VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

IDENTIFIKÁCIA MOBILNÝCH APLIKÁCIÍ S VYUŽITÍM TLS DÁT

MOBILE APPLICATION IDENTIFICATION BASED ON TLS DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RICHARD BORBÉLY**
AUTHOR

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

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Borbély Richard**
Program: Informační technologie
Název: **Identifikace mobilních aplikací s využitím TLS dat**
Mobile Application Identification Based on TLS Data
Kategorie: Data mining

Zadání:

1. Seznamte se s protokoly SSL/TLS.
2. Seznamte se s možnostmi identifikace aplikací prostřednictvím informací zjištěných ze síťové komunikace těchto aplikací.
3. Seznamte se s dostupnými datovými sadami, které obsahují SSL/TLS data z reálné komunikace mobilních aplikací.
4. Navrhněte postup pro extrakci klasifikačních pravidel z těchto datových sad a implementujte jej.
5. Implementujte klasifikátor pro identifikaci mobilních aplikací, který bude využívat vytvořená pravidla.
6. Klasifikátor otestujte na dostupné datové sadě a zhodnoťte dosažené výsledky.

Literatura:

- Matoušek, Petr: Síťové aplikace a jejich architektura, Brno: VUTIUM, 2014, 396 s., ISBN 978-80-214-3766-1
- Han, J., Kamber, M.: Data Mining: Concepts and Techniques, Third Edition,Morgan Kaufmann Publishers, 2012, 703 p., ISBN 978-0-12-381479-1
- Matoušek P., Burgetová I., Ryšavý O., Victor M.: On Reliability of JA3 Hashes for Fingerprinting Mobile Applications, In Proceedings of ICDF2C 2020.
- Razaghpanah A., Niaki A. A., Vallina-Rodriguez N., Sundaresan S., Amann J., Gill P.: Studying TLS Usage in Android Apps, In Proceedings of CoNEXT '17, ACM, New York

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 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. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 26. října 2020

Abstrakt

Táto práca sa venuje identifikácii mobilných aplikácií na základe dát zo sieťového protokolu TLS a to skúmaním hodnôt TLS handshake-u, konkrétne JA3, JA3S a SNI. Práca predstavuje aplikáciu, ktorej súčasťou je algoritmus vykonávajúci klasifikáciu nad dátami TLS protokolu. Výsledky klasifikácie reprezentujú informácie, na základe ktorých je možné rozhodnúť, či identifikácia daných aplikácií bola úspešná. Táto metóda umožnila identifikovať 17 z 18 poskytnutých aplikácií. Prínosom tejto práce je práve možnosť identifikácie aplikácií na základe hodnôt JA3, JA3S a SNI a jej využitie je možné napríklad v sieťovej administratíve.

Abstract

This thesis deals with identification of mobile applications based on data from network protocol TLS. It conducts a research of values from the TLS handshake, specifically of JA3, JA3S and SNI values. The work represents an application that includes an algorithm performing a classification over TLS data. The results of the classification represent information based on which we can decide, if the identification of the apps was successful. This method allowed to identify 17 of the 18 given applications. The benefit of this work is the ability to identify mobile apps based on JA3, JA3S and SNI values and for example, it can be used in network administration.

Klíčová slova

data mining, analýza dát, klasifikácia, klasifikačné pravidlo, data sety, TLS, TCP/IP model, fingerprinting, JA3, JA3S, SNI, aplikácia, algoritmus, mobilná aplikácia, TLS handshake, Python, identifikácia

Keywords

data mining, data analysis, classification, classification rule, data sets, TLS, TCP/IP model, fingerprinting, JA3, JA3S, SNI, application, algorithm, mobile application, TLS handshake, Python, identification

Citace

BORBÉLY, Richard. *Identifikácia mobilných aplikácií s využitím TLS dát.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivana Burgetová, Ph.D.

Identifikácia mobilných aplikácií s využitím TLS dát

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pani Ing. Ivany Burgetovej, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Richard Borbély
6. května 2021

Poděkování

Rád by som sa poďakoval vedúcej mojej práce Ing. Ivane Burgetovej, Ph.D za kvalitné odborné vedenie a cenné rady, ktoré mi v priebehu riešenia práce poskytovala.

Obsah

1 Úvod **3**

2 Sieťová architektúra, TLS, Fingerprinting a Klasifikácia **4**
2.1 Sieťová architektúra 4
2.1.1 Model ISO/OSI 4
2.1.2 Model TCP/IP 6
2.2 Protokol TLS 7
2.2.1 TLS Handshake 7
2.2.2 SNI 9
2.3 Fingerprinting 9
2.4 Klasifikácia 10
2.4.1 Binárna klasifikácia 11
2.4.2 Accuracy, Precision, Recall 12

3 Návrh **13**
3.1 Aplikácie pre identifikáciu 13
3.2 Dátové sady 14
3.3 Trénovacia množina 15
3.4 Testovacia množina 15
3.4.1 Kľúčové slová 15
3.4.2 Prahová hodnota podobnosti 15
3.5 Krížová validácia 16
3.6 Hierarchické vyhodnotenie 16
3.7 Kolízia pravdy 17
3.8 Vstupné hodnoty 17
3.9 Výstupné hodnoty 18

4 Implementácia klasifikátora **19**
4.1 Kľúčové slová 19
4.2 Výpočet a implementácia hodnoty podobnosti 20
4.2.1 Funkcia get_matching_blocks() 20
4.2.2 Funkcia ratio() 23
4.2.3 Porovnanie kľúčového slova s SNI 23
4.2.4 Integrácia hodnoty podobnosti 24
4.3 Užívateľské rozhranie 24
4.3.1 Vstupné okno 25
4.3.2 Výstupné okná 26
4.4 Spracovanie vstupných dát 26

4.4.1 Funkcie identify() . . . 26
4.4.2 Funkcie createTestingDict() a createTrainingDict() . . . 28
4.4.3 Funkcia adjustTrainingDict() . . . 29
4.5 Vyhodnotenie vytvoreného klasifikátoru . . . 30
4.5.1 Funkcia evaluate() . . . 30
4.5.2 Funkcia evaluateHierarchical() . . . 31
4.6 Zápis a reprezentácia vyhodnotení . . . 33
4.6.1 Funkcia fillListPositive() . . . 33
4.6.2 Funkcia fillListNegative() . . . 34
4.6.3 Využitie zoznamu trueCollision . . . 34
4.7 Znázornenie výstupu . . . 34
4.8 Znázornenie vstupu . . . 35
4.9 Zápis výsledkov do súboru . . . 35

5 Experimenty **36**
5.1 Porovnanie vyhodnotení s individuálnym využitím atribútov TLS . . . 37
5.1.1 Identifikácia na základe hodnoty JA3 . . . 37
5.1.2 Identifikácia na základe dvojice JA3 a JA3S . . . 39
5.1.3 Identifikácia na základe trojice JA3, JA3S a SNI . . . 40
5.2 Hierarchické vyhodnotenie . . . 41
5.3 Krížová validácia . . . 43
5.4 Prahová hodnota podobnosti . . . 44
5.5 Problémové aplikácie . . . 48
5.5.1 Chrome . . . 48
5.5.2 Facebook . . . 48
5.5.3 GoogleCalendarSync . . . 49
5.5.4 Telegram . . . 49
5.5.5 TikTok . . . 49
5.5.6 YouTube . . . 49
5.6 Použitie prahovej hodnoty podobnosti pri trénovaní . . . 49
5.7 Vyhodnotenie experimentov . . . 52

6 Záver **54**

Literatura **55**

A Obsah priloženého pamäťového média **56**

Kapitola 1

Úvod

Informačné technológie sa stávajú rok čo rok väčšou súčasťou nášho každodenného života. Je obdivuhodné, akým spôsobom si dnes človek vďaka internetu dokáže vyhľadať potrebné informácie a priam šokujúce, ak si dnešnú situáciu porovnáme so situáciou spred pár desaťročiami. V dnešnej dobe existuje mnoho druhov zariadení, ktoré nám umožňujú prístup na internet. Medzi tie najrozšírenejšie a najviac praktické, tie, ktoré vlastní dnes už hádam každý človek v čo i len minimálnej symbióze s technológiami, patria novodobé mobilné telefóny (smartfóny), poprípade tablety.

Vieme, že tieto mobilné zariadenia fungujú na báze operačného systému a ich jednotlivé funkcionality sú pre každého užívateľa individuálne doplniteľné za pomocou jednotlivých aplikácií stiahnuteľných z internetového obchodu konkrétnej platformy. Väčšina z týchto aplikácií aj po ich nainštalovaní do zariadenia komunikujú cez internet, často aj keď si to užívatelia vôbec neuvedomujú. Túto komunikáciu je potreba zabezpečiť, veď hádam nikto, kto si svoje súkromie nejakým spôsobom váži, by si neprial, aby ho bolo možné jednoducho odsledovať útočníkmi na internete, poprípade získať a následne zneužiť jeho dáta. K tomuto zabezpečeniu vypomáha protokol TLS, ktorý je šifrovacím protokolom v sieťovej architektúre. Jeho spôsob fungovania je popísaný v sekcii 2.2.

Práca je experimentálneho charakteru. Jej cieľom je zistiť, do akej miery je možné odchytené dáta z protokolu TLS využiť na to, aby sme jednotlivé mobilné aplikácie dokázali jednoznačne identifikovať. Práca zahŕňa teoretický úvod k sieťovej architektúre, TLS protokolu a to čo sa týka spôsobu jeho fungovania, tak aj konkrétnych získaných dát, ktoré sú dôležitou súčasťou tejto práce a ktoré sa pre experimentovanie využívajú. S týmito informáciami súvisia poskytnuté dátové sady, nad ktorými sa v práci vykonávajú rôzne operácie. Dátové sady si bližšie predstavíme v sekcii 3.2.

Výsledkom práce je skript v skriptovacom jazyku Python s rozšírením o grafické rozhranie pre jednoduchšie a intuitívnejšie zadávanie vstupných parametrov, ako aj na pohodlnejšie zobrazenie výstupných dát vo forme maticových tabuliek a výsledkami experimentov. Skript pracuje so súbormi viacerých typov ako .csv alebo .json. Kapitola 4 popisuje podrobnejšie implementačné detaily a detailnejší opis spôsobu pracovania s poskytnutými dátami.

Výsledky experimentov práce je možné využiť pri monitorovaní sieťového prenosu dát práve k detekcii toho, aké aplikácie užívateľom na ich mobilných zariadeniach bežia.

Táto práca naväzuje a je rozšírením práce rozobratej vo vedeckom článku [9] , ktorý sa venuje identifikácii mobilných aplikácií prostredníctvom dát získaných práve z TLS protokolu.

Kapitola 2

Sieťová architektúra, TLS, Fingerprinting a Klasifikácia

K porozumeniu fungovaniu TLS protokolu je vhodné sa vopred zoznámiť so sieťovou architektúrou, pod ktorú TLS, rovnako ako ostatné sieťové protokoly, spadá. Táto kapitola sa z tohto dôvodu venuje práve tejto problematike a poskytuje k nej teoretický úvod. Takisto obsahuje predstavenie problematík ako fingerprintig a klasifikácia.

2.1 Sieťová architektúra

Architektúra počítačovej siete reprezentuje veľmi podrobnú a presne definovanú infraštruktúru. Každá jej časť má svoje opodstatnenie a spĺňa určitú dôležitú úlohu v sieťovom systéme. Pre túto sekciu som čerpal informácie z knihy Síťové aplikace a jejich architektura [7]. Už od počiatku svojej existencie bola sieťová architektúra navrhovaná takým spôsobom, aby oddeľovala tri základné kategórie komunikačného systému. Nimi sú:

1. Fyzické technológie pre prenos signálu (metalická kabeláž, optické vlákna, Wi-fi)
2. Technológie vo forme protokolov pre spoľahlivý, resp. bezpečný prenos dát (TCP/IP, AppleTalk, IPX/SPX)
3. Technológie pre interakciu s aplikáciou, čiže nepriamo s užívateľom

Na základe týchto uvedených kategórií sa počas histórie sietí navrhovali isté sieťové modely, ktoré rozdeľujú sieťovú architektúru na viacero častí, presnejšie povedané úrovní. Niektoré na viac, niektoré na menej. Do súčasnej doby sa zachovali a aktívne používajú dva. Sú nimi model ISO/OSI a model TCP/IP.

2.1.1 Model ISO/OSI

Model ISO/OSI je model sieťovej architektúry, ktorý spľňa úlohu referenčného modelu. To znamená, že tento model sa používa ako názorný príklad riešenia komunikácie v počítačových sieťach. Využíva sa na porozumenie konceptu architektúry a funkcionality jednotlivých vrstiev. Model vypracovala organizáciia ISO (International Organization for Standardization) a v roku 1984 bol prijatý ako medzinárodná norma ISO 7498 [6]. Táto norma namiesto toho, aby špecifikovala implementáciu systémov, uvádza všeobecné princípy. Popisuje jednotlivé sieťové vrstvy a ich funkcie. Model sa skladá zo siedmich vrstiev, jeho štruktúru znázorňuje tabuľka 2.1:

Aplikačná vrstva
Prezentačná vrstva
Relačná vrstva
Transportná vrstva
Sieťová vrstva
Linková vrstva
Fyzická vrstva

Tabulka 2.1: Ukážka modelu OSI

Každá z uvedených vrstiev má vlastnosť, že pre svoj chod používa funkcie vrstvy, ktoré sa nachádzajú pod ňou a to bez toho, aby vedela, akým spôsobom sú služby pod ňou implementované a ako pracujú. Následne svoju funkcionalitu poskytuje nadradenej vrstve.

Vrstvy referenčného modelu ISO/OSI

- **Fyzická vrstva (Physical layer)** - táto vrstva definuje elektrické, mechanické a funkčné špecifikácie pre aktiváciu, priebeh a ukončenie fyzického spojenia medzi dvoma systémami. Zaoberá sa paramertrami ako časovanie, úroveň napätia, maximálna vzdialenosť medzi komunikujúcimi zariadeniami, počet a umiestnenie pinov na konektoroch a mnoho ďalšími.
- **Linková vrstva (Data Link layer)** - má na starosti poskytovanie spoľahlivého zasielania dát po jednotlivých médiách. Spadá sem fyzické adresovanie (pomocou MAC adries), fyzická sieťová topológia a regulácia zasielania dát.
- **Sieťová vrstva (Network layer)** - zabezpečuje spojenie a voľbu najlepšiej možnej cesty prenášaných dát medzi dvomi zariadeniami v sieti. Spadá pod ňu logická topológia. Jednotlivé zariadenia sa adresujú IP adresami.
- **Transportná vrstva (Transport layer)** - poskytuje spoľahlivý prenos dát, vykonáva detekciu chýb v sieti, možné opätovné posielanie dát v prípade straty.
- **Relačná vrstva (Session layer)** - slúži k udržiavaniu relácií medzi komunikujúcimi aplikáciami - nadväzuje, riadi a ukončuje relácie. Spravuje prenos dát medzi dvoma systémami a synchronizuje ich komunikáciu.
- **Prezentačná vrstva (Presentation layer)** - zaisťuje zobrazenie (prezentáciu) dát medzi rôznymi aplikáciami a architektúrami. Zahrňuje odlišné formáty dát, ich kompresiu a kódovanie.
- **Aplikačná vrstva (Application layer)** - definuje užívateľské procesy a aplikácie komunikujúce po sieti. Ide o vrstvu v najbližšom kontakte k užívateľovi, poskytuje

služby užívateľským aplikáciám. Medzi tieto služby patrí napríklad elektronická pošta, adresárové služby alebo systém DNS.

Model poskytuje dva druhy komunikácií, a to:

1. Komunikáciu medzi vrstvami systému
2. Komunikáciu medzi rovnakými vrstvami rôznych systémov (využívajú sa protokoly jednotlivých vrstiev)

Dátové jednotky jednotlivých modelov sa nazývajú PDU (Protocol Data Unit). Každá vrstva modelu obsahuje iný druh dát a týmpádom má aj iné značenie. PDU pre transportnú vrstvu sa tak nazýva Segment (Packet) a pre sieťovú napríklad IP datagram. Podrobné rozdelenie PDU pre jednotlivé vrstvy charakterizuje tabuľka 2.2:

Data	Aplikačná vrstva Prezentačná vrstva Relačná vrstva
Segment (Paket)	Transportná vrstva
IP datagram	Sieťová vrstva
Frame	Linková vrstva
Bit	Fyzická vrstva

Tabulka 2.2: Ukážka PDU - OSI model

2.1.2 Model TCP/IP

Architektúra modelu TCP/IP je výrazne jednoduchšia v porovnaní s referenčným modelom OSI. Model TCP/IP spojuje fyzickú a linkovú vrstvu do spoločnej vrstvy, nazývanou vrstvou fyzického rozhrania, ktorá je obvykle implementovaná na sieťovej karte zariadenia.

Vlastná implementácia architektúry TCP/IP je rozdelená do troch častí. Najnižšia vrstva, vrstva fyzického rozhrania, je implementovaná na sieťovej karte (NIC - Network Interface Card) a na ovládači karty (driver). Vyššie vrstvy - internetová a transportná - sú súčasťou sieťových modulov operačných systémov. Aplikačná vrstva pozostáva z aplikačných protokolov, ktoré sú implementované samostatne.

Vrstvy modelu TCP/IP

- **Vrstva fyzického rozhrania (Network interface layer)** - Táto vrstva popisuje štandardy pre fyzické médium a elektrické signály, napr. Ethernet, Token Ring alebo Frame Relay. Okrem fyzických technológií zahŕňa taktiež funkcie pre prístup k fyzickému médiu. Vrstva takisto zaisťuje zapúzdrenie IP datagramov do rámcov.

- **Internetová vrstva (IP layer)** - Vytvára datagramy, adresuje ich a smeruje na miesto určenia. Zaisťuje takzvané doručenie s najväčším úsilím (best-effort delivery), čo znamená, že IP vrstva sa snaží doručiť poslané dáta čo najvhodnejšou cestou. Ak na ceste príde k stráte dát, vysielajúci uzol je o strate datagramu informovaný. Patria sem protokoly ako ARP a RARP, ktoré slúžia na mapovanie IP adries na hardvérové MAC adresy (a obrátene) ako aj protokoly ICMP a IGMP.
- **Transportná vrstva (Transport layer)** - Prenáša dáta z aplikácie na zdrojovom zariadení do aplikácie na cieľovom zariadení. Vytvára takzvané logické spojenie medzi procesmi. Transportné protokoly rozdeľujú aplikačné dáta na menšie jednotky (pakety), ktoré posielajú po sieti. Patria sem protokoly TCP a UDP.
- **Aplikačná vrstva (Application layer)** - Je tvorená procesmi a aplikáciami, ktoré komunikujú po sieti. Vrstva zaisťuje spracovanie dát na najvyššej úrovni vrátane reprezentácie dát, kódovania a riadenia dialógu. Patria sem protokoly ako DNS, SMTP, Telnet, FTP, OSPF alebo SNMP.

Aplikačná vrstva
Transportná vrstva
Internetová vrstva
Vrstva fyzického rozhrania

Tabulka 2.3: Ukážka modelu TCP/IP

2.2 Protokol TLS

TLS (Transport Layer Security) je sieťový kryptografický protokol slúžiaci na zabezpečenie internetovej komunikácie pre služby ako elektronická pošta, WWW alebo IM (Instant Messaging). Je implementovaný ako medzivrstva medzi transportnou a aplikačnou vrstvou. Mnoho aplikácií túto vrstvu využíva pre bezpečný prenos dát. Protokol zaisťuje bezpečnú komunikáciu medzi klientom a serverom. Pomocou kryptografie poskytuje svojim koncovým bodom autentizáciu, dôvernosť a integritu dát. Komunikuje nad transportnou vrstvou prostredníctvom TCP paketov a zabezpečuje prenos dát protokolov aplikačnej vrstvy. Pred zahájením prenosu dát sa obidve strany potrebujú dohodnúť na verzii protokolu, používanom šifrovacom a autentizačnom algoritme a vymeniť si relačné kľúče, ktoré použijú pre šifrovanie.

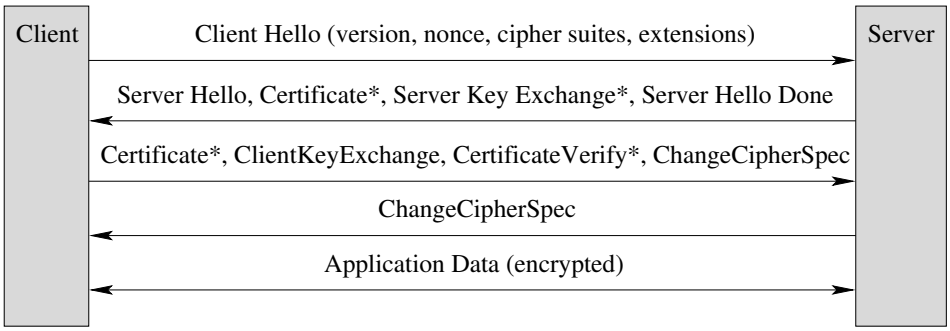
2.2.1 TLS Handshake

Komunikácia prebieha pomocou takzvaného TLS handshake, ktorý pozostáva z dvoch základných častí:

1. **TLS Handshake Protocol** - Slúži na vyjednávanie bezpečnostných parametrov - verziu protokolu, metódy pre výmenu kryptografických kľúčov, enkryptovanie, autentifikáciu a integritu dát. Táto komunikácia ešte nie je enkryptovaná.

2. **TLS Record Protocol** - Má na starosti enkryptovanie (zapúzdrenie) vysokoúrovňových protokolových dát a vysiela enkryptované pakety

Obrázok č. 2.1 [8] znázorňuje jednotlivé kroky TLS handshake-u:



Obrázek 2.1: Ukážka TLS handshake (informácie označené * môžu, ale nemusia byť v procese zahrnuté)

Jednotlivé kroky TLS handshake [12]:

1. **Client Hello** - klient ponúkne serveru set autentikačných metód, verziu protokolu, ktorú podporuje a algoritmy.
2. **Server Hello** - spracuje informácie a pošle odozvu s možnosťami, ktoré sú na jeho strane podporované a jeho výber parametrov z poskytnutého zoznamu klienta pre komunikáciu.
3. **Certificate** - server pošle svoj certifikát pre jeho overenie. Klient ho overí tým, že skontroluje jeho digitálny podpis a overí potenciálne problémy ako napríklad dátum jeho expirácie alebo nesprávne doménové meno.
4. **Server Key Exchange** - správa potrebná v prípade špecifických metód pre výmenu kľúčov, ktoré vyžadujú dodatočné informácie od serveru (napríklad algoritmus Diffie-Hellman).
5. **Server Hello Done** - správa, ktorá informuje klienta o tom, že server odoslal všetky dáta.
6. **Client Key Exchange** - klient odošle svoju časť informácií čo sa týka vytvoreniu kľúča.
7. **Change Cipher Spec** - klient dá vedieť serveru, že vygeneroval kľúč a plánuje prejsť na enkryptovanú komunikáciu.
8. **Finished (Client)** - informácia od klienta k serveru o tom, že handshake je dokončený na klientskej časti. Správa Finished je už enkryptovaná a sú to prvé dáta zabezpečené pomocou vytvoreného kľúča.
9. **Change Cipher Spec** - server dekryptuje správu a vypočíta kľúč. Následne pošle správu Change Cipher Spec, aby oznámil klientovi, že takisto prechádza na enkryptovanú komunikáciu.

10. **Finished (Server)** - server pošle svoju správu Finished zabezpečenú svojím vypočítaným kľúčom.

2.2.2 SNI

SNI (Server Name Indication) je rozšírenie protokolu TLS. TLS handshake, ako bolo spomenuté, začína správou Client Hello. Tá obsahuje potrebné údaje pre získanie certifikátu a ostatných informácií k serveru, ako je napríklad IP adresa. Servery z dôvodu šetrenia IP adresami môžu spravovať viacero webových domén pod jednou IP adresou, čiže klient potrebuje špecifikovať aj doménové meno, ku ktorému žiada prístup. Jednoduchá komunikácia HTTP prebieha tak, že klient pošle Client Hello message, server ho obdrží, pošie odpoveď, klient následne odošle HTTP GET request, ktorý už doménové meno obsahuje a týmpádom serveru umožní identifikovať špecifickú webovú stránku. Problém nastáva s príchodom enkryptovanej komunikácie. Po správach Client Hello a Server Hello sa celková komunikácia enkryptuje, vrátane HTTP GET správy obsahujúcou špecifické doménové meno. Preto vzniklo SNI, čo je jednoduchý reťazec charakterizujúci doménové meno webovej stránky, ku ktorej klient žiada prístup. Je pridané do nešifrovanej správy Client Hello.

2.3 Fingerprinting

Odtlačky (fingerprinty) sú, podobne ako v prípade ľudských odtlačkov prstov, jednoznačné identifikátory rôznych druhov informácií. Slúžia na identifikáciu veľkých dát náročných na spracovanie. Na vytvorenie takéhoto odtlačku pre isté dáta sa využíva hashovací algoritmus, čo je procedúra, ktorá premapuje dáta o vysokej veľkosti na oveľa kratší reťazec, takzvaný hash, ktorý je následne možné využiť na jednoznačnú identifikáciu daných dát. Väčšinou sa odtlačky využívajú na vyhnutie sa prenosu a porovnávaniu zbytočne veľkých dát. Fingerprinting využíva hashovacie algoritmy ako MD5 alebo SHA.

V tejto práci sa bude využívať fingerprinting metóda práve na dáta z TLS handshake protokolu, z ktorých vypočítané odtlačky budú použité na pokusy o identifikáciu jednotlivých mobilných aplikácií.

Ako som už spomínal v podkapitole 2.2.1, po istej fáze v TLS handshake sa dáta začnú enkryptovať a nie je možné ich žiadnym spôsobom využiť. Preto je nutné čerpať dáta z úvodných správ, ktoré ešte šifrované nie sú, a to sú Client Hello a Server Hello správy. Tieto správy obsahujú viacero charakteristických informácií pre konkrétny handshake. Na vypočítanie hashov z týchto správ existuje istá metodológia, nazývaná JA3 hash [3], respektíve JA3S hash [9].

1. **JA3 hash** - Hash vygenerovaný z Client Hello správy
2. **JA3S hash** - Hash vygenerovaný zo Server Hello správy

Na vypočítanie JA3 a JA3S hash sa z Client Hello a Server Hello správ používajú nassledovné informácie:

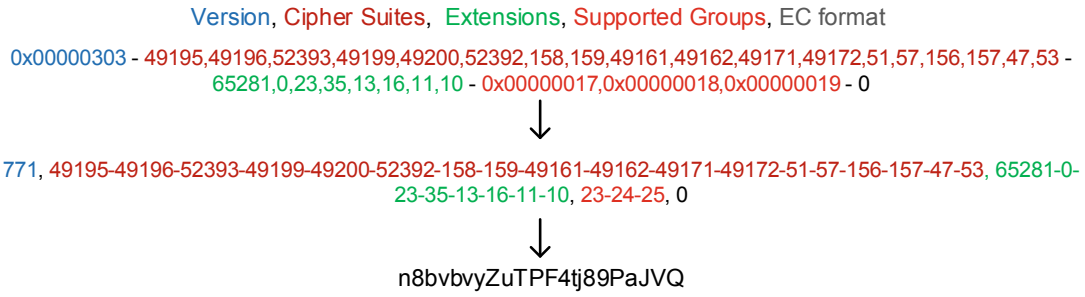
- **Version** - verzia protokolu.
- **Cipher Suites** - obsahujú zoznam podporovaných kryptografických algoritmov.
- **Extensions** - rozšírenia protokolu.

- **Supported Groups** - alebo eliptické krivky [1]. Jedná sa o kryptografický mechanizmus na zabezpečenie dát. Používa sa len pri generovaní JA3 hash.
- **EC format** - formát eliptickej krivky. Používa sa len pri generovaní JA3 hash.

Proces vytvárania TLS odtlačkov je rýchly, pretože zahŕňa prácu iba s hlavičkou TLS paketu. Spôsob získania TLS fingerprintu je nasledovný:

1. Extrakcia vybraných častí informácií z TLS Hello paketu.
2. Konkatenácia extrahovaných dát v decimálnej podobe do jedného reťazca.
3. Aplikovanie MD5 kryptografického algoritmu na získaný reťazec.

Výsledkom je reťazec v hexadecimálnom formáte o veľkosti 32 bitov. Znázornenie výpočtu JA3 hash je možné vidieť na obrázku č. 2.2 [9].



Obrázek 2.2: Ukážka výpočtu JA3 hash

2.4 Klasifikácia

Klasifikácia [5] je problematika, ktorej cieľom je zaradiť novú vzorku dát do jednej alebo viacerých tried na základe množiny dát, ktorej zaradenie do jednotlivých tried poznáme. Algoritmus, ktorý má na starosti vyhodnotenie a implementuje klasifikáciu, sa nazýva **klasifikátor**.

Existuje viac druhov klasifikácií, nimi sú [2]:

1. **Binárna klasifikácia** - Existujú dve triedy, do ktorých je možné dáta klasifikovať. Často sa triedy rozdeľujú ako pozitívna a negatívna.
2. **Diskrétna klasifikácia** - Prípad, ak je počet tried, do ktorých je možné dáta zaradiť, väčší ako dva.
3. **Fasetová klasifikácia** - Klasifikácia, ktorá umožňuje jednotlivé dáta zaradiť do viacerých tried súčasne.
4. **Jednotriedna klasifikácia** - Existuje len jedna trieda a klasifikátor detekuje takzvané anomálie, čiže dáta, ktoré do danej triedy nepatria.
5. **Fuzzy klasifikácia** - Pracuje s pravdepodobnosťou. Určuje pravdepodobnosť príslušnosti dát k jednotlivým triedam.

2.4.1 Binárna klasifikácia

Táto práca sa venuje klasifikácii do viacerých tried, avšak na základe spôsobu zaraďovania dát do jednotlivých tried v tejto práci je možné klasifikáciu zjednodušiť na binárnu klasifikáciu, čiže zaraďovaniu dát do jednej z dvoch rozdielnych tried. Tieto triedy nazveme pozitívnou a negatívnou triedou. Ak máme istú množinu dát zaradenú do pozitívnej triedy, klasifikátor sa potrebuje naučiť charakteristické atribúty, ktoré túto pozitívnu triedu identifikujú a na základe týchto charakteristík určiť, ktoré nasledovné dáta by mali patriť do rovnako pozitívnej triedy, a rozhodnúť o tom, že tam reálne patria. To isté platí aj pre negatívnu triedu.

Pri klasifikácii je obvykle cieľom zistiť, ako dobre sa klasifikátor naučil charakteristiky jednotlivých tried. Na toto zistenie slúži množina testovacích dát, nad ktorými prebieha testovanie. Výsledky získané testovaním sa následne porovnajú s pôvodnými dátami, takzvanými trénovacími.

V prípade binárnej klasifikácie pri rozhodovaní o tom, do ktorej z možných tried daná skupina dát patrí, môžme logicky dospieť k dvom rozdielnym výsledkom. Buď dáta zaradíme do pozitívnej, alebo negatívnej triedy. Tým, že pre prvky v testovacej množine vieme, do ktorej z tried patria, dokážeme dospieť k štyrom rôznym typom výsledku, a to:

1. **True Positive (TP)** - Dáta, ktoré majú patriť do pozitívnej triedy, sa identifikujú správne
2. **True Negative (TN)** - Dáta, ktoré majú patriť do negatívnej triedy, sa identifikujú správne
3. **False Positive (FP)** - Dáta, ktoré majú patriť do negatívnej triedy, sa identifikujú ako pozitívne
4. **False Negative (FN)** - Dáta, ktoré majú patriť do pozitívnej triedy, sa identifikujú ako negatívne

Pomocou týchto údajov je možné zhotoviť jednoduchú maticovú tabuľku, tzv. maticu zámien (confusion matrix), ktorá zaraďuje jednotlivé dáta podľa predikovanej a reálnej hodnoty, viď tabuľka č. 2.4. Tabuľka znázorňuje 1000 možných hodnôt, z ktorých dve patria do pozitívnej a zvyšok do negatívnej triedy. Klasifikátor v tomto prípade 999 hodnôt klasifikoval správne, avšak jednu z pozitívnych hodnôt určil ako negatívnu, týmpádom táto hodnota môže byť opísaná ako False Negative.

	Predicted Negative	Predicted Positive
Actual Negative	998	0
Actual Positive	1	1

Tabulka 2.4: Matica zámien

2.4.2 Accuracy, Precision, Recall

Na základe hodnôt získaných pomocou binárnej klasifikácie je možné dospieť k zaujímavým štatistikám. Ja sa v tejto práci budem zaoberať tromi najviac používanými metrikami, ktoré pomáhajú k vyhodnoteniu kvality a presnosti klasifikátora. Nimi sú nasledovné [13]:

1. **Accuracy** - Reprezentuje pomer správne a nesprávne klasifikovaných hodnôt. V podobe vzorca sa reprezentuje nasledovne:

$$\frac{TN + TP}{TN + TP + FN + FP}$$

   Pre dáta z tabuľky 2.4 Accuracy činí (998 + 1) / 1000 = 0.999, čiže 99,9%, keďže jeden z 1000 prvkov sa nepodarilo správne identifikovať.

2. **Precision** - Táto metrika reprezentuje počet pozitívne označených prvkov, ktoré sú aj reálne pozitívne. Hodnota sa získa nasledovným vzorcom:

$$\frac{TP}{TP + FP}$$

   Precision je v tomto prípade je 1 / (1 + 0) = 1, čiže 100%. Nedospeli sme k žiadnemu False Positive prvku.

3. **Recall** - Reprezentuje počet reálne pozitívnych vzoriek, ktorých sa podarilo správne identifikovať spomedzi všetkých pozitívnych vzoriek. Vzorec je nasledovný:

$$\frac{TP}{TP + FN}$$

   Recall je pre tento prípad 1 / (1 + 1) = 0.5, čiže 50%. Z dvoch reálne pozitívnych prvkov sa jeden podarilo identifikovať správne.

Kapitola 3

Návrh

V tejto kapitole sa budem venovať návrhu aplikácie, konkrétne jej vstupným a výstupným hodnotám. Kapitola čitateľa prevedie jednotlivými vstupnými a výstupnými hodnotami a objasní ich význam v identifikácii a experimentovaní. Takisto sa kapitola venuje obsahu jednotlivých dátových sád ako aj obsahu jednotlivých TLS handshake-ov, ktoré dátové sady obsahujú a ktoré sa využijú pre klasifikáciu. Kapitola vysvetľuje kľúčové časti klasifikátora ako sú trénovacia a testovacia množina alebo prahová hodnota podobnosti.

Naimplementovaná aplikácia bude obsahovať:

- 5 dátových sád, na základe ktorých bude vytvorený klasifikačný model
- vyhodnotenie vytvoreného modelu na základe vybraných testovacích sád
- možnosť určenia použitia dátových sád (ako trénovacie, testovacie alebo nevyužívané)
- možnosť voľby používaných TLS dát pre klasifikáciu
- možnosť krížovej validácie 3.5
- možnosť hierarchického vyhodnotenia 3.6
- vstup vo forme externého súboru obsahujúci kľúčové slová 3.4.1
- možnosť voľby prahovej hodnoty podobnosti, na základe ktorej klasifikácia prebehne 3.4.2
- výstup vo forme reprezentácie maticami v rozhraní aplikácie
- výstup vo forme exportovaného textového súboru

3.1 Aplikácie pre identifikáciu

Na identifikáciu máme dostupných 18 rozdielnych mobilných aplikácií. Každá z nich komunikuje individuálne a využíva služby TLS protokolu pre zabezpečenie komunikácie. Týmito aplikáciami sú:

- Boomplay
- Google Chrome

- CP (Cestovný poriadok)
- EquaBank
- Facebook
- Gmail
- Google Calendar
- KB Klíč
- Messenger
- Mobilní Banka
- Můj vlak
- Nextbike
- Reddit
- Seznam CZ
- Telegram
- TikTok
- WhatsApp
- YouTube

Nad každou z týchto aplikácií bolo prevedených viacero operácií, popri ktorých sa podarilo odchytiť viacero TLS handshake-ov a týmpádom aj neenkryptované informácie z nich. Jednotlivé informácie sú pre každú aplikáciu rozdelené do piatich dátových sád, čo rozširuje možnosti pre experimentovanie.

3.2 Dátové sady

Každá z piatich dátových sád obsahuje súbory vo formáte `.csv`. Každý súbor obsahuje určitý počet handshake-ov a reprezentuje konkrétnu aplikáciu. Každý riadok v `.csv` súbore reprezentuje jeden TLS handshake. Jeden handshake sa reprezentuje kombináciou týchto štyroch atribútov:

1. **JA3 hash** - hash vypočítaný z údajov nešifrovanej správy Client Hello.
2. **SNI** - doménové meno stránky, na ktorú sa klient pripája, nachádza sa v správe Client Hello.
3. **JASS hash** - hash vypočítaný z odchytených údajov nešifrovanej správy Server Hello.
4. **File name** - názov `.csv` súboru, v ktorom sa handshake nachádza. Obsahuje v sebe aj názov aplikácie, ku ktorej dáta patria.

3.3 Trénovacia množina

Cieľom je pre každú z aplikácií nájsť minimálne jeden handshake, ktorý by dokázal byť vyhodnotený ako reprezentujúci konkrétnu aplikáciu a týmpádom ju jednoznačne identifikovať. Trénovacia množina slúži ako model z istej, ideálne väčšej časti dát, po ktorej vyhodnotení sa klasifikátor naučí, ktoré handshake-y sú pre dané aplikácie jednoznačné a ktoré nie, inými slovami, do ktorej z dvoch možných skupín, to jest pozitívnej, alebo negatívnej triedy dáta patria.

3.4 Testovacia množina

Testovacia množina slúži na testovanie modelu získaného trénovaním, ktorý obsahuje trénovacia množina. Dáta z testovacej množiny sú rozdielne, avšak rovnakého charakteru s rovnakou štruktúrou. Testovacia množina plní rovnako dôležitú úlohu ako trénovacia a klasifikáciu ako aj výsledné vyhodnotenie výsledkov bez nej by v tejto práci nebolo možné uskutočniť.

Na vyhotovenie testovacej množiny z dátových sád použitých v tejto práci som potreboval pridať zopár parametrov, ktoré mi umožnili rozšíriť možnosti ako v experimentovaní, tak aj vo vytvorení samotnej množiny. Dátové sady obsahujú .csv súbory pre každú aplikáciu, kde každý súbor obsahuje radu handshake-ov, z ktorých niektoré nie sú pre danú aplikáciu špecifické. Túto vlastnosť je nutné poznať. Z tohto dôvodu som dátové sady rozšíril o parameter, ktorý nesie informáciu, či má byť handshake pre aplikáciu špecifický. Jedná sa o hodnotu podobnosti. Tá je vypočítaná porovnaním SNI a kľúčových slov pre každú z aplikácií.

3.4.1 Kľúčové slová

Kľúčové slová (keywords) sú reťazce charakterizujúce typicky doménové meno konkrétnej aplikácie. Ak doménové meno obsahuje ako podreťazec niektoré kľúčové slovo charakteristické pre konkrétnu aplikáciu, je veľmi vysoko pravdepodobné, že k nemu aj patrí. Charakterizujúce kľúčové slová pre domény jednotlivých apikácií sú voľne dostupné na internete, ale viacero z nich som čerpal aj z článku ohľadom klasifikácie pomocou TLS dát, ktorého je táto práca rozšírením [9].

3.4.2 Prahová hodnota podobnosti

Tým, že máme dostupné kľúčové slová pre jednotlivé aplikácie, je možné ich využiť na porovnávanie s jednotlivými handshake-mi. Tu prichádza na rad SNI, ktoré reprezentuje doménové meno serveru pristúpeného z aplikácie, s ktorou užívateľ pracuje. Toto porovnávanie prebehlo aj v článku [9], avšak spôsob bol rozdielny. Namiesto jednoduchšej kontroly toho, či sa kľúčové slovo v danom doménovom mene vyskytuje, som sa rozhodol využiť algoritmus programovacieho jazyku Python, ktorý v sebe obsahuje funkciu na výpočet hodnoty podobnosti dvoch reťazcov. Týmto spôsobom som získal rozdielne výsledky pre rozdielne handshake-y. Ak sa kľúčové slovo pre aplikáciu v doménovom mene z datasetu pre tú aplikáciu nachádza, hodnota podobnosti stúpne na vysokú hodnotu. Ak naopak nie, alebo obsahuje len podreťazec kľúčového slova, pre daný handshake sa vyhodnotí nízka hodnota, až nulová.

Tento spôsob vyhodnotenia podobnosti otvára dvere novým možnostiam experimentovania. Pred začiatkom klasifikácie je napríklad možné zadefinovať určitú prahovú hodnotu

podobnosti, od akej miery jednotlivé handshake-y radiť ako charakteristické pre aplikáciu a od akej už naopak nie. Túto hodnotu je týmpádom možné regulovať a dospieť k rozdielnym výsledkom.

Prahová hodnota podobnosti vypočítaná na základe porovnania kľúčových slov aplikácií a ich SNI sa okrem testovacej množiny využíva aj pri vyhotovení množiny trénovacej, a to ako pomocná hodnota. Tam však jej použitie z implementačného hľadiska nie je nutné. Použil som ju z dôvodu, pretože jednoducho zvyšuje kvalitu klasifikátora. Rozdiel v použití a nepoužití tejto hodnoty v trénovaní je podrobnejšie popísaný v sekcii 5.6.

3.5 Krížová validácia

Ako je už známe, k dispozícii je päť dátových sád. Ich štruktúra je totožná, každá z nich obsahuje rovnaký počet súborov s handshake-mi pre rovnaký počet aplikácií. Je logické, že čím viac dát máme prístupných pre vytvorenie istého obrazu a vytvorenia trénovacej množiny, tým bude klasifikácia presnejšia. Najpresnejší spôsob vypracovania kvalitných trénovacích dát je preto využitie štyroch dátových sád pre trénovanie a následne posledný, piaty použiť ako testovací a pokúsiť sa o identifikáciu handshake-ov, ktoré táto dátova sada obsahuje na základe výsledkov získaných z predošlých štyroch dátových sád. Krížová validácia toto spĺňa, akurát myšlienku rozšíri do širšej miery. Rozšíri ju takým spôsobom, že vyhodnotí všetky možnosti testovacích dátových sád súčasne. Inými slovami, keďže máme v tomto prípade 5 dátových sád, a každá z nich môže byť použitá ako testovacia, pri krížovej validácii prebehne klasifikácia päťkrát, a to v každom prípade pre inú množinu testovacích dát, zatiaľčo ako trénovacie budú využité vždy štyri zvyšné dátové sady. Keďže prebehne päťnásobne viacero operácií, máme v priemere päťkrát väčší počet testovacích dát, z čoho vyplýva, že sa oproti využitiu len jednej testovacej sady vyhodnotí zhruba päťkrát viac handshake-ov, inými slovami, vyhodnotí sa každý dostupný handshake spomedzi všetkých dátových sád.

3.6 Hierarchické vyhodnotenie

Pri vyhodnocovaní výsledkov na základe trojice hodnôt JA3, JA3S a SNI môže nastať situácia, keď sa určitú aplikáciu podarí identifikovať iba na základe jednej alebo kombináciou dvoch z týchto hodnôt. V tom prípade je zvyšná hodnota pre identifikáciu tejto konkrétnej aplikácie nepotrebná, jej využitie nemá žiadne výhody, ba priam naopak, jej využitie vedie k zbytočným výpočtom a môže viesť aj k vyššiemu počtu nepresných, False výsledkom. Práve to je dôvod, prečo som pridal funkcionalitu hierarchického vyhodnotenia, ktoré vyhodnocuje jednotlivé handshake-y na základe tejto trojice hodnôt jednotlivo namiesto toho, aby porovnával trojice ako celky. Algoritmus hierarchického vyhodnotenia je nasledovný:

1. Začne sa vyhodnotenie na základe hodnoty JA3 hash
2. V prípade, že sa aplikáciu podarí identifikovať, vyhodnotí sa úspešne. Ak nie, začne sa vyhodnotenie na základe dvojice hodnôt JA3 hash a JA3S hash
3. V prípade, že sa aplikáciu podarí identifikovať, vyhodnotí sa úspešne. Ak nie, začne sa vyhodnotenie na základe trojice hodnôt JA3 hash, JA3S hash a SNI, ako je to v prípade jednoduchej validácie.

3.7 Kolízia pravdy

Pri testovaní modelu trénovacej množiny môže nastať situácia, keď sa v trénovacej množine určitá kombinácia hodnôt reprezentujúca handshake vyhodnotí ako unikátna pre konkrétnu aplikáciu, zatiaľ čo v testovacej množine sa tá istá kombinácia hodnôt vyhodnotí ako unikátna pre rozdielnu aplikáciu. Tento jav som zazval kolíziou pravdy (True Collision). Táto situácia nám charakterizuje prípad, keď určitá skupinu dát, ktorá by podľa trénovacej množiny mala patriť do určitej triedy, testovacia množina vyhodnotí ako patriacu do triedy úplne rozdielnej. Tento druh problému nespadá pod binárnu klasifikáciu a je týmpádom ojedinelý pre túto prácu, výsledky z nej sa nepridávajú do vyhodnotenia hodnôt Accuracy, Precision a Recall. Z tohto dôvodu som ho nezaradil do všeobecnej teoretickej sekcie 2.4 ohľadom binárnej klasifikácie. Spôsob reprezentácie týchto typov výsledkov je uvedený v bode 3 sekcie 3.9 o rozšírenej matici a ich hodnoty sa takisto objavujú aj v kapitole 5 venovanej experimentom.

3.8 Vstupné hodnoty

Vstupné okno aplikácie potrebuje obsahovať možnosti voľby parametrov výpočtov, aby s nimi užívateľ dokázal jednoducho manipulovať, regulovať vstup a týmpádom získať viacero druhov výsledkov. Aplikácia z tohto dôvodu obsahuje tieto vstupy:

- Možnosť voľby použitia pre každú dátovú sadu individuálne. Jednotlivé dátové sady je možné zvoliť ako trénovacie, testovacie alebo nepoužívané. Táto možnosť umožňuje všetky možné kombinácie. Je možné zvoliť napríklad 2 trénovacie a 2 testovacie dátové sady, zatiaľčo piata sa vôbec nemusí použiť.
- Voľba jednotlivých hodnôt využitých na vytvorenie klasifikačného modelu a na samotnú klasifikáciu z trojice JA3, JA3S a SNI. Takisto umožňuje všetky možné kombinácie.
- Možnosť hierarchického vyhodnotenia. Klasifikátor sa pokúsi aplikácie identifikovať najskôr pomocou JA3, v prípade neúspechu pomocou JA3 a JA3S a následne pomocou štandardnej trojice JA3, JA3S a SNI.
- Možnosť krížovej validácie. Klasifikácia prebehne vo forme piatich operácií, pri ktorých v každej z nich sa použije rozdielna testovacia dátová sada. Je možné ju kombinovať s hierarchickým vyhodnotením.
- Voľba zápisu výsledkov do výstupného `.csv` súboru. Výsledky sa po klasifikácii okrem zobrazenia v užívateľskom rozhraní zapíšu taktiež do externého súboru.
- Nastavenie prahovej hodnoty. Umožnuje nastaviť prahovú hodnotu podobnosti použitú pre klasifikáciu. Má veľký dopad na klasifikáciu a výsledky.
- Zoznam kľúčových slov. Tento zoznam je ďalším vstupom, ktorý aplikácia využíva. Na rozdiel od ostatných vstupov sa nevolí v rozhraní aplikácie, ale je využívaný ako externý súbor `keywords.json`. Súbor je nutné uchovávať v jednej zložke s aplikáciou a zložkou s dátovými sadami.

3.9 Výstupné hodnoty

Aplikácia má štyri rôzne druhy reprezentácie výsledných hodnôt. Viacero druhov výstupu poskytuje širší prehľad o výsledkoch a prináša kvalitnejšie štatistiky ohľadom úspešnosti a presnosti klasifikátora. Týmito výstupnými reprezentáciami sú:

1. **Kompaktná matica** - Obsahuje počet vyhodnotených True Positive (TP), True Negative (TN), False Positive (FP) a False Negative (FN) hodnôt pre každú aplikáciu z testovacej množiny v porovnaní s hodnotou určenou klasifikátorom. Je kompaktnou verziou rozšírenej matice. Oproti rozšírenej obsahuje individuálny počet True Negative hodnôt pre každú aplikáciu, zatiaľčo rozšírená obsahuje celkový počet True Negative, ku ktorým klasifikátor dospel. Ukážku kompaktnej matice je možné vidieť na obrázku č. 3.1.
2. **Zjednodušená matica** - Inými slovami matica zámien binárnej klasifikácie. Jej obsahom je celkový počet získaných hodnôt TP, TN, FP a FN počas klasifikácie, nehľadiac na informáciu, o ktorú konkrétnu aplikáciu sa jedná. Ukážku tejto matice je možné vidieť na obrázku č. 3.2.
3. **Rozšírená matica** - Najväčšia matica zo všetkých. Obsahuje podrobný rozpis výsledkov porovnávania testovacích dát ohodnotenými klasifikátorom. Je možné z nej vyčítať dopad porovnania každého testovaného handshake-u, ktorý sa nachádza nad určenou prahovou hodnotou s handshake-mi ohodnotených klasifikátorom. Jej rozostavenie umožňuje podrobný prehľad prípadných hodnôt kolízie pravdy, čo z nej robí jedinú maticu reprezentujúcu tieto druhy hodnôt. Je navrhnutá, aby bola čo najvhodnejšia pre orientáciu vo výsledkoch. Ideálna pre najrýchlejšie získanie prehľadu pre užívateľa. Ukážka rozšírenej matice je na obrázku č. 5.1
4. **APR matica** - APR = Accuracy, Precision, Recall. Jedná sa o zobrazenie vypočítaných metrík klasifikátorom, vysvetlených v sekcii 2.4.2.

Compact Matrix

XX	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
TN	10	31	3	5	8	10	1	7	14	2	1	10	1	12	4	10	4	5
TP	5	3	1	4	2	2	4	1	0	3	1	8	2	12	0	10	3	6
FP	0	4	0	0	4	0	4	0	0	0	0	0	0	0	0	1	0	3
FN	0	1	0	1	1	2	4	0	0	1	0	1	3	2	0	5	1	6

Obrázek 3.1: Ukážka kompaktnej matice

Simple Matrix

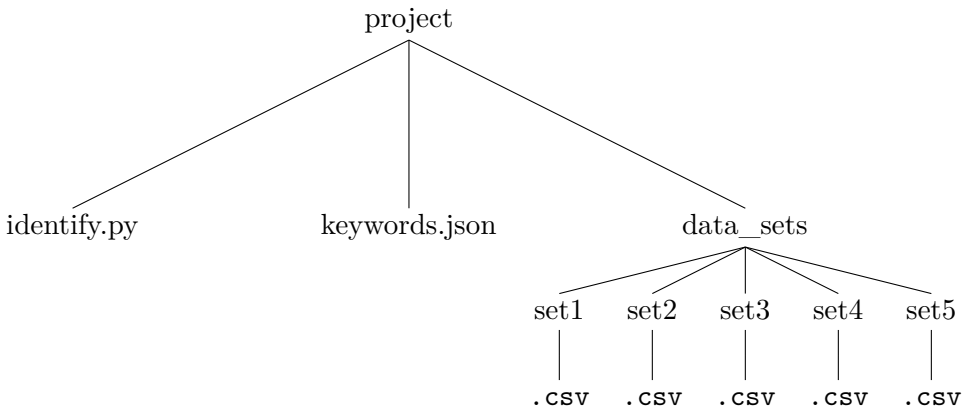
	Actual Positives	Actual Negatives
Predicted Positives	67	16
Predicted Negatives	28	138

Obrázek 3.2: Ukážka zjednodušenej matice (matica zámien)

Kapitola 4

Implementácia klasifikátora

Implementácia prebehla pomocou programovacieho jazyka Python [4]. Samotný klasifikátor, jednoduché užívateľské rozhranie a výpočet výsledných štatistických hodnôt ako napríklad Accuracy, Precision a Recall sú zahrnuté v jednom skripte `identify.py`. Skript je uložený v zložke `project`, spolu so súborom `keywords.json` obsahujúcim zadefinované kľúčové slová pre každú aplikáciu a zložkou obsahujúcou používané dátové sady. Štruktúra projektu je znázornená obrázku č. 4.1.



Obrázek 4.1: Štruktúra projektu

4.1 Kľúčové slová

Identifikácia aplikácií začína pri definovaní kľúčových slov, ktoré sú pre tento projekt neúprosnou potrebou pre vykonanie úspešnej klasifikácie. Tieto kľúčové slová charakterizujúce doménové meno servera jednotlivých aplikácií boli dohľadané na internete [10] alebo prevzaté z článku [9] o identifikácii aplikácií na základe protokolu TLS, ktorý táto práca rozširuje. Kľúčové slová pre jednotlivé aplikácie sú definované nasledovne:

- **Boomplay**: *boomplay*
- **Google Chrome**: *google*
- **CP (Cestovný poriadok)**: *crws*

- **EquaBank**: *equamobile, equa*
- **Facebook**: *facebook*
- **Gmail**: *mail, inbox*
- **Google Calendar**: *googleusercontent, googleapis, calendarsync*
- **KB Klíč**: *login*
- **Messenger**: *fbsbx*
- **Mobilní Banka**: *mojebanka, mobilnibanka, kb, trusteer.*
- **Můj vlak**: *ipws2, timetable.cz*
- **Nextbike**: *nextbike, nextbikeczech*
- **Reddit**: *reddit, redd.it*
- **Seznam CZ**: *seznam, sdn, imedia, szn*
- **Telegram**: *??? (nedefinované)*
- **TikTok**: *musical, tiktok, isnssdk, byteoversea*
- **WhatsApp**: *whatsapp*
- **YouTube**: *googlevideo, ytimg, youtube, youtu.be*

Súčasťou práce bolo takisto testovanie vhodnosti kľúčových slov kvôli dosiahnutiu čo najlepších výsledkov. Je možné si tiež všimnúť, že k aplikácii Telegram nie sú priradené žiadne kľúčové slová. Vysvetlenie k tomuto kroku sa nachádza v sekcii 5.5.4.

4.2 Výpočet a implementácia hodnoty podobnosti

Prahová hodnota podobnosti, ako už bolo spomenuté, je dôležitou súčasťou projektu. Jej implementáciu som sa rozhodol vyriešiť nasledovným spôsobom:

Výpočet hodnoty podobnosti stavia na porovnaní a vyhodnotení podobnosti dvoch reťazcov. Na toto som využil služby programovacieho jazyka `Python`, konkrétne funkciu triedy `SequenceMatcher` importovanej z knižnice `difflib`. Jedná sa o funkciu `SequenceMatcher.ratio()`, ktorá ako vstupné parametre vezme dva reťazce a vráti hodnotu typu `float` v intervale <0,1>.

4.2.1 Funkcia get_matching_blocks()

Funkcia `SequenceMatcher.ratio()` k výpočtu hodnoty podobnosti využíva služby sesterskej funkcie `SequenceMatcher.get_matching_blocks()`. Tá má tri vstupné parametre:

1. Parameter `isjunk`. Slúži na zadefinovanie takzvaných junk znakov alebo reťazcov, ktoré funkcia v prípade potreby nemusí brať do úvahy. Zvyčajne sa jedná o medzery a iné biele znaky. Tie sa v jednotlivých kľúčových slovách ani SNI nenachádzajú, takisto ako ani iné znaky, ktoré by bolo potrebné pre toto porovnávanie ignorovať, takže tento parameter pre tento projekt nie je potrebný ani využívaný. V takomto prípade sa doň predá hodnota `None`.

2. Reťazec, na základe ktorého sa kontroluje nasledovný reťazec
3. Reťazec, ktorý sa kontroluje na základe reťazca v predchádzajúcom parametri

Určite ste si všimli, že som zdôraznil rozdiel v jednotlivých parametroch funckie pre vstupné reťazce. Tieto parametre sa totiž líšia, funkcia s nimi pracuje rozdielnym spôsobom. Ak sa jednotlivé reťazce funkcii odovzdajú v opačnom poradí, s vysokou pravdepodobnosťou dospejeme aj k rozdielnemu výsledku. Algoritmus funkcie je nasledovný:

- Funkcia vracia podľa situácie jednu až viacero trojíc hodnôt, ich počet závisí od počtu nájdených zhodných podreťazcov. Každá trojica vždy charakterizuje jednu zhodu. Štruktúra jednej trojice je formy *(i, j, n)*, kde:
  - i reprezentuje index prvého reťazca, kde začína zhoda
  - j reprezentuje index druhého reťazca, kde začína zhoda
  - n reprezentuje dĺžku, respektíve počet znakov zhody.
- Funkcia vždy vráti minimálne jednu trojicu, a to aj v prípade, ak medzi vstupnými reťazcami je zhoda nulová. To z toho dôvodu, pretože v prípade ľubovoľného počtu zhôd, funkcia vždy vráti počet trojíc, ktoré sa budú rovnať počtu zhôd + 1. Trojica navyše je vždy posledná trojica, ktorá v každom prípade vstupu obsahuje namiesto hodnôt *(i, j, n)* nasledovné hodnoty:
  - celkový počet znakov prvého reťazca
  - celkový počet znakov druhého reťazca
  - hodnotu 0

  Táto trojica je v každom prípade jedinou trojicou, ktorá môže mať a zároveň vždy má hodnotu tretieho atribútu rovnú 0.
- Platí, že ak *(i, j, n)* a *(i', j', n')* sú susedné trojice a druhá trojica nie je poslednou trojicou, tak *i+n != i'*, ako ani *j+n != j'* . Inak povedané, susedné trojice v každom prípade reprezentujú zhodné podreťazce, ktoré nemôžu susediť. Ak sa nad tým človek zamyslí, nemalo by to žiaden zmysel, keďže nad všekymi susednými zhodnými znakmi vždy prebehne konkatenácia do jedného reťazca a informácie o ňom sa vráta v jedinej trojici.

Predpokladajme, že máme dva vstupné reťazce *'abcdef ABCf'* a *'abec ge AeCc'*. Funkcia `get_matching_blocks()` vráti nasledovné trojice:

1. (a=0, b=0, size=2)
2. (a=2, b=3, size=1)
3. (a=4, b=6, size=1)
4. (a=6, b=7, size=2)
5. (a=9, b=10, size=1)
6. (a=11, b=12, size=0)

Ak však zameníme poradie parametrov, výstup je rozdielny. V tomto prípade vráti funkcia nasledovné trojice:

1. (a=0, b=0, size=2)
2. (a=2, b=4, size=1)
3. (a=7, b=6, size=2)
4. (a=10, b=9, size=1)
5. (a=12, b=11, size=0)

Ako je možné vidieť na výstupe, funkcia vrátila v prvom prípade 6 trojíc, v druhom prípade 5. Stále však platí, že posledná trojica je navyše. Pohľadom na jednotlivé atribúty `size` je možné si všimnúť, že funkcia v prvom prípade identifikovala dokopy 7 znakov ako zhodných, v prípade druhého vstupu iba 6. Toto sa udeje z toho dôvodu, pretože algoritmus vyhodnotenia začína spôsobom, že vezme obidva reťazce a začne nimi prechádzať zľava doprava. Identifikuje prvú zhodu a jej dĺžku (počet znakov). Vyhodnotí prvú trojicu. Následne zaháji proces odzova, avšak znaky, ktoré sa nachádzajú pred naposledy identifikovanou dvojicou podreťazcov ignoruje, vrátane samotnej identifikovanej zhodnej dvojice podreťazcov. Týmto spôsobom môže dospieť k rozdielnym výsledkom. Prehľadná ukážka identifikovaných jednotných podreťazcov medzi dvomi vstupmi v oboch variantách je uvedená v tabuľke 4.1.

	Varianta 1	**Varianta 2**
Parameter 1	ab c d e f A B C f	ab e c ge A e C c
Parameter 2	ab e c g e A e C c	ab cd e f A B C f

Tabulka 4.1: Porovnanie rozdielu identifikácie na základe poradia vstupných reťazcov

4.2.2 Funkcia ratio()

Tu nastupuje na rad samotná funkcia `ratio()`, ktorá využíva informácie o reťazcoch získané funkciou `get_matching_blocks()`. Jej funkcionalita je pomerne jednoduchá, k výpočtu hodnoty podobnosti využíva nasledovný vzorec:

$$\frac{2 * M}{T}$$

1. **M** reprezentuje súčet počtu zhodných znakov medzi dvomi porovnávanými reťazcami získaných funkciou `get_matching_blocks()`
2. **T** reprezentuje súčet celkovej dĺžky oboch vstupných reťazcov

Výstupom funkcie `SequenceMatcher(None, "abcdef ABCf", "abec ge AeCc").ratio()` bude preto:

$$\frac{2 * 7}{23} = 0.61$$

Pre prípad s vymenenými reťazcovými parametrami to však bude:

$$\frac{2 * 6}{23} = 0.52$$

4.2.3 Porovnanie kľúčového slova s SNI

Predpokladajme prípad aplikácie Boomplay. Kľúčové slovo pre túto aplikáciu je jednoduché - názov aplikácie. V jednej z dátových sád pre Boomplay existuje handshake s nasledovným SNI. Spolu s kľúčovým slovom pre aplikáciu získame nasledovnú dvojicu pre porovnanie:

- **SNI:** source.boomplaymusic.com
- **kľúčové slovo:** boomplay

V prípade TLS dát je proces porovnávania reťazcov jednoduchší. Keďže rátame s tým, že kľúčové slovo pre aplikáciu je vždy podreťazec doménového mena, čiže SNI, vieme, že funkcia `get_matching_blocks()` nám vráti dve trojice, a z nich len jednu so zadefinovanou dĺžkou zhody. V opačnom prípade nám môže vrátiť rozkúskované znaky do trojíc, ktorým však funkcia `ratio()` vyhodnotí výrazne nižšiu hodnotu podobnosti oproti tým hodnotám SNI, v ktorých sa kľúčové slová našli.

Výsledné trojice vrátené funkciou `get_matching_blocks()` sú v tomto prípade nasledovné:

1. (a=7, b=0, size=8)
2. (a=24, b=8, size=0)

Z tohto výstupu môžme predpokladať, že kľúčové slovo boomplay, ktoré tvorí 8 znakov sa nachádza v danom SNI, keďže hodnota veľkosti v prvej trojici je taktiež 8. Následne v poslednej trojici vidíme, že počet znakov v SNI je 24, zatiaľčo u kľúčového slova spomínaných 8. Tieto hodnoty využíva funkcia `ratio()` pre výpočet hodnoty podobnosti:

$$\frac{2 * M}{T} = \frac{2 * 8}{24 + 8} = 0.5$$

4.2.4 Integrácia hodnoty podobnosti

Po získaní hodnoty podobnosti pre jednotlivé hodnoty SNI TLS handshake-u a k nemu príslušnému kľúčovému slovu, je na čase ho integrovať do klasifikátora. Keďže hodnoty podobnosti je potrebné vypočítať pre každý handshake individuálne, čo znamená individuálnu operáciu pre každý riadok v každej dátovej sade, mojím riešením je vytvorenie piateho stĺpca v dátových sadách, dedikovaného práve pre hodnotu podobnosti. Na to slúži mnou naimplementovaná funkcia `addFifthColumn()`. Funkcia má jeden vstupný parameter, a to absolútnu cestu k súboru z dátovej sady, s ktorou následne pracuje. Klasifikátor pracuje s jednotlivými dátovými sadami v cykle, ako aj s jednotlivými súbormi v konkrétnej dátovej sade. V situácii, keď sa zavolá `addFifthColumn()` na určitý súbor, piaty stĺpec sa do súboru dátovej sady pridá nasledovným spôsobom:

1. Funkcia `addFifthColumn()` otvorí súbor s kľúčovými slovami
2. Funkcia otvorí súbor špecifikovaný v jeho vstupnom parametri v režime `read`.
3. Na základe absolútnej cesty k súboru sa identifikuje názov súboru v dátovej sade, ktorý v sebe obsahuje aj názov aplikácie. Týmpádom klasifikátor vie, o akú aplikáciu sa jedná a ktoré špecifické kľúčové slová využiť pre porovnávanie na výpočet prahovej hodnoty.
4. Pre každý individuálny riadok v súbore prebehne výpočet hodnoty podobnosti na základe všetkých kľúčových slov charakterizujúcich aplikáciu, ku ktorej daný súbor patrí a ktorej TLS dáta obsahuje.
5. Do zoznamu (`Python list`) sa uloží každý riadok daného súboru a následne sa k nemu pripojí najvyššia dosiahnutá hodnota podobnosti spomedzi všetkých dostupných kľúčových slov pre danú aplikáciu.
6. Po získaní hodnoty podobnosti pre každý riadok v súbore sa súbor zatvorí. Program získal zoznam s prvkami identickými vzhľadom k jednotlivým riadkom súboru, avšak s pridanou prahovou hodnotou.
7. Súbor sa otvorí po druhýkrát v režime `read`. Každý súbor v dátovej sade má ako prvý riadok názvy jednotlivých hodnôt ktoré obsahuje, samotné hodnoty sa nachádzajú až od druhého riadku nižšie. V prípade, súbor už obsahuje vygenerovaný piaty stĺpec, funkcia to zistí skontrolovaním prvého riadku súboru. V takomto prípade funkcia končí. V opačnom prípade sa do zoznamu táto informácia uloží.
8. Súbor sa otvorí v režime `write`. Výsledný zoznam s hodnotami reprezentujúcimi riadky s pridanou hodnotou podobnosti je vytvorený vo formáte, aby štruktúra a poradie boli zhodné s tou v súbore. Následne prebehne jednoduchý prepis jednotlivých riadkov na nové, už obsahujúce vygenerované hodnoty podobnosti na základe kľúčových slov.

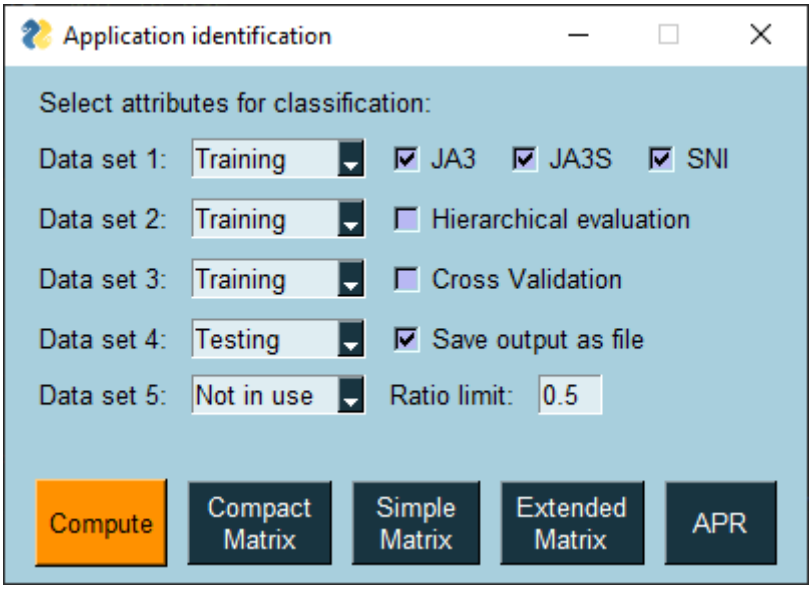
4.3 Užívateľské rozhranie

Pre zjednodušenie zadávania vstupných parametrov som sa rozhodol implementovať klasifikátor vo forme jednoduchej aplikácie, ktorá užívateľovi výrazne uľahčuje prácu s experimentovaním. Na jej implementáciu som využil technológiu `PySimpleGui` [11]. Jedná sa o balíček

rozšírenia programovacieho jazyka `Python`, v ktorom je celý klasifikátor implementovaný. Umožňuje vytvorenie jednoduchého grafického užívateľského rozhrania pre skripty, ktoré majú veľa možností vstupných parametrov a týmpádom poskytuje oproti možnosti zadávaní parametrov v príkazovom riadku obrovskú výhodu. S touto možnosťou sa skript spúšťa bez akýchkoľvek parametrov, všetky atribúty pre klasifikáciu sa špecifikujú pomocou grafického rozhrania. Grafické rozhranie takisto umožňuje reprezentáciu výsledkov získaných klasifikátorom vo forme vytvorených, na účel navrhnutých maticových tabuliek.

4.3.1 Vstupné okno

Po zapnutí skriptu sa objaví úvodné, vstupné okno rozhrania. Znázorňuje ho obrázok č. 4.2.



Obrázek 4.2: Ukážka vstupného okna grafického rozhrania

- Ako znázorňuje ukážka, pre každú dátovú sadu je možné individuálne zvoliť jeho rolu v klasifikácii. Na výber sú možnosti trénovanie - testovanie - nepoužité. V prípade zvolenia krížovej validácie, čo predstavuje 5 behov klasifikácie s vždy rozdielnou testovacou sadou sa tento výber neberie do úvahy a prebehne krížová validácia.
- Podobne pri výbere jednotlivých atribútov TLS handshake-u z trojice JA3 - JA3S - SNI pre klasifikáciu sú možné všetky možné kombinácie, avšak pri zvolení hierarchického vyhodnotenia, čo reprezentuje pokus o identifikáciu na základe JA3, následne kombináciou JA3 a JA3S a výsledne JA3, JA3S a SNI sa individuálny výber týchto hodnôt neberie do úvahy.
- Hierarchické vyhodnotenie a krížovú validáciu je možné kombinovať. Úvodné vstupné okno takisto obsahuje možnosť zápisu výsledkov do výstupného súboru `output.csv`.
- Prahová hodnota podobnosti musí byť udaná v desatinnom tvare, kde desatinné znamienko je reprezentované bodkou. Udáva sa v intervale (0,1).
- Vstupné okno obsahuje takisto tlačidlo `Compute` na spustenie klasifikácie a tlačidlá na zobrazenie výsledkov vo forme matíc.

4.3.2 Výstupné okná

Po zadaní vstupných hodnôt a kliknutí na tlačidlo `Compute` prebehne klasifikácia na základe zadaného vstupu a aktualizujú sa tabuľky, ku ktorým sa pristupuje pomocou jednotlivých tlačidiel. Je potrebné brať do úvahy, že užívateľské rozhranie a tlačidlo `Compute` nespúšťa samotný skript, ale celé tvoria súčasť skriptu. Po spustení skriptu na príkazovej riadke sa okno užívateľského rozhrania otvorí a zostane otvorené po dobu, kým ho užívateľ sám nezavrie. Skript je za túto dobu celý čas pochopiteľne aktívny. Táto dynamická vlastnosť skriptu umožňuje vyhodnotenie ľubovoľného počtu klasifikácií iba behom jediného spustenia. Pri spustení skriptu sú jednotlivé výstupné matice prázdne, čo je pochopiteľné, keďže pri danom spustení ešte žiadna klasifikácia neprebehla. Po vyhodnotení prvej klasifikácie po spustení sa údaje do výstupných matíc doplnia a po ďalších následných vyhodnotených klasifikáciách sa ich obsah na základe zvolených vstupných hodnôt dynamicky mení.

4.4 Spracovanie vstupných dát

Táto sekcia je venovaná spracovaniu vstupných dát, konkrétne sa jedná o vytvorenie trénovacieho a testovacieho slovníka na základe vybraných množín pre klasifikáciu.

Po spustení skriptu a následnom otvorení užívateľského rozhrania skript zostáva po celú dobu aktívny. Jednotlivé tlačidlá fungujú vo forme udalostí (`event`), ktoré sa aktivujú kliknutím na ne. Po vyplnení vstupných údajov a stlačení tlačidla `Compute` tlačidlo spustí udalosť s rovnakým názvom, ktorá v sebe zahŕňa klasifikáciu (vytvorenie modelu a jeho otestovanie na zvolených dátových sadách). Kľúčovou funkciou pre identifikáciu jednotlivých aplikácií na základe TLS dát je funkcia `identify()`, ktorá má viacero podôb, záležiac od druhu vstupu a týmpádom druhu klasifikácie. V tejto sekcii vysvetlím ich spôsob fungovania, ako aj spôsob fungovania funkcií `createTestingDict()`, `createTrainingDict()` a `adjustTrainingDict()`.

4.4.1 Funkcie identify()

Funkcia identify má 4 podoby, ktoré sú rozdelené do štyroch individuálnych funkcií. Týmito funkciami sú:

- `identify()` - Funkcia určená na klasifikáciu na základe individuálne zvolených atribútov TLS protokolu (JA3, JA3S, SNI) a individuálne zvolených rolí jednotlivých dátových sád.
- `identifyCross()` - Používa sa pri individuálnom výbere atribútov TLS protokolu v kombinácii s krížovou validáciou.
- `identifyHierarchical()` - Táto funkcia sa volá pri hierarchickom vyhodnotení a individuálnom výbere dátových sád.
- `identifyHierarchicalCross()` - Pracuje so vstupom, ktorý zahŕňa kombináciu hierarchického vyhodnotenia s krížovou validáciou.

Funkcie majú viacero vstupných parametrov, nimi sú:

- objekt rozšírenia `PySimpleGui`, ktorý slúži ako identifikátor dátovej sady a taktiež obsahuje jej užívateľom priradenú vlastnosť z trojice (Training, Testing, Not in use)

- prahovú hodnotu podobnosti zadanú užívateľom
- slovníky `Python dictionary` slúžiace pre uloženie informácií o dátových sadách a ich testovacej a trénovacej množine dát. Slovník s testovacími dátami je vždy len jeden. Počet trénovacích slovníkov sa odvíja od druhu klasifikácie, počet trénovacích dátových sád naň nemá vplyv, keďže sa množiny dát jednej kategórie vždy zjednotia. Ich počet sa líši v prípade hierarchického vyhodnotenia. Klasické vyhodnotenie využíva jeden slovník s trénovacími dátami. Hierarchické vyhodnotenie využíva tri, a to jeden pre identifikáciu na základe JA3, druhý pre JA3 + JA3S a tretí pre JA3 + JA3S + SNI.
- Tak, ako väčší počet trénovacích slovníkov charakterizuje funkcie využívané pre hierarchické vyhodnotenie, tak funkcie využívané pre klasické vyhodnotenie charakterizuje parameter `checkValue`. V prípade hierarchického vyhodnotenia ho nie je potreba, keďže toto vyhodnotenie samotné o sebe charakterizuje spôsob využitia atribútov TLS handshake-u. Hodnota `checkValue` totiž obsahuje informáciu o zvolenej kombinácii týchto atribútov vo vstupnom rozhraní v prípade, ak si užívateľ nezvolil možnosť hierarchického vyhodnotenia.
- Funkcie pre krížovú validáciu majú takisto svoj charakteristický parameter. Ním je index behu vyhodnotenia `i`. Keďže krížová validácia spočíva v únii piatich rozdielnych behov vyhodnotenia, jednotlivé behy sa identifikujú práve týmto parametrom.

Štyri funkcie s jednotlivými parametrami vyzerajú týmpádom nasledovne:

- `identify(dataSetVal, ratioLimit, trainingTempDict, testingTempDict, checkValue)`
- `identifyHierarchical(dataSetVal, ratioLimit, trainingTempDict1, trainingTempDict2, trainingTempDict3, testingTempDict)`
- `identifyCross(dataSetVal, ratioLimit, trainingTempDict, testingTempDict, checkValue, i)`
- `identifyHierarchicalCross(dataSetVal, ratioLimit, trainingTempDict1, trainingTempDict2, trainingTempDict3, testingTempDict, i)`

Nasledovný opis algoritmu charakterizuje funkcionalitu všetkých štyroch funkcií s ich opísanými rozdielmi:

1. V prípade, že nie je zvolená krížová validácia, čo sú prípady funkcií `identify()` a `identifyHierarchical()`, sa odignorujú dátové sady, ku ktorým bola vo vstupnom rozhraní priradená vlastnosť "Not in use".
2. Zaháji sa cyklus, v ktorom prebehne iterácia cez každý súbor v dátovej sade danej parametrom `dataSetVal`. Pre .csv súbor v iterácii sa zavolá funckcia `addFifthColumn()`, ktorá má na starosti pridanie vypočítanej hodnoty podobnosti do súboru vo forme piateho stĺpca. Jej funkcionalita je opísaná v sekcii 4.2.4. Následne v prípade, že je pre danú dátovú sadu priradená vlastnosť testovacej dátovej sady, cesta súboru sa predá do funkcie `createTestingDict()`. V opačnom prípade sa predá do funkcie `createTrainingDict()`. V prípade krížovej validácie sa namiesto tejto metódy rozhoduje na základe indexu behu vyhodnotenia, kde sa po jednotlivých behoch vystrieda každá dátová sada ako testovacia.

3. Pri predávaní parametrov do funkcií na vytvorenie slovníkov sa berie do úvahy aj využitie atribútov TLS handshake-u. V prípade hierarchického vyhodnotenia sa volá funkcia `createTrainingDict()` trikrát, jednotlivo pre každú kombináciu atribútov, ktoré toto vyhodnotenie charakterizujú. V opačnom prípade sa funkcia volá iba raz a predáva sa jej hodnota `checkValue`.

4. Týmto funkcia `identify()` končí a vytváranie množín pokračuje funkciami `createTestingDict()` a `createTrainingDict()`.

4.4.2 Funkcie createTestingDict() a createTrainingDict()

Tieto funkcie sa volajú iba vo funkciách `identify()` a týmpádom tvoria ich dôležitú súčasť.

Funkcia `createTestingDict()` sa vždy volá v iterácii po jednotlivých súboroch v cykle pre špecifickú dátovú sadu. Má nasledovné vstupné parametre:

- **`filepath`** - Absolútna cesta `.csv` súboru nachádzajúceho sa v niektorom z dátových sád.
- **`dic`** - Slovník, ktorý po skončení algoritmu bude obsahovať celú testovaciu množinu dát.
- **`ratioLimit`** - Prahová hodnota podobnosti definovaná užívateľom vo vstupe.

Algoritmus funkcie je nasledovný:

1. Otvorí sa súbor špecifikovaný absolútnou cestou vo vstupnom parametri funkcie.
2. Spustí sa cyklus skrz celý súbor s iteráciou riadok po riadku.
3. Od druhého riadku súboru (prvého riadku reprezentujúci TLS handshake) prebieha porovnávavanie už doplneného piateho stĺpca obsahujúcim vypočítanú hodnotu podobnosti s užívateľom určenou prahovou hodnotou.
4. Každý riadok reprezentujúci handshake sa uloží do slovníka ako kľúč. Ak má handshake hodnotu podobnosti väčšiu ako zadefinovaná prahová hodnota, ku kľúču slovníka obsahujúcemu riadok súboru sa priradí hodnota `known`, ktorá napovedá, že program podľa uvedenej prahovej hodnoty podobnosti na základe kľúčového slova vyhodnotil handshake ako unikátny pre danú aplikáciu. V opačnom prípade priradí k danému riadku hodnotu `unknown`.

Funkcia `createTrainingDict()` funguje podobným spôsobom, no je o isté aspekty rozsiahlejšia. Má štyri vstupné parametre, tri z nich sú parametrami funkcie `createTestingDict()`, zostávajúcim parametrom je parameter `checkValue`, ktorý obsahuje reťazec, ktorý informuje o použitých atribútoch TLS handshake-u pre klasifikáciu. Funkcia funguje na základe nasledovného algoritmu:

1. Otvorí sa súbor špecifikovaný absolútnou cestou vo vstupnom parametri funkcie.
2. Spustí sa cyklus skrz celý súbor s iteráciou riadok po riadku.
3. Od druhého riadku súboru (prvého riadku reprezentujúci TLS handshake) prebieha porovnávavanie už doplneného piateho stĺpca obsahujúcim vypočítanú hodnotu podobnosti s užívateľom určenou prahovou hodnotou.

4. Do parametrom uvedeného slovníka `dic` sa vytvorí nový kľúč s názvom aplikácie, získanej pomocou parametra `filepath`.

5. Na základe uvedenej hodnoty `checkValue` sa jednotlivé atribúty TLS handshake-u, ktoré riadok obsahuje zoskupia do jedného zoznamu, ktorý sa následne priradí k príslušnému kľúču v slovníku.

Týmto sa vytvoril slovník obsahujúci atribúty handshake-u zadefinované užívateľom, ktoré sú na základe kľúčových slov unikátne zoskupené do jedného zoznamu pre každú aplikáciu samostatne.

Pre zhrnutie, použitie funkcií `identify()` a následne funkcií volaných z nej, čiže `addFifthColumn()`, `createTestingDict()` a `createTrainingDict()` umožňuje pridať piaty stĺpec do `.csv` súborov jednotlivých dátových sád a následne vytvoriť slovník obsahujúcu testovaciu množinu dát, ako aj slovník obsahujúci množinu dát pre trénovanie, avšak zatiaľ nie v konečnej podobe.

4.4.3 Funkcia adjustTrainingDict()

Podobu slovníka s trénovacími dátami v momente po použití jednej z funkcií `identify()` by bolo možné opísať ako zoznam aplikácií s k nej príslušnými handshake-mi, ktoré sa dajú zaradiť ako jedinečné pre aplikáciu, ku ktorej patria. Táto podoba slovníka pre klasifikátor nie je vyhovujúca, keďže klasifikátor potrebuje vyhodnotiť jednotlivé handshake-y, ku ktorým následne priradí aplikáciu, nie opačne. Z tohto dôvodu je potreba trénovaciu množinu upraviť na rozdielny tvar. Na tento účel slúži funkcia `adjustTrainingDict()`.

Štruktúra trénovacej množiny získanej funkciou `createTrainingDict()` je nasledovná:

- Aplikácia A : ['TLS handshake 1', 'TLS handshake 2']
- Aplikácia B : [ ]
- Aplikácia C : ['TLS handshake 2', 'TLS handshake 3']
- ...

Predpokladajme, že nastane situácia ako na príklade. Aplikácie A a C majú jeden spoločný handshake, ktorý sa nejakým spôsobom podarilo priradiť k obom aplikáciám, napríklad zapríčinením podobnosti dvoch kľúčových slov. Aplikácia B nemá žiadne priradené handshake-y, týmpádom nemá v trénovacej množine žiadnu hodnotu. Funkcia `AdjustTrainingDict()` má dve fázy refaktorizácie slovníka. V prvej fáze obráti pohľad slovníka a na miesto kľúča namiesto názvu aplikácie nasadí jednotlivé handshake-y. Štruktúra príkladnej trénovacej množiny z ukážky po tejto fáze vyzerá nasledovne:

- TLS handshake 1 : ['Aplikácia A']
- TLS handshake 2 : ['Aplikácia A', 'Aplikácia C']
- TLS handshake 3 : ['Aplikácia C']
- ...

Ako je možné vyčítať z ukážky, táto štruktúra trénovacieho slovníka umožňuje vyfiltrovať aplikácie, pre ktoré sa žiadne charakteristické TLS handshake-y nenašli. Takisto odhalí handshake-y, ktoré boli priradené k viacerým aplikáciám ako charakteristické, ale nesprávne.

Ak sa daný handshake nachádza vo viacerých aplikáciách, je logické, že nemôže jednoznačne charakterizovať ani jednú z nich. Nasleduje druhá fáza funkcie, ktorá tento problém vyhodnotí, a prispôsobí hodnoty kľúčov potrebnej podobe:

- TLS handshake 1 : ['Aplikácia A']
- TLS handshake 2 : ['unknown']
- TLS handshake 3 : ['Aplikácia C']
- ...

Inými slovami, funkcionalitu funkcie `adjustTrainingDict()` je možné chápať ako úpravu štruktúry trénovacej množiny do formy vhodnej pre vyhodnotenie binárnej klasifikácie s testovacou množinou, tak aj ako druhú fázu filtrovania tých špecifických TLS handshake-ov, ktoré nie sú pre identifikáciu jednotlivých aplikácií vhodné a ktoré sa nepodarilo vyfiltrovať na základe kľúčových slov.

4.5 Vyhodnotenie vytvoreného klasifikátoru

Po vytvorení slovníkov pre trénovaciu a testovaciu množinu a ich následnej úprave do vhodného tvaru je na čase vyhodnotiť výsledky binárnej klasifikácie. Na to slúžia funkcie `evaluate()` a `evaluateHierarchical()`. Ich úlohou je porovnať a následne vyhodnotiť predpoveď danú klasifikátorom a označenie získané s využitím prahovej hodnoty podobnosti a následne získať potrebné výsledky vo forme TP (True Positive), TN (True Negative), FP (False Positive) a FN (False Negative).

4.5.1 Funkcia evaluate()

Funkcia `evaluate()` je jednoduchšou z dvoch variant funkcií pre vyhodnotenie výsledkov. Pracuje so slovníkmi získanými funkciami `identify()` a `identifyCross()`.

Vieme, že testovací slovník používa ako kľúč celý riadok `.csv` súboru, čo znamená, že obsahuje hodnoty JA3, JA3S, SNI a názov súboru, ktorého podreťazcom je aj názov aplikácie. Ako hodnotu kľúča používa jeho označenie na základe podobnosti, tzn. `known` / `unknown`. Naopak trénovací slovník obsahuje vždy tie atribúty TLS protokolu, ktoré si užívateľ zvolil pri zadávaní vstupných hodnôt klasifikátora, môže to byť napríklad iba JA3 hodnota, alebo JA3 a JA3S. V prípade, že je táto hodnota alebo kombinácia hodnôt v rámci trénovacej množiny unikátnou pre niektorú z aplikácií, ako hodnotu kľúča uchováva tento slovník názov danej aplikácie. V prípade, že funkcia `adjustTrainingDict()` detekovala viacero aplikácií pre daný kľúč, jeho hodnota v slovníku je namiesto názvu aplikácie `unknown`.

Funkcia evaluate má 2 vstupné parametre, a to `testingDict` a `trainingDict`, čiže slovníky s testovacími a trénovacími množinami vo finálnej podobe. Algoritmus funkcie je nasledovný:

1. Zaháji sa cyklus pre testovací slovník, iteruje sa pre každý testovaný handshake.
2. V každej iterácii cyklu pre testovací slovník sa zaháji druhý cyklus, tentoraz pre trénovací slovník.

3. V každej iterácii pre trénovací slovník sa jeho kľúč obsahujúci zvolené atribúty TLS protokolu rozdelí na jednotlivé časti, aby sa umožnila práca s každým atribútom samostatne. Pre každý atribút prebehne kontrola, či sa nachádza v kľúči testovacej množiny, v ktorej iterácii sa práve nachádza. Ak sa aspoň jeden z atribútov trénovacieho kľúča nezhoduje s porovnávanou hodnotou v testovacom kľúči, znamená to, že neprišlo k zhode handshake-ov. Pokračuje sa nasledujúcou iteráciou pre trénovací slovník, čiže nasledujúcim handshake-om v rade, až kým sa nenájde hľadaný handshake, alebo sa neminú všetky handshake-y v trénovacej sade. V prípade, že sa nájde zhoda medzi testovacím a trénovacím handshake-om, príde k jeho vyhodnoteniu. Vyhodnotenie je implementované na základe jednoduchej binárnej klasifikácie, čo znamená, že môžme dospieť k štyrom rôznym výsledkom:

    - Hodnota handshake-u v obidvoch slovníkoch je `unknown`. To znamená, že odhad klasifikátora bol úspešný, tento testovaný handshake nedokáže identifikovať aplikáciu. Klasifikátor získal novú hodnotu TN (True Negative)
    - Hodnota handshake-u sa ani v jednom zo slovníkov nerovná hodnote `unknown`. To znamená, že testovacia množina na základe podobnosti obsahuje pre tento handshake hodnotu `known` a zároveň trénovacia množina obsahuje názov aplikácie, ktorú identifikuje. Klasifikátor získal hodnotu TP (True Positive). V prípade, že v tomto porovnaní nastala nezhoda medzi identifikovanými aplikáciami, znamená to prípad kolízie pravdy a výsledok sa namiesto vyhodnotenia True Positive vyhodnotí touto cestou.
    - V prípade, že kľúč v testovacej sade obsahuje pre handshake hodnotu `unknown`, ale v trénovacej sade sa rovnaký handshake podarilo vyhodnotiť ako charakteristický pre jedinú aplikáciu, znamená to, že klasifikátor sa v trénovacej fáze vyhodnotenia daného handshake-u netrafil a handshake, ktorý je v skutočnosti `unknown`, charakterizoval ako unikátny pre niektorú z aplikácií. Tento prípad opisuje výsledok FP (False Positive).
    - V prípade, že je vyhodnocovaný kľúč testovacej sady označený ako `known`, čiže ako charakteristický pre určitú aplikáciu a zároveň ekvivalentný kľúč v trénovacej sade je označený ako `unknown`, znamená to, že sa klasifikátor takisto netrafil, akurát v opačnom prípade. Tento prípad charakterizuje FN (False Negative).

4. Ak nastane prípad, kedy sa v trénovacej množine nenájde ekvivalentný handshake k testovanému, jeho očakávaná hodnota je automaticky `unknown`. V prípade, že jeho kľúč obsahuje túto hodnotu, vyhodnotí sa ako True Negative. V opačnom prípade sa vyhodnotí ako False Negative.

4.5.2 Funkcia evaluateHierarchical()

Funkcia `evaluateHierarchical()` je obdobou funkcie `evaluate()` pre hierarchické vyhodnotenie. Týmpádom pracuje s množinami získanými funkciami `identifyHierarchical()` a `identifyHierarchicalCross()`. Obsahuje štyri vstupné parametre, a to:

- `testingDict` - Slovník obsahujúci testovaciu množinu dát
- `trainingDict` - Slovník obsahujúci trénovaciu množinu dát na základe hodnoty JA3

- `trainingDict2` - Slovník obsahujúci trénovaciu množinu dát na základe kombinácie hodnôt JA3 a JA3S
- `trainingDict3` - Slovník obsahujúci trénovaciu množinu dát na základe kombinácie hodnôt JA3, JA3S a SNI

Funkcia `evaluateHierachical()` je sama o sebe jednoduchá, pretože v sebe obsahuje iba jediný cyklus. A to iterácie na základe kľúčov slovníka obsahujúceho množinu s testovacími dátami. V tomto cykle sa však volá jej pomocná funkcia `evaluateHierarchicalKey()`, ktorá vykonáva už omnoho komplexnejšie vyhodnotenie. Vstupné parametre má takmer totožné, líši sa len jeden z parametrov, namiesto testovacieho slovníka v celej podobe sa do nej predá len samotný kľúč, s ktorým sa v aktuálnej iterácii pracuje. Inými slovami, funkcia `evaluateHierarchicalKey()` sa volá individuálne pre každý testovací kľúč a jej výsledkom je vždy vyhodnotenie predaného kľúča. Algoritmus funkcie `evaluateHierarchicalKey()` je nasledovný:

1. zaháji sa cyklus pre trénovací slovník na základe hodnoty JA3.
2. V prípade, že sa táto hodnota nachádza v handshake-u aktuálne testovaného kľúča, prebehne nasledovné vyhodnotenie:
   - V prípade, že kľúč daného trénovacieho slovníka je označený ako charakterizujúci určitú aplikáciu a zároveň testovací kľúč je označený ako `known`, handshake sa vyhodnotí ako True Positive alebo v prípade nezhode aplikácií ako kolízia pravdy a funkcia končí.
   - V prípade, že kľúč daného trénovacieho slovníka je označený ako charakterizujúci určitú aplikáciu a zároveň testovací kľúč je označený ako `unkown`, handshake sa vyhodnotí ako False Positive a funkcia končí.
   - V prípade, že kľúč daného trénovacieho slovníka je označený ako `unknown`, funkcia zaháji nový cyklus, tentoraz pre slovník trénovacej množiny na základe kombinácie hodnôt JA3 a JA3S:
     - Nasleduje rovnaký algoritmus porovnávania, ako v predchádzajúcom kroku. V prípade, že sa hodnoty JA3 a JA3S z trénovacieho kľúča nachádzajú v testovacom kĺúči a zároveň je trénovací kľúč označený ako charakterizujúci aplikáciu, prebehne vyhodnotenie rovnakým spôsobom ako v predchádzajúcom kroku.
     - Ak je kombinácia hodnôt JA3 a JA3S v trénovacej množine označená ako `unknown`, prebehne algoritmus pre tretí slovník obsahujúci kombinácie hodnôt JA3, JA3S a SNI. V tejto fáze sa hodnoty vyhodnotia rovnakým spôsobom ako v prípade jednoduchšej funkcie `evaluate()` a následne funkcia končí.
3. Ak nastal prípad, že sa nenašiel pre testovaný handshake žiaden zhodný v trénovacej množine, znamená to, klasifikátor rozhodne, že handshake takéhoto charakteru má mať charakteristiku `unknown`. Preto ak takýto handshake je označený ako `unknown`, vyhodnotí sa ako True Negative. V opačnom prípade sa vyhodnotí ako False Negative.

4.6 Zápis a reprezentácia vyhodnotení

Pri vyhodnotení istého handshake-u do jednej zo štvorice kategórií True Positive - False Positive - True Negative - False Negative funkciami `evaluate()` a `evaluateHierarchical` sa využívajú dva typy funkcií. Sú nimi funkcie `fillListPositive()` a `fillListNegative()`. Tieto funkcie využívajú globálne premenné vo forme zoznamov (Python list). Každý z týchto zoznamov má osemnásť prvkov, kde každý prvok reprezentuje jednu z aplikácií určených pre identifikáciu. Aplikácie sú zoradené v abecednom poradí. Týmito zoznamami sú:

- **appList** - zoznam názvov aplikácií. Jeho prvkami sú reťazce obsahujúce názvy jednotlivých aplikácií pre identifikáciu.
- **TP** - zoznam hodnôt True Positive. Pri deklarácii obsahuje 18 hodnôt rovných 0.
- **FP** - zoznam hodnôt False Positive. Pri deklarácii obsahuje 18 hodnôt rovných 0.
- **TN** - zoznam hodnôt True Negative. Pri deklarácii obsahuje 18 hodnôt rovných 0.
- **FN** - zoznam hodnôt False Negative. Pri deklarácii obsahuje 18 hodnôt rovných 0.
- Na uloženie hodnôt kolízie pravdy sa využíva takisto zoznam, je ním zoznam zoznamov **trueCollision**. Pri deklarácii obsahuje 18 prázdnych zoznamov.

4.6.1 Funkcia fillListPositive()

Táto funkcia slúži na evidenciu hodnôt True Positive a False Positive. Volá sa vo funkciách `evaluate()` a `evaluateHierarchicalKey()` v momente, keď príde k vyhodnoteniu testovaného handshake-u.

Predpokladajme, že funkcia pre vyhodnotenie vyhodnotí istý handshake ako True Positive. Týmpádom zavolá funkciu s nasledovnými parametrami: `fillListPositive(TP, trainingDict.get(trainingKey))`, kde:

- `TP` reprezentuje globálnu premennú TP obsahujúcu zoznam hodnôt typu `integer`
- `trainingDict.get(trainingKey)` je hodnota kľúča v trénovacom slovníku, čo je v prípade pozitívneho vyhodnotenia názov aplikácie, ktorú hodnoty v danom handshake-u charakterizujú.

Algoritmus funkcie `fillListPositive()` funguje nasledovne:

1. Zaháji sa cyklus pre globálnu premennú `appList`, ktorá obsahuje zoznam aplikácií.
2. V zozname sa nájde príslušná aplikácia, ktorá je špecifikovaná druhým parametrom funkcie a ktorú daný handshake charakterizuje.
3. V zozname premennej TP, ktorá bola špecifikovaná prvým parametrom, sa inkrementuje hodnota na danom indexe, ktorý reprezentuje aplikáciu. Hodnota indexu je získaná prostredníctvom zoznamu `appList`.

Týmto spôsobom uchováva klasifikátor hodnoty výsledkov klasifikácie. Zo zoznamu TP je po ukončení klasifikácie možné vyčítať presný počet True Positive handshake-ov, a to pre každú aplikáciu individuálne. Na rovnaký systém funguje aj vyhodnotenie prvkov False Positive.

4.6.2 Funkcia fillListNegative()

Funkcia `fillListNegative()` slúži na zápis hodnôt získaných vyhodnotením handshake-ov, pre ktoré bol dosiahnutý výsledok True Negative alebo False Negative.

Táto funkcia sa od funkcie `fillListPositive()` líši v druhom parametri. Čo sa týka algoritmu zápisu potrebnej hodnoty, sú totožné. Zatiaľ čo predošlá funkcia využíva ako parameter názov aplikácie z hodnoty kľúča v trénovacom slovníku, funkcia `fillListNegative()` využíva samotný kľúč slovníka s množinou dát pre testovanie. Je to z toho dôvodu, pretože v prípade False Negative výsledku, kde klasifikátor pri vytváraní trénovacej množiny dát identifikoval určitý handshake ako `unknown`, ako výsledok sa používa aplikácia, ktorá tento fakt vyvracia. Naopak v prípade predošlej funkcie `fillListPositive()`, ak dospejeme k výslednej hodnote False Positive, táto hodnota reperzentuje aplikáciu, ktorá bola v trénovcej množine identifikovaná nesprávne a spôsobila False výsledok.

4.6.3 Využitie zoznamu trueCollision

V prípade vyhodnotenia prípadu, keď obidve množiny obsahujú totožné handshake-y, ktoré sú však označené ako charakteristické pre rozdielne aplikácie, sa do zoznamu zoznamov `trueCollision` zapíšu hodnoty reprezentujúce jednotlivé aplikácie. Zoznam `trueCollision` obsahuje 18 vnútorných zoznamov, každý z nich reprezentujúci jednu z 18 aplikácií, konkrétne jednu z aplikácií, ktorá sa na základe trénovacej sady určila ako charakteristická pre práve vyhodnocovaný handshake. Jednotlivé vnútorné zoznamy obsahujú hodnoty, ktoré reprezentujú aplikáciu, pre ktorú je tento handshake unikátnym v testovacej sade.

Ak sa napríklad v trénovacej množine nachádza handshake, ktorý je v tejto množine unikátny pre aplikáciu Chrome, čiže v abecednom poradí druhú aplikáciu (keďže používam číslovanie od 0, je ním index 1), a zároveň je tento handshake v testovacej množine unikátnym pre aplikáciu Facebook, ktorá je v poradí piatou aplikáciou (index 4), do zoznamu zoznamov `trueCollision` sa do vnútorného zoznamu s indexom 1 zapíše hodnota 4. Ak by v testovacej sade boli dva takéto handshake-y, hodnota 4 sa do tohto zoznamu uloží dvakrát.

Index aplikácie sa získava pomocou funkcie `getAppIndex()`, ktorá má ako vstupný parameter názov aplikácie a ktorá následne vracia index žiadanej aplikácie z globálneho zoznamu `appList`.

4.7 Znázornenie výstupu

Po získaní zoznamov o jednotlivých výsledných hodnotách klasifikácie pre každý testovaný handshake (TP, FP, TN, FN) prichádza znovu na rad rozšítenie `PySimpleGui`. Toto rozšírenie umožňuje vytvorenie ľubovoľného počtu okien, ktorých implementácia je rozdelená do viacerých funkcií. Každá funkcia implementuje práve jedno výstupné okno obsahujúce príslušné výsledky vo forme napríklad maticovej tabuľky. Podrobný obsah tabuliek je opísaný v sekcii 3.9. Jedná sa o funkcie:

- **`compactMatrix()`** - jej výstupom je matica obsahujúca hodnoty jednotlivých zoznamov TP, FP, TN a FN získaných klasifikátorom.
- **`simpleMatrix()`** - využíva súčty hodnôt každého zoznamu individuálne na znázornenie celkového počtu TP, FP, TN a FN.

- `extendedMatrix()` - využíva hodnoty všetkých zoznamov, vrátane zoznamu `trueCollision` na prehľadné znázornenie výsledkov klasifikácie. Túto formu znázornenia výsledkov je možné nájsť aj v článku o klasifikácii TLS dát [9].
- `APRWindow()` - obsahuje výpočty a výpis metrík Accuracy, Precision a Recall na základe hodnôt zoznamov.

Každá z uvedených funkcií má rovnaké vstupné parametre - štvoricu zoznamov TP, FP, TN a FN, ktoré umožňujú všetky možné výpočty a výstupy programu.

4.8 Znázornenie vstupu

Medzi funkcie definujúce okná aplikácie patrí aj funkcia `inputWindow()`, ktorá implementuje objekty nachádzajúce sa v úvodnom, vstupnom okne, ako sú tlačidlá, položky pre vstupný text, alebo položky checkbox a combobox slúžiace na výber vstupných hodnôt. Vstupné okno je opísané v sekcii 4.3.1.

4.9 Zápis výsledkov do súboru

Program ponúka takisto zápis a následné uloženie výsledkov do externého súboru. Túto funkcionalitu vykonáva funkcia `writeToFile()`, ktorá sa volá v prípade zvolenia tejto možnosti vo vstupnom rozhraní. Ich zápis prebieha rovnakým spôsobom, ako pri funkciách na generovanie výstupných matíc do jednotlivých okien opísaných v predchádzajúcej sekcii 4.7. Po spustení klasifikácie s touto zvolenou možnosťou sa užívateľovi objaví okno pre zvolenie adresára, názvu súboru a jeho formátu. Táto vlastnosť je naimplementovaná pomocou knižnice `Python Tkinter`. Predvolený názov a formát súboru sú `output.csv`. Do daného súboru sa uložia všetky 3 druhy matíc, ako aj hodnoty Accuracy, Precision a Recall. Pre maximálnu prehľadnosť v maticiach odporúčam daný súbor otvárať v tabuľkovom programe, napríklad MS Excel.

Kapitola 5

Experimenty

V tejto kapitole sa budem venovať experimentom vykonaným pomocou naimplementovaného klasifikátora. Čitateľ sa v nej oboznámi s viacerými možnými výsledkami vstupov a ich kombinácií. V tejto kapitole takisto opisujem a charakterizujem problémy pri identifikácii špecifických aplikácií a vysvetľujem dôvod týchto problémov. Tiež sa môže čitateľ dočítať o rozdieloch v niektorých implementačných riešeniach a o odôvodnení voľby viac prospešnej implementačnej metódy.

V experimentoch sa pre maximálnu prehľadnosť na ukážku používa rozšírená matica a výpis hodnôt APR (Accuracy, Precision, Recall), ktoré sú vysvetlené v sekcii 2.4.2. Príklad rozšírenej matice je zobrazený na obrázku č. 5.1.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
C	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	0
F	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	8	-	-	-	-	-	-	-	-	-	-	-	7
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	6	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	0
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	1
X	0	0	0	1	2	0	1	0	1	0	0	4	0	0	0	5	2	1	152

Obrázek 5.1: Ukážka rozšírenej matice

Stĺpce v tejto matici reprezentujú reálne hodnoty, zatiaľčo riadky reprezentujú aplikácie, ku ktorým bol daný handshake priradený klasifikátorom ako ju charakterizujúci. Aplikácie sú radené v abecednom poradí identifikované ako písmená A-R, konkrétne tak, ako sú zoskupené v úvode kapitoly 3.1. Stĺpec X reprezentuje handshake-y, ktoré neidentifikujú ani jednu z aplikácií. Podobne riadok X reprezentuje handshake-y, ktoré túto vlastnosť predikujú. Z tejto matice týmpádom vyplýva, že ak napríklad bunka [riadok B, stĺpec X] má hodnotu 1, znamená to, že jeden handshake, ktorý reálne neidentifikuje žiadnu aplikáciu, bol označený, že identifikuje aplikáciu Chrome. Toto je samozrejme hodnota False Positive. Naopak bunka [riadok X, stĺpec E], ktorá má v tomto prípade hodnotu 2 informuje, že existujú 2 handshake-y, ktoré sú charakteristické pre Facebook, avšak boli označené ako také, ktoré žiadnu aplikáciu neidentifikujú. Sú to False Negative hodnoty. Pri podrobnejšom pohľade na maticu si človek všimne, že stĺpec X reprezentuje hodnoty False Positive, riadok X hodnoty False Negative a diagonála reprezentuje hodnoty True, a to od bunky [A,A] do [R,R] hodnoty True Positive, zatiaľčo bunka [X,X] reprezentuje celkový počet hodnôt True Negative spomedzi všetkých aplikácií.

5.1 Porovnanie vyhodnotení s individuálnym využitím atribútov TLS

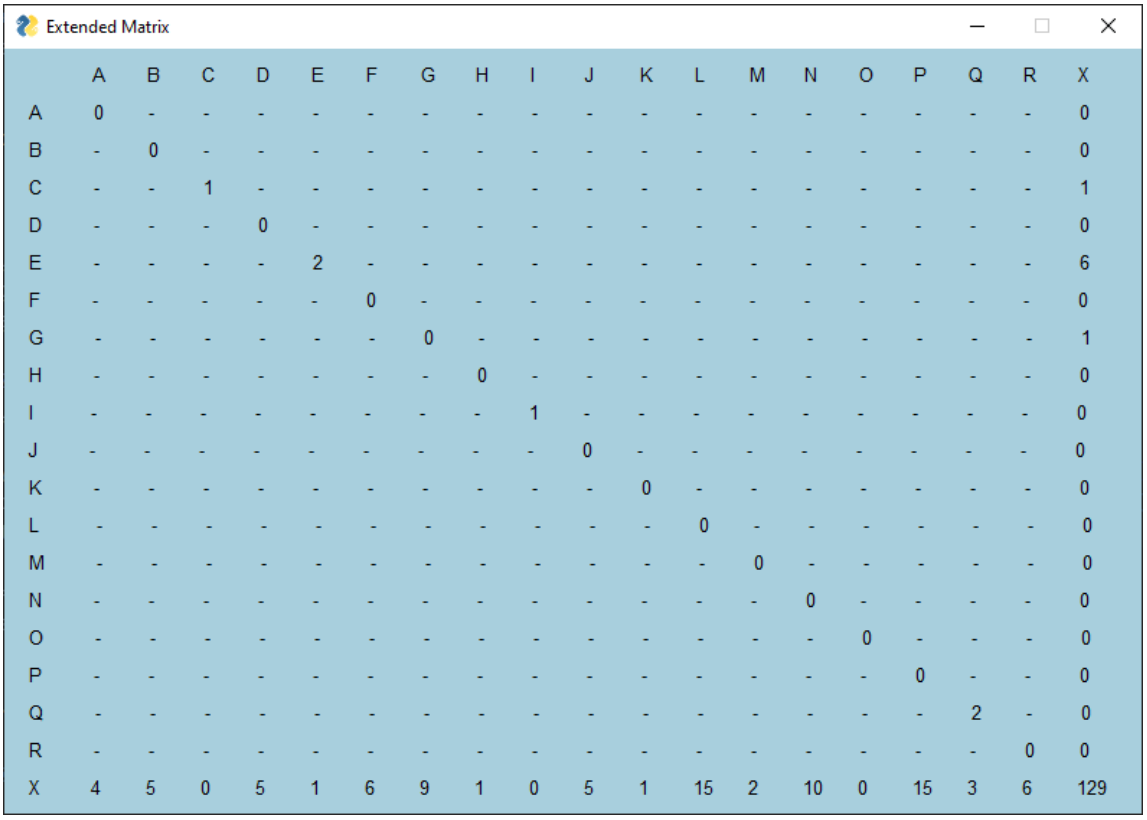
V tejto sekcii predvediem výsledky a ich porovnanie získané klasifikátorom na základe hodnoty JA3, kombináciou hodnôt JA3 + JA3S a kombináciou hodnôt JA3 + JA3S + SNI. Tak, ako v tejto sekcii, tak aj v celej experimentačnej kapitole budem ako prahovú hodnotu podobnosti využívať prioritne hodnotu 0.4. Jedná sa pre tieto experimenty o ideálnu hodnotu, viac o ideálnej prahovej hodnote v sekcii Prahová hodnota podobnosti 5.4 a v sekcii Vyhodnotenie experimentov 5.7.

5.1.1 Identifikácia na základe hodnoty JA3

Vstupné parametre prvej klasifikácie experimentu sú nastavené nasledovne:

- Trénovacia množina na základe dátových sád 2, 3, 4 a 5
- Testovacia množina na základe dátovej sady 1
- Zvolená hodnota JA3
- Prahová hodnota podobnosti nastavená na hodnotu 0.4

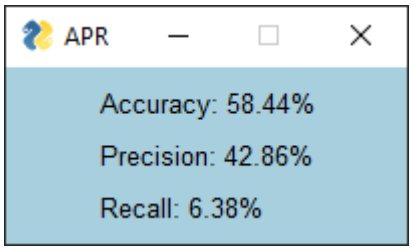
Výsledná rozšírená matica je znázorená na obrázku č. 5.2:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
D	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	6
F	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	1
H	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	0
K	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	0
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
X	4	5	0	5	1	6	9	1	0	5	1	15	2	10	0	15	3	6	129

Obrázek 5.2: Rozšírená matica - JA3

Ako je možné vidieť na ukážke, spomedzi všetkych aplikácií sa na základe hodnoty JA3 podarilo identifikovať iba štyri. Inými slovami, v tejto testovacej množine dát sa nachádzajú štyri aplikácie, ku ktorým bola priradená minimálne jedna hodnota JA3, ktorá je pre danú aplikáciu spomedzi všetkých dátových sád jedinečná. Rovnako si je možné všimnúť, že identifikácia na základe JA3 vedie k minimálnemu počtu hodnôt False Positive, čo je síce pozitívnou vlastnosťou, na druhej strane však obsahuje veľké množstvo False Negative prvkov, čo naznačuje, že JA3 hodnota sa o sebe jednoducho nepostačuje na priradenie týchto hodnôt k im príslušným aplikáciám.



Obrázek 5.3: Accuracy, Precision, Recall - JA3

APR metrika tiež nie je v tomto prípade veľmi pozitívna. Málo identifikovaných aplikácií drasticky znižuje hodnotu Recall.

5.1.2 Identifikácia na základe dvojice JA3 a JA3S

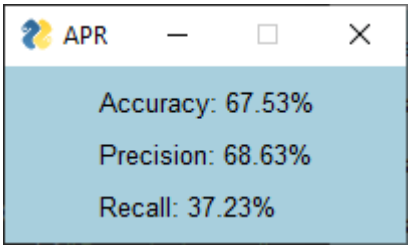
Nasledovný pokus o identifikáciu má totožné vstupné parametre od predošlého pokusu, s výnimkou toho, že sa používa kombinácia hodnôt JA3 a JA3S.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	5
F	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	4
G	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	1
H	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	2
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	2
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
X	0	5	0	4	1	5	9	1	1	4	0	4	0	2	0	14	3	6	121

Obrázek 5.4: Rozšírená matica - JA3 / JA3S

So zakomponovaním hodnoty JA3S k hodnote JA3 sa výsledky rapídne menia. V tomto prípade už máme 12 identifikovaných aplikácií na základe kombinácie týchto hodnôt. Keďže pridaný atribút zvyšuje danému handshake-u unikátnosť, je logické, že sa budú hodnoty pohybovať smerom hore. Toto platí vo väčšine prípadov, avšak nie je tomu tak vždy. Pri porovnaní tejto matice s predošlou si môžme všimnúť, že na diagonále identifikovaných aplikácií sa každý počet handshake-ov inkrementoval, až na jediný. Je ním aplikácia I, konkrétne Messenger. Ná základe samotnej hodnoty JA3 bolo možné túto aplikáciu identifikovať, no v kombinácii s JA3S to už neplatí z jedného dôvodu. Testovacia dátová sada 1 obsahuje handshake s unikátnou hodnotou JA3 pre aplikáciu Messenger. V trénovacej sade sa táto hodnota takisto našla ako unikátna pre Messenger. Žiaden handshake charakterizujúci inú aplikáciu túto hodnotu JA3 neobsahuje. Avšak v prípade, že prevedieme tento istý výpočet pre dvojicu hodnôt JA3-JA3S, klasifikátor zistí, že daná JA3S hodnota predošlo identifikovaného handshake-u sa v trénovacej sade nenachádza. Vyskytla sa iba raz, a to v testovacej sade. Týmpádom, že sa vyskytla pri aplikácii iba jednorázovo a v trénovacej sade sa nevyskytuje, klasifikátor sa ho nemôže naučiť a tak ani identifikovať.



Obrázek 5.5: Accuracy, Precision, Recall - JA3 / JA3S

V prípade použití dvojice hodnôt JA3 a JA3S je oproti použití samostatnej hodnoty JA3 možné spatriť mierny nárast metrík presnosti.

5.1.3 Identifikácia na základe trojice JA3, JA3S a SNI

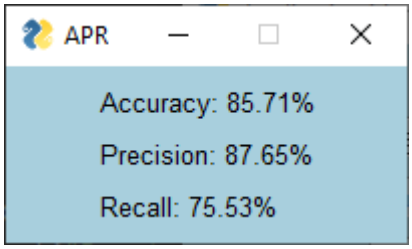
S následným pridaním hodnoty SNI nám program vygeneruje maticu, ktorá je zobrazená na obrázku 5.6.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	0
F	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	4
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	1
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	0
X	0	2	0	1	2	1	3	0	1	0	0	4	0	0	0	5	2	2	127

Obrázek 5.6: Rozšírená matica - JA3 / JA3S / SNI

Pri využití všetkých troch hodnôt dospejeme k maximálnej unikátnosti jednotlivých odchytených handshake-ov. Jednoznačne sa podarilo identifikovať 16 aplikácií. Takisto si môžme všimnúť, že jednotlivé hodnoty False Positive a False Negative sa blížia k minimu.



Obrázek 5.7: Accuracy, Precision, Recall - JA3 / JA3S / SNI

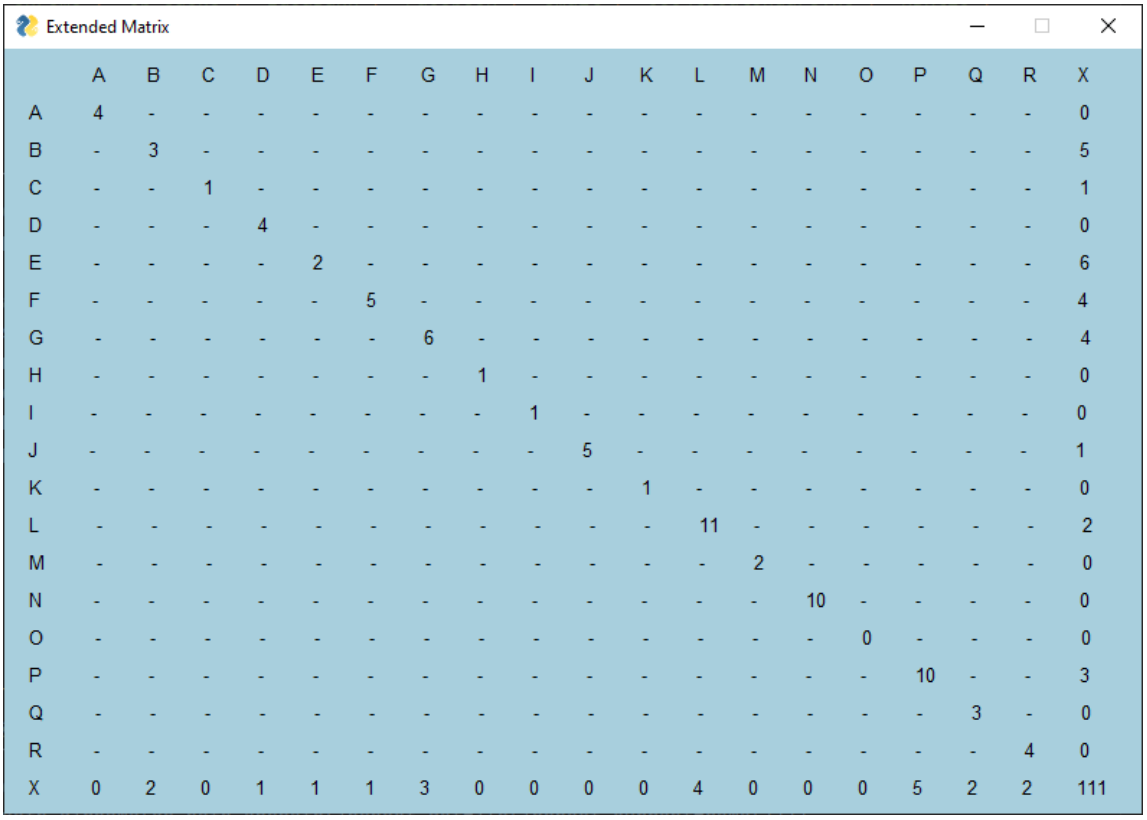
Vysokú úspešnosť a presnosť identifikácie charakterizujú aj hodnoty Accuracy, Precision a Recall.

5.2 Hierarchické vyhodnotenie

Hierarchické vyhodnotenie ako bolo spomenuté v sekcii 3.6, spočíva vo vyhodnocovaní na základe postupného pridávania atribútov TLS. Vstupné hodnoty klasifikátora v tomto experimente sú nasledovné:

- Trénovacia množina na základe dátových sád 2, 3, 4 a 5
- Testovacia množina na základe dátovej sady 1
- Zvolená možnosť hierarchického vyhodnotenia
- Prahová hodnota podobnosti nastavená na hodnotu 0.4

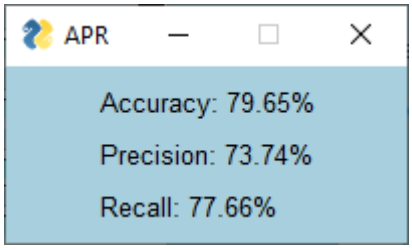
Parametre sa oproti sekcii 5.1 líšia znovu iba vo výbere atribútu TLS. Množiny a definovaná prahová hodnota zostávajú totožné. Rozšírenú maticu pre tento vstup zobrazuje obrázok č. 5.8.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	6
F	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	4
G	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	4
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	2
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	3
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	0
X	0	2	0	1	1	1	3	0	0	0	0	4	0	0	0	5	2	2	111

Obrázek 5.8: Rozšírená matica - Hierarchické vyhodnotenie

Na vygenerovanej matici je možné vidieť, že vďaka hiearchickému vyhodnoteniu sa podarilo úspešne identifikovať každú aplikáciu okrem Telegramu, ktorú identifikovať kvôli jeho nulovému počtu kľúčových slov nie je možné. Je možné si všimnúť, že sa úspešne podarilo identifikovať aj aplikáciu Messenger, a to práve vďaka výhode tohto spôsobu vyhodnotenia, keďže sa táto aplikácia identifikuje na základe samotnej JA3 hodnoty.



Obrázek 5.9: Accuracy, Precision, Recall - Hierarchické vyhodnotenie

Pri tomto typu vyhodnotenia sa jemne znížili hodnoty Accuracy a Precision. Tento jav je zapríčinený tým, že vyhodnotenia na základe hodnoty JA3 a dvojice hodnôt JA3 a JA3S majú tendenciu obsahovať vyšší počet hodnôt False Positive, čo je možné si všimnúť aj v predchádzajúcich príkladoch. V konečnom dôsledku je síce hierarchické vyhodnotenie oproti vyhodnotení trojicou JA3, JA3S a SNI jemne menej presným vyhodnotením, faktom však je, že má najvyššiu úspešnosť čo sa týka identifikácie aplikácií, čo je hlavným faktorom tejto práce.

5.3 Krížová validácia

Tento typ vyhodnotenia spočíva v prebehnutí piatich rozdielnych vyhodnotení, v každom prípade to je s použitím jednej dátovej sady ako testovacej a štyroch dátových sád ako trénovacích. Keďže máme k dispozícii 5 dátových sád, prebehne 5 vyhodnotení a ich výsledky sa pri ich konečnej reprezentácii sčítajú. Vstupné hodnoty sú pre tento experiment nasledovné:

- Zvolená možnosť krížovej validáie
- Identifikácia na základe trojice JA3 - JA3S - SNI
- Prahová hodnota podobnosti nastavená na hodnotu 0.4

Týmto vstupom program vygeneruje maticu zobrazenú na obrázku č. 5.10.

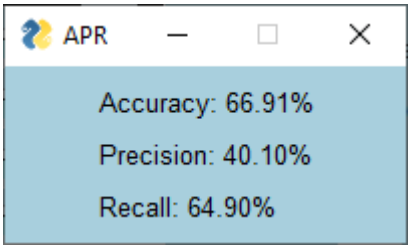
Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8
B	-	20	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	2	31
C	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	20	-	-	-	-	-	-	-	-	-	-	-	-	-	196
F	-	-	-	-	-	16	1	-	-	-	-	-	-	-	-	-	-	-	0
G	-	3	-	-	-	-	33	-	-	-	-	-	-	-	-	-	-	2	106
H	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	18	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	41	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	6	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	34	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	49	-	-	61
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	1
R	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	14	71
X	2	28	0	6	3	11	40	0	2	1	0	14	4	2	0	20	4	35	990

Obrázek 5.10: Rozšírená matica - Krížová validácia

Tento spôsob vyhodnotenia bol naimplementovaný čisto z experimentačných dôvodov, na získanie nového pohľadu na vyhodnotenie. Ako je možné vidieť na matici, tento experiment neprináša vyššiu kvalitu pri vyhodnotení, práve naopak, obsahuje veľké množstvo negatívnych čísiel. Jej pozitívum leží inde. Krížová validácia poskytuje široký prehľad o problémových aplikáciách. Na obrázku je možné vidieť, že klasifikátor má v istých prípadoch tendenciu identifikovať nejednoznačné handshake-y ako príslušné k istým aplikáciám, konkrétne sú to aplikáce B (Chrome), E (Facebook), G (GoogleCalendarSync), P (TikTok) a R (YouTube). Takisto je na prvý pohľad vidieť určitý počet prípadov kolízií pravdy, a nie

je náhodou, že v drvivej väčšine prípadov nastávajú práve pri aplikáciách, pri ktorých sa objavuje aj vysoký počet False hodnôt. Tieto negatívne čísla sú zapríčinené problémovými handshake-mi v istých aplikáciách, ktoré síce súvisia s aplikáciami, ku ktorým ich klasifikátor priradil, avšak sa k nim pristupuje z iných, rozdielnych aplikácií a týmpádom sú reálne nejednoznačné. Veľký dopad na výsledok má aj fakt, ktorú z dátových sád zvolíme ako testovaciu. Dátové sady 2, 3 a 4 obsahujú výrazne väčšie množstvo handshake-ov, ktoré odkazujú na istú aplikáciu práve prostredníctvom niektorej z iných aplikácií. Z tohto dôvodu pri výbere týchto dátových sád ako testovacích sa zvýši počet False Positive prvkov pri problémových aplikáciách, čo má za následok radikálne zníženie presnosti klasifikácie, čo sa výrazne odzrkadlí na hodnotách APR, ktorých hodnoty sú pre tento prípad znázornené na obrázku č. 5.11.



Obrázek 5.11: Accuracy, Precision, Recall - Krížová validácia

Ako je možné vidieť na hodnotách metrík klasifikačnej presnosti, krížová validácia nedisponuje presnou klasifikáciou, a to z toho dôvodu, že jednotlivé dátové sady nie sú jednotné. Dátové sady 2, 3 a 4 individuálne obsahujú v priemere dvojnásobný počet handshake-ov ako dátové sady 1 a 5 a väčšina z nich sú práve handshake-y tohto zavádzajúceho typu, ktoré komplikujú testovanie. Preto sú tieto dátové sady ideálne pre trénovanie, aj kvôli celkovému vyššiemu počtu handshake-ov. Naopak dátové sady 1 a 5 sú ideálnymi sadami pre vytvorenie testovacej množiny, pretože obsahujú minimum handshake-ov so zavádzajúcimi údajmi a týmpádom sa použitím týchto sád zvýši kvalita a pravdepodobnosť úspešnej klasifikácie.

5.4 Prahová hodnota podobnosti

Dôležitou súčasťou úspešnej identifikácie aplikácií je zvolenie vhodnej prahovej hodnoty podobnosti, na základe ktorej sa následne filtrujú hodnoty, ktoré nie sú pre danú aplikáciu charakteristické. Keďže jednotlivé kľúčové slová a SNI, na základe ktorých sa hodnota podobnosti generuje, sa pri každej aplikácii líšia, takisto sa medzi týmito aplikáciami líši hodnota, od ktorej je možné tvrdiť, že handshake charakterizuje danú aplikáciu. Po prevedení viacerých experimentov som však dospel k záveru, že hodnota podobnosti, ktorá sa začína pri určitých zhodách pri aplikáciách objavovať, je väčšia ako 0.4. Pri niektorých aplikáciách môžme dosiahnuť aj vyššie hodnoty, čo je jednoducho zapríčinené vyššou podobnosťou kľúčového slova s doménovým menom prístupového servera, čiže SNI.

Ako príklad uvediem výsledok nasledovného vyhodnotenia:

- Trénovacia množina na základe dátových sád 1, 2, 3, a 4
- Testovacia množina na základe dátovej sady 5
- Zvolená trojica hodnôt JA3 - JA3S - SNI

- Prahová hodnota podobnosti nastavená na hodnotu 0.4

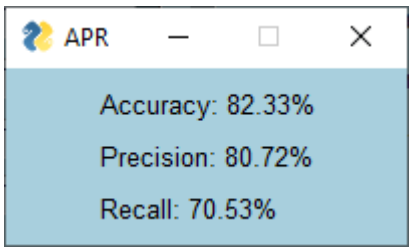
Výsledná matica je na obrázku č. 5.12.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	4
F	-	-	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	4
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	0
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	8	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	12	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	1
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6	3
X	0	1	0	1	1	2	4	0	0	1	0	1	3	2	0	5	1	6	138

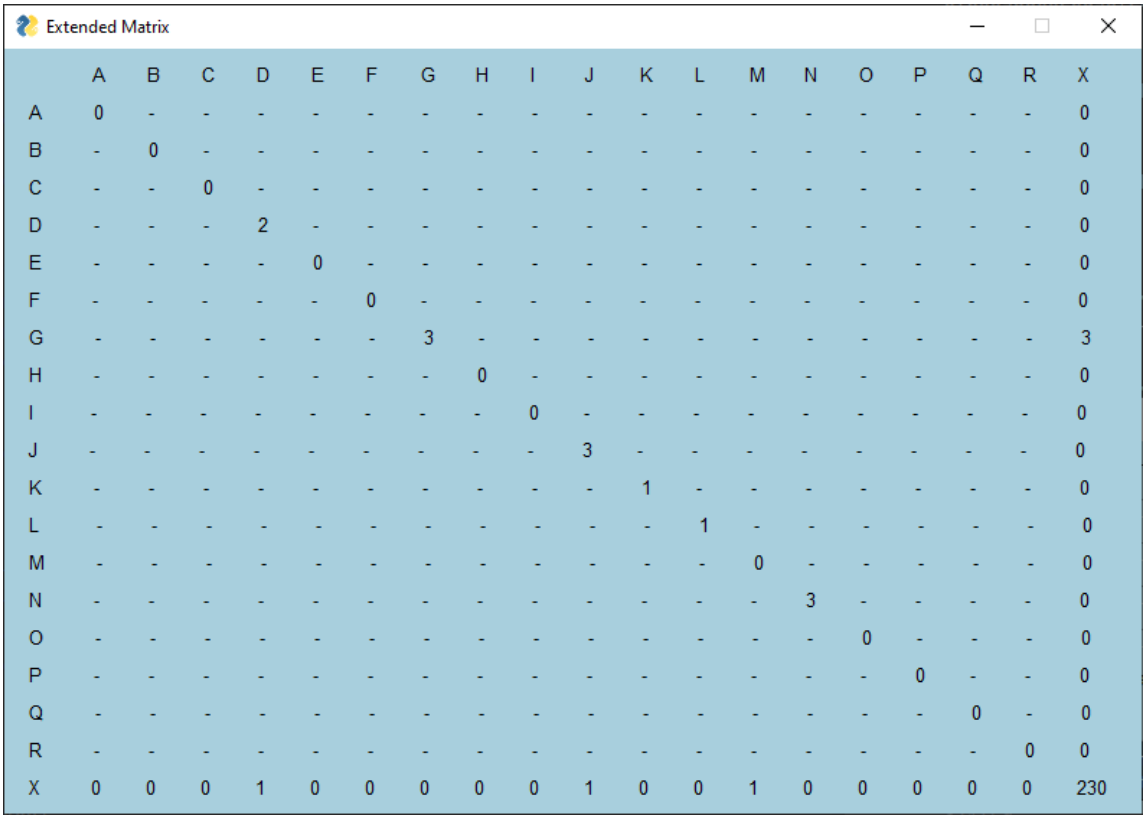
Obrázek 5.12: Rozšírená matica - Prahová hodnota 0.4

Ako je možné vidieť na obrázku, prahová hodnota nastavená na hodnotu 0.4 pri tomto vstupe umožňuje kvalitnú identifikáciu aplikácií, na druhej strane však obsahuje istý počet False hodnôt, ktoré znižujú celkovú hodnotu metrík, ktoré znázorňujú presnosť a týmpádom kvalitu klasifikácie.



Obrázek 5.13: Accuracy, Precision, Recall - Prahová hodnota 0.4

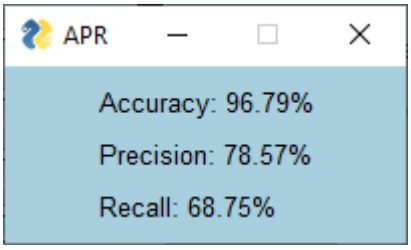
Zmena prahovej hodnoty na vysoké číslo zapríčiní to, že klasifikátor začne jednotlivé handshake-y filtrovať oveľa prísnejšou mierou. Nasledovná matica na obrázku č. 5.14 je výsledkom rovnakých vstupných parametrov, až na rozdiel, že je prahová hodnota nastavená na hodnotu 0.7:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
C	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	0
F	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	3
H	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	0
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	0
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
X	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	230

Obrázek 5.14: Rozšírená matica - Prahová hodnota 0.7

Pri tomto príklade si je na prvý pohľad možné všimnúť, že sa znížil počet identifikovaných aplikácií. Dobrou stránkou však je, že sa vďaka vysokej prahovej hodnote takmer všetky False hodnoty anulovali. Tým, že sa toto stane, sa nám podarí výrazne zvýšiť presnosť klasifikácie čo sa týka metriky Accuracy, ktorú znázorňuje APR tabuľka na obrázku č. 5.15.



Obrázek 5.15: Accuracy, Precision, Recall - Prahová hodnota 0.7

Toto je spôsobené tým, že isté handshake-y môžu mať hodnotu podobnosti charakterizujúcu zhodu vyššiu, ako ostatné, a pri voľbe nízkej hodnoty sa do vyhodnotenia môžu dostať handshake-y, ktoré nie sú pre aplikáciu charakteristické, keďže spravidla platí - čím menšie nastavená prahová hodnota, tým viac hodnôt k vyhodnoteniu. Zvyčajne preto platí, že čím vyššie prahovú hodnotu nastavíme, tým je klasifikácia na základe nej presnejšia, avšak sa nám zníži počet True Positive prvkov, čiže s nimi spravidla aj počet identifikovaných aplikácií. Veľký dopad na hodnoty APR má aj zvolená kombinácia dátových sád.

Nasledovný experiment znázorňuje rozdiel vo výsledkoch pri regulovaní prahovej hodnoty pri zachovaných zvyšných vstupných hodnotách. Vstupné hodnoty sú nasledovné:

- Trénovacia množina na základe dátových sád 2, 3, 4 a 5
- Testovacia množina na základe dátovej sady 1
- Zvolená trojica hodnôt JA3 - JA3S - SNI

Tabuľka č. 5.1 znázorňuje výsledky získané týmto druhom vstupu pri rozdielnej prahovej hodnote. Jej konštrukcia je nasledovná:

- Prvý stĺpec reprezentuje zadanú prahovú hodnotu podobnosti.
- Druhý stĺpec reprezentuje metriku Accuracy.
- Tretí stĺpec reprezentuje metriku Precision.
- Štvrtý stĺpec reprezentuje metriku Recall.
- Piaty stĺpec reprezentuje počet identifikovaných aplikácií.
- Šiesty stĺpec reprezentuje percentuálne vyjadrenie počtu identifikovaných aplikácií (keďže z 18 poskytnutých aplikácií aplikáciu Telegram ako jedinú nie je možné identifikovať, v tomto príklade 100% aplikácií reprezentuje hodnotu všetkých identifikovateľných aplikácií. Dôvod, pre ktorý aplikáciu Telegram nie je možné identifikovať, je vysvetlený v sekcii 5.5.4).

Podobnosť	Accuracy	Precision	Recall	Id. ap.	Id. ap. (%)
0.1	55.36%	92.93%	49.73%	17	100%
0.2	68.3%	85.71%	57.35%	16	94%
0.3	78.17%	82%	71.93%	16	94%
0.4	85.71%	87.65%	75.53%	16	94%
0.5	88.31%	83.87%	75.36%	13	76%
0.6	91.77%	78.26%	80%	10	58%
0.7	96.97%	72.22%	86.67%	6	35%
0.8	99.57%	75%	100%	1	6%
0.9	0%	0%	0%	0	0%

Tabulka 5.1: Hodnoty Accuracy, Precision, Recall a identifikované aplikácie pri rozdielnej prahovej hodnote podobnosti

Na ukážke je vidno, že s rastúcou prahovou hodnotou podobnosti sa hodnoty metrík APR v priemere zvyšujú, avšak počet identifikovaných aplikácií sa znižuje. Tiež je možné si všimnúť, že pri prahovej hodnote podobnosti definovanej ako 0.1 sa podarí identifikovať všetky aplikácie, ale po jej zvýšení tomu tak už nie je. Toto je zapríčinené tým, že nastavenie prahovej hodnoty na príliš nízku hodnotu môže viesť k tomu, že sa ako charakteristické handshake-y môžu označiť aj tie, ktoré vo svojom SNI kľúčové slovo v skutočnosti nemajú, avšak tam je istá, minimálna podobnosť, ktorá sa vyhodnotí ako tesne nad napríklad danú hodnotu 0.1. Následne daný handshake môže byť aj napriek nepresnému kľúčovému slovu pre danú aplikáciu unikátnym v rámci všetkých dátových sád, a tým ho klasifikátor vyhodnotí ako identifikujúci aplikáciu, čo však v skutočnosti tak nie je. Je to ukážka toho, ako príliš nízko definované prahové hodnoty podobosti nie sú z tohto dôvodu spoľahlivé. Naopak pri určení príliš vysokej prahovej hodnote ako napríklad 0.9 ľahko nastane situácia, keď žiadny handshake nebude označený ako charakteristický. V tomto prípade sa všetky hodnoty automaticky nastavia na hodnotu 0.

5.5 Problémové aplikácie

Spomedzi 18 aplikácií k dispozícii pre identifikáciu sa niekoľko z nich počas implementácie javilo ako problémové. Týmto aplikáciám som dedikoval túto sekciu, v ktorej opíšem, o aké problémy sa jedná a v čom sa líšia od ostatných aplikácií. Týmito aplikáciami sú:

- Chrome
- Facebook
- GoogleCalendarSync
- Telegram
- TikTok
- YouTube

5.5.1 Chrome

Google Chrome ako aplikácia vyniká zo všetkých tým, že ako jediná je internetovým prehliadačom. Toto spôsobuje problém, že pre túto aplikáciu neexistuje špecifické kľúčové slovo, ktoré by bolo možné nájsť v jej doménovom mene. Ako kľúčové slovo som použil jednoducho reťazec "Google", ktorý síce jednoznačne neidentifikuje aplikáciu Chrome, hlavne, ak sa medzi ostatnými nachádzajú aplikácie ako Gmail alebo GoogleCalendarSync, avšak aspoň poskytuje filtrovanie na nejakej úrovni, a umožní nájsť jednoznačné handshake-y spomedzi všetkých dátových sád práve pre túto aplikáciu. Všeobecné kľúčové slovo ako "Google"môže byť samozrejme spojené aj s viacerými aplikáciami, čo zvýši počet False prvkov pre túto aplikáciu.

5.5.2 Facebook

Facebook je jedna z aplikácií, ktoré majú vysokú tendenciu vyskytovať sa v iných aplikáciách vo forme viacerých druhov odkazov. V týchto prípadoch SNI daného handshake-u síce obsahuje kľúčové slovo pre Facebook, vyskytuje sa však v rozdielnych aplikáciách ako

napríklad YouTube, GoogleCalendarSync alebo Whatsapp, vrátane Facebooku samotného. Takéto typy handshake-ov sa v trénovacej sade podarí označiť ako unknown. Avšak, handshake s rovnakými hodnotami (JA3, JA3S, SNI) sa môže pritom v testovacej sade javiť ako unikátny, a to práve pre aplikáciu Facebook. V týchto prípadoch sa handshake vyhodnotí ako False Positive. Táto negatívna vlastnosť sa vo výsledkoch odzkrkadľuje hlavne v prípadoch, ak sa použije ako testovacia dátová sada práve sada 2, 3 alebo 4.

5.5.3 GoogleCalendarSync

Pri tejto aplikácii sa jedná o rovnakú situáciu ako v prípade aplikácie Facebook. Handshakey charakteristické pre GoogleCalendarSync majú tendenciu objavovať sa vo viacerých aplikáciách, ktoré môžu vyžadovať synchronizáciu kalendára a podobné služby. Väčší počet False hodnôt sa dá všimnúť v rovnakej situácii ako pri aplikácii Facebook, čiže pri zvolení testovacích sád 2, 3 alebo 4.

5.5.4 Telegram

Chatovacia aplikácia známa vďaka svojej nevystopovateľnosti s enkryptovanou komunikáciou ako jediná nemá v tomto projekte definované žiadne kľúčové slová a to z jednoduchého dôvodu - žiadne neexistujú. Tým, že pre ňu nie sú zadefinované žiadne kľúčové slová, klasifikátor každý handshake spadajúci pod túto aplikáciu vyhodnotí ako nešpecifikovaný. Týmpádom úspešnou klasifikáciou pre túto aplikáciu sa rozumie tá, ak sa každý jej handshake zaradí do kategórie True Negative. Inými slovami, aplikácia je neidentifikovateľná a po úspešnej klasifikácii v rozšírenej matici sa v stĺpci aj riadku pre aplikáciu Telegram nachádza vždy hodnota 0.

5.5.5 TikTok

Aplikácia sociálnej siete TikTok sa podobne ako väčšina problémových aplikácií takisto vyskytuje vo viacerých aplikáciách ako napríklad Whatsapp. Rovnako aj v aplikácii TikTok sa často pristupuje na aplikácie YouTube alebo Facebook, čo znamená, že aj pri tejto aplikácii sa dajú jednoducho objaviť False Positive, resp. False Negative výsledky.

5.5.6 YouTube

Obľúbená aplikácia od spoločnosti Google je takisto úzko spájaná s aplikáciami Facebook a ostatnými Google službami, čo mimo úspešnej identifikácie znovu zapríčiní aj vysoký počet False prvkov.

5.6 Použitie prahovej hodnoty podobnosti pri trénovaní

Prahová hodnota podobnosti je základným a jediným faktorom rozhodovania v testovacej množine o tom, či je daný handshake pre aplikáciu charakteristický, alebo nie. Tento spôsob označovania vznikol takisto kvôli testovacej množine. Prahová hodnota sa však využíva aj pri vytváraní trénovacej množiny, ešte predtým, ako v trénovacej množine prebehne kontrola unikátnosti jednotlivých handshake-ov medzi aplikáciami, na základe čoho sa výsledky v trénovacej sade následne označujú. Spôsob označovania a vyhodnotenia handshake-ov a následné vytvorenie slovníkov s množinami je podrobne opísaný v implementačnej sekcii 4.4.

Pôvodne som vypočítanú hodnotu podobnosti do procesu vytvárania trénovacej množiny nemal v pláne využiť, ale po získavaní menej priaznivých klasifikačných výsledkov som sa rozhodol zmeniť názor. Ako príklad uvádzam klasifikačný experiment s nasledovnými vstupnými parametrami:

- Trénovacia množina na základe dátových sád 2, 3, 4 a 5
- Testovacia množina na základe dátovej sady 1
- Klasifikácia na základe trojica JA3 - JA3S - SNI
- Prahová hodnota podobnosti nastavená na hodnotu 0.4
- **Trénovacia množina bez využitia hodnoty podobnosti**

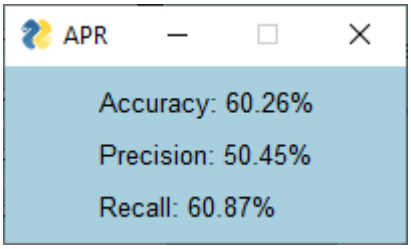
Rozšírená matica pre tento príklad je na obrázku 5.16.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	7
B	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	20
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1
E	-	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-	-	-	2
F	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	5
G	-	-	-	-	-	-	3	-	-	-	-	-	-	-	-	-	-	-	3
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	1	-	-	-	0	-	-	-	-	-	-	-	-	-	2
J	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	1	-	-	2
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	2
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	-	-	6
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5	-	-	2
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	1
X	2	4	0	1	2	1	6	0	1	0	0	4	0	0	0	9	3	3	82

Obrázek 5.16: Rozšírená matica - Trénovanie bez prahovej hodnoty

Je možné vidieť, že klasifikácia dopadla pomerne úspešne. Spomedzi 17 identifikovateľných aplikácií (nerátajúc Telegram) sa podarilo identifikovať 15. Môžme si však všimnúť dva prípady kolízie pravdy a takmer každá z aplikácií so sebou prináša aj určitý počet False hodnôt, ktoré znižujú hodnoty APR, čiže celkovú presnosť klasifikácie:



Obrázek 5.17: Accuracy, Precision, Recall - Trénovanie bez prahovej hodnoty

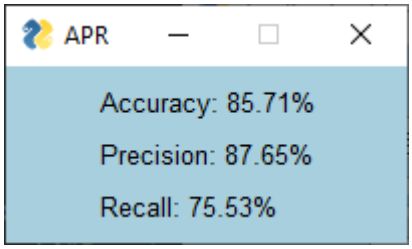
Po naimplementovaní zahadzovania trénovacích handshake-ov pod prahovou hodnotou podobnosti som za rovnakých okolností dosiahol výsledok zobrazený na obrázku 5.18.

Extended Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	X
A	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
B	-	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4
C	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
D	-	-	-	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0
E	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	0
F	-	-	-	-	-	5	-	-	-	-	-	-	-	-	-	-	-	-	0
G	-	-	-	-	-	-	6	-	-	-	-	-	-	-	-	-	-	-	4
H	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	-	-	-	0
I	-	-	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-	-	0
J	-	-	-	-	-	-	-	-	-	5	-	-	-	-	-	-	-	-	1
K	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-	0
L	-	-	-	-	-	-	-	-	-	-	-	11	-	-	-	-	-	-	0
M	-	-	-	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	0
N	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	-	-	0
O	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	-	-	-	0
P	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	10	-	-	1
Q	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3	-	0
R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4	0
X	0	2	0	1	2	1	3	0	1	0	0	4	0	0	0	5	2	2	127

Obrázek 5.18: Rozšírená matica - Trénovanie s prahovou hodnotou

Výsledky s touto pridanou funkcionalitou sú neporovnatelne priaznivejšie. Pri porovnaní s predošlou tabuľkou si je možné všimnúť, že sme na True Positive diagonále neprišli ani o jeden charakteristický handshake, priam naopak, hodnoty True Positive sa na takmer každej aplikácií inkrementovali, dokonca sa podarilo identifikovať aj o jednu aplikáciu navyše. Okrem toho sa klasifikátoru podarilo obísť hodnoty kolízie pravdy a takisto sa výrazne znížil počet False Positive, ako aj False Negative hodnôt. Výsledok sa najviac odzrkadlí na metrikách presnosti - Accuracy, Precision a Recall, ktoré sú pre tento prípad zobrazené na obrázku č. 5.19.



Obrázek 5.19: Accuracy, Precision, Recall - Trénovanie s prahovou hodnotou

Využitie prahovej hodnoty podobnosti pri vytváraní trénovacej množiny nie je k úspešnej identifikácii aplikácií nutné, avšak odfiltrovávaním zbytočných handshake-ov v trénovacej množine prináša výhodu vo forme oveľa presnejších a tým pádom kvalitnejších klasifikačných výsledkov.

5.7 Vyhodnotenie experimentov

Experimenty potvrdili niekoľko predpokladaných výsledkov:

- Použitie jedinej hodnoty JA3 alebo kombinácie hodnôt JA3 a JA3S môže postačiť k identifikácii určitého počtu aplikácií, avšak na úkor vyššieho počtu nepresne vyhodnotených hodnôt vo forme False výsledkov a hodnôt kolízie pravdy. S pridaním SNI k týmto hodnotám sa však presnosť klasifikácie zvýši a počet nepresne vyhodnotených hodnôt sa zníži na minimálny.
- Pri identifikácii na základe samotnej hodnoty JA3 je možné v istom prípade aplikáciu Messenger identifikovať, avšak pri kombinácii tejto hodnoty s inými už tomu tak nie je.
- Hierarchické vyhodnotenie poskytuje najvyšší počet pozitívnych výsledných hodnôt, keďže jeho funkcionalita v sebe zahŕňa aj práve identifikáciu na základe samotnej hodnoty JA3, ktorá má výhodu napríklad pri identifikácii aplikácie Messenger. Táto funkcionalita však so sebou prináša aj nevýhody, čím je vyšší počet nepresných vyhodnotení, to jest False výsledkov, čo je zapríčinené práve vyhodnotením bez hodnoty SNI.
- Krížová validácia slúži ako experimentačný spôsob vyhodnotenia, jej zmysel je hlavne v získaní celkového prehľadu o napríklad problémových aplikáciách, pri ktorých tento druh vyhodnotenia odhalí vysoké množstvo negatívnych čisiel. Takisto však tento druh vyhodnotenia môže byť použitý na identifikáciu, avšak celková presnosť klasifikácie je pri tomto druhu vyhodnotení veľmi nízka.
- Prahová hodnota podobnosti hrá veľkú rolu vo výsledkoch vyhodnotenia. Vo väčšine prípadov platí miera - čím vyššie definovaná prahová hodnota podobnosti, tým presnejšia klasifikácia, no aj menej identifikovaných aplikácií. Ako ideálne zvolenú mieru prahovej hodnoty na základe testovania a vykonaných experimentov k dosiahnutiu maximálneho počtu pozitívnych hodnôt a zároveň udržania akceptovanej presnosti hodnotím interval <0.3 , 0.5>.

- Využitie prahovej hodnoty podobnosti pri trénovaní je dôležitou súčasťou procesu vytvárania trénovacej množiny, keďže sa na základe nej filtrujú zbytočné hodnoty, ktoré spôsobojú negatívne výsledky a nemajú dopad na získanie výsledkov pozitívnych.
- Problémovosť niektorých aplikácií, čím mám na mysli väčší počet negatívnych hodnôt, je zapríčinená ich vlastnou funkcionalitou. Užívateľ síce môže používať rozdielnu aplikáciu, avšak daná aplikácia môže využívať napríklad služby aplikácie Facebook pre prihlasovanie, čo zapríčiní, že sa handshake-y patriace k jednej aplikácie objavia u druhej. Tieto negatívne hodnoty majú však informačnú vlastnosť. Vďaka nim vieme, že síce na danom zariadení aplikácia beží, napriek tomu sa mimo nej komunikuje s rozdielnymi servermi.

Kapitola 6

Záver

Cieľom tejto bakalárskej práce bolo implementovať klasifikátor, ktorý sa na základe odchytených dát TLS protokololu uložených v dátových sadách pokúsi identifikovať mobilné aplikácie. Vytvoril som užívateľskú aplikáciu, ktorá umožňuje definovanie vstupných hodnôt pre klasifikáciu, čo zahŕňa voľbu dátových sád alebo výber z viacerých druhov vyhodnotenia. Aplikácia umožňuje skúmanie použiteľnosti hodnôt TLS protokolu JA3, JA3S a SNI vzhľadom na identifikáciu aplikácií, ktoré tieto protokol k zabezpečeniu dát používajú. Výstupné hodnoty klasifikátora aplikácia umožňuje reprezentovať dvomi rôznymi spôsobmi, a to buď v rozhraní samotnej aplikácie alebo pri voľbe exportu výsledkov aj v externom súbore vo zvolenom formáte.

Klasifikácia stavia na kľúčových slovách aplikácií charakterizujúcich doménové meno ich servera. Tie bolo nutné si dohľadať. Úspešne sa podarilo identifikovať 17 aplikácií z celkovo 18 poskytnutých. Aplikáciu Telegram totiž nebolo možné idenfitikovať práve z toho dôvodu, že som nemal prístup k žiadnemu kľúčovému slovu, ktoré by túto aplikáciu charakterizovalo. Podrobnejšie sú výsledky prevedených experimentov zhrnuté v sekcii 5.7.

Z osobného hľadiska som vďaka práci získal nové skúsenosti s Data mining-om a klasifikáciou. Takisto mi práca umožnila rozšíriť svoje schopnosti s programovacím jazykom `Python`, keďže veľa jeho schopností som prvýkrát využil práve v implementovaní tejto práce.

Literatura

[1] *Eliptická krivka*. [Online]. Dostupné z: https://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/.

[2] *Druhy klasifikácie*. [Online]. Dostupné z: https://cs.wikipedia.org/wiki/Klasifikace_(um%C4%9Bl%C3%A1_inteligence).

[3] *JA3 hash*. [Online]. Dostupné z: https://www.defensive-security.com/blog/hiding-behind-ja3-hash.

[4] *Dokumentácia jazyka Python*. [Online]. Dostupné z: https://docs.python.org/3.5/library/.

[5] *Klasifikácia a klasifikačné pravidlo*. [Online]. Dostupné z: https://en.wikipedia.org/wiki/Classification_rule.

[6] *ISO/OSI - ISO 7498*. [Online]. Dostupné z: https://www.iso.org/standard/20269.html.

[7] Matoušek P.. *Síťové aplikace a jejich architektura*. VUTIUM, 2014. ISBN 978-80-214-3766-1.

[8] Matoušek P.. *ISA: Zabezpečení počítačové komunikace*. Vysoké učění technické v Brně, fakulta informačních technologií, Október 2020.

[9] Matoušek P. Burgetová I. Ryšavý O. Victor M.. *On Reliability of JA3 Hashes for Fingerprinting Mobile Applications* [online]. 2021. Dostupné z: https://doi.org/10.1007/978-3-030-68734-2_1.

[10] *Online zdroj kľúčových slov určitých aplikácií*. [Online]. Dostupné z: https://www.netify.ai/resources/applications.

[11] *Manuál rozšírenia PySimpleGUI*. [Online]. Dostupné z: https://pysimplegui.readthedocs.io/en/latest/.

[12] *Podrobný opis fungovania TLS handshake-u*. [Online]. Dostupné z: https://www.thesslstore.com/blog/explaining-ssl-handshake/#key-exchange.

[13] *Accuracy, Precision, Recall*. [Online]. Dostupné z: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

Příloha A

Obsah priloženého pamäťového média

Priložené CD obsahuje nasledovné súbory:

- **doc** - adresár obsahujúci zdrojový tvar písomnej správy
- **project** - adresár obsahujúci súbor so zdrojovoým skriptom, súbor obsahujúci kľúčové slová pre aplikácie a adresár s dátovými sadami
- **readme.md** - užívateľský manuál na spustenie skriptu
- **xborbe00_BP.pdf** - správa v elektronickej forme