům pro stejnou identifikovanou entitu**. Tato nejednotnost bránila efektivnímu a spolehlivému sdílení otisků.
- **Omezený rozsah**: *JA3* se zaměřovaly pouze na prvky v rámci *Hello* zpráv *TLS* spojení. Tento omezený rozsah často postrádal kontext o prostředí klienta. Navíc, s rostoucí popularitou novějších transportních protokolů, jako například *QUIC*, se ukázalo, že **identifikace je neúčinná**.

3.1.1 Spolehlivost a omezení JA3 otisků

Mobilních aplikací pomocí *TLS* otisků lze považovat za praktickou metodu s potenciálním využitím v digitální forenzní analýze. Studie ukázaly, že samotné použití *JA3* otisku není

pro přesnou identifikaci mobilních aplikací dostačující. Spolehlivějších výsledků je dosaženo kombinací *JA3*, *JA3S* a *SNI*, přičemž všechny tyto charakteristiky lze jednoduše získat z *TLS* handshake zpráv [18]. Rovněž byla zkoumána volatilita *TLS* otisků – experimenty ukázaly, že jejich variabilita není výrazná [17].

3.1.2 Shrnutí a přechod na JA4+

JA3 otisky poskytují jednoduchou a rychlou identifikaci *TLS* spojení; ovšem s příchodem nové verze *TLS 1.3* se stávají zastaralými a jejich přesnost klesá. Řešením je využít jiné metody klasifikace nebo použít nového nástupce těchto otisků. V následující sekci 3.2 je probírána sada metod *JA4+*, která slouží jako novější verze *JA3* a přináší s sebou také další možnosti analýzy, a to nejen *TLS* spojení.

3.2 Sada metod JA4+

Kvůli již zmíněným důvodům, které vedly k nepřesnosti identifikace, byl vyvinut nástupce *JA3* otisků, a to *JA4*. Přesněji se jedná o sadu metod *JA4+*, které dokáží identifikovat i jinou komunikaci než pouze *TLS*. V této sadě se nachází jak metody *JA4* a *JA4S* pro tvorbu otisků *TLS*, tak například *JA4HTTP*, *JA4Latency*, *JA4SSH*, *JA4TCP* a další... V rámci této práce je pozornost zaměřena pouze na metody **JA4** a **JA4S**.

3.2.1 Metoda JA4

JA4 přistupuje k identifikaci podobně jako metoda JA3. Využívá určité hodnoty z polí *Hello* zpráv, které buď zřetězí, nebo z nich vypočítá hash pomocí hashovací funkce *SHA256*. Formát otisku je rozdělen do tří částí (a, b, c), oddělených podtržítkem (_). Každou z těchto částí lze označit, což zjednodušuje použití v situacích, kdy je k identifikaci potřeba pouze konkrétní část [3].

3.2.2 Formát otisku JA4

Tato podsekce se zaměřuje na formát otisku *JA4*. Na základě uvedených příkladů lze vidět, jak je otisk strukturován a z jakých částí se skládá. Příklady formátu otisku jsou převzaty z [3]. Formát otisku *JA4* je následující:

$$\text{JA4} = \text{JA4a\_JA4b\_JA4c}$$

Kde:

- **JA4a** představuje konkatenaci vybraných informací o TLS spojení, jako je:
  - Protokol, označený jako `„t"` (*TCP*) nebo `„q"` (*QUIC*),
  - Verze TLS: 1.2 = `„12"`, 1.3 = `„13"`,
  - SNI, `„d"` pro doménu nebo `„i"` pro *IP* adresu,
  - Počet šifrovacích sad,
  - Počet rozšíření,
  - První hodnota *ALPN*.
- **JA4b** je zkrácený hash (12B) (*SHA-256*) nad seřazeným seznamem šifrovacích sad.

- **JA4c** je také zkrácený hash (12B) (*SHA-256*) nad seřazeným seznamem rozšíření a algoritmů pro podpisy (*signature algorithms*) v pořadí, ve kterém se objevují.

Konkrétní příklad může vypadat takto:

$$\text{JA4} = \texttt{t13d1516h2\_acb858a92679\_e5627efa2ab1}$$

Podobně jako u svého předchůdce *JA3*, umožňuje sada *JA4* identifikaci serverové strany *TLS* komunikace prostřednictvím analýzy zprávy *Server Hello*. Tato zpráva, odesílaná v nezašifrované formě, reflektuje volbu serveru na základě parametrů uvedených v předchozí zprávě *Client Hello*. Obsahuje mimo jiné jednu šifrovací sadu vybranou ze seznamu nabízeného klientem a sadu rozšíření, které server zvolil pro navázání spojení. Formát otisku **JA4S** je obdobný:

$$\text{JA4S} = \text{JA4Sa\_JA4Sb\_JA4Sc}$$

Kde:

- **JA4Sa** obsahuje následující vybrané hodnoty:
  - Protokol, označený jako `„t"` (*TCP*) nebo `„q"` (*QUIC*),
  - Verze TLS: 1.2 = `„12"`, 1.3 = `„13"`,
  - Počet rozšíření,
  - První hodnota *ALPN*.
- **JA4Sb** představuje zvolenou šifrovací sadu (*cipher suite*).
- **JA4Sc** je zkrácený hash (*SHA-256*) nad seřazeným seznamem rozšíření.

Konkrétní příklad může vypadat takto:

$$\text{JA4S} = \text{t1204000\_c030\_4e8089b08790}$$

3.2.3 Využití otisků JA4

Použitelnost *JA4* otisku je různorodá. Přesná identifikace jednotlivých aplikací je stále obtížná. Otisk však dokáže poskytnout množinu možných kandidátů, přičemž mohutnost této množiny může být poměrně vysoká. I přesto tato informace napomáhá přesnější klasifikaci [6]. *JA4* lze také využít pro identifikaci škodlivých aplikací (*malwaru*) v síťovém provozu. Ukazuje se, že za použití dalších informací o TLS spojení, jako je kombinace *JA4*, *JA4S* a *SNI*, se přesnost detekce zlepšuje [19].

3.3 Srovnání otisků JA3 a JA4

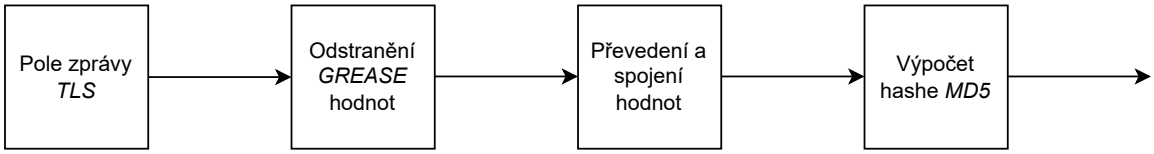
Pro účely porovnání je níže uveden zkrácený výpis z *TLS* spojení se zvýrazněním prvků, které jsou využívány v otiscích *JA3* a *JA4*. (Položka `Extensions` reprezentuje ostatní rozšíření, která nejsou jmenovitě využita ani v jedné z verzí *JA* otisků.)

Legenda

- JA3 – prvek použitý pouze v otisku JA3
- JA4 – prvek použitý pouze v otisku JA4
- JA3 + JA4 – prvek společný pro oba otisky

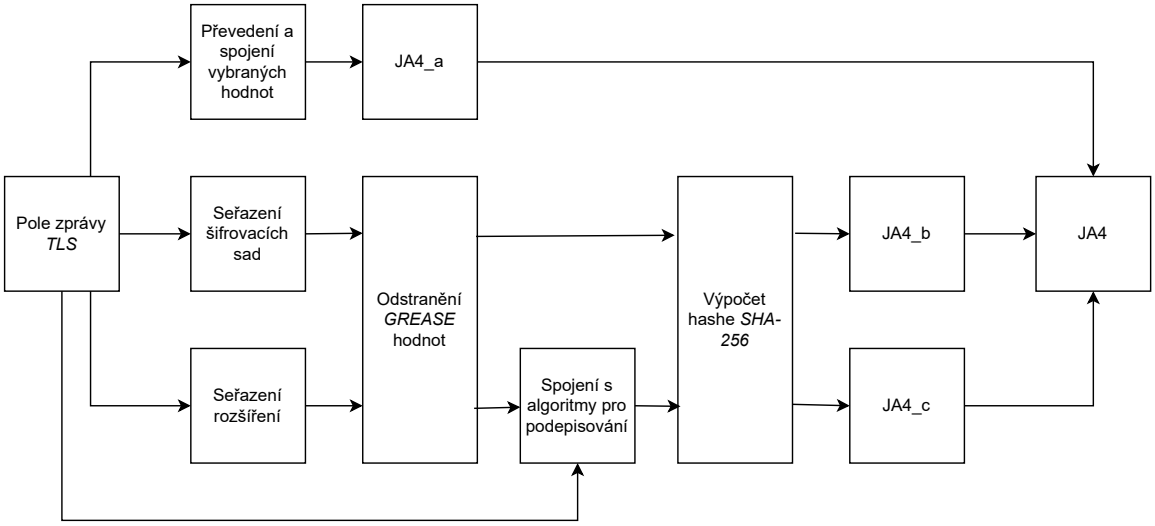
```
Transport Protocol : TCP
Handshake Type: Client Hello (1)
Version: TLS 1.2 (0x0303)
Cipher Suites Length: 34
Cipher Suites (17 suites)
Compression Methods Length: 1
Compression Methods (1 method)
Extensions Length: 560
Extension: server_name name=tasks-pa.clients6.google.com
Extension: elliptic_curves (len=2)
Extension: ec_point_formats (len=2)
Extension: application_layer_protocol_negotiation (len=14)
Extension: signature_algorithms (len=24)
...
Extensions...
```

Jak je patrné, otisky *JA4* využívají širší spektrum informací, například transportní protokol nebo konkrétní rozšíření *TLS*, což vede k jejich vyšší unikátnosti a spolehlivější identifikaci klientů [3]. Rozdíl je také v tom, jak se s těmito atributy nakládá. Starší verze *JA3* odebere tzv. *GREASE* hodnoty, spojí je v daném pořadí, přičemž používá znak „,“ k oddělení jednotlivých polí a znak „-“ k oddělení hodnot v každém poli. Tento řetězec je poté vstupem pro hashovací funkci *MD5* [4]. Jak lze spatřit v diagramu 3.1, jednotlivé kroky tohoto procesu jsou znázorněny sekvenčně.



Obrázek 3.1: Proces tvorby otisků *JA3*

Novější verze *JA4* vytváří otisky komplexnějším způsobem. První část otisku obsahuje zkonkatenizované hodnoty, druhá a třetí část obsahují zkrácený hash *SHA-256* nad vybranými poli zprávy (seřazenými seznamy). Bližší popis je uveden v sekci 3.2 nebo znázorněn diagramem 3.2



Obrázek 3.2: Proces tvorby otisků *JA4*

Z technického hlediska představuje sada metod *JA4+* významný posun oproti svému předchůdci. Zatímco *JA3* představuje jednoduchý a stále využívaný nástroj, zaměřuje se pouze na omezenou množinu atributů TLS zpráv. Naproti tomu *JA4* zohledňuje širší spektrum informací, čímž zvyšuje unikátnost výsledného otisku.

Navíc *JA4* implementuje vícefázový proces tvorby otisku – první část je tvořena explicitními hodnotami, následují hashované části, které zároveň eliminují vliv náhodného přehazování rozšíření a šifrovacích sad. Použití *SHA-256* namísto *MD5* zvyšuje odolnost vůči kolizím.

Díky těmto změnám poskytuje *JA4* výrazně přesnější identifikaci a vyšší robustnost vůči technikám obfuskace, což z něj činí užitečný nástroj pro moderní síťovou analýzu.

Kapitola 4

Teoretické základy použitých algoritmů

Za účelem zvýšení přesnosti identifikace aplikací, a tedy zúžení kandidátní množiny generované pomocí metod *JA3* či *JA4*, je využit algoritmus pro vyhledávání frekventovaných vzorů těchto otisků v kontextu okolních *TLS* spojení. Tato kapitola obsahuje obecný přehled algoritmů pro vyhledávání frekventovaných vzorů, se zvláštním zaměřením na princip a využití algoritmu *Apriori* a jeho optimalizace.

Důvodem volby algoritmu *Apriori* je jeho schopnost efektivně nacházet časté vzory v transakčních datech, což lze analogicky aplikovat na otisky *TLS* spojení. Tento přístup umožňuje identifikovat opakující se kombinace otisků, které jsou charakteristické pro specifické aplikace, a tím efektivně eliminovat kandidáty, jejichž otisky s těmito vzory nekorespondují.

4.1 Algoritmy vyhledávající frekventované vzory

Rozpoznávání frekventovaných vzorů v datech představuje klíčovou a nedílnou součást procesu dolování dat (*data mining*). Tento krok umožňuje identifikovat opakující se vzory a vztahy v rozsáhlých datových souborech, které by jinak zůstaly skryté. V rámci dolování dat se frekventované vzory využívají například pro tvorbu asociačních pravidel, klasifikaci či shlukování.

Typickými příklady využití jsou:

- **Analýza nákupního chování zákazníků**, kde transakce představují množiny položek, které se společně vyskytují v rámci nákupního chování [1].
- **Analýza webových dat**, při které se vyhledávají frekventované vzory ve webových záznamech. Tyto vzory lze následně využít k návrhu či vylepšení uživatelského rozhraní nebo ke zlepšení personalizace obsahu [7].
- **Analýza softwarových chyb**, kde lze běh softwarových programů reprezentovat pomocí grafů obsahujících typické vzory. Logické chyby se často projevují jako specifické vzory, které je možné dále těžit pro účely ladění a diagnostiky [15].

Většina algoritmů pro hledání frekventovaných vzorů je navržena v rámci tradičního modelu založeného na podpoře (*support*) a důvěře (*confidence*). Existují však i specializované

přístupy, které využívají pokročilejší míry zajímavosti, modelují záporná pravidla nebo pracují s omezeními pro nalezení relevantnějších vzorů [1]. Tato práce je zaměřena především na algoritmus *Apriori*, který je založen na přístupu využívajícím podporu a důvěru.

4.2 Základní pojmy a notace

Před popisem samotného algoritmu *Apriori* je vhodné uvést základní pojmy a notace, které se v oblasti dolování častých vzorů a asociativních pravidel běžně používají.

4.2.1 Transakce, položky a jejich množiny

Nechť množina položek $I = \{i_1, i_2, i_3, \ldots, i_n\}$ reprezentuje například zboží v obchodě či *JA3/4* otisky *TLS* spojení. Transakce T je podmnožinou těchto položek, tedy $T \subseteq I$. Množina všech transakcí tvoří transakční databázi $D = \{T_1, T_2, T_3, \ldots, T_m\}$. Množina položek (angl. *itemset*) je libovolná kombinace prvků z množiny I. Pokud množina položek (*itemset*) obsahuje právě k položek, nazýváme jej *k-itemset*.

Množina častých položek (angl. *frequent itemset*) je taková množina položek, která se v transakční databázi vyskytuje alespoň s minimální zadanou četností, zvanou podpora (*support*). Míra *podpory* (*support*) je klíčovým kritériem pro určení toho, zda je daná množina položek považována za častou. Podpora množiny položek vyjadřuje, v kolika transakcích z celé databáze se daná množina vyskytuje. Formálně je definována jako:

$$\text{support}(X) = \frac{\text{počet transakcí obsahujících } X}{\text{celkový počet transakcí}}$$

Kandidátní množiny, jejichž podpora nedosahuje zvoleného prahu *minsup* (minimální podpora), jsou z dalšího zpracování vyřazeny.

4.3 Algoritmus Apriori

Apriori je algoritmus určený pro těžbu častých vzorců a generování asociativních pravidel v transakční databázích. Postupuje tak, že nejprve identifikuje časté jednotlivé položky v databázi a následně je rozšiřuje na větší množiny, dokud se tyto množiny objevují v databázi dostatečně často. Tyto časté vzory mohou být poté použity pro generování asociativních pravidel, která reprezentují trendy a závislosti v databázi.

Algoritmus využívá přístup shora dolů pro generování kandidátů, přičemž kandidáti jsou postupně rozšiřováni o jednotlivé prvky. Skupiny kandidátů jsou následně ověřovány vůči databázi. Při prohledávání se používá prohledávání do šířky. Pro efektivní počítání kandidátních množin je použita struktura hash stromu. Kandidátní množiny položek délky k se nejprve vytvoří na základě množin častých položek délky $k-1$. Jsou odstraněny položky, které neobsahují podmnožinu s častými položkami, a prohledá se databáze transakcí, aby se mezi zbývajícími kandidáty určily skutečné množiny frekventovaných položek [2]. Postup algoritmu *Apriori* lze stručně shrnout pomocí následujícího pseudokódu, jenž byl převzat z [2].

Algoritmus 1: Apriori algoritmus

Vstup : Transakční databáze D, Práh minimální podpory *minsup*
Výstup: Množina častých položek L
$L_1 \leftarrow$ všechny časté množiny délky 1 v D;
$k \leftarrow 2$;
while $L_{k-1} \neq \emptyset$ **do**
  $C_k \leftarrow$ kandidátní množiny položek délky k vygenerované z L_{k-1};
  foreach *transakce* $t \in D$ **do**
    foreach *kandidát* $c \in C_k$ **do**
      if $c \subseteq t$ **then**
        zvýšit počet výskytů kandidáta c;
      end
    end
  end
  $L_k \leftarrow \{c \in C_k \mid support(c) \geq minsup\}$;
  $k \leftarrow k + 1$;
end
return $\bigcup_k L_k$;

Výhodou algoritmu *Apriori* je jeho systematický přístup založený na teorii množin. Nevýhodou je však značná výpočetní náročnost při práci s velkým množstvím transakcí nebo položek, zejména při generování velkého počtu kandidátních množin.

4.3.1 Optimalizace algoritmu Apriori

Hlavní nevýhodou algoritmu *Apriori* je jeho vysoká výpočetní náročnost při generování a vyhodnocování kandidátních množin, zejména v rozsáhlých transakčních databázích. Tento problém se projevuje zejména při práci s dlouhými množinami (*itemsets*) a vysokým počtem unikátních položek (*unique items*), kdy počet potenciálních kombinací roste exponenciálně. V extrémních případech může generování všech možných kombinací položek vést k obrovskému množství kandidátů, z nichž většina se nakonec neukáže jako častá. Tento jev, známý jako kandidátní exploze (*candidate explosion*), významně zatěžuje paměťové i výpočetní zdroje a představuje hlavní slabinu algoritmu *Apriori* [2].

Za účelem omezení výpočetní zátěže bylo navrženo několik optimalizačních technik, které se snaží snížit počet generovaných kandidátních množin (*candidate itemsets*).

Následující nejvýznamnější strategie jsou převzaty z [12]:

Počítání výskytů kandidátů s využitím hašování

Strategie založená na hašování množin položek do odpovídajících bloků může být použita k omezení velikosti kandidátních množin položek C_k pro $k > 1$. Při procházení každé transakce v databázi za účelem generování častých 1-položkových množin je možné vygenerovat všechny 2-položkové kombinace položek pro danou transakci a namapovat je do odpovídajících bloků v hašovací tabulce, čímž se zvýší počet výskytů v daném bloku.

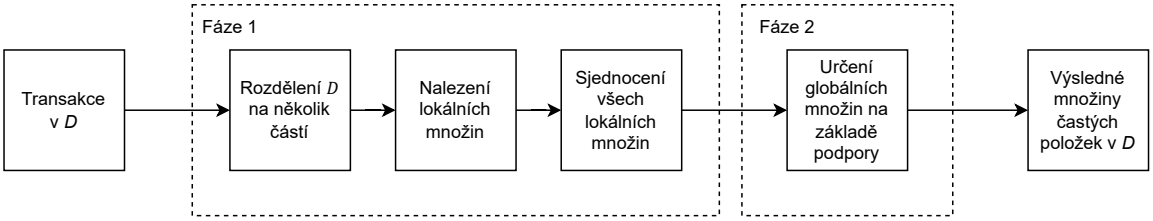
2-položková množina, která má v příslušném bloku počet výskytů nižší než je prahová hodnota podpory, nemůže být častá a může být proto odstraněna z kandidátní množiny. Tato strategie může výrazně snížit počet položkových množin, které je třeba následně vyhodnocovat (zejména v případě, kdy $k = 2$).

Redukce transakcí

Redukce transakcí spočívá ve snížení počtu transakcí, které je nutné prohledávat v následujících iteracích. Transakce, která neobsahuje žádnou častou množinu položek o velikosti k (angl. *frequent k-itemset*), nemůže obsahovat ani žádnou častou množinu položek o velikosti $k + 1$. Takovou transakci je tedy možné zcela vyřadit, neboť při dalším průchodu databází při hledání množin častých položek o velikosti j , kde $j > k$, již nebude relevantní.

Rozdělení (partitioning dat)

Technika rozdělení může být využita ke zjištění množin častých položek při dvou průchodech databází. Skládá se ze dvou fází, které jsou znázorněny na diagramu 4.1.



Obrázek 4.1: Ilustrace principu techniky rozdělení (partitioning), převzat z [12]

1. **Fáze I:** Databáze je rozdělena do n nepřekrývajících se částí (partitions). Pro každou z těchto částí se vypočítá nový lokální práh minimální podpory, definovaný jako *minsup* $\times$ počet transakcí v dané části databáze. Dále se pro každou část nalezne lokální množina častých položek.

   Lokální množina častých položek může, ale nemusí být častá s ohledem na celou databázi D . Nicméně každá množina, která může být potenciálně častá v celé databázi, se musí vyskytnout jako častá alespoň v jedné z částí. Sjednocením častých množin ze všech částí vznikne globální množina kandidátů vůči databázi D .

2. **Fáze II:** Proběhne druhý průchod databází D , při kterém je pro každou kandidátní množinu spočítána její skutečná podpora (*support*) s cílem určit globálně časté množiny položek. Velikost částí a jejich počet jsou voleny tak, aby se každá část vešla do hlavní paměti a bylo ji možné načíst pouze jednou v každé fázi.

Kapitola 5

Návrh a implementace identifikace s využitím JA3/JA4 otisků a frekventovaných vzorů

Tato kapitola se zaměřuje na návrh a implementaci systému pro identifikaci síťových aplikací s využitím otisků *JA3* a *JA4* v kombinaci s analýzou frekventovaných vzorů. Nejprve jsou představeny použité datové sady, jejich původ, struktura a proces předzpracování. Následně je podrobně popsán návrh systému, jeho architektura a zvolené metody pro extrakci charakteristických rysů síťové komunikace. Důraz je kladen na praktickou implementaci řešení, včetně využití algoritmů pro dolování častých vzorů a zpracování otisků v prostředí jazyka *Python*. Cílem této kapitoly je nejen popsat samotný postup realizace systému, ale také zdůvodnit výběr konkrétních technologií a metod, které byly v rámci vývoje zvoleny.

5.1 Seznámení s datovými sadami

Pro účely této práce byly poskytnuty tři datové sady: `mobile_desktop_apps_raw.csv` (autor – Fakulta informačních technologií, Vysoké učení technické v Brně), `iscx.csv` a `iscx-raw.csv` (autor – Univeristy of New Brunswick, Canadian Institute for Cybersecurity). Tyto soubory obsahují záznamy o šifrované síťové komunikaci navázané pomocí protokolu *TLS*, konkrétně o jednotlivých spojeních generovaných různými aplikacemi v kontrolovaném testovacím prostředí. Každý řádek reprezentuje jedno *TLS* spojení a popisuje informace získané jak z klientské zprávy `Client Hello`, tak ze zprávy `Server Hello`, které se vyměňují při navazování šifrovaného spojení.

Datové sady byly využity v předchozích výzkumech zaměřených na analýzu šifrovaného provozu a identifikaci síťových aplikací [18] [6].

Následuje popis jednotlivých souborů:

- **iscx.csv** – Základní verze datové sady, která obsahuje agregované informace o spojeních. Klíčové sloupce zahrnují:
  - `SrcIP`, `DstIP`, `SrcPort`, `DstPort` – informace o zdrojové a cílové *IP* adrese a portech,
  - `SNI` – *Server Name Indication*, doménové jméno cílového serveru,

  - `OrgName`[1] – organizace provozující cílový server (např. Google, Skype),
  - `JA3hash`, `JA4hash` – otisky *TLS* klienta dle metod *JA3* a *JA4*,
  - `JA3Shash`, `JA4Shash` – otisky *TLS* serveru dle metod *JA3S* a *JA4S*,
  - `AppName`[2] – zkratka názvu aplikace generující provoz (např. Skype, Hangouts),
  - `Type` – typ provozu: `0` (běžná aplikace), `M` (škodlivý software), `A` (reklamní či analytický provoz),
  - `Filename`[3] – název původního souboru, ze kterého byl záznam extrahován,
  - `Version` – verze aplikace, (nepoužívaná, vždy `0`).

- **iscx-raw.csv** – Rozšířená verze předešlé sady, obsahující detailní *TLS* metadata včetně obsahu jednotlivých *TLS* parametrů:
  - `Proto` – typ transportního protokolu (např. TCP = `6`),
  - `TLSVersion`, `ClientCipherSuite`, `ClientExtensions`, `SignatureAlgorithms`, `ClientSupportedGroups` – detailní *TLS* parametry zaslané klientem,
  - `ServerCipherSuite`, `ServerExtensions`, `ServerSupportedVersions` – odpovídající parametry na straně serveru,
  - `JA4_raw`, `JA4S_raw` – surové podoby otisků.

- **mobile_desktop_apps_raw.csv** – Podobná struktura jako `iscx-raw.csv`, avšak data pochází z oddělené studie zaměřené na mobilní a desktopové aplikace [18]. Přítomny jsou také sloupce jako:
  - `ALPN` – protokoly vyjednané při *TLS handshake* (např. `http/1.1`, `h2`),
  - `ClientSupportedVersions` – seznam verzí *TLS* podporovaných klientem,

Tabulka 5.1 níže uvádí statistické charakteristiky vybraných atributů obsažených v datové sadě `iscx.csv`[4]. Pro každý atribut je uveden počet unikátních hodnot, podíl unikátních hodnot vůči celkovému počtu záznamů (v %), celkový počet výskytů atributu, podíl výskytů vzhledem k celkovému počtu řádků v datové sadě (v %). Tato statistika slouží k posouzení informační hodnoty jednotlivých atributů pro účely identifikace.

[1] OrgName – Organizace odpovídající *IP* adresy serveru, získané z databáze *WHOIS*.

[2] Nejedná se o přesné názvy aplikací ale pouze o jejich zkratky.

[3] Reprezentuje název souboru, ze kterého byla datová sada poskládána. V rámci jednoho souboru je zaznamenáno několik spuštění vždy jedné aplikace.

[4] Kromě atributu `ClientExtensions` jsou všechny uvedené atributy pouze z datové sady `iscx.csv`, neboť `iscx-raw.csv` ji pouze rozšiřuje o dodatečná metadata.

Položky	Počet unikátních hodnot	(v %)	Celkový počet	(v %)
SNI	207	8,50	2092	85,88
OrgName	72	2,96	2300	94,42
ClientExtensions	32	1.31	2436	100.00
JA3hash	46	1,89	2436	100,00
JA4hash	42	1,72	2436	100,00
AppName	16	0,66	2436	100,00
JA3Shash	117	4,80	2399	98,48
JA4Shash	127	5,21	2399	98,48
Filename	109	4,47	2399	98,48
Type	2	0.08	2436	100.00

Tabulka 5.1: Přehled položek v datové sadě `iscx.csv`

Tabulka 5.2 rozšiřuje statistickou analýzu na datovou sadu `mobile_desktop_apps_raw`. Opět jsou uvedeny klíčové atributy relevantní pro identifikaci aplikací.

Položky	Počet unikátních hodnot	(v %)	Celkový počet	(v %)
SNI	929	4.36	21259	99.80
OrgName	197	0.92	21273	99.87
ClientExtensions	10462	49.12	21301	100.00
ALPN	9	0.04	19505	91.57
JA3hash	10495	49.27	21301	100.00
JA4hash	123	0.58	21301	100.00
AppName	78	0.37	21301	100.00
JA3Shash	83	0.39	21180	99.43
JA4Shash	103	0.48	21180	99.43
Filename	1746	8.20	21180	99.43
Version	1	0.00	21180	99.43

Tabulka 5.2: Přehled položek v datové sadě `mobile_desktop_apps_raw.csv`

V prvních dvou datových sadách, `iscx.csv` a `iscx-raw.csv`, se nachází **celkem 2436 záznamů**, které byly vygenerovány **16 různými aplikacemi**. Tyto záznamy pocházejí ze **109 různých spuštění** (atribut *Filename*). Otisky *JA3* a *JA4* jsou dostupné pro každý záznam, zatímco otisky *JA3S* a *JA4S* chybí přibližně u 1,52 % záznamů.

Lze si rovněž povšimnout, že atribut *SNI*, ačkoliv chybí u 14,12 % záznamů, obsahuje **největší počet unikátních hodnot**. Díky tomu se jeví jako ideální kandidát pro přesnější identifikaci aplikací – zejména v případech, kdy dochází ke zlepšení identifikace kombinací více atributů.

Datová sada `mobile_desktop_apps_raw.csv` obsahuje **celkem 21301 záznamů**, které byly vygenerovány **78 různými aplikacemi**. Dále zahrnuje **1746 odlišných spuštění** (atribut `Filename`). Stejně jako u předchozích datových sad nechybí žádný z otisků *JA3* ani *JA4*. Naopak otisky *JA3S* a *JA4S* chybí ve 0,57 % případů.

Zajímavý je poměr počtu otisků *JA3* ku *JA4*, který činí přibližně **85:1** (10495 ku 123). Tento výrazný nepoměr naznačuje, že informační hodnota otisků *JA3* může být silně ovlivněna přímo použitými rozšířeními (např. `ClientExtensions`), kterých je v datasetu evi-

dováno 10462, viz 3.1. Na rozdíl od *JA4*, jehož konstrukce počítá s filtrováním redundancí pomoci seřazení a výpočtu zkráceného hashe, se *JA3* otisk generuje ze všech rozšíření bez jakýchkoli úprav, což může vést ke snížení jeho spolehlivosti pro účely identifikace.

Opět i v tomto případě se položka *SNI* jeví jako **vhodný kandidát pro přesnější identifikaci**, neboť vykazuje vysoký počet unikátních hodnot a chybí pouze ve 0,2 % záznamů, což svědčí o její vysoké informační hodnotě.

Tabulky 5.3 a 5.4 poskytují přehled o rozložení aplikací v jednotlivých použitých datových sadách. Každá tabulka uvádí název aplikace, počet zaznamenaných spojení a relativní podíl na celkovém počtu spojení v dané datové sadě. Tabulka 5.3 shrnuje statistiky z datové sady `iscx.csv`, kde dominují zejména aplikace jako *Hangouts* a *Skype*. Taktéž tabulka 5.4 zachycuje rozložení aplikací v rozsáhlejší datové sadě `mobile_desktop_apps_raw.csv`, která obsahuje širší spektrum moderních mobilních i desktopových aplikací.

Pro přehlednost jsou v obou tabulkách uvedeny pouze vybrané položky – prostřední části byly nahrazeny výpustkami.

`iscx.csv`		
Aplikace	**Počet**	**Podíl (%)**
Hangouts	553	22.70
Skype	436	17.90
Email	315	12.93
⋮	⋮	⋮
SCP	9	0.37
BitTorrent	8	0.33
ICQ	7	0.29

Tabulka 5.3: Přehled aplikací v `iscx.csv`

`mobile_desktop_apps_raw.csv`		
Aplikace	**Počet**	**Podíl (%)**
AirDroid	1774	8.33
Oer	1441	6.76
BiduTerBox	1252	5.88
⋮	⋮	⋮
Milbird	15	0.07
Telegram	12	0.06
TelegramDesktop	1	0.01

Tabulka 5.4: Přehled aplikací v 3. sadě

Všechny tři datové sady se všemi atributy, včetně kompletního seznamu aplikací použitých v těchto sadách, jsou uvedeny v příloze B.

5.2 Motivace a obecný popis

Tato sekce poskytuje obecný přehled o navrženém řešení, jeho struktuře a zvoleném přístupu. Hlavním cílem této práce je zlepšit identifikaci aplikací prostřednictvím kontextového přístupu. Při aplikaci otisku na identifikované spojení je vygenerována kandidátní množina aplikací, tedy množina všech aplikací, ke kterým je daný otisk přiřazen v trénovací fázi. Tato kandidátní množina je tedy výsledkem metod, jako jsou *JA3* nebo *JA4*, pokud jsou použity k identifikaci *TLS* spojení. Identifikace založená výhradně na otiscích, přestože může na první pohled působit poměrně přesně, není ideální – průměrná velikost kandidátní množiny bývá často příliš rozsáhlá, což snižuje její informační hodnotu.

Například v situaci, kdy 16 odlišných aplikací generuje síťovou komunikaci, může klasifikace založená na otiscích vrátit kandidátní množinu o velikosti 14. Takový výsledek je z hlediska jednoznačné identifikace aplikace prakticky bezcenný, protože neposkytuje dostatečně úzký výběr možných odpovědí.

Cílem této práce je pokusit se zúžit kandidátní množinu tak, aby bylo možné dosáhnout přesnější identifikace aplikace, a to na základě kontextu dalších *TLS* spojení, která se vy-

skytují paralelně s daným spojením. Využití kontextu dalších *TLS* spojení je možné díky tomu, že většina aplikací při spuštění generuje více souběžných spojení [21]. V rámci těchto okolních *TLS* spojení se hledají frekventované vzory, založené na otiscích a případně i dalších atributech komunikace. Na základě shody mezi aktuálním okolím spojení a předem známými frekventovanými vzory je následně možné určit nejpravděpodobnější kandidáty pro identifikaci aplikace.

5.3 Návrh řešení

Tato kapitola se věnuje detailnímu návrhu systému. Na základě analýzy současných metod a problémů spojených s identifikací na základě otisků je zde představen nový přístup, který využívá informace o souběžně navázaných *TLS* spojeních.

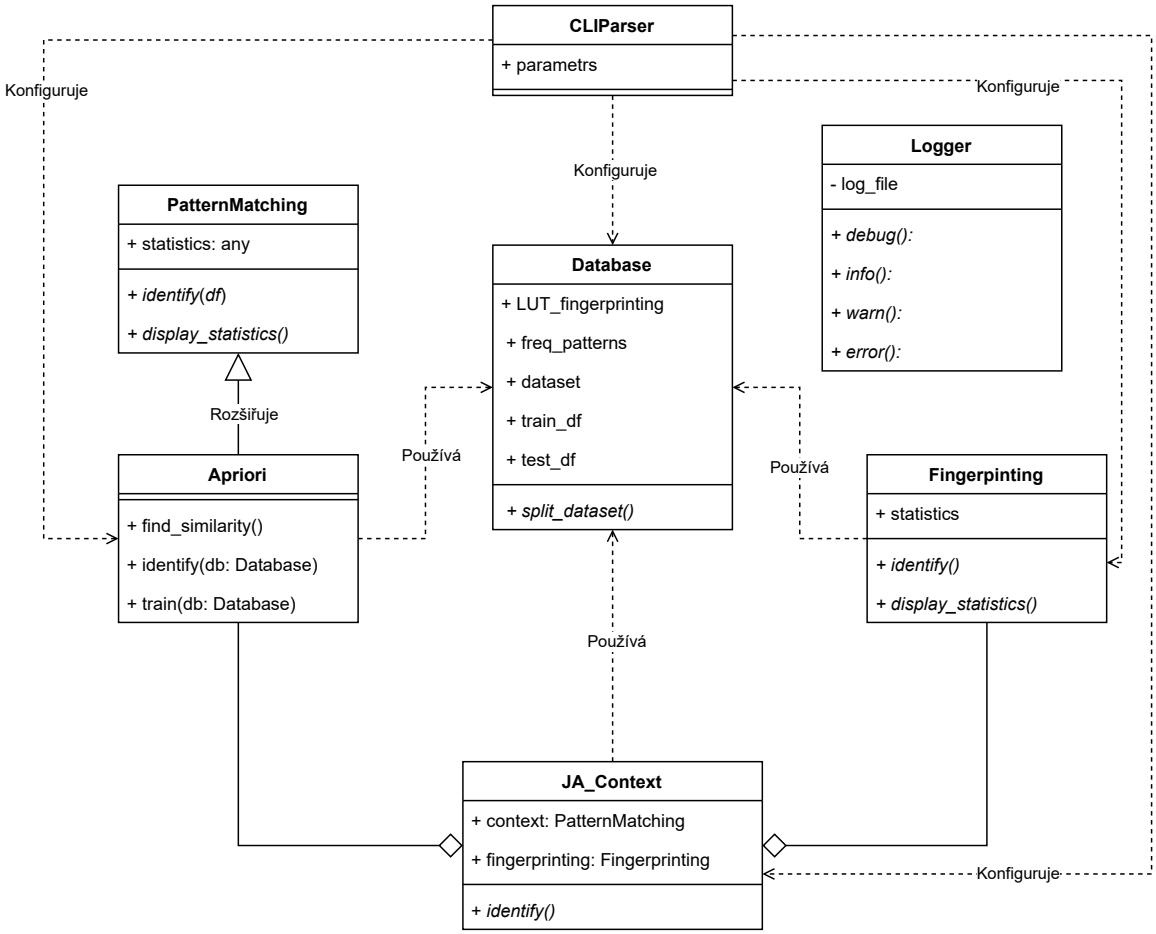
Cílem návrhu je vytvořit systém, který dokáže efektivně využít kontextové informace k zúžení kandidátní množiny aplikací, a tím zlepšit přesnost výsledné identifikace. Sekce popisuje požadavky na systém, jeho architekturu, zvolené algoritmy a metodiky, předpoklady pro jeho fungování i jednotlivé klíčové části řešení.

5.3.1 Požadavky a cíle

Jak již bylo zmíněno, hlavním cílem této práce je především zúžení kandidátní množiny pomocí kontextového přístupu. Zároveň je žádoucí, aby byla zachována co nejvyšší přesnost identifikace a aby zpracování spojení nebylo časově náročné. Je proto nutné nalézt ideální kompromis mezi přesností a časovou náročností, což samo o sobě může být náročné.

5.3.2 Architektura systému

Z důvodu rozšiřitelnosti a snadnější manipulace byl zvolen přístup objektově orientovaného programování. Systém je rozdělen do několika vzájemně spolupracujících tříd, které odpovídají jednotlivým částem řešení – načítání a uchovávání dat, identifikace založená čistě na otiscích, učení frekventovaných vzorů a kombinace identifikace otisků s kontextovým přístupem. Součástí systému jsou i třídy pro logování, správu nastavení a parsování parametrů z příkazové řádky. Tento přístup zajišťuje lepší modularitu, snadnější rozšiřování funkcionality a přehlednou správu celkového kódu. Diagram tříd 5.1 poskytuje podrobný přehled jednotlivých tříd a jejich metod.



Obrázek 5.1: Diagram tříd

V příloze C je uvedena rozšířená verze diagramu tříd, která obsahuje všechny atributy a metody.

Navržené řešení se bude skládat ze sedmi tříd, přičemž každá z nich pokryje jasně vymezenou část systému. Ovládání systému bude zajištěno prostřednictvím příkazové řádky, což vyžaduje existenci třídy `CLIParser`, která zpracuje parametry programu a předá je příslušným instancím ostatních tříd, jak je znázorněno v diagramu 5.1.

Pro účely snadného ladění a lepšího porozumění běhu programu se každý významný krok bude zaznamenávat prostřednictvím instance třídy `Logger`.

Třída `Database` bude sloužit jako hlavní zdroj dat a poskytne přístup k vyhledávací tabulce otisků, datové struktuře frekventovaných vzorů a dalším potřebným údajům z trénovací i testovací části datové sady. Přístup k těmto datům bude mít každá komponenta, která je bude vyžadovat.

Třída `Fingerprinting` nabídne metody pro identifikaci *TLS* spojení pomocí otisků *JA3*, *JA4* či jejich kombinací ($JA + JAS + SNI$), přičemž bude využívat vyhledávací tabulku uloženou v databázi. Zároveň bude zajišťovat generování statistik, které zaznamenají například přesnost identifikace nebo velikost kandidátní množiny.

Třída `PatternMatching` vytvoří základní rozhraní pro implementaci algoritmů určených k dolování informací z dat. Její návrh umožní snadné rozšíření systému o další metody data miningu a zajistí tak jeho flexibilitu a rozšiřitelnost.

Třída `Apriori` rozšíří třídu `PatternMatching` o konkrétní implementaci algoritmu *Apriori*. Tento algoritmus umožní vyhledávání frekventovaných vzorců v trénovacích datech a bude spolupracovat s databází pro jejich ukládání a vyhledávání. Porovnávat nalezené vzory se zkoumaným *TLS* spojením a vracet N[5] nejpravděpodobnějších kandidátů na základě podobnosti, vypočítané například porovnáním otisků nebo frekvence vzorců, bude metoda `find_similarity()`.

Třída `JA_Context` bude koordinovat využití tříd `Apriori` a `Fingerprinting` s cílem zúžit kandidátní množinu pro identifikaci aplikace. Nejprve provede iteraci přes testovací data v rámci předem definovaného okna a tím specifikuje okolí analyzovaného spojení. Pomocí otisků vytvoří počáteční kandidátní množinu aplikací, které budou relevantní pro dané spojení. Následně předá odpovídající frekventované vzory těchto kandidátů instanci třídy `Apriori`, která se pokusí nalézt vzory charakteristické pro aplikace v dané oblasti. Výsledkem tohoto procesu bude dále zúžená kandidátní množina obsahující aplikace nejlépe odpovídající analyzovanému spojení.

5.3.3 Předpoklady a omezení

Pro správné spuštění a fungování systému je nutné splnit několik základních předpokladů:

- **Dostupnost datové sady**: Program vyžaduje existenci pojmenované datové sady obsahující alespoň čtyři TLS spojení, přičemž každé musí obsahovat otisky *JA3* a *JA3S* (případně *JA4* a *JA4S*) a atribut *SNI*. Bez datové sady nebude možné program spustit.

- **Kvalita a rozsah dat**: Platí, že čím více unikátních spuštění aplikací je v datové sadě zaznamenáno, tím rozsáhlejší a přesnější frekventované vzory je možné extrahovat. Malé množství dat povede ke snížení přesnosti identifikace. Ovšem také velmi rozsáhlé datové sady mohou zvýšit nároky na paměť a výpočetní výkon. Návrh systému je optimalizován pro běžné datové sady (například `mobile_desktop_apps_raw.csv`, která obsahuje 21,301 řádků, je v tomto případě považována za běžnou datovou sadu).

5.4 Implementace

Tato sekce popisuje implementační detaily, použité technologie a poskytuje podrobný popis klíčových částí systému.

5.4.1 Použité technologie a nástroje

Pro implementaci byl zvolen programovací jazyk *Python* verze 3.13 kvůli vysoké flexibilitě a dostupnosti knihoven jak pro práci s daty, tak pro matematické výpočty a další účely. *Python* také usnadňuje implementaci objektově orientovaného návrhu.

Za účelem efektivní práce s datovou sadou byla zvolena knihovna *Pandas* verze 2.2.3. Algoritmus *Apriori* byl importován z knihovny *mlxtend* verze 0.23.4, stejně jako *TransactionEncoder*, který je potřebný pro předzpracování dat. Knihovna *scikit_learn* verze 1.6.1 poskytla metodu `train_test_split` pro rozdělení datové sady na testovací a trénovací část, a také metodu `cosine_similarity` pro výpočet kosinové podobnosti. Knihovna *numpy* verze 2.2.5 byla také využita pro různé matematické operace. Při tvorbě této technické

[5] Kde N reprezentuje předem určený maximální počet kandidátů.

zprávy byla využita umělá inteligence, konkrétně *ChatGPT*, pro úpravu stylistiky a opravu gramatiky.

5.4.2 Použití programu a konfigurace

K zajištění správného běhu programu a korektního zpracování dat byl vytvořen konfigurační soubor `config.py`, ve kterém lze upravovat jednotlivé názvy sloupců v případě použití jiných datových sad. V tomto souboru lze také povolit ladicí výpisy, které budou propsány do logů. Byl umístěn na stejné úrovni zanoření jako `main.py`, pomocí kterého se celý program spouští. Všechny navrhované třídy jsou implementovány v jednotlivých souborech ve složce `identify`, s výjimkou tříd `PatternMatching` a `Apriori`, které jsou implementovány v jednom společném souboru. Struktura celého projektu je uvedena v příloze A. Program lze spustit následujícím příkazem:

```
python3 main.py -d <ds> -f <3|4> -w <win> -m <min_sup> -c <num_can>
```

Kde:

- `-d <ds>`: Reprezentuje cestu k datové sadě (např. `data/iscx.csv`),
- `-f <3|4>`: Volí verzi otisků (`3` pro *JA3*, `4` pro *JA4*),
- `-w <win>`: Určuje šířku posuvného okna (celé číslo),
- `-m <min_sup>`: Specifikuje minimální podporu pro *Apriori* (desetinné číslo v rozsahu 0,0–1,0),
- `-c <num_can>`: Nastavuje počet kandidátů ve množině (celé číslo),
- a `-h`, `-help`: Zobrazí nápovědu.

5.4.3 Popis klíčových částí

Tato podsekce popisuje klíčové a zajímavé části programu, které se objevily v průběhu implementace.

Rozdělení a čištění datové sady

Nejprve dochází k čištění datové sady, během něhož jsou odstraněny všechny záznamy typu `A` a `M` reprezentující reklamní provoz či malware. Tímto krokem se zachovají pouze záznamy odpovídající běžnému provozu. Následně je datová sada rozdělena v poměru 75:25 na trénovací a testovací část.

Rozdělení probíhá na základě hodnoty atributu `Filename`, který reprezentuje jednotlivá spuštění aplikací. V rámci každého spuštění je 75% záznamů zařazeno do trénovací množiny a zbývajících 25% do množiny testovací. Tím je zajištěno, že každé spuštění je zastoupeno v obou částech sad.

Pokud dané spuštění obsahuje pouze jeden záznam (tj. daná aplikace vygenerovala pouze jedno *TLS* spojení), je tento záznam přiřazen automaticky do trénovací části, aby bylo možné z něj vytěžit potřebné vzory.

Těžba frekventovaných vzorů

Spojení v tomto kroku jsou reprezentována jako kombinace několika diskrétních hodnot, zejména otisků *JA3*, *JA3S*, *JA4*, *JA4S*, a dále atributů jako *SNI* nebo *OrgName*. Tyto prvky slouží jako základní položky pro tvorbu transakcí, nad nimiž je následně aplikován algoritmus *Apriori* k identifikaci často se vyskytujících vzorů charakteristických pro jednotlivé aplikace.

Před samotnou aplikací algoritmu jsou data převedena do binární reprezentace pomocí objektu `TransactionEncoder` z knihovny `mlxtend`, který transformuje kategorická data do formátu, se kterým algoritmus *Apriori* pracuje. Následně je volána funkce `apriori()`, která na základě zadané minimální podpory (`min_support`) vyhledá všechny časté vzory, jež se vyskytují ve spojeních dané aplikace.

Algoritmus *Apriori* se aplikuje na trénovací část datové sady, konkrétně na seskupená data podle jednotlivých aplikací, pro které se frekventované vzory vyhledávají. Následně jsou všechny nalezené vzory očištěny na základě parametrů, které budou popsány v následující kapitole 6 v rámci experimentů.

Výsledkem tohoto kroku je pro každou aplikaci množina častých vzorů, které jsou dále použity v kontextové fázi identifikace, kde slouží k porovnání s okolními spojeními.

Volba kontextu identifikovaného spojení

Identifikace probíhá nad testovacími daty, která jsou seskupena za sebou podle jednotlivých spuštění aplikací. Tyto skupiny jsou následně deterministicky promíchány v takovém pořadí, aby žádné dvě po sobě jdoucí skupiny nepocházely od stejné aplikace. Toto opatření má za cíl předejít příliš výraznému kontextu, který by mohl zkreslit výsledky. Cílem je simulovat realistický scénář běžného provozu, kdy je pravděpodobnost opakovaného spuštění stejné aplikace bezprostředně po sobě relativně nízká.

Kontext, neboli okolí zkoumaného spojení, se určuje pomocí posuvného okna (angl. sliding window) s předem definovanou šířkou, které iteruje přes celou testovací část dat. Velikost okna by ideálně měla být lichá, aby bylo možné udržet zkoumané spojení uprostřed a zároveň zajistit symetrický počet sousedních spojení na obou stranách. V případech, kdy se zkoumané spojení nachází příliš blízko začátku nebo konce testovací sady a běžné centrování by vedlo k neúplnému oknu, je pozice spojení v rámci okna upravena tak, aby bylo možné zachovat úplný a vyvážený kontext. V takových situacích zůstává pozice okna v datech pevná a posouvá se pouze relativní umístění zkoumaného spojení uvnitř tohoto okna.

Velikost posuvného okna má zásadní vliv na kvalitu identifikace. Příliš malé okno nemusí zachytit dostatečný kontext pro spolehlivé porovnání s frekventovanými vzory, což vede ke ztrátě informační hodnoty. Naopak příliš široké okno může zahrnout irelevantní nebo rušivé spojení od jiných aplikací, čímž dochází ke zhoršení přesnosti. Z tohoto důvodu je volba optimální velikosti okna předmětem experimentálního vyhodnocení, které je popsáno v sekci 6.3.

Výpočet podobnosti

Při identifikaci se ve zkoumaném kontextu vyhledávají frekventované vzory jednotlivých aplikací. Pouhé přesné porovnání těchto vzorů však není efektivní, protože často dochází k částečnému překryvu mezi kontextem a naučenými vzory. Vhodnějším přístupem je proto výpočet míry podobnosti mezi množinami prvků – tedy mezi frekventovanými vzory dané aplikace a aktuálním obsahem kontextového okna.

Tato podobnost byla zprvu kvantifikována pomocí metriky kosinové podobnosti, která umožňuje zohlednit nejen přesné shody, ale i dílčí překryvy. Později se však ukázalo, že při rozsáhlé databázi vzorů a velkém počtu spojení není její časová náročnost ideální. Mnohem příznivější výsledky z hlediska výpočetní efektivity přinesly množinové operace, jako například *Jaccardova* či *Diceho* podobnost. Výsledkem je „skóre“ podobnosti pro každou aplikaci, na jehož základě je následně sestavena kandidátní množina nejpravděpodobnějších aplikací.

Samotný výpočet skóre probíhá následovně: pro každou aplikaci se prochází seznam jejích frekventovaných vzorů. Každý vzor je reprezentován jako množina položek (např. kombinace *JA3*, *SNI* a *JA4S*), a jeho výskyt je zaznamenán napříč celou trénovací množinou. Následně je vypočteno skóre relevance vůči aktuálnímu *TLS* spojení na základě *Jaccardovy* podobnosti mezi vzorem a zkoumanou množinou obsahující atributy spojení. *Jaccardova* podobnost je vyobrazena v rovnici 5.1.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

Tento dílčí výsledek je dále upraven pomocí *IDF* faktoru, který zvýhodňuje méně časté vzory podle vztahu 5.2:

$$\text{idf}(p) = \log\left(1 + \frac{N}{\text{df}(p)}\right) \tag{5.2}$$

kde N je celkový počet aplikací a $\text{df}(p)$ je počet aplikací obsahujících daný vzor p , jak je zapsáno rovnicí 5.3.

Pokud je vzor podmnožinou atributů spojení, skóre se ještě násobí podporou vzoru a jeho délkou, viz 5.4, což zvýhodňuje přesné a zároveň delší shody. Výsledná skóre všech aplikací jsou nejprve normalizována pomocí *min-max* normalizace, aby bylo možné jednotlivé hodnoty porovnávat na stejné škále. Následně jsou nejpravděpodobnější kandidáti určeni na základě nejvyšších skóre – k jejich efektivnímu výběru se využívá funkce `heapq.nlargest`, která díky použití haldy umožňuje rychlý výběr `N` nejlépe hodnocených položek bez nutnosti seřadit celý seznam.

Výpočet skóre podobnosti Skóre aplikace a je počítáno jako vážený součet příspěvků jednotlivých vzorů $p \in P_a$, viz 5.5:

1. **Základní skóre pomocí *Jaccardovy* podobnosti:**

$$s_1(p, C) = (J(p, C) + 1) \cdot \text{idf}(p) \tag{5.3}$$

2. **Doplňkové skóre při úplné shodě (podmnožině):**

$$s_2(p, C) = \begin{cases} |p| \cdot 10 \cdot \text{idf}(p) \cdot (\text{sup}_{\text{norm}}(p) + 1), & \text{pokud } p \subseteq C \\ 0, & \text{jinak} \end{cases} \tag{5.4}$$

3. **Celkové skóre aplikace a :**

$$\text{score}(a) = \sum_{p \in P_a} [s_1(p, C) + s_2(p, C)] \tag{5.5}$$

kde:

- P_a je množina frekventovaných vzorů pro aplikaci a ,
- C je množina atributů kontextu,
- $J(p, C)$ je *Jaccardova* podobnost mezi vzorem p a kontextem C ,

V případě, že první fáze identifikace (využití samotných otisků bez kontextu) vrátí kandidátní množinu požadované délky, je i přesto proveden výpočet podobnosti. Pokud již nedojde ke zkrácení množiny, slouží tento výpočet alespoň k jejímu seřazení podle nejpravděpodobnější aplikace.

Je také možné, že první fáze identifikace neposkytne žádné kandidátní aplikace, použije se pro vyhledávání v kontextu celá databáze frekventovaných vzorů. Pokud ani v okolí daného spojení (v rámci daného kontextu) nebude nalezen žádný frekventovaný vzor odpovídající zúžené databázi, provede se rozšíření vyhledávání i na vzory aplikací, které tvoří komplement původní množiny. Tento postup zajišťuje, že bude vždy existovat alespoň pokus o identifikaci, i v případech, kdy selže primární výběr podle otisků.

Vyhodnocení úspěšnosti

Identifikace je vyhodnocována na základě pevně zvoleného počtu kandidátů N . Pro každou pozici x , kde $x \leq N$, je zaznamenána úspěšnost – tedy podíl případů, kdy se skutečná aplikace nachází na dané pozici v kandidátní množině. Jinými slovy, pokud je hodnota N nastavena na 3, jsou vyhodnoceny statistiky přesnosti pro 1., 2. a 3. pozici výsledné kandidátní množiny. Celková úspěšnost je počítána jako součet dílčích úspěšností pro jednotlivé pozice v kandidátní množině. Chybovost je následně vyjádřena jako doplněk celkové úspěšnosti do jedné. Kromě toho jsou vypočítávány i charakteristiky velikosti kandidátních množin, jako je průměr, medián a modus. Tyto metriky umožňují posoudit nejen přesnost identifikace, ale i kompaktnost výsledných množin, což je zásadní pro praktickou použitelnost a analýzu navrženého systému.

5.5 Zhodnocení návrhu a rozdíly v implementaci

Navržená architektura systému byla převážně dodržena, včetně rozdělení systému do samostatných tříd. Objektový návrh se osvědčil zejména při udržování přehlednosti a při výraznějších změnách v kódu. Některé třídy však byly oproti původnímu návrhu mírně rozšířeny – například třída `Fingerprinting` byla doplněna o dodatečné statistiky. Naopak některé pomocné funkce byly z důvodu přehlednosti nebo znovupoužitelnosti přesunuty mimo své původní třídy.

Při implementaci bylo nutné učinit několik kompromisů. Nejvýraznějším z nich byla nutnost nahradit původně zamýšlené výpočetně náročnější metriky pro výpočet podobnosti jednoduššími množinovými operacemi. Konkrétně došlo k nahrazení kosinové podobnosti *Jaccardovou* podobností, což přineslo výrazné zrychlení a umožnilo splnit nefunkční požadavek na časovou náročnost, a to i přes zanedbatelné zhoršení přesnosti identifikace.

Mezi hlavní výzvy patřila práce s rozsáhlými datovými sadami, zejména při opakovaném trénování algoritmu *Apriori* pro desítky různých aplikací. Bylo nutné optimalizovat datové struktury a přistupovat k datům selektivně. Dále bylo důležité nalézt vhodný kompromis mezi přesností identifikace a rychlostí zpracování. Počet vytěžených častých vzorů na aplikaci se oproti navrhovanému řešení snížil na pouhé desítky.

Možnosti budoucího rozšíření zahrnují paralelizaci dat, což by umožnilo rychlejší trénování a hodnocení na více-jádrových systémech. Dalším krokem by mohlo být přidání algoritmu *FP-Growth* nebo úplné nahrazení algoritmu *Apriori*. Také by bylo možné zkoumat alternativní metriky podobnosti nebo zavést váhování jednotlivých atributů podle jejich informační hodnoty. V neposlední řadě lze systém doplnit o heuristiky pro automatickou kalibraci parametrů nebo o adaptivní učení na nových datech.

Kapitola 6

Experimenty za účelem zlepšení identifikace

Tato kapitola představuje soubor experimentů zaměřených na zefektivnění procesu identifikace aplikací. Hlavním cílem je minimalizace mohutnosti kandidátní množiny aplikací při současném zachování co nejvyšší přesnosti. Zároveň jsou respektovány všechny relevantní nefunkční požadavky, zejména s ohledem na časovou náročnost.

Identifikace a použití otisků probíhá ve dvou fázích, jak bylo popsáno v sekci 5.3.2. Nejprve je aplikována metoda fingerprintingu *JA3*, *JA4* či jejich kombinace, na jejímž základě je získána kandidátní množina (proběhne jednoduché vyhledání aplikací, které odpovídají danému otisku). Tato množina kandidátů je dále v textu označována jako **počáteční kandidátní množina**.

Počáteční kandidátní množina přímo určuje, které získané vzory (tj. vzory získané z trénovací množiny) se budou používat pro výpočet podobnosti. V modelové situaci je pro dané spojení vygenerována počáteční kandidátní množina obsahující `App1`, `App2` a `App3`. Výpočet podobnosti mezi kontextem daného spojení a frekventovanými vzory tedy probíhá pouze pro vzory aplikací `App1`, `App2` a `App3`.

Tímto způsobem se zefektivní identifikace aplikací, jelikož není potřeba iterovat přes časté vzory, které nejsou v počáteční kandidátní množině. Použití kontextu zde slouží pouze ke zúžení kandidátní množiny. Tato množina bude dále v textu označována jako **finální kandidátní množina**.

6.1 Volby položek pro získání vzorů

Cílem tohoto experimentu je vyhodnotit, které otisky a případně další atributy spojení, například *SNI* či *ORG*, přinášejí nejvyšší přesnost identifikace, a to jak při tvorbě počáteční kandidátní množiny, tak i jako vstupní data pro algoritmus *Apriori*. Ideální kombinace by zároveň měla generovat co nejkratší finální kandidátní množinu aplikací a udržet výpočetní náročnost v přijatelných mezích.

Pro tyto experimenty byly ostatní parametry zvoleny následovně: **minimální podpora** algoritmu *Apriori* byla nastavena **na 0,25**, **počet finálních kandidátů na 3**, **šířka okna na 15** spojení u sady `iscx.csv` a **25** u sady `mobile-desktop-apps-raw.csv`. Metody generující počáteční kandidátní množinu byly: *JA3*, *JA3+JA3S+SNI*, *JA4* a *JA4+JA4S+SNI*.

Tabulka 6.1 shrnuje přesnost a průměrnou délku počáteční kandidátní množiny po aplikaci metod *JA3*, *JA4* a jejich kombinací. Výsledky odpovídají čistému fingerprintingu da-

ných spojení, tedy bez využití kontextu a frekventovaných vzorů. Tabulka slouží jako výchozí přehled přesnosti základního fingerprintingu na jednotlivých datových sadách a jako referenční bod pro následné porovnání s výsledky identifikace využívající kontext.

Otisky	`iscx.csv`		`mobile_desktop_apps_raw.csv`	
	Přesnost	**Délka**	**Přesnost**	**Délka**
JA3	0.975	3.146	0.495	4.325
JA3+JA3S+SNI	0.924	2.099	0.855	5.953
JA4	0.975	3.146	0.966	14.298
JA4+JA4S+SNI	0.924	2.094	0.854	3.382

Tabulka 6.1: Přesnost otisků a jejich kombinací

V následujících tabulkách jsou uvedeny výsledky identifikace s využitím kontextu a vzorů. V záhlaví každé tabulky je vždy uvedena metoda, pomocí které byla určena počáteční kandidátní množina. Kombinace pak specifikuje položky, ze kterých algoritmus *Apriori* získává časté vzory. Výsledky zahrnují přesnost identifikace, počítanou jako podíl případů, kdy byla správně určena aplikace nacházející se ve finální kandidátní množině, vůči celkovému počtu identifikovaných spojení. Dále je uvedena délka, která reprezentuje průměrnou velikost finální kandidátní množiny. Je zaznamenán i čas, potřebný pro identifikaci, pouze pro datovou sadu `mobile-desktop-apps-raw.csv`. Časové hodnoty byly měřeny na testovacím zařízení s následujícími parametry: CPU: `AMD Ryzen 7 5700U (16) @ 1.80 GHz`, OS: `EndeavourOS x86_64`, RAM: `16 GB`. Přestože je časová náročnost závislá na výkonu konkrétního zařízení, poskytují tyto hodnoty orientační představu o relativní složitosti jednotlivých variant identifikace. Z typografických důvodů byla pro kombinaci položek *JA3+JA3S+JA4+JA4S* zvolena zkratka *JA34S**. Všechny výsledky testované kombinace položek jsou uvedeny v příloze D.

Počáteční kandidátní množina získaná pomocí metody *JA3*					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas [s]**
JA3	0.837	2.128	0.279	2.778	74.470
JA4	0.837	2.128	0.390	2.792	60.340
JA4+JA4S	0.842	2.128	0.487	2.796	112.910
JA3+JA3S+SNI	0.852	2.128	0.371	2.796	163.680
JA4+JA4S+SNI	0.847	2.128	0.490	2.796	185.050
JA34S*+SNI	0.842	2.128	0.441	2.796	575.500
JA34S*+SNI+ORG	0.849	2.128	0.423	2.796	1123.730

Tabulka 6.2: Výsledky experimentu s kombinacemi položek, které byly použity jako vstupní data pro získání častých vzorů pomocí algoritmu *Apriori*, při použití počáteční kandidátní množiny získané pomocí metody *JA3*.

V první části experimentu, při použití metody *JA3* pro generování počáteční kandidátní množiny jsou výsledky uvedeny v tabulce 6.2. U datové sady `iscx.csv` dosahuje většina kombinací přesnosti kolem 0,84, přičemž průměrná délka finální kandidátní množiny činí 2,128 kandidátů na spojení, a to i bez výrazného navýšení počtu položek.

Naproti tomu `mobile_desktop_apps_raw.csv` vykazuje výrazně nižší přesnost identifikace. Ačkoliv kombinace *JA4+JA4S+SNI* dosahuje obdobné přesnosti jako samotné použití otisku *JA3* (0,490 při délce 2,796 oproti 0,495 při délce 4,325), jak je uvedeno v tabulce 6.1, ukazuje se, že tato datová sada je **mnohem citlivější na volbu kombinace otisků**. Rozdíl mezi nejhorším a nejlepším výsledkem činí **0,211**, což poukazuje na značný dopad správného výběru položek na výslednou přesnost.

Při použití metody *JA4* pro vytváření počáteční kandidátní množiny jsou výsledky uvedené v tabulce 6.3, pro datovou sadu `iscx.csv`, identické s výsledky dosaženými při použití počáteční množiny vytvořené pomocí *JA3* (viz.Tabulka 6.2). V případě této datové sady tedy zvolená metoda pro získaní počáteční kandidátní množiny nemá vliv na výslednou úspěšnost identifikace. Tento jev lze pravděpodobně přičíst nižšímu počtu různých aplikací.

Počáteční kandidátní množina získaná pomocí metody *JA4*					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas [s]**
JA3	0.837	2.128	0.304	2.781	53.660
JA4	0.837	2.128	0.387	2.785	32.340
JA4+JA4S	0.842	2.128	0.558	2.791	58.500
JA3+JA3S+SNI	0.852	2.128	0.489	2.791	66.340
JA4+JA4S+SNI	0.847	2.128	0.548	2.791	84.030
JA34S*+SNI	0.842	2.128	0.549	2.791	225.220
JA34S*+SNI+ORG	0.849	2.128	0.533	2.791	400.770

Tabulka 6.3: Výsledky experimentu s kombinacemi položek, které byly použity jako vstupní data pro získání častých vzorů pomocí algoritmu *Apriori*, při použití počáteční kandidátní množiny získané pomocí metody *JA4*.

U druhé datové sady došlo oproti předchozí metodě k mírnému zlepšení přesnosti identifikace. Přesto však i s nejpřesnější kombinací *JA3+JA3S+JA4+JA4S* je dosaženo pouze přesnosti 0,577 při průměrné délce finální kandidátní množiny 2,791, zatímco při použití samotného fingerprintingu pomocí *JA4* je dosaženo výrazně vyšší přesnosti 0,975, i když za cenu výrazně delší průměrné kandidátní množiny 14,298 (viz. Tabulka 6.1).

Přesnost identifikace při využití metody *JA3+JA3S+SNI* pro získání počáteční kandidátní množiny vykazuje výrazné zlepšení, zejména u druhé datové sady (viz. Tabulka 6.4).

Nejvíce se to projevuje u kombinací *JA4+JA4S* a *JA4+JA4S+SNI*, kde přesnost identifikace dosahuje hodnoty **0,800** při průměrné délce finální kandidátní množiny **1,734**. Tato přesnost se tak výrazně přibližuje hodnotě 0,855 dosažené při použití fingerprintingu, avšak při podstatně větší průměrné délce kandidátní množiny 5,953, jak je uvedeno v Tabulce 6.1. Toto zlepšení však přináší vyšší dobou trvání identifikace.

Tento přístup zároveň přináší mírné zlepšení také u datové sady `iscx.csv`, kde průměrná délka kandidátní množiny **klesá z 2,128 na 1,583** kandidátů.

Počáteční kandidátní množina získaná pomocí metody *JA3+JA3S+SNI*					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas [s]**
JA3	0.867	1.583	0.385	2.287	74.470
JA4	0.867	1.583	0.771	1.747	60.340
JA4+JA4S	0.869	1.583	0.800	1.734	112.910
JA3+JA3S+SNI	0.872	1.583	0.790	1.734	163.680
JA4+JA4S+SNI	0.867	1.583	0.798	1.734	185.050
JA34S*+SNI	0.867	1.583	0.792	1.734	575.500
JA34S*+SNI+ORG	0.869	1.583	0.774	1.734	1123.730

Tabulka 6.4: Výsledky experimentu s kombinacemi položek, které byly použity jako vstupní data pro získání častých vzorů pomocí algoritmu *Apriori*, při použití počáteční kandidátní množiny získané pomocí metody*JA3+JA3S+SNI*.

Tabulka 6.5 uvádí výsledky identifikace s počáteční kandidátní množinou generovanou použitím kombinace *JA4+JA4S+SNI*. Přesnost a průměrná délka finální kandidátní množiny vykazují **mírné zlepšení** oproti dříve uvedeným metodám, hlavním přínosem této kombinace je však **výrazně kratší doba trvání** identifikace – ve srovnání s kombinací *JA3+JA3S+SNI* je přibližně poloviční, v některých případech dokonce až třetinová.

Počáteční kandidátní množina získaná pomocí metody *JA4+JA4S+SNI*					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas [s]**
JA3	0.869	1.585	0.385	2.277	53.660
JA4	0.869	1.585	0.770	1.727	32.340
JA4+JA4S	0.869	1.585	0.798	1.714	58.500
JA3+JA3S+SNI	0.874	1.585	0.795	1.714	66.340
JA4+JA4S+SNI	0.869	1.585	0.804	1.714	84.030
JA34S*+SNI	0.869	1.585	0.797	1.714	225.220
JA34S*+SNI+ORG	0.872	1.585	0.782	1.714	400.770

Tabulka 6.5: Výsledky experimentu s kombinacemi položek, které byly použity jako vstupní data pro získání častých vzorů pomocí algoritmu *Apriori*, při použití počáteční kandidátní množiny získané pomocí metody *JA4+JA4S+SNI*.

Hlavním důvodem tohoto zrychlení je nižší počet kandidátních aplikací (mohutnost počáteční kandidátní množiny je menší viz. Tabulka 6.1), a tedy i menší množství vzorů, které je potřeba porovnat. Při použití fingerprintingu s kombinací *JA4+JA4S+SNI* bez kontextu nad datovou sadou `mobile_desktop_apps_raw.csv` je dosaženo přesnosti 85,4% při průměrné délce kandidátní množiny 3,382. Při identifikaci využívající kontext **klesá přesnost přibližně o 5%**, avšak průměrná délka finální kandidátní množiny je **více než poloviční**.

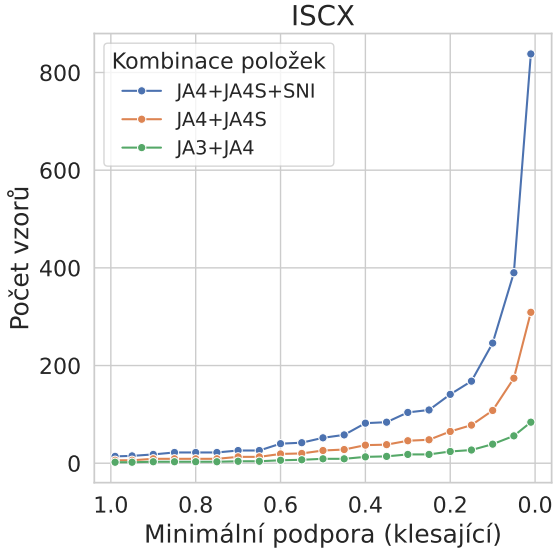
Testování identifikace s různými položkami a jejich kombinacemi přineslo cenný pohled na jejich využitelnost. Experiment byl rovněž proveden s položkou *ClientExtension*, avšak vzhledem k tomu, že tato položka nepřinesla žádné významné zlepšení, není zde podrobně uvedena. Podrobné výsledky lze nalézt v příloze D.1.

Na základě provedených experimentů se jako nejefektivnější pro identifikaci aplikací pomocí kontextu ukazují otisky *JA4* a jejich kombinace. Tyto kombinace se osvědčily jak při vyhledávání v kontextu, tak při generování počáteční kandidátní množiny. Mezi nejlépe hodnocené kombinace pro získávání častých vzorů patří: *JA4+JA4S*, *JA4+JA4S+SNI* a *JA3+JA4* v kombinaci s metodami *JA4* a *JA4+JA4S+SNI* pro generování počáteční kandidátní množiny. Tyto přístupy dosahují nejvyšší přesnosti identifikace při zachování kratší délky kandidátní množiny a zároveň přispívají k výraznému zkrácení doby trvání identifikace.

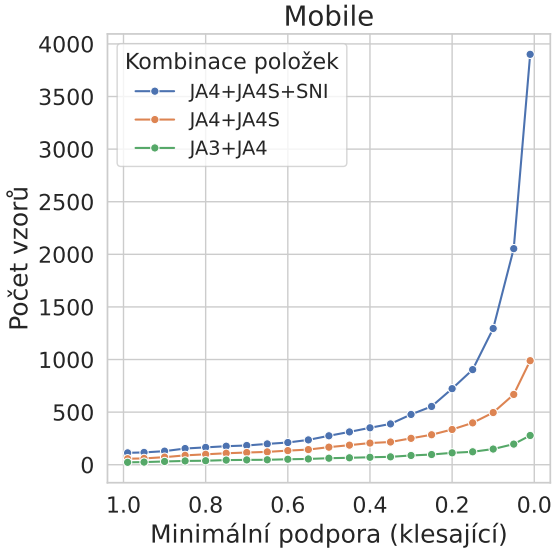
6.2 Minimální podpora algoritmu *Apriori*

Tento experiment zkoumá závislost mezi volbou minimální podpory algoritmu *Apriori*, přesností identifikace a průměrnou délkou finální kandidátní množiny. Testování probíhá nad třemi nejlépe hodnocenými kombinacemi otisků z předchozího experimentu popsaného v sekci 6.1, konkrétně *JA3+JA4*, *JA4+JA4S+SNI* a *JA4+JA4S* v kombinaci s metodou *JA4+JA4S+SNI* pro generování počáteční kandidátní množiny.

Cílem je určit, jak výrazně výběr prahové hodnoty podpory ovlivňuje kvalitu výsledné množiny kandidátů. Příliš vysoká hodnota podpory může vést k zanedbání užitečných, avšak méně frekventovaných vzorů, zatímco příliš nízká může zahrnout i irelevantní kombinace, čímž může narůstat velikost kandidátní množiny a zvyšuje se výpočetní náročnost. Výsledky experimentu by tak měly napomoci při optimalizaci nastavení algoritmu.



Obrázek 6.1: Počet vzorů v závislosti na podpoře pro `iscx.csv`.



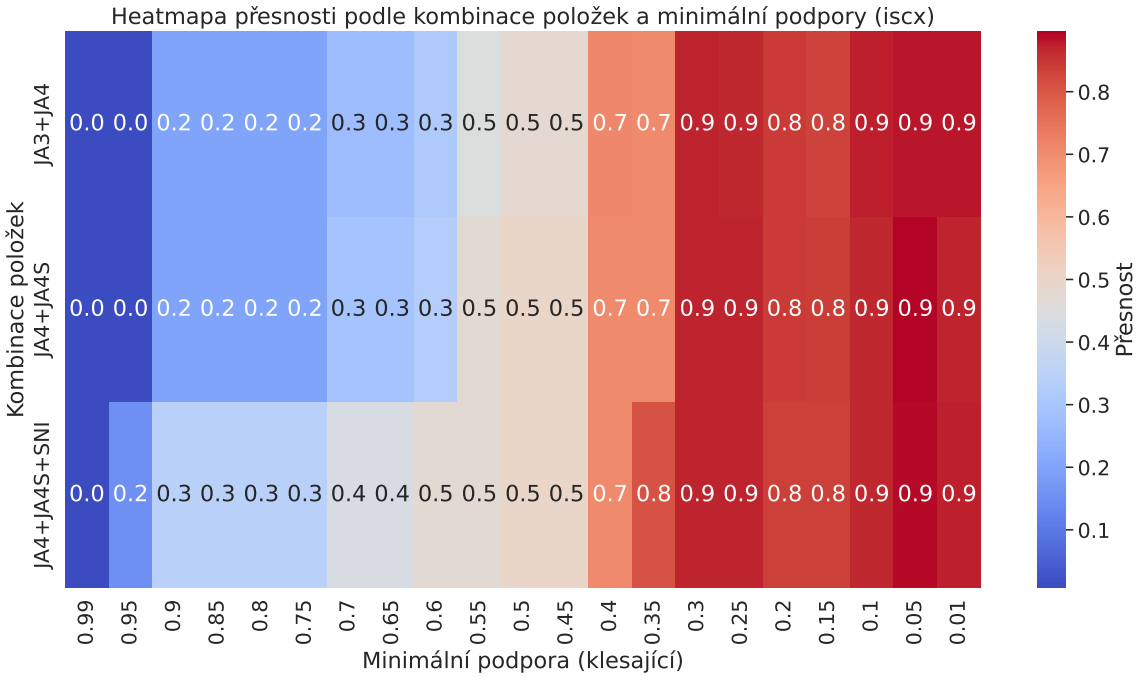
Obrázek 6.2: Počet vzorů v závislosti na podpoře pro `mobile_desktop_apps_raw.csv`.

Ostatní parametry pro tento experiment byly nastaveny následovně: **šířka okna 15 spojení**, maximální **počet finálních kandidátů na 3**.

Je zřejmé, že volba minimální podpory přímo ovlivňuje počet identifikovaných vzorů v rámci okolních spojení, jak ilustruje graf na obrázku 6.1 pro datovou sadu `iscx.csv` a graf na obrázku 6.2 pro datovou sadu `mobile_desktop_apps_raw.csv`. Lze rovněž očekávat, že celkový počet vzorů výrazně narůstá s velikostí datové sady, což potvrzuje sada

`mobile_desktop_apps_raw.csv`, kde zaznamenaný počet vzorů **při minimální podpoře 0,01 dosahuje téměř 4000**. Naproti tomu u sady `iscx.csv` celkový počet vzorů mírně **přesahuje hodnotu 800**.

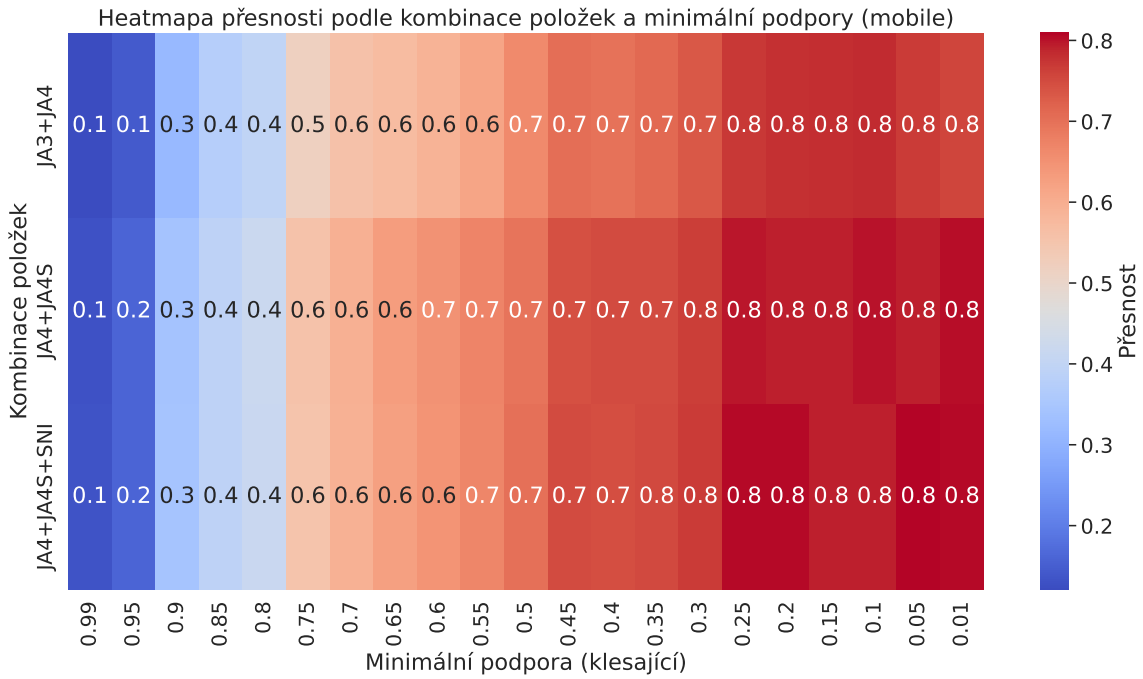
V obou datových sadách je přesnost identifikace při nastavení minimální podpory v intervalu $\langle 0{,}30; 0{,}99\rangle$ poměrně nízká a roste s klesající hodnotou podpory. V nejlepších případech dosahuje maximálně hodnoty 0,7. Při podpoře **nižší než 0,3** vykazují obě datové sady výrazně lepší úroveň identifikace.



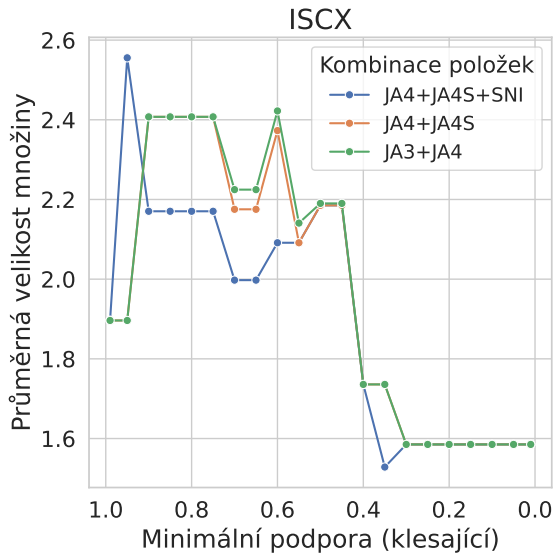
Obrázek 6.3: Heatmapa znázorňuje testované kombinace položek pro algoritmus *Apriori* v závislosti na volbě minimální podpory a dosažené přesnosti pro datovou sadu `iscx.csv`, při použití metody *JA4+JA4S+SNI* pro nalezení počáteční kandidátní množiny.

U datové sady `iscx.csv` (viz.obr. 6.3) lze pozorovat, že všechny testované kombinace dosahují nejvyšší míry identifikace při hodnotách podpory 0,3, 0,25 a 0,05. Druhá datová sada `mobile_desktop_apps_raw.csv` (viz.obr. 6.4) si od podpory 0,3 udržuje podobnou úroveň přesnosti napříč všemi kombinacemi. Nejlepších výsledků však dosahuje kombinace *JA4+JA4S+SNI* při hodnotách podpory 0,25, 0,2, 0,05 a 0,01. Přesnější identifikace od těchto hodnot je pravděpodobně způsobena tím, že při hodnotě 0,26 již všechny aplikace mají minimálně jeden vzor.

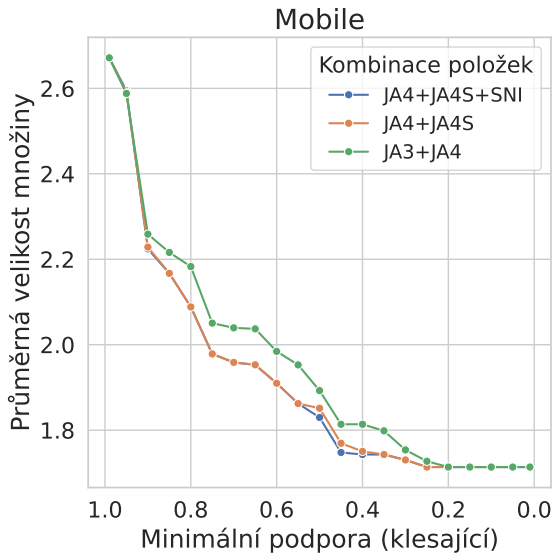
Experiment byl rovněž proveden s využitím metody *JA4* pro generování kandidátní množiny určené k následné úpravě. Přestože se kombinace *JA4+JA4S+SNI* dosud jeví jako nejvhodnější varianta, výsledky ukazují, že samotný otisk *JA4* zatím nedosahuje požadované úrovně přesnosti. Kompletní výsledky experimentu jsou uvedeny v příloze D.2.



Obrázek 6.4: Heatmapa zobrazující testované kombinace položek pro algoritmus *Apriori* v závislosti na volbě minimální podpory a dosažené přesnosti pro datovou sadu `mobile desktop apps raw.csv`, při použití metody *JA4+JA4S+SNI* pro nalezení počáteční kandidátní množiny.



Obrázek 6.5: Průměrná velikost finální kandidátní množiny v závislosti na podpoře pro `iscx csv`.



Obrázek 6.6: Průměrná velikost finální kandidátní množiny v závislosti na podpoře pro `mobile desktop apps raw.csv`.

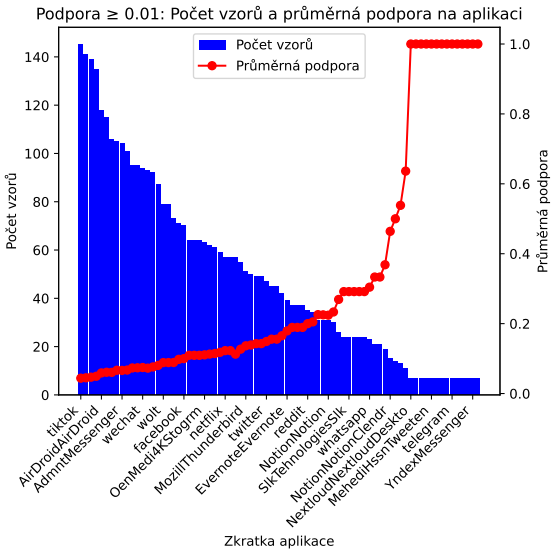
Průměrná velikost finální kandidátní množiny klesá se snižující se minimální podporou u obou datových sad. U datové sady `iscx.csv` (viz. graf na Obrázku 6.5) jsou roz-

díly mezi jednotlivými kombinacemi výraznější než u sady `mobile_desktop_apps_raw.csv` (viz. graf na Obrázku 6.6). Nicméně při hodnotách podpory 0,2 a nižších konvergují všechny kombinace k téměř stejné, nejnižší zaznamenané hodnotě (v případě `iscx.csv` k druhé nejnižší zaznamenané hodnotě).

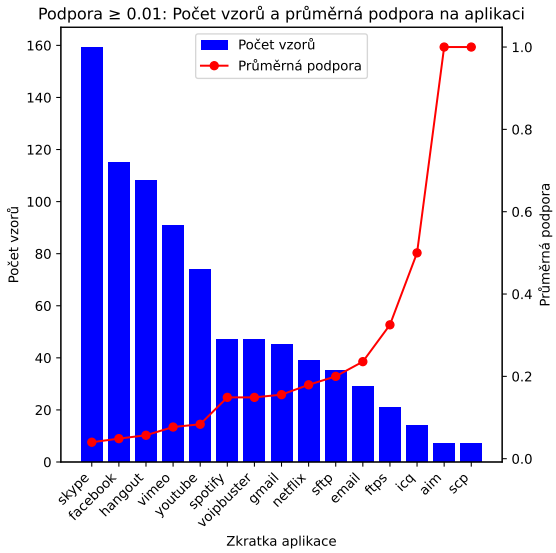
V této fázi z experimentu vyplývá, že nejlepší výsledky v rámci nastavení minimální podpory se pohybují kolem **hodnot 0,2, 0,25 , 0,05 nebo 0,01**. S ohledem na počet vytěžených vzorů se však hodnoty **0,2 a 0,25 jeví jako ideální volba** pro budoucí experimenty. Nabízí totiž srovnatelnou úroveň identifikace jako nižší hodnoty (například 0,05), avšak **s výrazně menším počtem vzorů**, což snižuje výpočetní náročnost i složitost následného zpracování.

6.2.1 Velikost a počet vzorů na aplikaci

Tato část představuje druhou fázi experimentu, která se zaměřuje na analýzu počtu a délky vzorů přiřazených jednotlivým aplikacím. Cílem je detailněji prozkoumat získané vzory a jejich vlastnosti s ohledem na optimalizaci následné identifikace. Analyzovány jsou všechny vzory získané při použití hodnot minimální podpory, které se v předchozí fázi ukázaly jako nejefektivnější: **0,01**, **0,05**, **0,2** a **0,25**. Ostatní parametry zůstávají beze změny – konkrétně se jedná o šířku okna na **15 spojení**, omezení maximálního počtu finálních **kandidátů na 3** a využití metody *JA4+JA4S+SNI* jak pro **generování počáteční kandidátní množiny**, tak pro **nalezení frekventovaných vzorů**. Vzory jsou dále zkoumány za účelem jejich selekce – tedy filtrování na základě jejich počtu a délky, s cílem snížit výpočetní náročnost a zároveň zvýšit efektivitu procesu identifikace.



Obrázek 6.7: Počet vzorů a průměrná podpora na aplikaci – `mobile desktop apps.csv`



Obrázek 6.8: Počet vzorů a jejich průměrná podpora na aplikaci – `iscx.csv`

Graf na obrázku[1] 6.7 znázorňuje počet získaných vzorů při prahu minimální podpory 0,01 pro každou aplikaci z datové sady `mobile_desktop_apps_raw.csv`. Je patrné, že

[1] U všech následujících grafů jsou z důvodu přehlednosti zobrazeny popisky pouze pro každou čtvrtou aplikaci z datové sady `mobile_desktop_apps_raw.csv`.

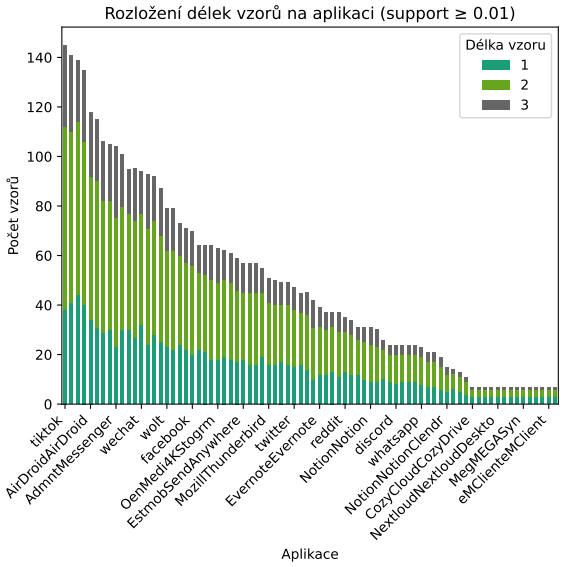
mezi aplikací s nejnižším a nejvyšším počtem vzorů je výrazný rozdíl — konkrétně až **130 vzorů**. V grafu je rovněž vyznačena průměrná podpora vytěžených vzorů pro jednotlivé aplikace, přičemž lze pozorovat, že některé aplikace obsahují vzory s podporou rovnou 1,0. To znamená, že se tyto vzory vyskytují ve všech jejich instancích. Tento jev lze pravděpodobně přičíst malému testovacímu vzorku daných aplikací.

Obdobně je v grafu na obrázku 6.8 znázorněn průměrný počet vzorů a jejich podpora pro jednotlivé aplikace v datové sadě `iscx.csv`. Také v tomto případě je patrný výrazný rozdíl v počtu vzorů mezi jednotlivými aplikacemi, který dosahuje až **152 vzorů**. Kompletní přehled, počtu vzorů a jejich průměrné podpory pro dříve zmíněné hodnoty minimální podpory, je uveden v Příloze D.2.

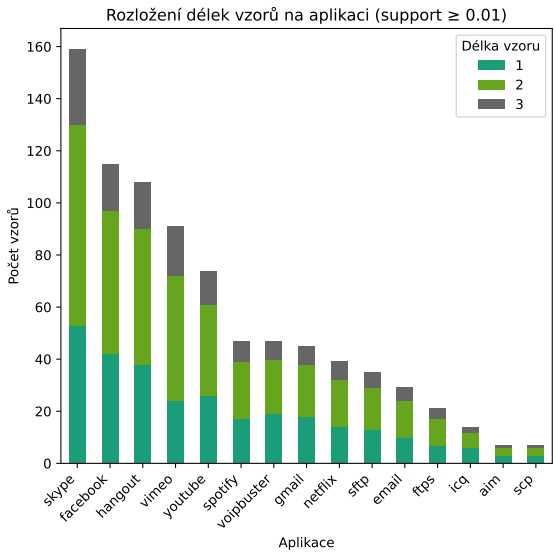
Kvůli tomuto výraznému nepoměru v počtu vzorů napříč aplikacemi bylo nutné při výpočtu podobnosti zohlednit nejen samotnou přítomnost vzorů, ale i jejich podporu (frekvenci výskytu v rámci dané aplikace) a unikátnost (míru výskytu napříč všemi aplikacemi), jak je popsáno v podsekci 5.4.3.

Alternativním přístupem je normalizace počtu vzorů mezi aplikacemi, například filtrováním na přibližně stejný počet vzorů podle stanovených kritérií (např. délka či podpora). Tento krok by mohl předejít zkreslení, které vzniká v případech, kdy některé aplikace disponují nepřiměřeně vysokým počtem vzorů s nízkou podporou, zatímco jiné obsahují pouze několik málo specifických a silně reprezentativních vzorů. Při vyšší minimální podpoře již není rozdíl v počtu získaných vzorů mezi aplikacemi tak markantní. Nicméně identifikace při stejném počtu informativně rovnocenných vzorů by měla teoreticky být vyváženější a tím pádem přesnější.

Z tohoto důvodu je provedena analýza délky vzorů pro jednotlivé aplikace. Kratší vzory mají tendenci vést k častějším, ale obecnějším identifikacím, zatímco delší vzory mohou poskytnout specifičtější identifikaci, avšak s nižší frekvencí výskytu.



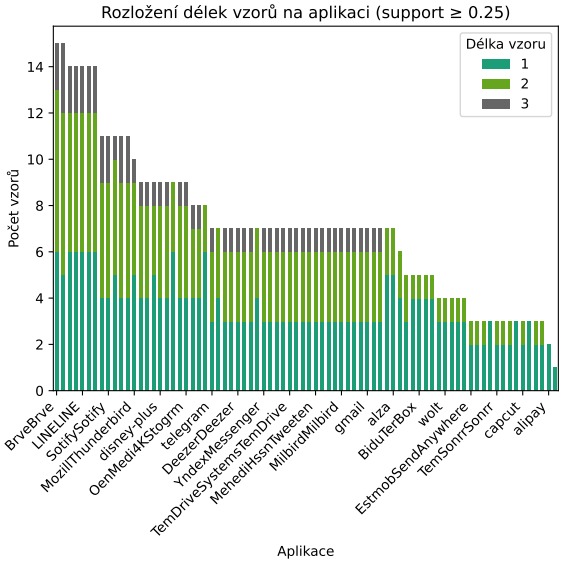
Obrázek 6.9: Rozložení délek vzorů na aplikaci při minimální podpoře 0,01 pro `mobile desktop apps raw.csv`



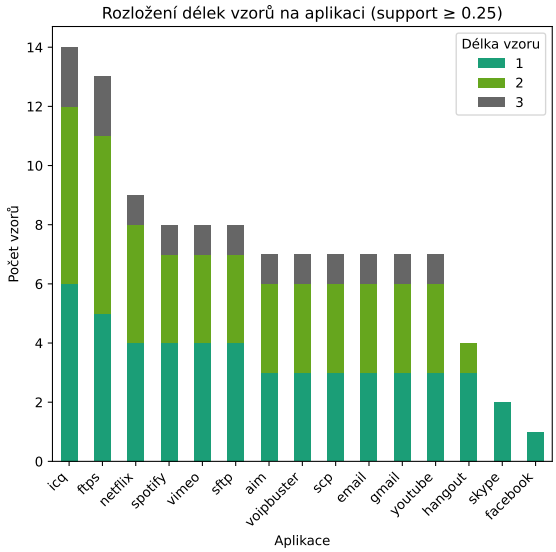
Obrázek 6.10: Rozložení délek vzorů na aplikaci při minimální podpoře 0,01 pro `iscx.csv`

Grafy na obrázcích 6.9 a 6.10 znázorňují rozložení délek získaných vzorů aplikací při minimální podpoře 0.01 pro datové sady `mobile_desktop_apps_raw.csv` a `iscx.csv`. V obou případech převažují vzory o délce 1 a 2. Jak bylo již uvedeno, vzory o délce 3 jsou méně časté, ale mohou poskytnout informativně cennější hodnotu díky své specifičnosti.

Viz.grafy na obrázcích 6.11 a 6.12, které znázorňují stejnou závislost při změně hodnoty minimální podpory, která je nyní nastavena na 0.25. Při této podpoře jsou častější obecnější vzory o délce 1, zatímco vzory o délce 3 jsou zde zastoupeny ve výrazně menším počtu, v některých případech již vůbec. Hodnota podpory 0.26 představuje hranici pro oba datové soubory. Při této hodnotě již pro některé aplikace nebyly nalezeny žádné vzory. Konkrétně se jedná o aplikaci *facebook* v datové sadě `iscx.csv` a *alipay* v sadě `mobile_desktop_apps_raw.csv`. Tento jev je patrný i v grafech, kde je zřetelný pokles počtu vzorů, přičemž u těchto aplikací, jako je *facebook* a *alipay*, se v tomto rozmezí nachází pouze jeden vzor.



Obrázek 6.11: Rozložení délek vzorů na aplikaci při minimální podpoře 0,25 pro `mobile desktop apps raw.csv`



Obrázek 6.12: Rozložení délek vzorů na aplikaci při minimální podpoře 0,25 pro `iscx.csv`

Vzory byly filtrovány dle délky, přičemž v rámci každé délkové skupiny byly seřazeny podle podpory. Byla vytvořena sada filtrů na základě předchozí analýzy počtu a délky vzorů. Následně bylo z každé množiny vybráno N vzorů podle zvoleného filtru.

Různé varianty filtrů byly otestovány pro čtyři hodnoty minimální podpory: **0,01**, **0,05**, **0,2** a **0,25**. Jako vstupní množina pro algoritmus *Apriori* byly konzistentně použity otisky *JA4*, *JA4S* a atribut spojení *SNI*. Cílem této fáze bylo porovnat přesnost výsledné kandidátní množiny vůči referenčnímu stavu bez jakékoli filtrace (viz Obrázky 6.3 a 6.4).

Následující tabulky shrnují nejzajímavější výsledky jednotlivých filtrů z hlediska identifikační přesnosti a počtu použitých vzorů. **Kombinace** označují konkrétní otisky a atributy spojení použitých pro generování počáteční kandidátní množiny. **Podpora** udává hodnotu minimální podpory použitou při těžbě vzorů. **Filtr** reprezentuje způsob filtrování vzorů ve formátu $len(n)Xm$, kde $len(n)$ označuje délku vzoru a Xm specifikuje počet takových

vzorů. Pokud je parametr Xm vynechán, znamená to, že jsou použity všechny vzory dané délky. Dále byly testovány i kombinace více filtrů, které jsou označeny symbolem +.

iscx.csv				
Kombinace	Podpora	Filtr	Přesnost	Prům. délka množiny
JA4	0.01	len(1)x5+len(3)x10	0.91	2.128395
JA4	0.01	len(2)x5	0.91	2.128395
JA4	0.01	len(3)	0.91	2.128395
JA4+JA4S+SNI	0.05	len(3)x10	0.89	1.585185
JA4+JA4S+SNI	0.05	len(3)	0.89	1.585185
JA4+JA4S+SNI	0.05	len(3)x5	0.89	1.585185

Tabulka 6.6: Top 3 výsledky, pro každou kombinaci generující počáteční množinu, podle přesnosti s aplikací filtrů u datové sady `iscx.csv`

V tabulce 6.6 jsou uvedeny výsledky přesnosti jednotlivých filtrů pro datovou sadu `iscx.csv`. Kombinace *JA4*, která vytváří počáteční množinu, vykazuje velmi mírné zlepšení u výše zmíněných filtrů, přičemž průměrná velikost kandidátní množiny zůstává nezměněna. Druhá kombinace *JA4+JA4S+SNI* nevykazuje žádné zlepšení.

mobile_desktop_apps_raw.csv				
Kombinace	Podpora	Filtr	Přesnost	Prům. délka množiny
JA4+JA4S+SNI	0.01	len(1)x2+len(3)x2	0.81	1.713638
JA4+JA4S+SNI	0.05	len(1)x2+len(3)x2	0.81	1.713638
JA4+JA4S+SNI	0.20	len(1)x5+len(2)x10	0.81	1.713638
JA4	0.01	len(2)x5+len(3)x10	0.80	2.790678
JA4	0.01	len(2)x5+len(3)x5	0.78	2.790678
JA4	0.01	len(3)	0.78	2.790678

Tabulka 6.7: Top 3 výsledky, pro každou kombinaci generující počáteční množinu, podle přesnosti s aplikací filtrů u datové sady `mobile desktop apps raw.csv`

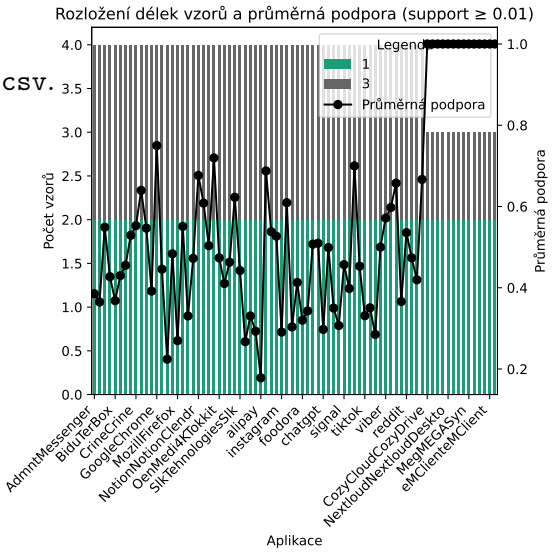
Datová sada `mobile_desktop_apps_raw.csv` při použití filtrů taktéž nevykazuje žádné významné zlepšení v přesnosti identifikace ani ve zmenšení velikosti kandidátní množiny (viz. Tabulka 6.7). Přestože se mohou výsledky jevit jako málo přínosné, z pohledu optimalizace procesu identifikace přinášejí určitý benefit – zejména v podobě redukce počtu vzorů, které je nutné porovnávat, a jejich částečné normalizace napříč aplikacemi.

Z tabulek 6.7 a 6.6 je patrné, že nejlepších výsledků přesnosti je dosahováno při různých kombinacích filtrů a parametrů pro každou z datových sad. Zatímco pro `iscx.csv` se jako nejefektivnější jeví selekce zaměřená na delší vzory (např. `len(3)` nebo `len(2)x5`) a nižší hodnoty podpory, u datové sady `mobile_desktop_apps_raw.csv` dosahují srovnatelné přesnosti filtry kombinující obecné a specifické vzory (např. `len(1)x2+len(3)x2`) i při vyšších hodnotách podpory.

Použitím nižší hodnoty minimální podpory je zároveň možné zachytit i méně časté, ale informačně hodnotnější vzory. Ty mohou být klíčové pro přesnější rozlišení mezi jednotlivými spojeními, která nelze efektivně identifikovat pomocí obecnějších vzorů vznikajících

při vyšších prahových hodnotách. Aplikace filtrů pak umožňuje tyto specifické vzory v databázi ponechat, čímž přispívá k vyváženějšímu kompromisu mezi výpočetní efektivitou a identifikační schopností.

Graf na obrázku 6.13 znázorňuje aplikovanou selekci pomocí filtru `len(1)x2 + len(3)x2` a minimální podpory **0,01** pro datovou sadu `mobile_desktop_apps_raw.csv`. Toto nastavení dosahuje přesnosti **0,8129** při průměrné velikosti finální kandidátní množiny **1,7**. Modus i medián této velikosti jsou rovny **1**. Většina aplikací si uchovává právě dva vzory délky 1 a dva vzory délky 3. U menšiny aplikací však nebyly nalezeny dva vzory délky 3, ale pouze jeden.



Obrázek 6.13: Rozložení délek vzorů a průměrná podpora po aplikaci filtrace `len(1)x2 a len(3)x2` pro `mobile desktop apps raw.csv` s minimální podporou 0,01

Tento experiment ukázal, že selektivní filtrování vzorů může vést k efektivnější (méně náročnější) identifikaci, přičemž negativní dopad na přesnost se projevuje až při použití vyšších hodnot minimální podpory. Při nízké podpoře a vhodně zvolených filtrech je tento dopad téměř zanedbatelný (viz D.2).

Přestože se nejedná o univerzálně aplikovatelné řešení, výsledky potvrzují, že pro různé datové sady je možné nalézt vhodnou kombinaci parametrů (podpora, délka, počet vzorů), která zajistí optimální rovnováhu mezi výpočetní náročností a kvalitou identifikace. Kompletní výsledky se všemi použitými filtry pro obě datové sady a obě kombinace jsou uvedeny v příloze D.

6.3 Šířka okna určující okolí spojení

Kontext každého identifikovaného spojení je určen velikostí posuvného okna, jak je popsáno v podsekci 5.4.3. Ve většině případů se ve středu tohoto okna nachází právě identifikované spojení, zatímco jeho okolí může tvořit buď další spojení stejné aplikace (například v rámci jednoho spuštění), nebo spojení patřící jiným aplikacím. Vzhledem k náhodnému uspořádání testovací množiny může dojít k tomu, že do okna spadají i nesouvisející spojení, což způsobuje informační šum a může negativně ovlivnit kvalitu identifikace.

Cílem tohoto experimentu je určit optimální velikost okna, která zachytí dostatek relevantního kontextu pro přesnou identifikaci, aniž by zahrnovala rušivá data z jiných aplikací.

V rámci experimentu byly parametry nastaveny na použití metody ***JA4+JA4S+SNI*** jak pro generování **počáteční** kandidátní množiny, tak i pro nalezení **častých vzorů**. **Minimální podpora** byla nastavena na **0,01** a byla využita selekce filtrů z předchozího experimentu. Maximální počet kandidátů byl zvolen na **3**. Testovány byly různé hodnoty šířky kontextového okna (vždy liché číslo, aby byl kontext z obou stran vyvážen), konkrétně **3, 5, 7, 9, 11, 21** a **31**. V tabulce 6.8 jsou uvedeny jednotlivé použité filtry, které byly zvoleny na základě nejvyšší dosažené přesnosti v předchozím experimentu.

Kombinace	`mobile_desktop_apps_raw.csv`	`iscx.csv`
JA4	len(2)x5+len(3)x10	len(1)x5+len(3)x10
JA4+JA4S+SNI	len(1)x2+len(3)x2	len(3)x5

Tabulka 6.8: Použité filtry pro experiment

Následující tabulky uvádějí použitou šířku okna a přesnost pro daný otisk nebo kombinaci otisků, které byly použity pro generování počáteční množiny. Nejvyšší dosažené přesnosti jsou v tabulkách vyznačeny tučně.

`mobile_desktop_apps_raw.csv`		
Šířka okna	Přesnost při kombinacích	
	JA4	JA4+JA4S+SNI
3	0.7706	**0.8157**
5	0.8058	0.8133
7	0.8289	0.8133
9	**0.8332**	0.8131
11	0.8280	**0.8157**
21	0.7644	0.8013
31	0.6877	0.7874

Tabulka 6.9: Přesnost identifikace pro různé velikosti okolí pro datovou sadu `mobile desktop apps raw.csv`.

`iscx.csv`		
Šířka okna	Přesnost při kombinacích	
	JA4	JA4+JA4S+SNI
3	0.9111	0.8815
5	0.9012	0.8840
7	0.9012	0.8864
9	0.9062	0.8864
11	0.9062	**0.8914**
21	**0.9185**	0.8840
31	0.8864	0.8691

Tabulka 6.10: Přesnost identifikace pro různé velikosti okolí s *JA4+JA4S+SNI* a *JA4* pro datovou sadu `iscx.csv`.

Na základě analýzy bylo zjištěno, že optimální šířka okna pro jednotlivé datové sady se částečně odvíjí od průměrného počtu spojení při jednom spuštění aplikace. U sady `mobile_desktop_apps_raw.csv`, kde je průměrný **počet spojení 3,15 na jedno spuštění aplikace**, byly nejlepší výsledky dosaženy při **menší šířce** okna 9 nebo 11 (viz. Tabulka 6.9). Naproti tomu u sady `iscx.csv`, kde je průměrný **počet spojení v rámci jednoho spuštění vyšší a to 4,55**, se optimální přesnost posunula k **širším oknům 11 a 21** (viz. Tabulka 6.10). I velmi úzké **okno o velikosti 3** přináší konkurenceschopné výsledky, což lépe odpovídá průměrnému počtu spojení v jednotlivých spuštěních aplikací v testovací množině.

Vzhledem k tomu, že testovací množina byla sestavena tak, že obsahuje přibližně 25 % spojení z každého spuštění (viz.podsekce 5.4.3), může nedostatek širšího kontextu v testovacích datech negativně ovlivnit konečné výsledky klasifikace. Chybějící návaznosti mezi spojeními mohou omezit schopnost zachytit frekventované vzory, které by při plnějším kontextu mohly být lépe rozpoznatelné.

Je však třeba poznamenat, že ve skutečném síťovém provozu zpravidla nebývá možné zachytit čistě oddělená spuštění jednotlivých aplikací bez jakéhokoli informačního šumu pocházejícího od jiných procesů. Volba strategie, kdy je testovací množina sestavena pouze ze zlomku spojení napříč jednotlivými spuštěními, tak do určité míry simuluje reálnou situaci, kdy je třeba aplikace identifikovat v prostředí s překrývající se či souběžně probíhající komunikací. V budoucnu by bylo možné zkoumat například vliv typu aplikace na ideální šířku okna, nebo adaptivní strategie, které dynamicky upravují velikost okna podle délky nebo hustoty síťové komunikace dané aplikace.

6.4 Volba maximálního počtu kandidátů a její vliv na přesnost identifikace

Posledním parametrem, který přímo ovlivňuje přesnost identifikace a délku finální kandidátní množiny, je maximální počet kandidátů – tedy výsledná maximální délka finální množiny kandidátů. Experiment zaměřený na volbu tohoto parametru se soustředí na finální přesnost identifikace a její porovnání s výsledky fingerprintingu pomocí metod *JA4* a *JA4+JA4S+SNI*. Počáteční kandidátní množina je v rámci experimentu určena jak metodou ***JA4***, tak metodou ***JA4+JA4S+SNI***. Vstupem pro *Apriori* jsou položky ***JA4***, ***JA4S*** a ***SNI***. Šířka okna určující kontext spojení byla nastavena na **3 spojení**, jelikož experiment popsaný v sekci 6.3 ukazuje na tuto šířku jako na univerzální pro obě datové sady i obě testované kombinace. Testování identifikace proběhlo **s minimální podporou 0.01 a 0.2** pro soubor `mobile_desktop_apps_raw.csv` a **0.01 a 0.25** pro soubor `iscx.csv`. Tyto hodnoty se ukázaly jako ideální v předchozím experimentu popsaném v sekci 6.2. Pro práh podpory 0.01 při těžbě jsou uvedeny dvě verze výsledků – **s filtrem** (každý filtr byl zvolen shodně jako v předchozím experimentu v sekci 6.3) **a bez filtru**. Maximální zkoumaný počet kandidátů je **4** pro `iscx.csv` a **9** pro `mobile_desktop_apps_raw.csv`. Časové hodnoty byly měřeny na testovacím zařízení s následujícími parametry: CPU: `AMD Ryzen 7 5700U (16) @ 1.80 GHz`, OS: `EndeavourOS x86_64`, RAM: `16 GB`.

6.4.1 Výsledky pro `iscx.csv`

V této sekci se budou experimenty zabývat pouze datovou sadou `iscx.csv`.

`iscx.csv`		
	JA4	**JA4+JA4S+SNI**
Přesnost	0.975	0.924
Délka	3.146	2.094

Tabulka 6.11: Přesnost fingerprintingu *JA4* a jejich kombinace pro `iscx.csv`

Tabulka 6.11 uvádí přesnost a průměrnou délku kandidátní množiny po aplikaci otisků *JA4* a *JA4+JA4S+SNI* (bez využití kontextu). Na tuto tabulku bude dále odkazováno při porovnání jednotlivých verzí v rámci testování.

`iscx.csv` s minimální podporou 0,01						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.635	16.567	1.000	0.679	16.567	1.000
2	0.827	13.919	1.649	0.832	13.919	1.385
3	0.899	16.752	2.128	0.874	16.752	1.585
4	0.951	16.847	2.454	0.923	16.847	1.751

Tabulka 6.12: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře 0,01, porovnávající metody *JA4* a *JA4+JA4S+SNI* pro generování počáteční kandidátní množiny nad datovou sadou `iscx.csv`.

Při minimální podpoře 0,01 bez použití filtrů (viz.Tabulka 6.12) dosahuje jednoznačná identifikace úspěšnosti mezi 63,5 % a 67,8 %. Při maximálním počtu kandidátů rovném 4 je přesnost identifikace založená pouze na otisku *JA4* o **2,4 %** nižší, nicméně průměrná velikost finální kandidátní množiny je menší – **2,5** oproti **3,1**. Při využití kombinace *JA4+JA4S+SNI* je přesnost **téměř identická**, avšak velikost kandidátní množiny se snižuje na **1,75 oproti 2,094**. Celková doba trvání identifikace se pohybuje v rozmezí **13 až 17** sekund.

`iscx.csv` s minimální podporou 0,25						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.607	1.370	1.000	0.751	1.370	1.000
2	0.701	1.385	1.649	0.817	1.385	1.385
3	0.859	1.480	2.128	0.872	1.480	1.585
4	0.904	1.861	2.454	0.881	1.861	1.751

Tabulka 6.13: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře 0,25, porovnávající metody *JA4* a *JA4+JA4S+SNI* pro generování počáteční kandidátní množiny nad datovou sadou `iscx.csv`.

V případě vyšší minimální podpory, jak je uvedeno v Tabulce 6.13, klesá doba trvání identifikace na pouhých **1,3 až 1,8 sekundy**. Jednoznačná identifikace s použitím kombinace otisků dosahuje vyšší úspěšnosti. Při použití maximálně čtyř kandidátů vykazuje metoda *JA4* **ztrátu 7,1%**, zatímco kombinace ztrácí pouze **3,3 %**. Průměrná délka finální kandidátní množiny zůstává nezměněna, což je dáno jejím omezením maximálním počtem kandidátů.

Ideálním kompromisem se jeví použití minimální podpory **0,01 s aplikací filtrů** (viz. Tabulka 6.14). Metoda *JA4* vykazuje ztrátu **4,4%** v přesnosti, přičemž doba trvání identifikace se pohybuje mezi **2 až 3** sekundami. Kombinace *JA4+JA4S+SNI* ztrácí méně, konkrétně **2%**, a zároveň je **nejrychlejší** z testovaných verzí, s dobou trvání kolem **1,1 až 1,2 sekundy**.

`iscx.csv` s minimální podporou 0,01 s filtry						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.647	2.100	1.000	0.664	1.168	1.000
2	0.847	3.576	1.649	0.805	1.146	1.385
3	0.906	2.217	2.128	0.881	1.191	1.585
4	**0.931**	**2.355**	**2.454**	**0.904**	**1.281**	**1.751**

Tabulka 6.14: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře **0,25**, porovnávající metody *JA4* a *JA4+JA4S+SNI* pro generování počáteční kandidátní množiny a aplikaci selekce vzorů pomocí filtrů nad datovou sadou `iscx.csv`. Aplikovaný filtr pro *JA4* `len(1)x5+len(3)x10`, zatímco pro kombinaci *JA4+JA4S+SNI* byl zvolen filtr `len(3)x5`.

6.4.2 Výsledky pro `mobile_desktop_apps_raw.csv`

V této sekci se budou experimenty zabývat pouze datovou sadou `mobile_desktop_apps_raw`.

`mobile_desktop_apps_raw.csv`		
	JA4	**JA4+JA4S+SNI**
Přesnost	0.966	0.854
Délka	14.298	3.382

Tabulka 6.15: Přesnost otisků *JA4* a jejich kombinací pro `mobile desktop apps raw.csv`

Tabulka 6.15 uvádí přesnost a průměrnou délku kandidátní množiny po aplikaci otisků *JA4* a *JA4+JA4S+SNI* (bez kontextu). Na tuto tabulku bude dále odkazováno při porovnání jednotlivých verzí v rámci testování.

Vysoká přesnost identifikace by mohla být očekávána u následující varianty (minimální podpora 0,01 bez filtrování vzorů), avšak dosažené výsledky naznačují opak (viz. 6.16). U metody *JA4* dochází při maximálním počtu kandidátů, tedy 9, k poklesu přesnosti o **7,1 %**, přičemž průměrná velikost finální kandidátní množiny byla snížena z **14,3 na 7,1**. I přes redukci množiny přibližně na polovinu nebyla dosažena uspokojivá přesnost.

`mobile_desktop_apps_raw.csv` s minimální podporou 0,01						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.590	486.053	1.000	0.693	486.053	1.000
3	0.759	518.081	2.791	0.781	518.081	1.714
5	0.842	482.950	4.399	0.816	482.950	2.180
7	0.852	516.947	5.774	0.824	516.947	2.486
9	0.895	555.791	7.142	0.845	555.791	2.781

Tabulka 6.16: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře 0,01, porovnávající metody *JA4* a *JA4+JA4S+SNI* (pro generování počáteční kandidátní množiny) nad datovou sadou `mobile desktop apps raw.csv`.

Zdá se, že při kombinaci nízké minimální podpory a malého kontextového okna dochází k tvorbě velkého množství příliš specifických vzorů, což negativně ovlivňuje celkovou přesnost identifikace.

Kombinace *JA4+JA4S+SNI* zaznamenala pouze mírné zhoršení přesnosti o **0,9 %**, přičemž průměrná délka finální kandidátní množiny klesla z **3,4 na 2,8**. Nejvýraznějším nedostatkem této varianty je však výrazná časová náročnost – identifikace trvá přibližně **8 až 9 minut**.

Ani varianta s minimální podporou 0,2 nedosahuje příliš přesných výsledků (viz. 6.17). Je zaznamenán pokles přesnosti o **11,8 %** při devíti kandidátech a velikosti finální kandidátní množiny **7,1** oproti původním **14,3**. Kombinace *JA4+JA4S+SNI* vykazuje horší výsledky v identifikaci o pouhých **0,8 %**, při snížení průměrné velikosti finální kandidátní množiny na **2,8** z **3,4**. Časová náročnost klesá na **90 sekund**.

`mobile_desktop_apps_raw.csv` s minimální podporou 0,2						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.404	90.781	1.000	0.672	90.781	1.000
3	0.622	92.644	2.791	0.787	92.644	1.714
5	0.742	90.124	4.399	0.815	90.124	2.180
7	0.813	87.807	5.774	0.833	87.807	2.486
9	0.848	91.002	7.142	0.846	91.002	2.781

Tabulka 6.17: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře 0,2, porovnávající metody *JA4* a *JA4+JA4S+SNI* (pro generování počáteční kandidátní množiny) nad datovou sadou `mobile desktop apps raw.csv`.

Tabulka 6.18 uvádí výsledky pro minimální podporu 0,01 a aplikaci filtrů. Snížení přesnosti, při volbě 9 kandidátů, je pouhých **3,5 %**. Spolu s **eliminací poloviny kandidátu** z finální kandidátní množiny. Doba potřebná pro identifikaci je v průměru **2 minuty**. Kombinace *JA4+JA4S+SNI* dosahuje identické přesnosti s fingerprintingem a zároveň nejkratší doby identifikace ze všech variant. Stejně jako u sady *iscx.csv* je tato varianta ideální volbou.

`mobile_desktop_apps_raw.csv` s minimální podporou 0,01 s filtry						
Počet kandidátů	JA4			JA4+JA4S+SNI		
	Přesnost	Čas [s]	Prům. délka	Přesnost	Čas [s]	Prům. délka
1	0.573	111.425	1.000	0.670	53.268	1.000
3	0.771	120.797	2.791	0.815	51.985	1.714
5	0.845	105.540	4.399	0.839	50.342	2.180
7	0.902	120.590	5.774	0.845	49.652	2.486
9	**0.931**	**118.449**	**7.142**	**0.855**	**53.959**	**2.781**

Tabulka 6.18: Výsledky experimentu s různým nastavením maximálního počtu kandidátů při minimální podpoře 0,01, porovnávající metody *JA4* a *JA4+JA4S+SNI* (pro generování počáteční kandidátní množiny) a aplikaci filtru `len(2)x5+len(3)x10` pro *JA4* a `len(1)x2+len(3)x2` pro *JA4+JA4S+SNI* s `mobile desktop apps raw.csv`.

Vysoký počet kandidátů neumožňuje jednoznačnou identifikaci ve většině případů avšak, použití této metody **výrazně snižuje velikost finální kandidátní množiny**, omezuje její maximální rozsah a zároveň zaručuje, že každá množina obsahuje **alespoň jednoho** kandidáta (viz. Tabulka 6.19).

Použití	Kombinace	Průměr	Medián	Modus	Maximum	Minimum
Otisky	JA4	14,3	16	23	23	0
Kontext	JA4	7,14	9	9	9	1
Otisky	JA4+JA4S+SNI	3,4	1	1	51	0
Kontext	JA4+JA4S+SNI	2,78	1	1	9	1

Tabulka 6.19: Statistiky velikosti finálních kandidátních množin (otisky vs. kontext)

Kapitola 7

Závěr

Cílem této bakalářské práce byla identifikace aplikací v síťovém provozu na základě TLS spojení a jeho okolí, přinášející zkrácení délky finální kandidátní množiny aplikací, které generují otisky *JA3* a *JA4*. Zároveň měla být zachována co nejvyšší přesnost s ohledem na časovou náročnost, aby bylo možné výsledné řešení využít v reálném provozu.

Pro poskytnuté datové sady se zdá být tento cíl splněn. Síťová komunikace, zejména protokol *TLS*, byla podrobně analyzována. Datové sady byly prostudovány a vyhodnoceny, stejně jako možnosti identifikace na základě těchto dat. Pro identifikaci na základě okolních spojení byl navržen a implementován vlastní systém. Hodnocení úspěšnosti identifikace bylo převzato z předchozích studií a mírně upraveno pro potřeby navrženého přístupu. V závěrečné fázi byl systém i jeho výsledky analyzován a zhodnocen.

Navržený způsob identifikace spolu s aplikací selekce vzorů dosahuje dobrých výsledků při eliminaci nadbytečných kandidátů, a to při zachování požadované přesnosti. Celý proces přitom probíhá v přijatelných časových mezích.

Pro datovou sadu *iscx.csv* bylo dosaženo přesnosti 93,1 %, přičemž průměrná velikost kandidátní množiny se zúžila z 3,1 na 2,5 kandidáta na jedno identifikované spojení. Stejné přesnosti bylo dosaženo i u datové sady `mobile_desktop_apps_raw.csv`, u níž průměrná délka kandidátní množiny klesla na méně než polovinu původní hodnoty.

Všechny zaznamenané statistiky rovněž potvrzují výrazné snížení velikosti kandidátních množin. Tyto množiny jsou navíc normalizovány i v okrajových případech – tedy situacích, kdy by jinak mohlo být kandidátů příliš mnoho nebo naopak žádný.

Práci by bylo možné dále rozšířit například o jiný nebo doplňující data-miningový algoritmus, který by dokázal získávat informace jiným způsobem — toto možné rozšíření bylo v návrhu systému zohledněno. Rovněž by bylo přínosné otestovat další kombinace vstupních dat a různá nastavení filtrů. V případě dalšího testování by dalším krokem mohlo být nasazení systému v reálné síťové komunikaci s cílem ověřit jeho efektivitu při zpracování reálných dat – a to jak z hlediska rychlosti, tak přesnosti.

Literatura

[1] Aggarwal, C. C. a Han, J. *Frequent pattern mining*. 1. vyd. Cham: Springer, září 2014. ISBN 978-3-319-07820-5.

[2] Agrawal, R. a Srikant, R. Fast Algorithms for Mining .... In: Bocca, J. B., ed. *Proceedings of the 20th International Conference on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Září 1994, s. 487–499. VLDB '94. ISBN 1558601538.

[3] Althouse, J. JA4+ Network Fingerprinting. *FoxIO Blog* online. FoxIO, září 2023. Dostupné z: https://blog.foxio.io/ja4%2B-network-fingerprinting. [cit. 2024-10-21].

[4] Althouse, J. a Lindeman, L. TLS Fingerprinting with JA3 and JA3S – Salesforce Engineering Blog. *Salesforce Engineering Blog* online. Salesforce, duben 2022. Dostupné z: https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967. [cit. 2024-10-21].

[5] Benjamin, D. *RFC 8701: Applying Generate Random Extensions And Sustain Extensibility (GREASE)...*. 8701. IETF, leden 2020. Request for Comments: 8701.

[6] Burgetová, I.; Matoušek, P. a Ryšavý, O. Towards Identification of Network Applications.... In: *The Proceedings of the 8th Cyber Security in Networking Conference (CSNet 2024)*. Paris: IEEE Communications Society, 2024, sv. 8, s. 213–221. IEEE Explore. ISBN 979-8-3315-3410-3.

[7] Chen, M.-S.; Park, J. S. a S.Yu, P. Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, 1998, sv. 10, č. 2, s. 209–221.

[8] Eastlake, D. E. *RFC 6066: Transport Layer Security (TLS) Extensions: Extension Definitions*. 6066. IETF, leden 2011. Request for Comments: 6066.

[9] Eddy, W. M. *RFC 9293: Transmission Control Protocol (TCP)*. 9293. IETF, srpen 2022. Request for Comments: 9293.

[10] Enghardt, R.; Pauly, T.; Perkins, C. et al. *RFC 8922: A Survey of the Interaction between Security Protocols and Transport Services*. 8922. IETF, říjen 2020. Request for Comments: 8922.

[11] Friedl, S.; Popov, A.; Langley, A. a Stephan, E. *RFC 7301: Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension*. IETF, červenec 2014. Request for Comments: 7301.

[12] Han, J.; Kamber, M. a Pei, J. *Data mining: concepts and techniques*. 3. vyd. Amsterdam: Morgan Kaufmann, červen 2012. ISBN 9780123814807.

[13] Iyengar, J. a Thomson, M. *RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport*. 9000. Květen 2021. Request for Comments: 9000.

[14] Kurose, J. F. a Ross, K. W. *Computer Netrowking: a top-down approach*. 8. vyd. Harlow: Pearson Education Limited, 2021. 750 s. ISBN 978-1-292-40546-9.

[15] Liu, C.; Yan, X.; Yu, H. et al. Mining Behavior Graphs for "Backtrace" of Noncrashing Bugs. In: *Proceedings of the 2005 SIAM International Conference on Data Mining (SDM)*. 2005, s. 286–297.

[16] Matoušek, P. *Síťové služby a jejich architektura*. 1. vyd. Brno: Nakladatelství Vysokého učení technického v Brně VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1.

[17] Matoušek, P.; Burgetová, I.; Ryšavý, O. et al. On Reliability of JA3 Hashes .... In: *Digital Forensics and Cyber Crime. ICDF2C 2020*. Boston: Springer International Publishing, 2021, sv. 351, s. 1–22. Lecture Notes of the Institute for CS. ISBN 978-3-030-68733-5.

[18] Matoušek, P.; Burgetová, I. a Victor, M. *Mobile Device Fingerprinting*. Brno: Fakulta informačních technologií VUT v Brně, 2020. 65 s.

[19] Matoušek, P.; Ryšavý, O. a Burgetová, I. Experience Report: Using JA4+ Fingerprints for Malware Detection in Encrypted Traffic. In: *Proceedings of 20th International Conference on Network and Service Management*. Prague: [b.n.], 2024, s. 1–5.

[20] Nir, Y.; Josefsson, S. a Pégourié Gonnard, M. *RFC 8422: Elliptic Curve Cryptography (ECC) Cipher Suites for TLS...*. 8422. IETF, srpen 2018. Request for Comments: 8422.

[21] Oppliger, R. *SSL and TLS: Theory and Practice, Third Edition*. 3. vyd. University of Berne, Switzerland: Artech House, 2023. Artech House Information Security and Privacy Library. ISBN 9781685690168.

[22] Popov, A. *RFC 7465: Prohibiting RC4 Cipher Suites*. 7465. IETF, únor 2015. Request for Comments: 7465.

[23] Rescorla, E. *The Transport Layer Security (TLS) Protocol Version 1.3*. 8446. IETF, srpen 2018. Request for Comments: 8446.

[24] Rescorla, E. a Dierks, T. *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. 5246. IETF, srpen 2008. Request for Comments: 5246.

[25] Rescorla, E.; Oku, K.; Sullivan, N. et al. *TLS Encrypted Client Hello*. Internet-Draft draft-ietf-tls-esni-22. IETF, září 2024. Dostupné z: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/22/. [cit. 2024-10-14]. Work in Progress.

[26] Stallings, W. *Cryptography and network security: principles and practice*. 7. vyd. Harlow: Pearson Education Limited, 2017. 767 s. ISBN 978-1-292-15858-7.

[27] Stevens, M.; Bursztein, E.; Karpman, P. et al. *SHAttered: The first collision for full SHA-1.* Amsterdam: Google Research a CWI Amsterdam, 2017.

[28] Thomson, M. a Turner, S. *RFC 9001: Using TLS to Secure QUIC.* 9001. Květen 2021. Request for Comments: 9001.

Příloha A

Obsah odevzdaný na úložiště *NextCloud*

Tato kapitola znázorňuje strukturu odevzdaných souborů na úložišti *NextCloud*.

```
.
|--- src
|   |---aggregate.py......(skript sloužící pro~agregaci dat z datových sad)
|   |--- config.py.................................(konfigurační soubor)
|   |--- data/.....................................(složka s datovými sadami)
|   |   |--- gh-repos.md
|   |   |--- iscx.csv
|   |   |--- iscx-raw.csv
|   |   |-- mobile_desktop_apps_raw.csv
|   |--- identify/..............(složka s moduly pro identifikaci aplikací)
|   |   |--- command_line_parser.py..(zpracování parametrů příkazové řádky)
|   |   |--- database.py...(správa přístupu k datům a jejich strukturování)
|   |   |--- fingerprinting.py ........(identifikace spojení pomocí otisků)
|   |   |--- __init__.py
|   |   |--- ja_context.py.....................(kombinovaná identifikace)
|   |   |--- logger.py...........................(logování chodu programu)
|   |   |-- pattern_matching.py...(implementace  PatternMatching a Apriori)
|   |--- main.py.............................(hlavní spustitelný skript)
|   |--- Makefile............................(automatizace běžných příkazů)
|   |--- out/................................(složka s výýsledky experiemntů)
|   |--- README.md.........................(popis a dokumentace projektu)
|   |-- requirements.txt.....................(seznam závislostí projektu)
|--- thesis-src/.....................(zdrojové soubory technické zprávy)
|-- xpomsa00-BP.pdf.....................................(technická zpráva)
```

Příloha B

Rozšířené seznámení s datovými sadami a strukturou

V této části zprávy jsou uvedeny kompletní tabulky obsahující všechny atributy datových sad. Pro úplnost a transparentnost jsou všechny hodnoty, včetně jejich unikátních a celkových počtů, uvedeny v tabulkách bez jakýchkoli úprav.

B.1 Kompletní přehled hodnot v datových sadách

Tabulky jsou následující: Tabulka B.1 obsahuje přehled atributů v datové sadě `iscx.csv`, tabulka B.2 uvádí přehled atributů v datové sadě `iscx-raw.csv` a tabulka B.3 poskytuje přehled atributů v datové sadě `mobile_desktop_apps_raw.csv`.

B.1.1 iscx.csv

`iscx.csv`				
Atribut	**Počet unikátních hodnot**	**(v %)**	**Celkový počet**	**(v %)**
SrcIP	17	0.70	2436	100.00
DstIP	426	17.49	2436	100.00
SrcPort	2297	94.29	2436	100.00
DstPort	159	6.53	2436	100.00
SNI	207	8.50	2092	85.88
OrgName	72	2.96	2300	94.42
JA3hash	46	1.89	2436	100.00
JA4hash	42	1.72	2436	100.00
AppName	16	0.66	2436	100.00
Type	2	0.08	2436	100.00
JA3Shash	117	4.80	2399	98.48
JA4Shash	127	5.21	2399	98.48
Filename	109	4.47	2399	98.48
Version	1	0.04	2399	98.48

Tabulka B.1: Přehled všech atributů v `iscx.csv`

B.1.2 iscx-raw.csv

iscx-raw.csv				
Atribut	**Unikátní hodnoty**	**(v %)**	**Celkový počet**	**(v %)**
SrcIP	17	0.70	2436	100.00
DstIP	426	17.49	2436	100.00
SrcPort	2297	94.29	2436	100.00
DstPort	159	6.53	2436	100.00
Proto	1	0.04	2436	100.00
SNI	207	8.50	2092	85.88
OrgName	72	2.96	2300	94.42
TLSVersion	3	0.12	2436	100.00
ClientCipherSuite	24	0.99	2436	100.00
ClientExtensions	32	1.31	2436	100.00
ClientSupportedGroups	5	0.21	2342	96.14
EC_fmt	2	0.08	2342	96.14
ALPN	6	0.25	1310	53.78
SignatureAlgorithms	8	0.33	2398	98.44
ClientSupportedVersions	0	0.00	0	0.00
JA3hash	46	1.89	2436	100.00
JA4hash	42	1.72	2436	100.00
JA4_raw	42	1.72	2436	100.00
AppName	16	0.66	2436	100.00
Type	2	0.08	2436	100.00
ServerCipherSuite	22	0.90	2399	98.48
ServerExtensions	45	1.85	2399	98.48
ServerSupportedVersions	0	0.00	0	0.00
JA3Shash	117	4.80	2399	98.48
JA4Shash	127	5.21	2399	98.48
JA4S_raw	127	5.21	2399	98.48
Filename	109	4.47	2399	98.48
Version	1	0.04	2399	98.48

Tabulka B.2: Přehled všech atributů v `iscx-raw.csv`

B.1.3 mobile_desktop_apps_raw.csv

mobile_desktop_apps_raw.csv				
Atribut	**Unikátní hodnoty**	**(v %)**	**Celkový počet**	**(v %)**
SrcIP	1254	5.89	21301	100.00
DstIP	1686	7.92	21301	100.00
SrcPort	5823	27.34	21301	100.00
DstPort	2	0.01	21301	100.00
Proto	2	0.01	21301	100.00
SNI	929	4.36	21259	99.80
OrgName	197	0.92	21273	99.87
TLSVersion	1	0.00	21301	100.00
ClientCipherSuite	36	0.17	21301	100.00
ClientExtensions	10462	49.12	21301	100.00
ClientSupportedGroups	61	0.29	21301	100.00
EC_fmt	2	0.01	20522	96.34
ALPN	9	0.04	19505	91.57
SignatureAlgorithms	17	0.08	21301	100.00
ClientSupportedVersions	50	0.23	18938	88.91
JA3hash	10495	49.27	21301	100.00
JA4hash	123	0.58	21301	100.00
JA4_raw	123	0.58	21301	100.00
AppName	78	0.37	21301	100.00
Type	2	0.01	21266	99.84
ServerCipherSuite	11	0.05	21180	99.43
ServerExtensions	56	0.26	21180	99.43
ServerSupportedVersions	2	0.01	15977	75.01
JA3Shash	83	0.39	21180	99.43
JA4Shash	103	0.48	21180	99.43
JA4S_raw	103	0.48	21180	99.43
Filename	1746	8.20	21180	99.43
Version	1	0.00	21180	99.43

Tabulka B.3: Přehled všech atributů v `mobile_desktop_apps_raw.csv`

B.2 Kompletní přehled aplikací v datových sadách

Dále jsou uvedeny nezkrácené tabulky obsahující všechny aplikace v daných datových sadách spolu s jejich zastoupením. Tabulky jsou následující:

- B.4 – Přehled aplikací v datové sadě `mobile_desktop_apps_raw.csv`.
- B.5 – Přehled aplikací v datové sadě `iscx.csv`.

B.2.1 Aplikace v mobile_desktop_apps_raw.csv

mobile_desktop_apps_raw.csv					
App	**Počet**	**Podíl (%)**	**App**	**Počet**	**Podíl (%)**
AirDroidAirDroid	1774	8.33	google-play	135	0.63
OerOer	1441	6.76	facebook	134	0.63
BiduTerBox	1252	5.88	gmail	130	0.61
AdmntMessenger	1180	5.54	instagram	129	0.61
accuweather	1052	4.94	mapy-cz	108	0.51
OenMedi4KStogrm	1020	4.79	viber	100	0.47
OenMedi4KTokkit	1012	4.75	LINELINE	95	0.45
aliexpress	719	3.38	regiojet	94	0.44
TemSonrrSonrr	717	3.37	snapchat	85	0.40
MozillFirefox	637	2.99	signal	83	0.39
EstmobSendAnywhere	607	2.85	packeta	82	0.38
HedsetHedset	554	2.60	whatsapp	81	0.38
EvernoteEvernote	469	2.20	TymnixEletorrent	79	0.37
capcut	413	1.94	spotify	78	0.37
shein	368	1.73	CloudAGCloudDrive	78	0.37
TIDALMusiASTIDAL	365	1.71	OenWhiserSystemsSignl	77	0.36
BeeerBeeer	342	1.61	disney-plus	77	0.36
RkutenViber	300	1.41	ZoomZoom	75	0.35
temu	299	1.40	MirosoftEdge	75	0.35
GoogleChrome	287	1.35	messenger	74	0.35
waze	267	1.25	eMClienteMClient	72	0.34
NotionNotion	263	1.23	TrillinTrillin	72	0.34
DeezerDeezer	262	1.23	YndexMessenger	72	0.34
MozillThunderbird	253	1.19	ProtonProtonDrive	72	0.34
AsnAsn	248	1.16	CozyCloudCozyDrive	72	0.34
CrineCrine	232	1.09	netflix	71	0.33
tiktok	217	1.02	twitter	64	0.30
SlkTehnologiesSlk	214	1.00	reddit	54	0.25
trello	205	0.96	wechat	54	0.25
SotifySotify	199	0.93	discord	50	0.23
wolt	179	0.84	muj-vlak	46	0.22
alipay	177	0.83	NextloudNextloudDeskto	36	0.17
alza	167	0.78	MehediHssnTweeten	36	0.17
BrveBrve	150	0.70	MegMEGASyn	35	0.16
NotionNotionClendr	145	0.68	SlshedIoInssist	27	0.13
chatgpt	144	0.68	TemDriveSystemsTemDrive	21	0.10
foodora	142	0.67	MilbirdMilbird	15	0.07
BiglySoftwreBiglyBT	141	0.66	telegram	12	0.06
youtube	137	0.64	TelegrmTelegrmDeskto	1	0.00

Tabulka B.4: Přehled všech aplikací v mobile_desktop_apps_raw.csv

B.2.2 Aplikace v iscx.csv

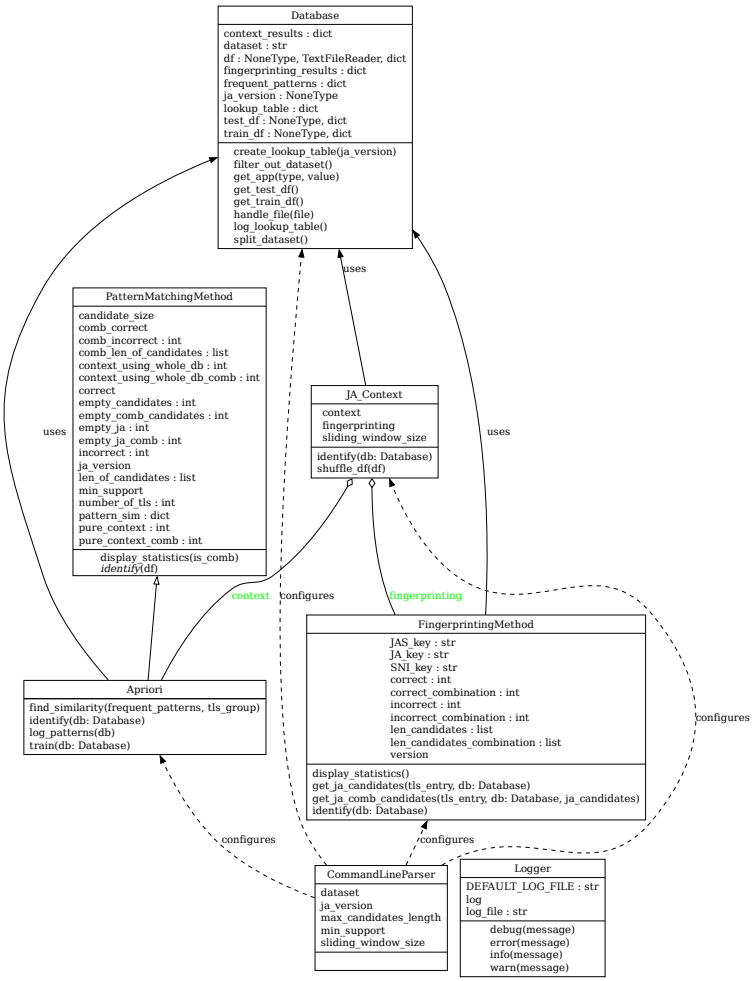
iscx.csv		
Aplikace	**Počet**	**Podíl (%)**
hangout	553	22.70
skype	436	17.90
email	315	12.93
facebook	229	9.40
ftps	229	9.40
vimeo	211	8.66
youtube	209	8.58
voipbuster	64	2.63
spotify	60	2.46
netflix	39	1.60
gmail	35	1.44
sftp	21	0.86
aim	11	0.45
scp	9	0.37
bittorrent	8	0.33
icq	7	0.29

Tabulka B.5: Přehled všech aplikací v `iscx.csv`

Příloha C

Rozšířený diagram tříd

Tento diagram C.1 tříd byl automaticky vygenerován pomocí nástroje `pyreverse`, který je součástí balíku `pylint`. Diagram vizuálně zachycuje strukturu systému, hierarchii tříd a jejich vzájemné vztahy, včetně dědičnosti a závislostí.



Obrázek C.1: Rozšířený diagram tříd

Příloha D

Rozšířené výsledky experimentů

Tato kapitola obsahuje podrobné výstupy z provedených experimentů. Jsou zde uvedeny grafy, tabulky a další doplňující materiály, sloužící k hlubší analýze a interpretaci dosažených výsledků.

D.1 Experiment 1

Následují neupravené a nezkrácené tabulky uvádějící přesnost, průměrnou délku a čas potřebný k identifikaci pro všechny testované kombinace.

Zúžení databáze kandidátů pomocí JA3					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas[s]**
JA3	0.837	2.128	0.279	2.778	74.470
JA4	0.837	2.128	0.390	2.792	60.340
JA3+JA4	0.837	2.128	0.298	2.796	98.680
JA3+JA3S	0.847	2.128	0.365	2.796	101.050
JA4+JA4S	0.842	2.128	0.487	2.796	112.910
JA3+JA3S+SNI	0.852	2.128	0.371	2.796	163.680
JA4+JA4S+SNI	0.847	2.128	0.490	2.796	185.050
JA34S*	0.842	2.128	0.444	2.796	363.750
JA34S*+SNI	0.842	2.128	0.441	2.796	575.500
JA34S*+SNI+ORG	0.849	2.128	0.423	2.796	1123.730
JA34S*+SNI+ORG+CE	0.849	2.128	0.379	2.796	2157.100

Tabulka D.1: Výsledky experimentu s kombinacemi položek při zúžení databáze kandidátů pomocí otisku *JA3*

Zúžení databáze kandidátů pomocí JA4					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas[s]**
JA3	0.837	2.128	0.304	2.781	53.660
JA4	0.837	2.128	0.387	2.785	32.340
JA3+JA4	0.837	2.128	0.361	2.789	48.100
JA3+JA3S	0.847	2.128	0.507	2.791	46.220
JA4+JA4S	0.842	2.128	0.558	2.791	58.500
JA3+JA3S+SNI	0.852	2.128	0.489	2.791	66.340
JA4+JA4S+SNI	0.847	2.128	0.548	2.791	84.030
JA34S*	0.842	2.128	0.577	2.791	150.820
JA34S*+SNI	0.842	2.128	0.549	2.791	225.220
JA34S*+SNI+ORG	0.849	2.128	0.533	2.791	400.770
JA34S*+SNI+ORG+CE	0.849	2.128	0.507	2.791	840.660

Tabulka D.2: Výsledky experimentu s kombinacemi položek při zúžení databáze kandidátů pomocí otisku *JA4*

Zúžení databáze kandidátů pomocí JA3+JA3S+SNI					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas[s]**
JA3	0.867	1.583	0.385	2.287	74.470
JA4	0.867	1.583	0.771	1.747	60.340
JA3+JA4	0.872	1.583	0.764	1.745	98.680
JA3+JA3S	0.872	1.583	0.793	1.734	101.050
JA4+JA4S	0.869	1.583	0.800	1.734	112.910
JA3+JA3S+SNI	0.872	1.583	0.790	1.734	163.680
JA4+JA4S+SNI	0.867	1.583	0.798	1.734	185.050
JA34S*	0.867	1.583	0.793	1.734	363.750
JA34S*+SNI	0.867	1.583	0.792	1.734	575.500
JA34S*+SNI+ORG	0.869	1.583	0.774	1.734	1123.730
JA34S*+SNI+ORG+CE	0.869	1.583	0.761	1.734	2157.100

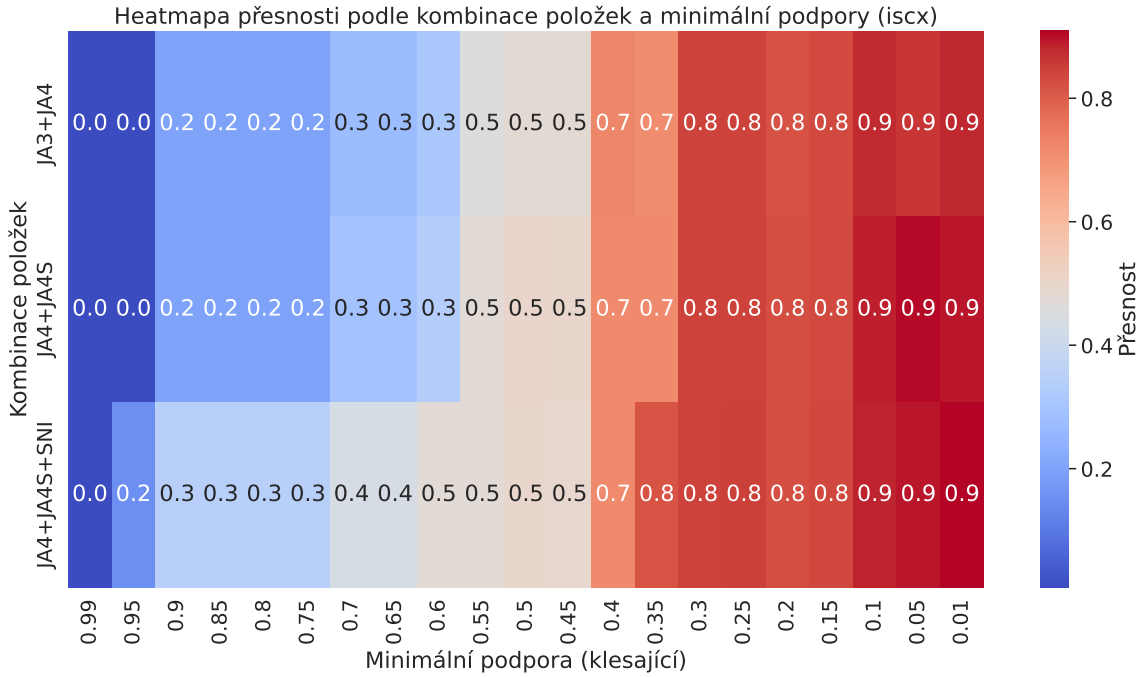
Tabulka D.3: Výsledky experimentu s kombinacemi položek při zúžení databáze kandidátů pomocí kombinace *JA3+JA3S+SNI*

Zúžení databáze kandidátů pomocí JA4+JA4S+SNI					
Kombinace položek	`iscx.csv`		`mobile_desktop_apps_raw.csv`		
	Přesnost	**Délka**	**Přesnost**	**Délka**	**Čas[s]**
JA3	0.869	1.585	0.385	2.277	53.660
JA4	0.869	1.585	0.770	1.727	32.340
JA3+JA4	0.869	1.585	0.768	1.726	48.100
JA3+JA3S	0.874	1.585	0.793	1.714	46.220
JA4+JA4S	0.869	1.585	0.798	1.714	58.500
JA3+JA3S+SNI	0.874	1.585	0.795	1.714	66.340
JA4+JA4S+SNI	0.869	1.585	0.804	1.714	84.030
JA34S*	0.869	1.585	0.795	1.714	150.820
JA34S*+SNI	0.869	1.585	0.797	1.714	225.220
JA34S*+SNI+ORG	0.872	1.585	0.782	1.714	400.770
JA34S*+SNI+ORG+CE	0.872	1.585	0.774	1.714	840.660

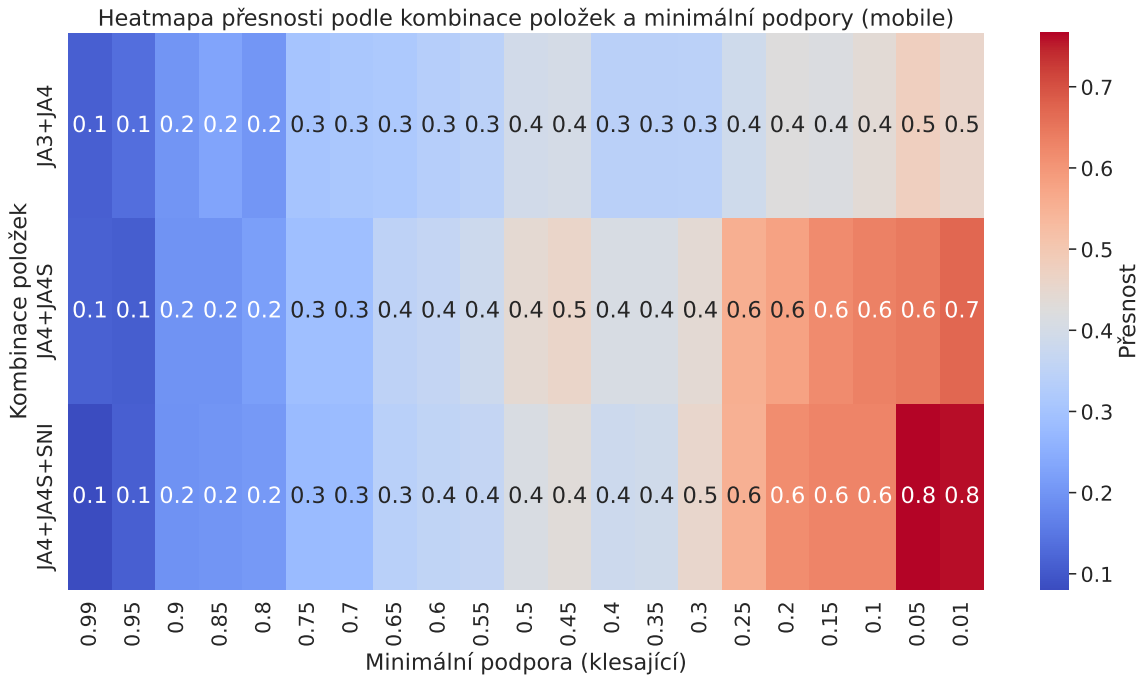
Tabulka D.4: Výsledky experimentu s kombinacemi položek při zúžení databáze kandidátů pomocí kombinace *JA4+JA4S+SNI*

D.2 Experiment 2

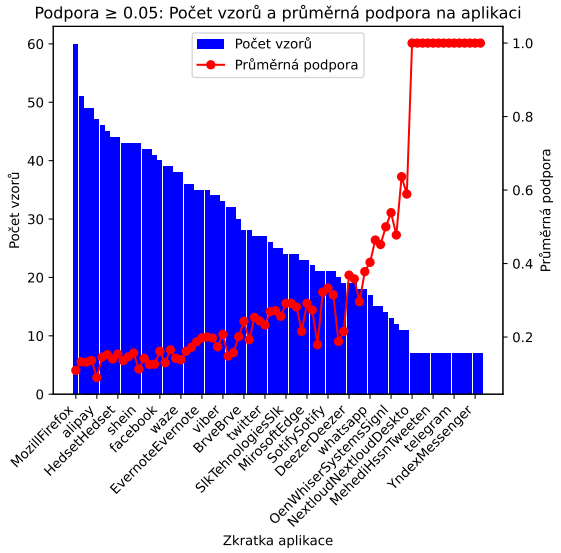
Zde jsou uvedeny heatmapy znázorňující výsledky při použití otisků `JA4` jako základní metody pro generování počáteční kandidátní množiny.



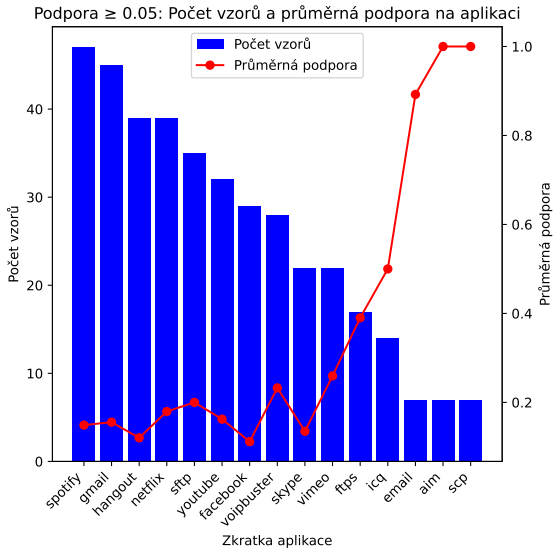
Obrázek D.1: Heatmapa zobrazující testované kombinace položek v závislosti na volbě minimální podpory a dosažené přesnosti pro datovou sadu `iscx.csv` při použití otisků *JA4*.



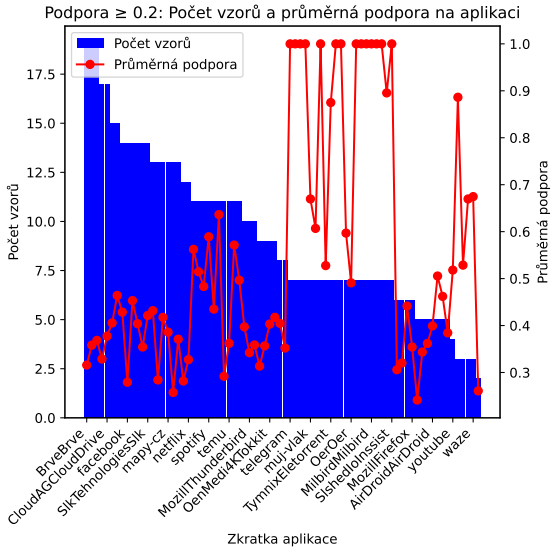
Obrázek D.2: Heatmapa zobrazující testované kombinace položek v závislosti na volbě minimální podpory a dosažené přesnosti pro datovou sadu `mobile desktop apps raw.csv` při použití otisků *JA4*.



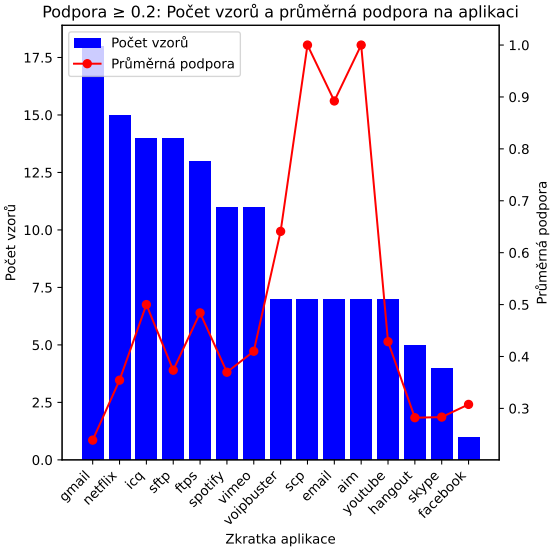
Obrázek D.3: Počet vzorů a průměrná podpora na aplikaci pro `mobile desktop apps raw.csv` při minimální podpoře 0.05.



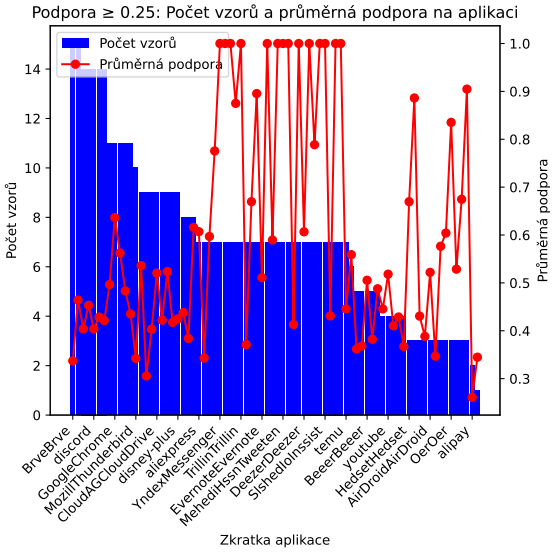
Obrázek D.4: Počet vzorů a průměrná podpora na aplikaci pro `iscx.csv` při minimální podpoře 0.05.



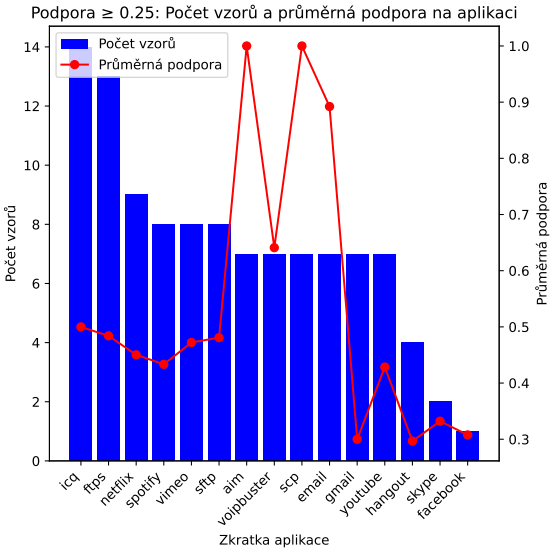
Obrázek D.5: Počet vzorů a průměrná podpora na aplikaci pro `mobile desktop apps raw.csv` při minimální podpoře 0.2.



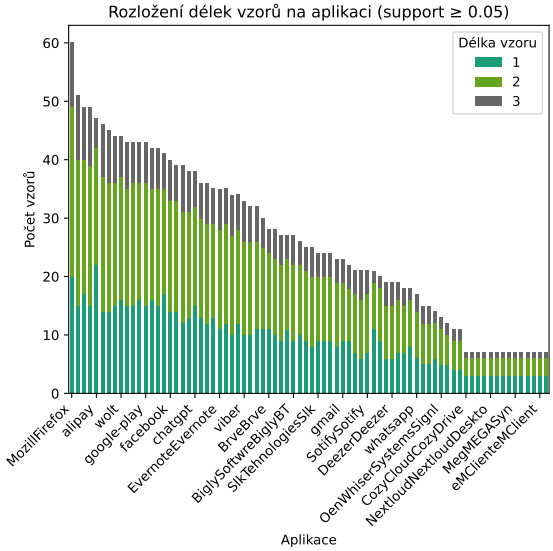
Obrázek D.6: Počet vzorů a průměrná podpora na aplikaci pro `iscx.csv` při minimální podpoře 0.2.



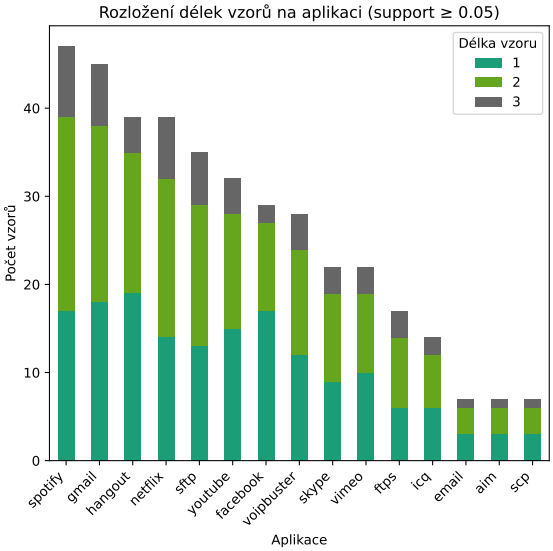
Obrázek D.7: Počet vzorů a průměrná podpora na aplikaci pro `mobile desktop apps raw.csv` při minimální podpoře 0.25.



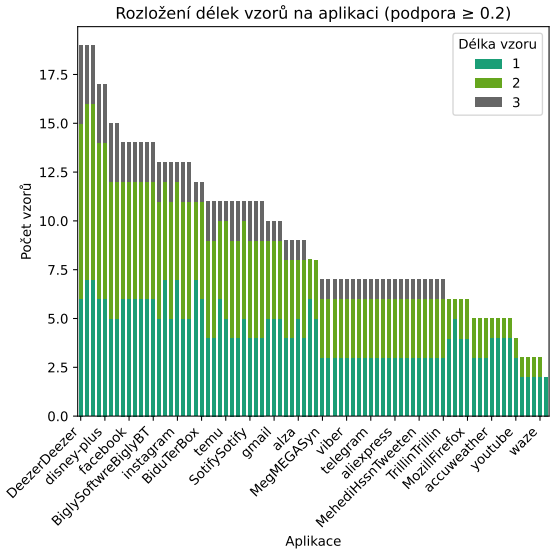
Obrázek D.8: Počet vzorů a průměrná podpora na aplikaci pro `iscx.csv` při minimální podpoře 0.25.



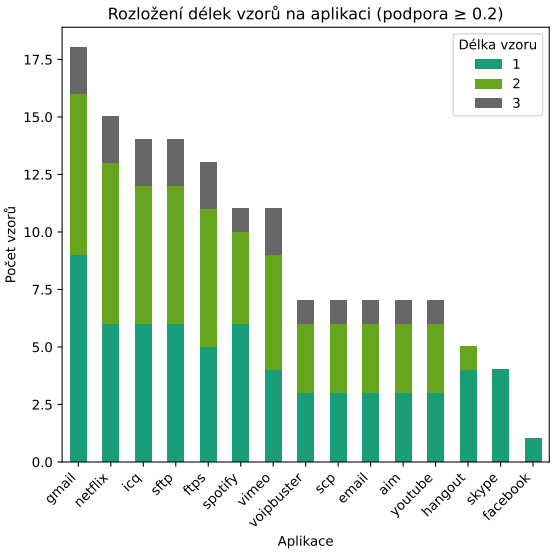
Obrázek D.9: Počet vzorů a rozložení délek vzoru na aplikaci pro `mobile desktop apps raw.csv` při minimální podpoře 0.05.



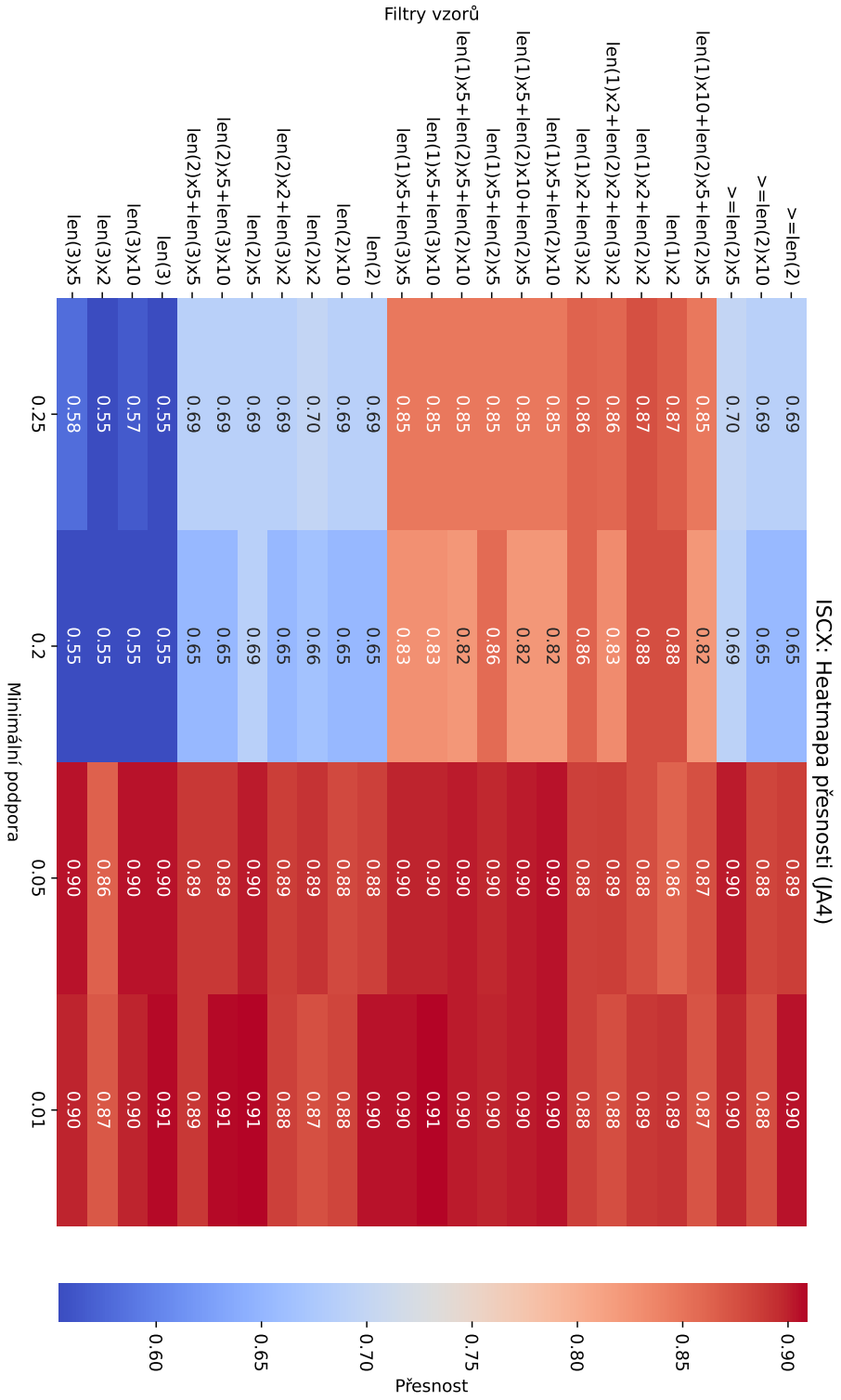
Obrázek D.10: Počet vzorů a rozložení délek vzoru na aplikaci pro `iscx.csv` při minimální podpoře 0.05.



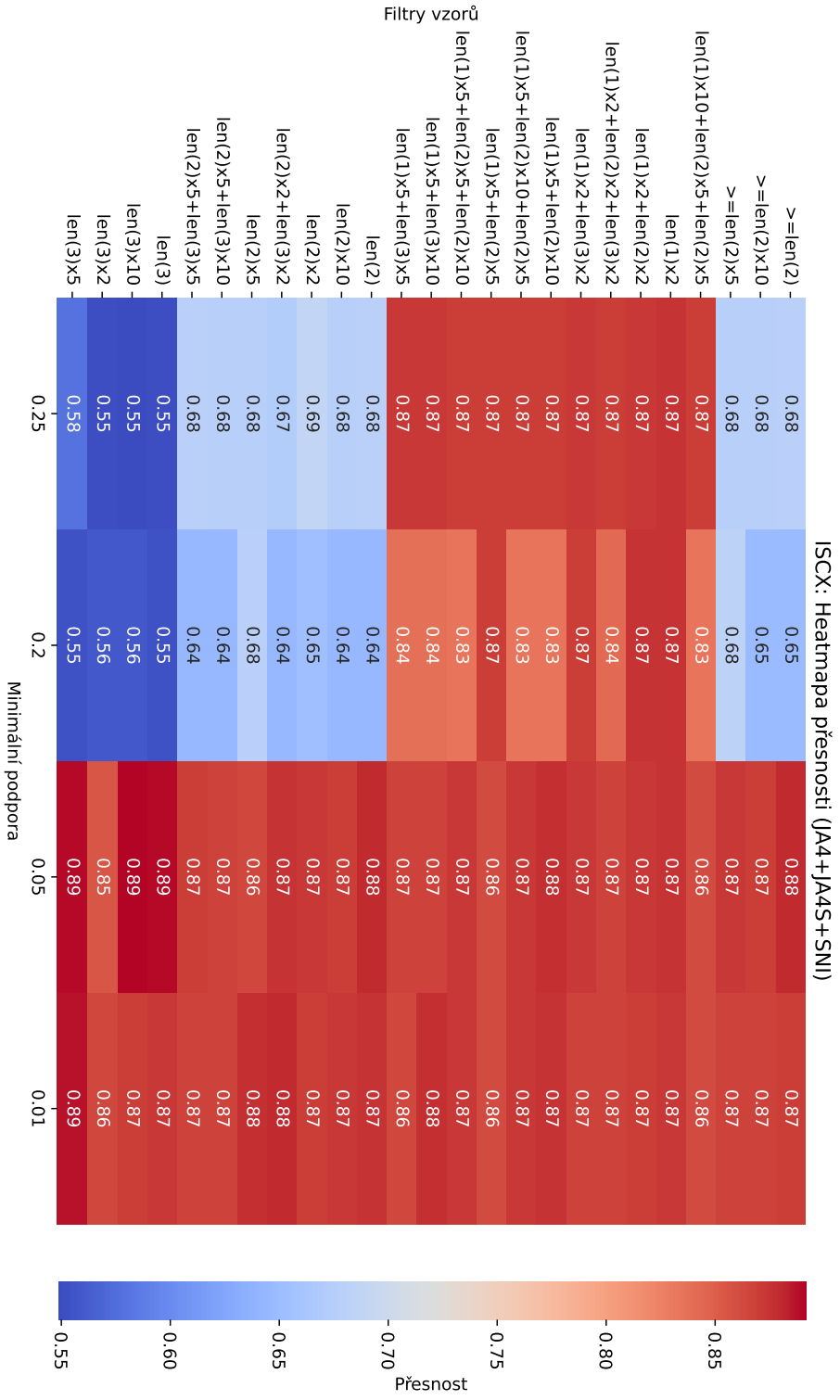
Obrázek D.11: Počet vzorů a rozložení délek vzoru na aplikaci pro `mobile desktop apps raw.csv` při minimální podpoře 0.2.



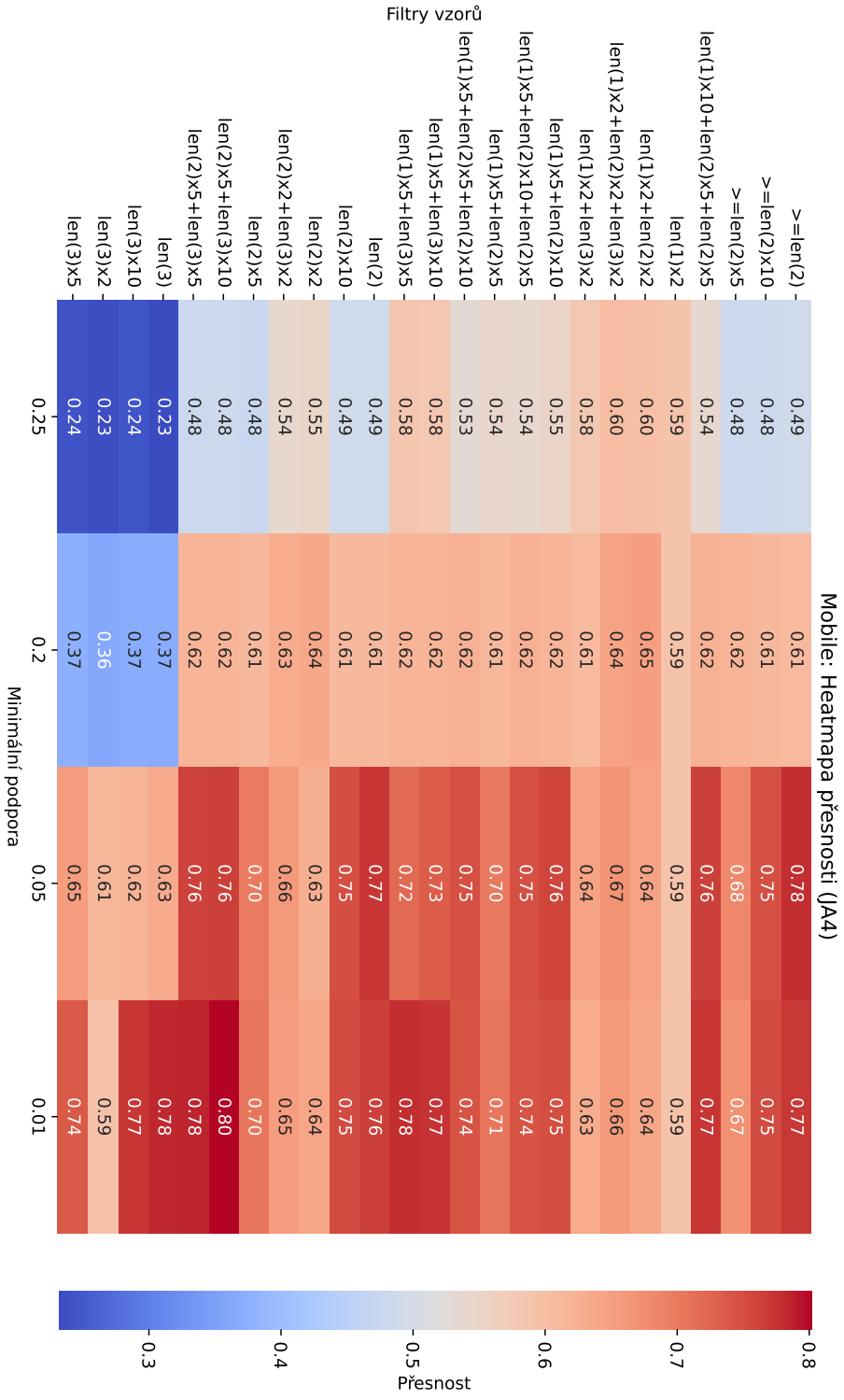
Obrázek D.12: Počet vzorů a rozložení délek vzoru na aplikaci pro `iscx.csv` při minimální podpoře 0.2.



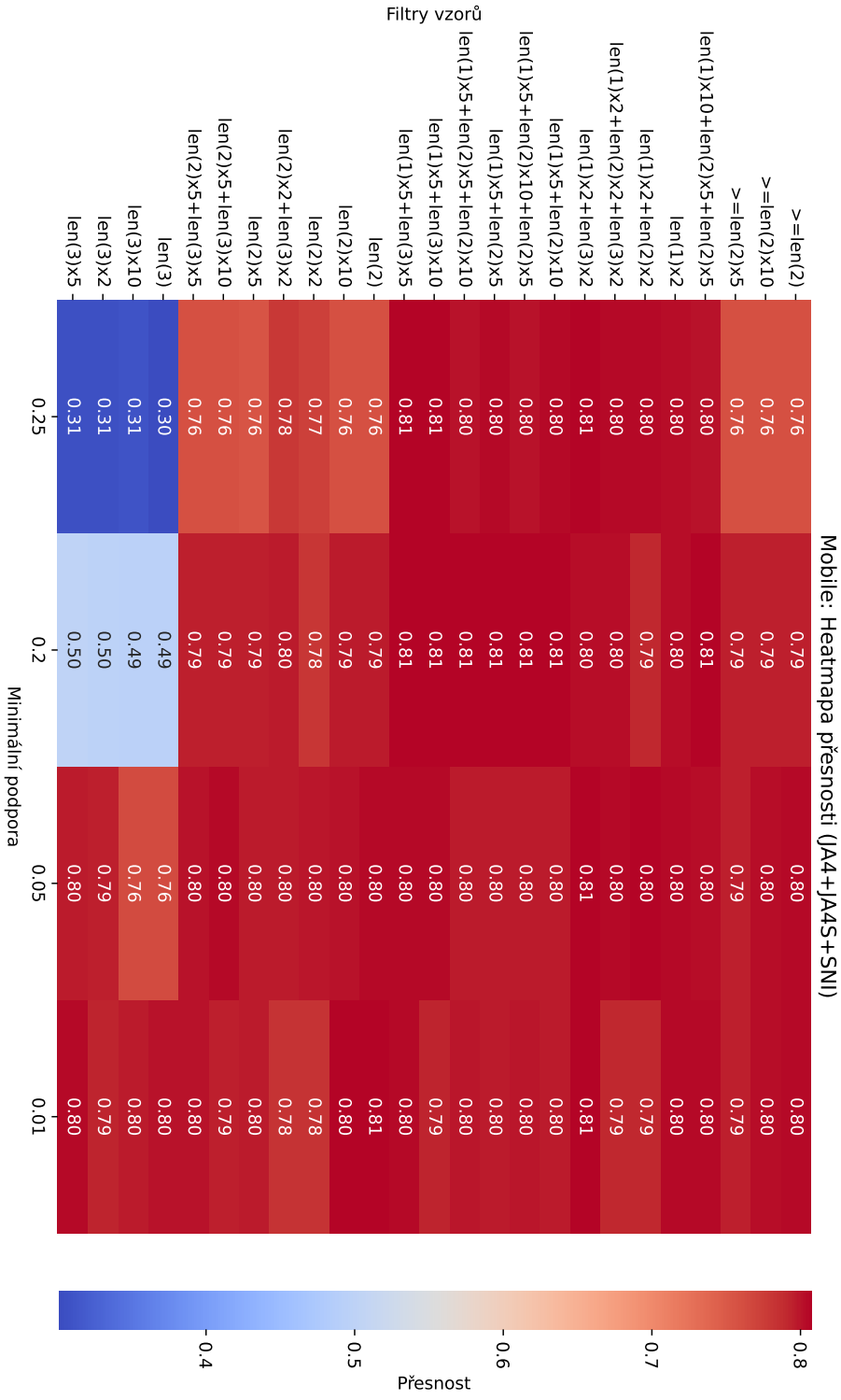
Obrázek D.13: Heatmapa zobrazuje testované kombinace otisků a filtrů při použití různých selekčních strategií nad datovou sadou `iscx.csv`, konkrétně pro kombinaci otisků *JA4*. Barevné škálování reprezentuje dosaženou přesnost identifikace pro každou konfiguraci.



Obrázek D.14: Heatmapa zobrazuje testované kombinace otisků a filtrů při použití různých selekčních strategií nad datovou sadou `iscx.csv`, konkrétně pro kombinaci otisků *JA4+JA4S+SNI*. Barevné škálování reprezentuje dosaženou přesnost identifikace pro každou konfiguraci.



Obrázek D.15: Heatmapa zobrazuje testované kombinace otisků a filtrů při použití různých selekčních strategií nad datovou sadou `mobile_desktop_apps_raw.csv`, konkrétně pro kombinaci otisků *JA4*. Barevné škálování reprezentuje dosaženou přesnost identifikace pro každou konfiguraci.



Obrázek D.16: Heatmapa zobrazuje testované kombinace otisků a filtrů při použití různých selekčních strategií nad datovou sadou `mobile_desktop_apps_raw.csv`, konkrétně pro kombinaci otisků *JA4+JA4S+SNI*. Barevné škálování reprezentuje dosaženou přesnost identifikace pro každou konfiguraci.