VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

PROFILOVANIE SIEŤOVÝCH ENTÍT PRE ZLEPŠENIE SITUAČNÉHO POVEDOMIA

PROFILING OF NETWORK ENTITIES TO IMPROVE SITUATIONAL AWARENESS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RENÉ BOLF**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN ŽÁDNIK, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Bolf René**
Program: Informační technologie
Název: **Profilování síťových entit pro zlepšení situačního povědomí**
Profiling of Network Entities to Improve Situational Awareness
Kategorie: Počítačové sítě
Zadání:

1. Seznamte se s měřením síťového provozu a s využitím profilů služeb, IP adres a podsítí v síťovém provozu (obecně síťových entit) v oblasti síťové bezpečnosti.
2. Nastudujte relevantní literaturu z pohledu vytváření a uchování profilů entit, zaměřte se na použité příznaky v profilu a jejich aplikace pro odvozování informací v oblasti situačního povědomí.
3. Navrhněte metodu a příznaky pro profilování síťových entit na různých úrovních detailu.
4. Implementujte navrženou metodu formou prototypu.
5. Řešení vyhodnoťte na reálných síťových datech.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Žádník Martin, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Mať dobré situačné povedomie je dôležitou súčasťou počítačovej bezpečnosti. Vedomosť o tom, čo sa v sieti nachádza, kde sa to nachádza a kto v sieti komunikuje dokáže pomôcť robiť lepšie a rýchlejšie rozhodnutia pri vzniku bezpečnostných incidentov. Táto práca sa zaoberá profilovaním sieťových entít na úrovni zariadení. Presnejšie sa zameriava na pasívnu identifikáciu operačných systémov. Každý paket vložený do siete nesie vo svojej hlavičke paketu špecifické informácie, ktoré odrážajú počiatočné nastavenie operačného systému. Sada týchto informácií tvorí „odtlačok prsta“ operačného systému. V práci je popísaná implementácia klasifikátoru strojového učenia využívajúceho metódu rozhodovacích stromov. Klasifikátor pri klasifikácii využíva príznaky z TCP a IP hlavičiek. Klasifikátor bol vyhodnotení na dátovej sade, ktorá obsahovala dáta reálneho sieťového prenosu a pri klasifikácii do 9 tried operačných systémov dosiahol presnosť 96 %.

Abstract

Having a good situational awareness is an important part of computer security. Knowing what is connected to the network, where it is located, and who is communicating can help make better and faster decisions when security incidents occur. This thesis is focusing on the profiling of network entities at the device level. More specifically, it focuses on the passive identification of operating systems. Every packet transferred in the network carries a specific information in its packet header that reflects the initial settings of a host's operating system. The set of these information is called the "fingerprint" of an operating system. In the thesis, there is described an implementation of a machine learning classifier using the decision tree method, which uses features from TCP and IP headers. The classifier was evaluated on a data set containing data from real network traffic and has achieved accuracy of 96 % when classifying into 9 classes of operating systems.

Kľúčové slová

Situačné povedomie, Profilovanie, operačné systémy, IP tok, monitorovanie, identifikácia operačného systému

Keywords

Situational Awareness, Profiling„ operating systems, IP flow, monitoring, operating system identification

Citácia

BOLF, René. *Profilovanie sieťových entít pre zlepšenie situačného povedomia.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Žádnik, Ph.D.

Profilovanie sieťových entít pre zlepšenie situačného povedomia

Prehlásenie

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

.....
René Bolf
11. mája 2021

Poďakovanie

Ďakujem vedúcemu práce Ing. Martinovi Žádnikovi, Ph.D. za jeho čas, cenné rady a usmerňovanie pri tvorbe tejto práce. Tiež by som sa chcel poďakovať Ing. Václavovi Bartošovi za jeho cenné rady.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Zlepšenie situačného povedomia sa stalo dôležitou úlohou vo viacerých odvetviach. Dôležitosť tejto úlohy stúpala spolu so zložitosťou jednotlivých systémov. Technológie sa vyvíjali a začali vznikať zložitejšie systémy, v ktorých si ľudia už nedokázali udržať vedomie o situácii len pomocou manuálnej činnosti. Ako sa technológie vyvíjali, vyvíjala sa aj počítačová sieť a stávala sa stále zložitejšia. Zároveň sa počítáčová sieť stala dôležitou súčasťou každodenného života.

Počítačová sieť poskytuje služby, ktoré sú kritické pre spoločnosť, ako napríklad finančné systémy, riadiace systémy elektrární alebo služby, ktoré sú využívané v každodennom živote, ako napríklad sociálne siete, internetové obchody. Takéto služby poskytujú citlivé informácie, ktoré sa stali cieľom mnohých útočníkov a podvodníkov. Počítačová bezpečnosť sa stala dôležitou témou a nevyhnutnou súčasťou počítačovej bezpečnosti je dosiahnutie situačného povedomia. Inak povedané je dôležité dosiahnuť schopnosť vnímať procesy v sieti, chápať ich význam, vzťahy a predpovedať budúci stav siete. Efektívny spôsob, s ktorým je možné získať situačné povedomie je analýzou dát získaných technikou monitorovania sieťových tokov.

Cieľom tejto práce je zlepšiť situačné povedomie v sieti pomocou profilovania sieťových entít. Práca sa zameriava na profilovanie sieťových entít na úrovni zariadení a presnejšie sa zameriava na pasívnu detekciu operačných systémov z dát získaných pasívnou technikou monitorovania tokov. Identifikácia operačných systémov je dôležitou témou počítačovej bezpečnosti, pretože moderné počítačové siete umožňujú priniesť a pripojiť akékoľvek zariadenie do siete, čo môže predstavovať nové bezpečnostné riziká.

Kapitola 2 sa venuje súčasnému stavu. V sekcii 2.1 je zhrnuté situačné povedomie v počítačových sieťach. Situačné povedomie je možné získať pomocou monitorovania siete, preto sa v tejto kapitole ďalšia sekcia 2.2 venuje monitorovaniu počítačových sietí. Ďalšia sekcia 2.3 v tejto kapitole sa venuje profilovaniu sieťových entít a presnejšie sa venuje pasívnej detekcii operačných systémov. V kapitole 3 je popísaná navrhnutá metóda a implementácia klasifikátora, ktorý identifikuje operačné systémy z monitorovaných sieťových tokov. V kapitole 4 sa nachádza vyhodnotenie klasifikátora a experimenty. V tejto kapitole sa nachádza aj porovnanie klasifikátora s ďalšími dvomi metódami, a to metóda, ktorá identifikuje operačný systém pomocou tabuľky známych príznakov operačných systémov a metóda, ktorá klasifikuje skupinu tokov. Okrem toho sa v sekcii 4.4 nachádzajú ďalšie experimenty, ktoré vyhodnocujú úspešnosť klasifikátora za rôznych podmienok.

Kapitola 2

Súčasný stav

2.1 Situačné povedomie v počítačových sieťach

Zvýšenie situačného povedomia sa stalo dôležitou úlohou vo viacerých odvetviach. Dôležitosť tejto úlohy stúpala spolu so zložitosťou jednotlivých systémov. Technológie sa vyvíjali a začali vznikať zložitejšie systémy, v ktorých si ľudia nedokázali udržať vedomie o situácii [11]. Uvedomenie si situácie bolo vždy potrebné aj v každodennom živote, aby ľudia mohli efektívne vykonávať úlohy. Mnoho rokov stačil na získanie situačného povedomia tréning, vďaka ktorému je možné nadobudnúť skúsenosti o tom, čo je potrebné sledovať a čo znamenajú rôzne dôležité signály. Postupom času sa technológie vyvíjali a udržanie si situačného vedomia už nebolo také jednoduché, pretože ľudia museli začať vnímať veľký obsah dát, ktoré sa rýchlo menia [11]. Dnešné systémy sú schopné produkovať obrovské množstvo údajov, a to ako aj o stave ich vlastných komponentov, tak aj o stave externého prostredia.

Počítačové siete sú čoraz viac zložitejšie a sú nevyhnutné pre rôzne služby alebo kritické systémy. Takéto služby poskytujú citlivé dáta, ktoré sa stali cieľom mnohých útočníkov. Aby bolo možné brániť systémy pred útočníkmi a dosiahnuť vedomosť o tom čo sa v sieti nachádza je potrebné dosiahnuť situačné povedomie. Je dôležité porozumieť procesom v sieti, identifikovať hrozbu a prijať vhodné opatrenia [18]. Prvky, ktoré sa vyskytujú v situačnom povedomí počítačových sietí sú časti fyzickej infraštruktúry (prepínače, smerovače, koncoví užívatelia, servery) a sieťový prenos, ktorý je prenášaný cez sieť. Cieľom je poskytnúť operátorovi dostatok informácií, na základe ktorých vie vykonávať informované rozhodnutia. Informácie sa získavajú napríklad z detekčných systémov vniknutia, antivírusov, detektorov škodlivého softvéru, protokolov, tokov a ďalších informačných zdrojov [18].

Koncept situačného povedomia začal byť študovaný a stal sa užitočným v mnohých oblastiach. Vzniklo viacero definícií ale najznámejšia a najpoužívanejšia definícia situačného povedomia, ktorá je použiteľná v širokej škále domén, znie takto

> *„Uvedomenie si situácie je vnímanie prvkov v prostredí v objeme času a priestoru, pochopenie ich významu a projekcia ich stavu v blízkej budúcnosti.“* [9]

Definícia pozostáva z troch úrovní:

- **Úroveň 1: Vnímanie prvkov**. Prvým krokom k dosiahnutiu situačného povedomia je vnímanie stavu, atribútov a dynamiky dôležitých prvkov v prostredí. Atribúty prostredia je možné vnímať buď priamo z prostredia, alebo nepriamo pomocou senzorov [10]. Bezpečnostní analytici musia vnímať výstrahy, ktoré sú hlásené napríklad zo systémov detekcie narušenia (IDS), záznamy z brány firewall, správy o skenovaní, čas kedy k tomu došlo a konkrétne ovládacie prvky, ktoré hlásili [30]. Rozhodujúce je správne vnímanie, pretože ak by získané údaje boli nepresné, tak by operátor mohol získať chybnú predstavu o situácií. Prvá úroveň teda poskytuje len počiatočný príjem informácií [10].
- **Úroveň 2: Pochopenie súčasnej situácie**. Cieľom situačného povedomia nie je len situáciu vnímať, ale aj pochopiť. Pochopenie situácie je založené na súhrne rôznorodých prvkov z úrovne 1. V druhej úrovni ide o pochopenie významu prvkov z prvej úrovne, ktoré sú dôležité pre splnenie príslušných cieľov operátora. Na základe týchto poznatkov sa vytvára obraz prostredia, ktorý chápe význam objektov a udalostí. Napríklad pochopenie informáciám obsiahnutých v dátach získaných z monitorovaného prostredia [10].
- **Úroveň 3: Projekcia budúceho stavu**. Projekcia budúceho stavu je treťou a najvyššou úrovňou situačného povedomia. Projekcia budúceho stavu vychádza z poznatkov nadobudnutých z predchádzajúcich dvoch úrovni. Presnosť projekcie budúceho stavu závisí od presnosti vnímania a pochopenia situácie. Projekcia budúceho stavu je dôležitá, pretože poskytuje operátorom vedomosti vďaka, ktorým vedia predvídať udalosti a ich dôsledky a na základe toho sa včasne a správne rozhodovať. Ak by boli poznatky z prechádzajúcich úrovní nepresné, došlo by k nepresnej projekcii. Okrem toho presnosť predpovede závisí aj od skúsenosti operátora a znalosti dynamiky situácie [10, 19]. Projekcia odpovedá na otázky - aké sieťové útoky sú možné a aké kontroly môžu byť potrebné? [30]

Sú to teda tri úrovne, ktoré tvoria situačné povedomie. Prvá úroveň - vnímanie slúži pre získanie údajov z monitorovaného prostredia, úroveň 2 - pochopenie má za cieľ porozumieť informáciám obsiahnutých v dátach a cieľom tretej úrovne - projekcie je predpovedať budúci stav prostredia [19]. Monitorovanie tokov IP predstavuje široko používaný prístup na získanie viditeľnosti siete, a teda budovanie situačného povedomia.

2.2 Monitorovanie siete

Prístupy k monitorovaniu siete

V priebehu rokov boli navrhnuté a vyvinuté rôzne prístupy na monitorovanie siete, z ktorých každý slúžil k inému účelu. Monitorovanie siete je možné rozdeliť na dva hlavné prístupy:

- aktívne monitorovanie siete
- pasívne monitorovanie siete

Aktívne monitorovanie siete

Aktívne monitorovanie siete funguje na princípe vnášania testovacieho prenosu do počítačovej siete. Jednoducho povedané administrátor nečaká len na správy, ktoré mu pošlú sieťové zariadenia alebo servery samy od seba, ale aktívne sa dopytuje na dostupnosť liniek, aplikácií alebo sieťových prvkov a na základe odozvy určuje stav siete. Jednoduchý spôsob aktívneho monitorovania siete je pomocou nástrojov Ping alebo Traceroute [26].

Pasívne monitorovanie siete

Pasívne prístupy monitorovania siete fungujú na základe sledovania prebiehajúcej sieťovej komunikácie, a to pri jej prechode cez pozorovací bod v sieti. Z prebiehajúcej komunikácie zbierajú informácie o stave siete a zariadeniach v sieti [26]. Pasívny prístup sa používa hlavne na monitorovanie sieťovej prevádzky. Okrem toho slúži na zhromažďovanie, ukladanie a analýzu sieťového prenosu. Sieťový prenos je sledovaný pomocou prístupových metód monitorovania sieťového prenosu.

Prístupy k monitorovaniu sieťovej prevádzky je možno rozdeliť podľa úrovne abstrakcie informácií, ktoré sú získané zo sieťovej komunikácie:

- **Simple Network Management Protocol (SNMP)** - tento prístup nevyžaduje analýzu paketu, štatistiky sa získavajú zo sieťových zariadení pomocou jednoduchého protokolu správy siete (SNMP) alebo zo systému RMON. Tento prístup neposkytuje detailné informácie, poskytuje len objemové informácie o sieťovej prevádzke ako napríklad informácie o počte prenášaných alebo prijímaných paketov alebo o počte chýb. Zvyčajne sa tento prístup používa len na základné sieťové účtovníctvo [19].

- **Paketová analýza** - výhoda tohto prístupu je, že poskytuje najväčší prehľad o sieťovej prevádzke, pretože je možné zachytiť a analyzovať obsah celého paketu, to znamená hlavičku paketu a aj dátovú časť paketu. Nevýhodou tohto prístupu je, že vo vysokorýchlostných sieťach je potrebné mať drahý hardvér a rozsiahlu infraštruktúru na ukladanie a analýzu. Tento prístup je vhodný skôr na analýzu cieleného sieťového prenosu ako na sledovanie celej siete. Okrem toho sa objem informácií dostupný z paketovej analýzy neustále znižuje so zvyšujúcim sa šifrovaným prenosom [14, 19].

- **Monitorovanie sieťových tokov** - tento prístup je vhodnejší na použitie vo vysokorýchlostných sieťach. Objem informácií, ktoré sú získané pomocou tohto prístupu, je menší ako pomocou paketovej analýzy. Informácie sú zredukované len na základné metadáta o pripojení v sieti. Avšak na druhej strane tento prístup umožňuje vysokú priepustnosť a relatívne nízke požiadavky na zdroje, pretože sa informácie analyzujú len z hlavičiek paketov. Monitorovanie toku teda predstavuje kompromis medzi objemom dostupných informácií a mierou analýzy [19].

Monitorovanie tokov

Monitorovanie sieťových tokov je pasívny prístup monitorovania siete, vďaka ktorému je možné dosiahnuť situačné povedomie v počítačovej sieti. Monitorovanie sieťových tokov je prístup, ktorý je možno použiť vo vysokorýchlostných sieťach, a ktorý poskytuje informácie o tom čo sa v počítačovej sieti deje, ako sa to deje, a kde sa to deje. Pri monitorovaní sieťových tokov sa zaznamenávajú metadáta o každom pakete, ktorý prešiel sieťou. Záznamy o tokoch sa používajú pre správu sieťovej prevádzky a vďaka tomu je možné identifikovať komunikáciu v danom čase pri vzniku bezpečnostného incidentu.

Sieťový tok

Sieťový tok angl. network flow je v RFC 3954 [7] definovaný ako - *sada IP paketov, ktoré majú spoločnú vlastnosť a prechádzajú v určitom časovom intervale pozorovacím bodom v sieti.* To znamená, že všetky IP pakety patriace konkrétnemu toku majú sadu spoločných vlastností. Tieto spoločné vlastnosti môžu obsahovať polia hlavičky paketu, ako zdrojové a cieľové IP adresy, zdrojové a cieľové čísla portov, metadáta a dokonca aj obsah paketu [14]. Počiatočné práce, ktoré sa zaoberajú pasívnym prístupom monitorovania sieti pomocou exportu tokov sa datujú do deväťdesiatych rokov a stali sa základom pre protokoly NetFlow a IPFIX [14].

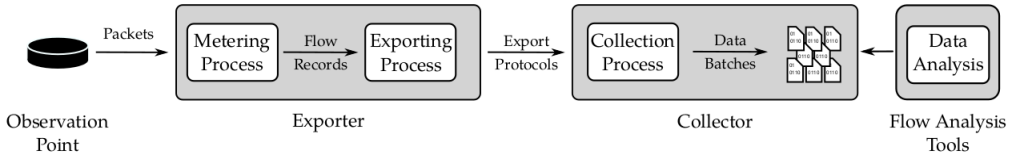
Výhody monitorovania tokov oproti klasickému zachytávaniu paketov

- možnosť použitia vo vysokorýchlostných sieťach.
- sú široko nasadené - vďaka integrácii do zariadení ako sú smerovače, prepínače.
- použiteľné v súlade so zákonmi o uchovávaní údajov.
- významné zníženie dát, pretože pakety sa agregujú až po ich zachytení.
- poskytuje viac súkromia [14].

Architektúra NetFlow

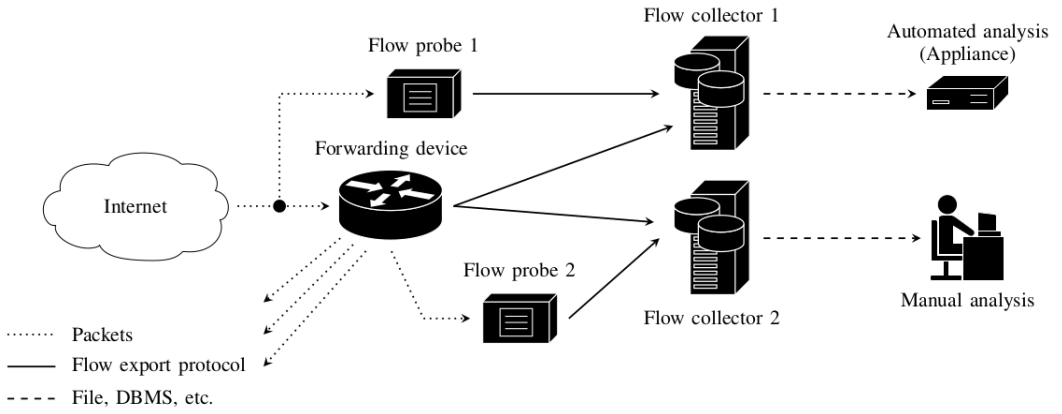
Architektúra pre monitorovanie toku typicky pozostáva z niekoľkých fáz, ktoré sú znázornené aj na obrázku 2.1:

- **Prvá fáza: pozorovanie paketov (packet observation)** - v tejto časti sú pakety zachytené z pozorovacieho bodu (observation point) a sú vopred spracované. Pozorovacími bodmi môžu byť sieťové karty alebo rozhrania zariadení na preposielanie paketov [14].
- **Druhá fáza: meranie a export toku** - táto fáza pozostáva z procesu merania a z procesu exportu. V procese merania sú pakety agregované do tokov, potom čo sa tok považuje za ukončený, je umiestnený do datagramu použitého protokolu na export toku [14].
- **Tretia fáza: zber údajov** - táto fáza zaisťuje príjem, ukladanie a predspracovanie údajov o toku (napríklad agregácia, kompresia, filtrovanie)
- **Posledná fáza: analýza dát** - manuálne alebo plne automaticky - viz. obrázok 2.2



Obr. 2.1: Architektúra monitorovania tokov [19].

Architektúra NetFlow môže byť popísaná aj iným spôsobom a to tak, že sa budú líšiť niektoré fázy. Napríklad často proces zachytávania paketov a proces merania a exportu tokov je spojený v jednom zariadení, ktoré sa nazýva Exportér [14]. Takýto spôsob architektúry je možné vidieť na obrázku 2.2. Flow probe 1 a Flow probe 2 predstavujú na obrázku exportér ako sondu a Forwarding device predstavuje exportér ako smerovač.



Obr. 2.2: Rôzne nastavenia architektúry monitorovania tokov [14].

Pozorovanie paketov

Pozorovanie paketov sa skladá z dvoch hlavných fáz, ktoré sa vykonávajú pre každý pozorovaný paket. Zo zachytávania paketov a vytvárania časovej značky. Zachytávanie paketov sa vykonáva zvyčajne pomocou sieťovej karty (NIC) a v tejto fáze sa tiež vykonáva niekoľko kontrol (napríklad kontrola chyby kontrolného súčtu). Druhou fázou je vytvorenie časovej pečiatky. Počas tejto fázy je každému paketu priradená presná časová pečiatka pozorovania. Vytvorenie presnej časovej pečiatky je nevyhnutné pre všetky nasledujúce procesy. Nepresné časové pečiatky môžu viesť k vytvoreniu nesprávnych záznamov toku IP alebo ku skresleným výsledkom vo fáze analýzy tokov IP [19]. Ďalšími fázami sú vzorkovanie a filtrovanie. Hlavnou funkciou vzorkovania a filtrovania je preposielať len určité pakety do ďalšej fázy merania a exportu toku. Pri vzorkovaní sa vyberá taká podmnožina paketov, pri ktorej je stále možné odhadnúť vlastnosti celého toku paketov. Cieľom je znížiť zaťaženie nasledovných fáz. Pri filtrovaní sa jedná o odstránenie všetkých paketov, ktoré nie sú zaujímavé [14].

Exportér

Exportér je sieťové zariadenie, a to sonda alebo smerovač viz. obrázok 2.2, ktoré je umiestnené na dôležitých miestach v sieti a monitoruje sieťovú prevádzku. Exportér vytvára záznamy o tokoch [17]. Exportér tvoria dva hlavné procesy, a to proces merania a proces exportu. V niektorých architektúrach Exportér môže zaobstarávať aj fázu pozorovania paketov viz. obrázok 2.2.

Proces merania

Fáza merania je proces, ktorý je zodpovedný za agregáciu jednotlivých paketov do tokov IP a za vytvorenie záznamov toku IP. Všetky aktívne toky IP sa ukladajú do tabuľky medzipamäte toku tzv. flow cache. Táto tabuľka obsahuje položky, ktoré sú zložené z kľúčových a nekľúčových polí. Kľúčové polia tvorí zvyčajne pätica *(zdrojová IP, cieľová IP, zdrojový port, cieľový port, číslo protokolu)*. Nekľúčové polia obsahujú ďalšie informácie o zázname toku IP, súčet bajtov, pakety. S každým prichádzajúcim paketom z fázy pozorovania paketov sa v medzipamäti vyhľadávajú záznamy pomocou zodpovedajúcich kľúčových polí. Ak sa záznam v medzipamäti toku nenájde, znamená to, že ide o nový záznam a jedná sa o prvý paket toku, a preto sa založí nový záznam zo zodpovedajúcimi kľúčovými poľami paketu. Následne ak príde paket daného toku tak sa pridá do tohto záznamu. Platnosť záznamu v medzipamäti končí, keď sa považuje zodpovedajúci tok IP za ukončený [19]. Zistenie či je tok ukončený nie je až tak jednoduché. V prípade TCP komunikácii je možné koniec toku určiť pomocou sledovania príznakov FIN a RST, ale ak pôjde paket inou cestou alebo sa stratí tak záznam bude v pamäti ešte dlho. Problém nastáva pri ostatných protokoloch ako sú napríklad UDP alebo ICMP, v takýchto protokoloch príznaky ako FIN alebo RST neexistujú. Preto platnosť týchto záznamov často končí podľa daných parametrov časového limitu alebo podľa výskytu konkrétnych udalostí [14]:

- **aktívny časový limit** - aktívny časový limit udáva, že tok bol aktívny po zadanú časovú dobu. Typické hodnoty časového limitu sú v intervale od 120 sekúnd do 30 minút. Dôležité je, že záznamy sa z medzipamäte neodstránia, ale počítadlá sa vynulujú a čas začiatku a konca sa aktualizuje.
- **časový limit nečinnosti** - počas stanovenej doby neboli pozorované žiadne pakety patriace k toku. Typické hodnoty časového limitu sú od 15 sekúnd až do 5 minút.
- **obmedzenie zdrojov** - používajú sa špeciálne heuristiky
- **prirodzené vypršanie platnosti** - pre tok bol pozorovaný paket TCP s príznakmi FIN alebo RST, preto sa tok považuje za ukončený.
- **núdzové vypršanie platnosti** - po zaplnení medzipamäte (flow cache).

Proces exportu

Potom čo je IP tok ukončený, tak sa celý záznam o toku odstráni z medzipamäte a odovzdá sa na proces exportu. Počas odovzdávania záznamov o toku prebieha proces vzorkovania a filtrovania záznamov IP tokov. Rovnako ako je to pri vzorkovaní a filtrovaní paketov je cieľom vzorkovania a filtrovania IP tokov znížiť požiadavky na spracovanie v nasledujúcich fázach znížením počtu záznamov IP tokov, ktoré sa majú spracovať. Vzorkovanie a filtrovanie sa vykonáva po procese merania, a preto namiesto paketov pracujú so záznamami tokov [14]. Proces exportu spracováva odosielanie záznamov IP toku generovaných jedným alebo viacerými procesmi merania do jedného alebo viacerých procesov zhromažďovania (kolektorov). Záznamy toku sú agregované do správ a odoslané cez vybraný protokol do zariadenia na zber údajov (kolektora). Štruktúra správy je odvodená od protokolu použitého na export toku. Najznámejšie protokoly sú napríklad NetFlow v5, NetFlow v9, IPFIX. Všetky tieto protokoly slúžia na odosielanie štatistík o tokoch z exportéru na kolektor [19].

Kolektor

Kolektor je významná súčasť monitorovacej architektúry. Na kolektore sa prijímajú, ukladajú a predbežne spracovávajú záznamy o tokoch, ktoré môžu pochádzať buď od jedného exportéru alebo od viacerých exportérov v sieti. Pred spracovaním záznamov sa bežne vykonávajú procesy ako sú kompresia dát, agregácia dát, anonymizácia údajov, filtrovanie a generovanie súhrnov [14]. Funkčnosť a výkon kolektoru závisí od toho aký formát úložiska je použitý, pretože to určuje ako rýchlo sa budú údaje čítať a zapisovať. Pri formáte úložiska sa rozlišujú dva druhy úložisk a to krátkodobé úložisko a trvalé úložisko. Krátkodobé úložisko môže byť využité pri spracovaní údajov skôr ako sa zapíšu do trvalého úložiska. Trvalé úložisko slúži na dlhodobé uchovanie údajov ale je výrazne pomalšie ako krátkodobé úložisko [14].

Pre dočasné úložisko sa rozlišujú nasledovné typy úložísk:

- **ploché súbory (flat files)** - ploché súbory môžu byť napríklad binárne alebo textové súbory. Pri tomto type je ukladanie a čítanie veľmi rýchle. Naopak poskytuje obmedzené možnosti dopytovania [14].
- **riadkovo orientované databázy** - údaje sa ukladajú do tabuliek. Sú to typy databázových technológií ako napríklad MySQL. Údaje sa ukladajú pomocou riadkov, a preto pri čítaní sa načítavajú celé riadky a nie len potrebná časť údajov [14].
- **stĺpcovo orientované databázy** - údaje sa ukladajú podľa stĺpcov, preto je možný prístup, len k potrebným dátam [14].

2.3 Profilovanie sieťových entít

Vo veľkých počítačových sieťach, kde sa nachádzajú stovky alebo viac zariadení a sú usporiadané do desiatok podsietí, je pre správcu nemožné aby získal situačné povedomie len manuálnou kontrolou. Preto je pre správcu počítačových sietí obrovskou výhodou použitie nástrojov, ktoré vytvárajú presné a aktuálne informácie o sieti. Vďaka takýmto nástrojom môže správca rýchlejšie a presnejšie zareagovať na vzniknuté incidenty. Profilovanie sieťových entít je možné vykonávať na dvoch hlavných úrovniach detailov - na úrovni zariadení alebo na úrovni topológie siete:

- **Profilovanie na úrovni zariadení (Device fingerprinting)** - Pri tejto činnosti sa získavajú informácie špecifické pre zariadenie. Informácie sa získavajú nezávisle od siete, kde je zariadenie umiestnené. Informácie, ktoré je možné získať v tejto činnosti sú informácie ako napríklad operačný systém zariadenia, služby, ktoré bežia na zariadení, protokoly, používané porty a pod. [29]
- **Profilovanie na úrovni topológie siete** - Pojem topológia siete predstavuje spôsob akým sú v sieti koncové zariadenia alebo sieťové zariadenia navzájom pospájané medzi sebou a ako je sieť rozdelená na segmenty. Počas získavania informácií o topológii siete je potrebné zhromaždiť údaje, ako sú napríklad IP adresa každého zariadenia, ich geografické umiestnenie, množina podsiete alebo lokálnej siete. Je potrebné vytvoriť vzťah medzi zariadením a podsieťou, v ktorej sa nachádza. Je potrebné aby počas zisťovania topológie boli podsiete úplne definované. To znamená, že by mal byť známy celý rozsah IP adries zahrnutých v podsieti, alebo je potrebné aby bola známa prvá IP

adresa a maska podsiete. Objavovanie topológie siete predstavuje proces identifikácie sieťovej štruktúry. Ide o objavovanie informácií ako sú podrobnosti o podsieti, informácie o koncových užívateľoch napríklad ich fyzické umiestnenie v sieti, ich IP adresa, podsieť a lokálna sieť, ku ktorým patrí. Metódy objavovania topológie je možné rozdeliť na aktívny prístup a na pasívny prístup, rovnako ako pri identifikácií operačných systémov [29].

Detekcia operačného systému

Princíp moderných počítačových sietí je, že si užívateľ môže priniesť a pripojiť akékoľvek zariadenie. To je pre používateľov veľká výhoda, avšak pre sieť to môže predstavovať nové bezpečnostné riziká [22]. Preto je detekcia operačného systému v počítačovej siete dôležitou úlohou pre zlepšenie situačného povedomia v sieti, a je to jeden z hlavných záujmov počítačovej bezpečnosti. Motiváciu detekcie operačného systému je možné sledovať z dvoch pohľadov. Z pohľadu správcu siete a z pohľadu útočníka. Z pohľadu správcu siete sa na detekciu operačného systému dá pozerať z pozitívneho hľadiska, pretože je dôležité aby správca siete mal čo najviac informácií o tom čo sa v sieti deje, čiže mal lepšie situačné povedomie. Napríklad je dôležité aby mal správca siete informáciu napríklad o tom, že v jeho sieti sa nachádza zariadenie, ktoré používa starú verziu operačného systému, to je napríklad taký operačný systém, ktorému už nie sú poskytnuté bezpečnostné aktualizácie. Takéto zariadenie sa môže stať ľahkým cieľom pre útočníka, ktorý môže využiť jeho zraniteľnosť a vniknúť tak do siete [2].

Identifikácia operačných systémov je tiež užitočná pri udržiavaní bezpečnostných politík. Pravidlá bezpečnostných politík môžu určovať, že kto sa môže pripojiť do siete, zakázanie starých verzii operačných systémov alebo takých, ktoré sieťová infraštruktúra nepodporuje. Vytvorené povedomie o tom aké operačné systémy sa v sieti nachádzajú môže pomôcť správcovi siete posúdiť možnú hrozbu [24]. Lastovička a kol. [22] určili prípady použitia, kedy správcovia siete profitujú z detekcie operačných systémov:

- **Nepodporovaný operačný systém** - technológie sa rýchlo vyvíjajú a rovnako rýchlo sa vyvíjajú aj nové verzie operačných systémov. To vedie k tomu, že veľa používateľov používa stále zastaranú verziu, ktorá už nemá k dispozícii bezpečnostné aktualizácie, čo môže využiť útočník a použiť takéto zariadenie ako vstupný bod do siete. Tento problém nastáva hlavne v mobilných operačných systémoch.
- **Rozhodovanie** - na základe vedomosti o tom aké zariadenia sú pripojené v sieti, umožňuje správcom sietí sa správne rozhodovať a presnejšie reagovať na bezpečnostné incidenty.
- **Bezpečnostná politika** - je možné vytvárať bezpečnostnú politiku, ktorá rozhoduje o tom, kto sa môže pripojiť a s akým zariadením sa môže pripojiť.
- **Statické siete** - akékoľvek zariadenie, ktoré do siete nepatrí sa môže považovať za bezpečnostné riziko, ktoré by malo byť preskúmané.

Na detekciu operačného systému sa dá pozerať aj z negatívneho hľadiska. Vďaka vedomosti o operačnom systéme si útočník, môže zvoliť stratégiu a identifikovať zraniteľné miesta a využiť ich ako vstup do siete [2].

Aktívny vs pasívny prístup

Detekcia operačných systémov funguje na základe identifikácie odtlačku operačného systému. Odtlačok operačného systému je možné definovať ako skúmanie určitých charakteristík a správania sa pri sieťovej komunikácii s cieľom vzdialene identifikovať operačný systém a jeho verziu bez priameho prístupu k samotnému zariadeniu [20]. Existujú dva hlavné prístupy k detekcii operačného systému, a to aktívny prístup a pasívny prístup:

- **Aktívny prístup** - pri tomto prístupe sa posiela séria špeciálne vytvorených paketov cieľovému zariadeniu s cieľom vyvolať odpoveď, ktorá odhalí operačný systém. Toto umožňuje aktívnemu prístupu získať presnejšie výsledky ako v prípade použitia pasívneho prístupu. Ak je v sieti systém pre detekciu vniknutia (IDS) alebo brána firewall aktívny prístup sa použiť nedá [20].
- **Pasívny prístup** - pri pasívnom prístupe sa nevytvárajú žiadne špeciálne pakety, a tak sa negeneruje žiadny ďalší prenos. Analýza prebieha na základe existujúcej komunikácie, kde sa analyzujú informácie z paketov v sieti [20].

Jednou z výhod pasívneho prístupu je, že právna povaha tohto prístupu nemúsí byť spochybnená, kde na druhej strane aktívny prístup je v niektorých krajinách označený za ilegálny [12]. Pasívny prístup len analyzuje prenos, ktorý už bol odoslaný. Okrem toho pasívna metóda môže byť vykonaná úplne off-line. Nevýhoda pasívneho prístupu je jeho menšia presnosť v porovnaní s aktívnym prístupom, pretože má menšiu kontrolu nad dátami, ktoré sa analyzujú. Na druhej strane pri pasívnom prístupe nie je možné zistiť, že prebieha nejaká analýza odtlačku prstov operačných systémov, kde pri aktívnom prístupe to možné je [20].

Ďalšia nevýhoda pasívneho prístupu je, že je závislý od existujúcej sieťovej prevádzky, to znamená, že ak koncový užívateľ neodošle očakávaný typ údajov ako sú TCP SYN pakety, DHCP požiadavky, HTTP požiadavky alebo ak vôbec nekomunikuje, tak nemožno použiť pasívny prístup [27].

Výhodou aktívneho prístupu je, že sa zvyčajne výsledky generujú s vysokou presnosťou, pretože majú schopnosť získať prístup alebo požadovať informácie, ktoré sú užitočné pre identifikáciu operačného systému. Nevýhody sú, že aktívne nástroje môžu byť blokované bránou firewall alebo systémami na detekciu vniknutia a nebudú schopné posielať pakety do sondovacieho systému [1]. Aktívne a pasívne prístupy majú veľa spoločného, pokiaľ ide o detekciu operačného systému, pretože využívajú podpisy, na základe, ktorých identifikujú operačný systém. V tabuľke 2.1 sú stručne zhrnuté výhody a nevýhody pasívneho a aktívneho prístupu.

Tabuľka 2.1: Stručné porovnanie aktívneho prístupu a pasívneho prístupu

Aktívny	**Pasívny**
+ presnejší	+ nezistiteľný
+ informácie na požiadanie	+ nedotieravý
- generuje ďalší sieťový prenos	+ negeneruje ďalší sieťový prenos
- blokovaný IDS, bránou firewall	- menej presný
- dotieravý	- závislí od existujúcej sieťovej prevádzky
- orientovaný na jednu požiadavku	

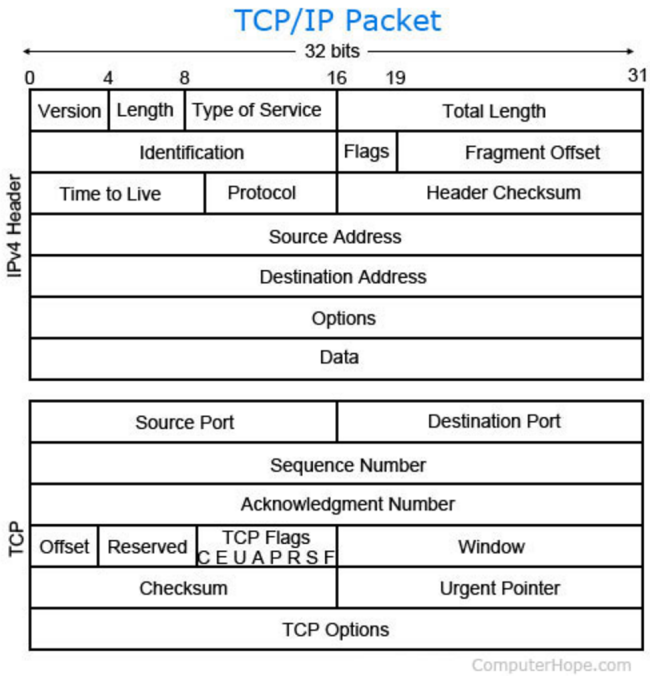
Pasívny prístup

Už ako bolo spomenuté pasívny prístup identifikácie operačných systémov funguje na princípe, že na rozdiel od aktívneho prístupu nevytvára žiadne špeciálne pakety a neposiela ich vzdialenemu zariadeniu, ale analyzuje existujúcu sieťovú komunikáciu, kde získava informácie z paketov. Odtlačky operačných systémov je možné získať pomocou rôznych metód najčastejšie z parametrov hlavičiek TCP/IP paketu. Počítače majú určité pravidlá, ktoré musia dodržiavať pri komunikácií v sieti, z ktorých najbežnejšia je skupina protokolov TCP a IP. Protokol IP(Internetový protokol) zaisťuje prideľovanie a správu logických adries pre každého hostiteľa v sieti. Protokol TCP (Transmission Control Protocol) zaisťuje správne doručenie všetkých paketov [20]. Tieto protokoly musia byť implementované v akomkoľvek operačnom systéme aby bol schopný komunikovať na internete. Protokoly TCP a IP sú popísané v oficiálnych dokumentoch RFC 791 [31] pre IP protokol a RFC 793 [32] pre TCP protokol. Vývojári by mali pri implementácii operačného systému dodržať odporúčania, ktoré sú popísané v príslušných RFC. Avšak existujú oblasti v RFC, kde sa nechávajú určité rozhodnutia na vývojárov. Napríklad v príslušnom RFC dokumente pre IP protokol nie je stanovená počiatočná hodnota pre TTL pole, a teda vývojári sú obmedzení jedine na maximálnu veľkosť poľa 8 bitov, čo sa rovná číslu 255. Preto záleží na vývojároch akú počiatočnú hodnotu zvolia [20]. Rovnaký problém nastáva aj u iných polí z TCP/IP paketu. Vďaka tomu sa vytvára takzvaný „odtlačok prsta“ operačného systému (angl. operating system fingerprinting). Pasívny prístup zisťovania odtlačkov prsta využíva teda skutočnosť, že rôzne operačné systémy majú odlišne naimplementované zásobníky TCP a IP protokolu a vytvára sa tak jedinečný podpis.

Väčšina pasívnych nástrojov na zisťovanie odtlačkov prstov operačných systémov využíva hlavičky paketu TCP SYN. Keď je paket TCP SYN zachytený, extrahujú sa príznaky a porovnajú sa s tabuľkou, ktorá obsahuje podpisy rôznych operačných systémov. Takýmto spôsobom je možné určiť z akého operačného systému paket pochádza. Takýto spôsob sa nazýva TCP/IP Fingerprinting a využíva ho väčšina nástrojov na pasívnu detekciu operačných systémov. Na detekciu operačného systému je možné použiť niekoľko polí z TCP/IP paketu ako napríklad TTL, veľkosť okna, typ služby a pod. Existuje viacej spôsobov ako sa dá vykonať pasívne snímanie odtlačkov prstov a vzniklo niekoľko nástrojov, ktoré využívajú rôzne nástroje. Metódy využívajú nasledovne typy paketov [12]:

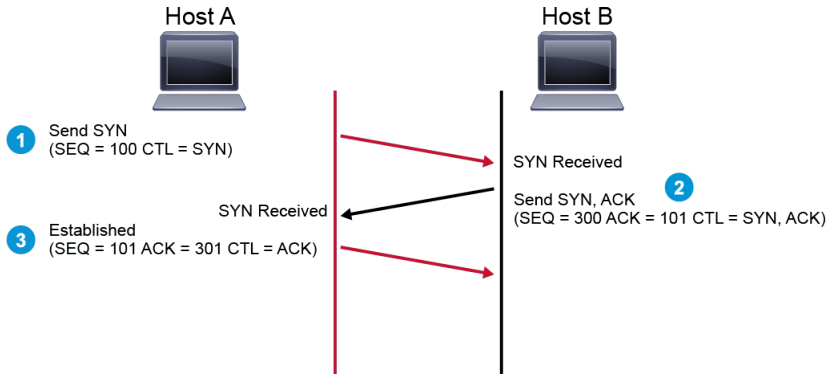
- TCP SYN
- TCP SYN + ACK
- TCP RST
- TCP FIN

Na obrázku 2.3 môžeme vidieť z akých polí sa skladá TCP/IP paket. V TCP hlavičke je možné vidieť niekoľko príznakov označených ako TCP Flags, ktoré označujú stav TCP spojenia.



Obr. 2.3: TCP/IP hlavička paketu [15]

Vytvorenie TCP spojenia sa vždy začína pomocou mechanizmu trojfázovej synchronizácie zvanej tiež ako „trojfázové podávanie ruky“ (angl. three-way handshake), ktorý zahrňuje výmenu paketov SYN, SYN+ACK, ACK, viz. obrázok 2.4 [26]. Najskôr iniciátor komunikácie odošle paket TCP, ktorý má nastavený príznak SYN. Ak je server pripravený na nadviazanie komunikácie odpovie paketom TCP, ktorý má nastavený príznak SYN+ACK, ak nie je pripravený na zahájenie komunikácie odpovie paketom s príznakom RST. Následne ako posledný paket v trojfázovom podaní ruky je odoslaný paket od iniciátora s príznakom ACK a vytvorí sa tak spojenie. Dôvod, prečo sa pozerá na trojcestné overenie je, že pri pasívnej metóde odtlačkov prstov sa využívajú rôzne stavy pripojenia TCP, ako SYN, SYN+ACK, RST a FIN. Ak má paket nastavený jeden z týchto príznakov, tak je možné vykonať detekciu operačného systému. Často stačí prvý paket TCP SYN na určenie operačného systému [12].



Obr. 2.4: Trojfázové podávanie ruky - three-way handshake [6]

Príznaky z hlavičiek TCP/IP paketu, ktoré sa zvyknú používať na detekciu operačného systému a sú používané v nástrojoch pre pasívnu detekciu operačného systému [24]:

- Veľkosť TCP okna (Window) - veľkosť okna TCP segmentu je počet bajtov, ktoré je prijímateľ momentálne ochotný prijať [24].
- IP Time to Live (TTL) - počet povolených smerovacích skokov pred zahodením paketu, hodnota je znížená vždy o jedna každým smerovacím zariadením. Motivácia je zabrániť vzniku smerovacej sľučky [24].
- IP zákaz fragmentácie - Don´t Fragment (DF) - dáva smerovačom pokyn, aby tento paket IP nefragmentovali, ale aby ho vyhodili, ak je pre nasledujúci sieťový segment príliš veľký [24], nachádza sa v IP vrstve v poli Flags.
- TCP Maximálna veľkosť segmentu (MSS) - maximálna veľkosť príjmaného segmentu, ktorú prijímateľ dokáže prijať. Toto pole sa musí odoslať iba v úvodnej žiadosti o pripojenie (t. j. V segmentoch s nastaveným kontrolným bitom SYN). Ak táto možnosť nie je použitá, je povolená ľubovoľná veľkosť segmentu [31]. Nachádza sa v poli TCP options.
- TCP Window Scaling Options Flag - príznak, ktorý označuje, že sa možnosť škálovania TCP používa na získanie väčších okien TCP [24].
- TCP možnosť selektívneho potvrdenia - Príznak, ktorý označuje, kedy bola nastavená možnosť selektívneho potvrdenia TCP [24].
- TCP NOP Option Flag - príznak, ktorý označuje jeden alebo viac NOP, boli pridané na zarovnanie ďalších možností na hranici slova [24].
- Veľkosť paketu - celková veľkosť paketu t. j. TCP hlavička + IP hlavička
- TCP Timestamp Option Flag - príznak, ktorý označuje jednu z časových značiek TCP, časová pečiatka sa niekedy používa na určenie poradia, v ktorom boli pakety odoslané

Ďalšou pasívnou metódou, ktorá sa často využíva na získanie vedomosti o operačnom systéme je metóda získania operačného systému z poľa hlavičky užívateľského agenta (*User-Agent*). Užívateľský agent je reťazec od používateľa, ktorý pochádza z HTTP požiadavky, kvôli poskytnutiu serveru informácie o operačnom systéme a o prehliadači. Účelom takejto vedomosti je, aby webový server mohol poskytnúť obsah prispôsobený pre konkrétne zariadenie alebo softvér a poskytol tak lepšiu užívateľskú skúsenosť [22]. Užívateľský agent patrí do aplikačnej vrstvy sieťovej komunikácie, a preto konštrukcia reťazca je plne pod kontrolou aplikačného softvéru nezávisle od operačného systému. Užívateľský agent by nemal obsahovať zbytočné presné detaily. Príliš dlhé a podrobné hodnoty užívateľského agenta zvyšuje latenciu požiadaviek a riziko identifikácie používateľa. Avšak v praxi je bežné, že užívateľský agent obsahuje podrobné informácie o operačnom systéme ako jeho hlavnú a vedľajšiu verziu a často aj s konkrétnym zostavením tejto verzie [22, 34].

Analýza predchádzajúcich prác

V priebehu rokov vzniklo niekoľko nástrojov či už aktívneho prístupu alebo pasívneho prístupu detekcie operačného systému. Medzi najznámejšie nástroje aktívneho prístupu patria nástroje Nmap [25] a Xprobe2. Nmap je voľne dostupný nástroj, pomocou ktorého je možné zistiť, ktorí koncoví užívatelia sú v sieti k dispozícii, aké služby (názov a verzia aplikácie) títo koncoví užívatelia využívajú a aké operačné systémy používajú. Nmap vysiela až 15 testovacích sond TCP, UDP a ICMP vďaka tomu sa považuje za jeden z najkomplexnejších dostupných nástrojov. Nevýhoda počtu testov je, že vďaka tomu je Nmap ľahko detekovateľný a často blokovaný systémami na detekciu vniknutia alebo bránou firewall [25, 20]. Xprobe2 je ďalší aktívny nástroj, ktorý na detekciu operačného systému, nevyužíva protokol TCP, ale používa kombináciu ICMP protokolu a protokolu UDP.

V pasívnom prístupe sú to nástroje p0f[1], Ettercap, siphon. Nástroj p0f je možno považovať za jeden z najpokročilejších a najznámejších nástrojov pasívnej detekcie operačného systému. Nástroj p0f sa postupne vylepšuje a od vydania v roku 2000 autorom Michalom Zalewskim existujú tri hlavné verzie. Najnovšia verzia p0fv3 využíva okrem príznakov z TCP/IP paketu aj údaje z aplikačnej vrstvy.

Lipmann a kol. [24] sa zameriavajú na hodnotenie bežných nástrojov pasívnej detekcie operačných systémov, ako sú nmap, siphon, p0f a ettercap. Predstavili tabuľku príznakov z TCP/IP hlavičky paketu, ktoré používajú tieto nástroje. Rozoberajú kvalitu rôznych funkcií, počet funkcií, ktoré sa majú použiť na klasifikáciu a rozoberajú štruktúru podpisových databáz operačných systémov, ktoré tieto nástroje poskytujú. Lipmann a kol. tvrdia, že pasívna identifikácia operačného systému z informácií hlavičky TCP/IP paketu je možná, ale nízku chybovosť je možné dosiahnuť len s použitím malého počtu tried. Pomocou metód strojového učenia preukázali, že mnohé z bežných príznakov, ktoré sú použité v bežných nástrojoch pasívneho prístupu, neposkytujú najlepšie výsledky. Najmenšia chybovosť bola dosiahnutá pomocou binárnych stromov za použitia príznakov veľkosti okna TCP, TTL a maximálnej veľkosti segmentu (MSS).

Autori Blake Anderson a David McGrew zo spoločnosti Cisco Systems [3] predstavili nové techniky a ich informačný zisk. Okrem toho sa venovali detekcii zraniteľných operačných systémov a technikám zahmlievania operačných systémov. V tejto štúdií autori porovnali tri techniky na detekciu operačného systému a vytvorili novú metódu, pri ktorej integrovali všetky tieto typy údajov do jedného modelu a vďaka tomu dokázali identifikovať operačný systém a ich hlavnú a vedľajšiu verziu. V metóde získania príznakov z TCP/IP paketu využili príznaky TTL, hodnoty z poľa TCP možnosti (TCP options), maximálnu veľkosť segmentu (MSS) a vlastnosti mierky okna (WS). V metóde TLS využili údaje zo správy ClientHello a v metóde HTTP získali operačný systém z reťazca užívateľského agenta (User-Agent). V práci analyzovali samostatné sieťové toky, pri tom použili vlastný program a pri analýze nahromadených sieťových tokov v časovom okne využili klasifikačnú metódu náhodných lesov (Random forrest). V oboch experimentoch vyškolili tri klasifikátory. Najhoršie výsledky dosahovala metóda TCP/IP, ktorá dosiahla presnosť 55,54 %. Metódy TLS a HTTP dosiahli lepšie výsledky. TLS metóda dosiahla presnosť 76,39 % a metóda HTTP dosiahla najlepšiu presnosť, a to 87,41%. V modely, ktorý zhromažďuje všetky odtlačky prstov z koncového bodu v rámci 60 minútového okna a následne toto okno klasifikuje pomocou náhodných lesov zistili, že tento model zlepšuje TCP/IP a TLS metódu, ale zostáva rovnaký pre HTTP metódu. TCP/IP metóda dosiahla presnosť 62,30 %, metóda TLS 93,67% a metóda HTTP 87,69 %. Zároveň vytvorili model, ktorý je spojením jed-

[1] https://lcamtuf.coredump.cx/p0f3/

notlivých metód a dosiahli presnosť 97,50 %. Experimentami zistili vplyv zmeny veľkosti okna od 5. minút do 55. minút na výkon klasifikátora. TLS a TCP/IP metódy sa výrazne zlepšujú, a to medzi 25. minútou a 40. minútou. Metóda, ktorá vznikla spojením sa zlepšuje neustále, keď sa zväčšuje veľkosť okna, a to od 95,67 % do hodnoty presnosti 97,5 %.

Tyagi a kol.[37] využívajú príznaky z paketu, ktorý je označený príznakom SYN a využívajú vlastnosti ako TTL, dĺžka paketu, veľkosť TCP okna (TCP window size) a možnosti TCP (TCP options). Ich klasifikátor je založený na euklidovskej vzdialenosti a vykazuje 95,5 % presnosť z takmer 2000 paketov SYN. Zistené operačné systémy porovnávajú s podnikovou databázou operačných systémov, aby zistili prítomnosť neoprávneného operačného systému.

Matoušek a kol.[27] predstavili prístup, ktorý používa kombináciu TCP SYN paketov s HTTP hlavičkou. Vylepšujú štandardné záznamy IPFIX o ďalšie informácie o operačnom systéme. Postup, ktorý navrhli je odvodený od princípu, ktorý použil Michal Zalewski v p0f nástroji. Použili podpisovú databázu, ktorá bola založená na databáze podpisov nástroja p0f, pozorovaniach z druhých publikácií a vlastných experimentov. Tabuľka podpisov operačných systémov obsahuje osem príznakov z TCP/IP paketu a k tomu priradený operačný systém. Klasifikácia operačného systému sa vykonáva ak sa nájde dokonalá zhoda aspoň šiestich funkcii paketu SYN. Ak sa takáto zhoda nepodarí, a ak je súčasťou komunikácie HTTP pole užívateľského agenta tak sa na základe tohto údaju vygeneruje nový podpis, ktorý obsahuje vlastnosti paketov a takto sa pridá do tabuľky podpisov operačných systémov. Vďaka tomuto prístupu dokázali vyriešiť problém, ktorý pri pasívnej detekcií operačných systémov nastáva, a to je neaktuálna verzia databázy podpisov. Ako vylepšenie tejto metódy uvádzajú použitie metódy strojového učenia bez učiteľa pomocou metódy K-means. Pri takejto metóde nie je potrebná dokonalá zhoda a ani pevná databáza operačných systémov. Ako najdôležitejšie príznaky uvádzajú celkovú veľkosť paketu SYN, bit značiaci nulovú operáciu, mierku okna (Window scale). V publikácii diskutujú o dopade protokolu IPV6 na pasívnu detekciu operačného systému. Rovnako dopad protokolu IPV6 na pasívnu detekciu operačného systému teoreticky opisuje aj Eckstein [8].

Lastovička a kol.[22] vo svojej publikácii vyhodnotili presnosť dvoch známych metód, a to metódu založenú na odtlačku prsta získaného z TCP/IP paketu a metódu, ktorá získava informáciu o operačnom systéme pomocou HTTP hlavičky, ktorá obsahuje reťazec užívateľského agenta. Pomocou tohto reťazca je možné získať názov operačného systému aj s jeho verziou. Tieto dve metódy doplnili o novú metódu, ktorá identifikuje operačný systém zo špecifických domén. Princíp metódy spočíva v tom, že operačné systémy sa z času na čas pripájajú k špecifickým doménam. Takéto pripájanie môžu vykonávať napríklad kvôli kontrole konektivity alebo pri aktualizáciách operačného systému. Autori porovnali výsledky všetkých troch metód a vytvorili kombináciu týchto metód. Na experimenty využili veľkú dátovú sadu, ktorá bola zozbieraná z univerzitnej bezdrôtovej siete. Najvyššiu presnosť dosiahla metóda, ktorá identifikuje operačný systém z reťazca užívateľského agenta. Táto metóda je presná ale pokrýva len nízku časť prenosu. Pri metóde založenej na príznakoch získaných z TCP/IP paketu sa zamerali na paket s príznakom SYN. V tomto pakete sa zameriavajú na príznaky TTL, veľkosť okna (Window size), a veľkosť celého paketu. Vytvorili tabuľku, ktorá obsahuje zmienené príznaky a k tomu priradený názov operačného systému s jeho hlavnou a vedľajšou verziou. Aby udržali tabuľku aktuálnu, využili prístup, ktorý navrhol Matoušek a kol. [27]. Tabuľka obsahovala 2078 unikátnych záznamov a 51 unikátnych operačných systémov. Svoje výsledky porovnali so známymi nástrojmi p0f a Ettercap. Vyhodnotili, že táto metóda je menej presná ako metóda založená na reťazci užívateľského agenta, ale je schopná pokryť veľkú časť sieťového prenosu. Metódu založenú na špecific-

kých doménach vyhodnotili ako porovnateľnú s ostatnými metódami. Okrem toho vytvorili metódu, ktorá bola kombináciou týchto metód. Táto kombinovaná metóda dosiahla najlepšie výsledky z hľadiska pokrytia, kde dokázala identifikovať 93,4 % operačných systémov. Vo výsledkoch presnosti najlepšie obstála metóda založená na užívateľskom agentovi, ktorá dosiahla presnosť 91 %.

V publikácii [23] autori porovnali štyri známe metódy strojového učenia a zistili, ktorá metóda je najviac vhodná pre klasifikáciu operačného systému. Experimenty vykonali na rozsiahlych reálnych dátach. Pri jednotlivých metódach porovnávali čas, a to z hľadiska času potrebného na trénovanie klasifikátora a čas potrebný na klasifikáciu. Ďalej porovnávali pamäťové požiadavky a presnosť jednotlivých klasifikátorov. Na identifikáciu operačného systému, použili prístup, ktorý je založený na získavaní odtlačku prsta operačného systému z TCP/IP paketu. Príznaky získali z paketu, ktorý je označený ako SYN a zvolili si tri parametre, a to TTL, veľkosť celého paketu, veľkosť okna. Metódy strojového učenia, ktoré sa rozhodli porovnať boli naivný bayesovský klasifikátor, rozhodovacie stromy, metóda K najbližších susedov (KNN), metóda pomocných vektorov (SVM). Pri vyhodnotení najlepšieho času na trénovanie najlepšie obstáli metódy naivný bayesovský klasifikátor a rozhodovacie stromy. Tým postačila menej než jedna sekunda na natrénovanie klasifikátora. Metóda SVM potrebovala 5 minút času na trénovanie a KNN dokonca až 15 minút. V čase, ktorý je potrebný na klasifikáciu opäť obstáli najlepšie metódy naivný bayesovský klasifikátor a rozhodovacie stromy. Zložitosť klasifikátora vôbec neovplyvnila čas klasifikácie týchto dvoch metód, avšak pri metódach SVM a KNN nastalo značné spomalenie. V práci autori vyhodnotili, že metódy KNN a SVM sú náchylné na vytváranie príliš komplikovaných modelov. Z hľadiska vyhodnotenia pamäte obstála najhoršie metóda SVM, ktorá na trénovanie jedného milióna vzoriek spotrebovala 429 MB pamäte. Ostatné tri metódy naivný bayesovský klasifikátor, rozhodovacie stromy a KNN 40MB pamäte. Z týchto výsledkov autori usúdili, že požiadavky na pamäť nie je potrebné brať ako primárny faktor pri rozhodovaní sa, ktorú metódu strojového učenia použiť, pretože novodobé počítačové sady disponujú s oveľa väčšími zdrojmi, ako boli spotrebované.

Pri porovnávaní presnosti klasifikátorov, najlepšie obstáli metódy KNN, SVM a rozhodovacie stromy. Všetky tri metódy dosiahli presnosť 97 %. Najmenšiu presnosť dosiahla metóda naivného bayesovského klasifikátora, ktorá dosiahla presnosť 81 %. Vyhodnotili, že metódu rozhodovacích stromov vnímajú ako najlepšiu voľbu pre klasifikáciu operačných systémov, pretože táto metóda dosiahla vysokú presnosť, rovnakú ako metódy SVM a KNN, ale rozhodovacie stromy dosiahli lepšie výsledky pri vyhodnotení času trénovania klasifikátora a čas potrebný na klasifikáciu. Metódy SVM a KNN sú náročné na čas a pri stúpajúcom počte tréningových vzoriek stúpa aj čas potrebný na trénovanie a klasifikáciu.

Aksoy a kol. [1] využívajú techniky strojového učenia na generovanie nových príznakov, ktoré sú vhodné na identifikáciu operačného systému. Presnejšie využili genetické algoritmy pre získanie príznakov, ktoré budú vhodné pre detekciu operačného systému. Vďaka kombinácii automatického výberu príznakov a algoritmov strojového učenia znížili počet paketových funkcií potrebných na klasifikáciu operačného systému. Vďaka tomuto prístupu sú schopní vykonať klasifikáciu jedného paketu s vysokou presnosťou. Genetický algoritmus dokáže nájsť menšie a tým pádom efektívnejšie príznaky, ktoré je možné použiť na detekciu. Ich prístup je schopný na určenie operačného systému použiť akýkoľvek paket TCP/IP nielen SYN paket. Výsledky ukazujú, že genetický algoritmus významne znižuje počet funkcií paketov, ktoré sú potrebné pri klasifikácii a dokáže tak zvýšiť výkonnosť klasifikácie. Ako ďalšiu výhodu použitia genetických algoritmov uvádzajú nezávislosť od podpisov, ktoré sú vytvorené odborníkmi, ale skôr, že sú tieto podpisy vytvorené vďaka technikám strojového

učenia. Dynamicky sa dokážu prispôsobiť rôznym operačným systémom a ďalším dynamickým zmenám prostredia.

Richardson a kol.[35] identifikujú štyri hlavné výzvy automatickej detekcie OS. V tejto publikácií Richardson a kolegovia nadviazali na skoršiu prácu Caballera a kolegov a zhodnotili presnosť automatických nástrojov na detekciu odtlačkov prstov operačných systémov. Skúmali obmedzenia a výzvy automaticky generovaných nástrojov v realistickejších a náročnejších scenároch. Zistili, že automatické generovanie odtlačkov prstov operačných systémov čelí štyrom hlavným výzvam. Prvou výzvou je, že pre akýkoľvek súčasný nástroj je ťažké nájsť zovšeobecniteľné a dostatočné diskriminačné pravidlá klasifikácie pre rôzne verzie operačných systémov, pretože varianty operačných systémov sú od seba jednoducho ťažko odlíšiteľné. Druhou výzvou je, že neúčinnosť nástrojov v reálnych sieťach. Ďalšími výzvami sú, že plne automatické nástroje nemôžu ľahko využiť sémantické znalosti protokolov a preťaženie detekčných techník.

Al-Shehari a Shahzad [2] využili existujúce metódy porovnávania podpisov operačných systémov a rozšírili ich o techniky strojového učenia. Algoritmus, ktorý navrhli pozostáva z viacerých fáz. V prvej fáze sa príznaky vybraté zo zachytených TCP/IP paketov porovnávajú s existujúcou tabuľkou podpisov operačných systémov. Použili tabuľku nástroja p0f a rozšírili ju o nové príznaky, ktoré boli získané z paketov SYN a paketov, ktoré sú označené príznakom FIN+ACK. Tabuľku prerobili na relačnú databázu, pre lepšie vyhľadávanie. Ak sa nájde zhoda algoritmus nájde operačný systém a ukončí sa. Ak sa zhoda nenájde použije sa strojové učenie a operačný systém sa pokúsi nájsť najbližšiu zhodu pomocou natrénovaného klasifikátora rozhodovacích stromov.

Husák a kol. [16] využívajú prístup, kde sú schopní odhadnúť užívateľského agenta v šifrovanej komunikácii a na základe tohto zistenia sú schopní identifikovať operačný systém. Zo šifrovaného prenosu pomocou protokolov SSL/TLS využívajú zoznam šifrovacích balíkov, ktoré získavajú zo správy ClientHello, ktorá sa posiela pri zahájovaní komunikácie (three-way handshake). Správa ClientHello ešte neobsahuje šifrované údaje, a preto je možné získať informácie o šifrovaných balíkoch. Tieto šifrovacie balíky sa líšia medzi rôznymi klientskymi aplikáciami a ich verziami, vďaka čomu je možné s nimi identifikovať klienta. Spárovali zoznamy šifrovacích balíkov s reťazcom užívateľského agenta, ktorý sa nachádza v HTTP hlavičke. Spárovanie, ktoré vymysleli bolo vytvorené na základe úsudku, že klient komunikuje cez oba protokoly HTTP aj HTTPS. Následne hľadali pre HTTP a HTTPS spojenie, ktoré malo rovnakú zdrojovú IP adresu a z HTTPS použili šifrovacie balíčky a spárovali ich s reťazcom užívateľského agenta z HTTP spojenia, ktoré bolo najbližšie v čase.

Gagnon a Esfandiari [13] popisujú hybridný prístup k detekcií operačného systému. Tento hybridný prístup je kombináciou pasívneho prístupu s aktívnymi prístupmi, kvôli zlepšeniu presnosti identifikácie odtlačkov prstov operačných systémov. Tento hybridný prístup, ale nefunguje, keď sa cieľové zariadenie nachádza za sieťovými zariadeniami, ako napríklad brána firewall alebo zariadenie na preklad adries (NAT). V takom prípade aplikácia nie je schopná poslať sondovacie pakety k cieľu. Okrem toho autori popisujú pasívny a aktívny prístup, nástroje týchto prístupov a ich výhody a limitácie.

V [4] pomocou naivného Bayesovho klasifikátora dokázali pasívne odhaliť operačný systém z hlavičiek paketov. Klasifikátor porovnávali s pasívnymi nástrojmi. V tejto publikácií autori okrem toho identifikujú počet klientov, ktorí sa nachádzajú za zariadeniami na preklad adries (NAT).

Kohno a kol.[21] predstavili prístup kde vedia identifikovať operačný systém na základe skreslenia hodín, ktoré sú zachytené z časových značiek TCP paketu (TCP timestamps options). Vďaka tomuto spôsobu sú schopní identifikovať odtlačok prsta operačného systému,

aj keď je zariadenie vzdialené tisíce kilometrov, a aj keď je zariadenie pripojené k internetu z rôznych miest a pomocou rôznych prístupových technológií. Rovnako tento spôsob môžu použiť, aj keď je zariadenie umiestnené za technológiou NAT alebo za bránou firewall alebo keď je systémový čas udržiavaný prostredníctvom NTP alebo SNTP.

Matunaka a kol. [28] navrhli novú pasívnu metódu na získanie odtlačkov prstov operačných systémov. Metóda si vyžaduje len analýzu DNS prenosu. Metóda na základe DNS požiadaviek, ktoré každý operačný systém odosiela so špecifikovanými vzormi v časových intervalov dokáže rozpoznať operačný systém.

Kapitola 3

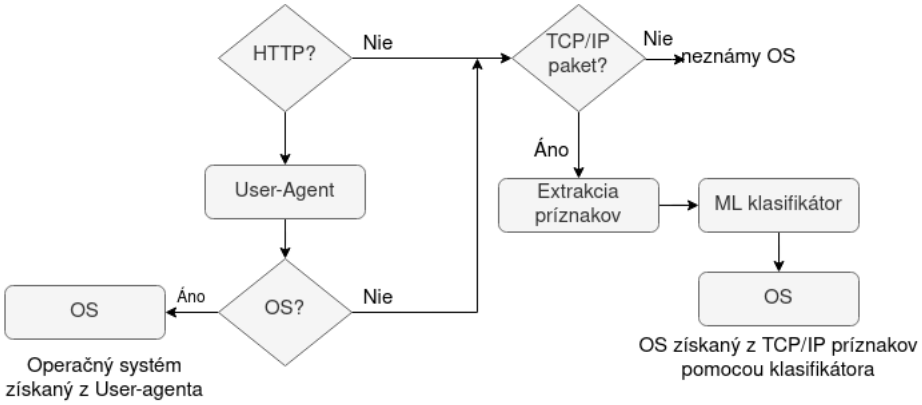
Návrh a implementácia

V tejto kapitole bude popísaná navrhnutá metóda pre identifikáciu operačného systému z monitorovaných sieťových tokov, ktoré boli získané pasívnou metódou monitorovania tokov. Metóda, ktorá bola navrhnutá využíva dva prístupy. Jeden z prístupov, ktorý bol použitý je prístup identifikácie operačného systému na základe príznakov z hlavičky TCP/IP paketu. Druhá metóda je identifikácia operačného systému pomocou reťazca užívateľského agenta.

3.1 Návrh metódy

Metóda, ktorá bola navrhnutá využíva kombináciu dvoch známych metód pre identifikáciu operačného systému. Jedna z použitých metód je metóda, ktorá identifikuje operačný systém na základe parametrov z hlavičiek TCP/IP paketu. Druhá metóda je metóda, ktorá dokáže identifikovať operačný systém z reťazca užívateľského agenta, ktorý sa nachádza v požiadavke HTTP protokolu. Obe metódy majú nejaké výhody a nevýhody a vďaka vytvoreniu metódy, ktorá je kombináciou je možné profitovať z oboch metód a dosiahnuť tak lepšie výsledky.

Spôsob akým je metóda navrhnutá je nasledovný - ak sa v dátach získaných pasívnou metódou monitorovania tokov nachádza reťazec užívateľského agenta z HTTP požiadavky, využi túto vedomosť a identifikuj operačný systém z tohto reťazca. Ak nie je k dispozícií HTTP komunikácia (napríklad komunikácia je šifrovaná) alebo nebolo možné identifikovať operačný systém z užívateľského agenta, tak v tom prípade sa skontroluje či sa jedná o TCP komunikáciu, a ak áno tak sú údaje spracované technikami strojového učenia pomocou natrénovaného klasifikátora, ktorý využije príznaky z hlavičiek TCP/IP paketu a nájde najbližšiu zhodu. Metóda navrhnutá takýmto spôsobom dokáže identifikovať operačný systém zo šifrovaného prenosu a aj z prenosu, ktorý nie je šifrovaný. Metóda je znázornené aj na obrázku 3.1.



Obr. 3.1: Navrhnutá metóda identifikácie operačného systému

Návrh príznakov TCP/IP paketu

Príznaky z hlavičiek TCP/IP paketu, ktoré sú navrhnuté obsahujú príznaky, ktoré výlučne závisia od samotného jadra operačného systému a zvyčajne sa líšia medzi jednotlivými implementáciami jadra to sú príznaky, ako TTL, TCP veľkosť okna, či celková veľkosť paketu SYN. Okrem toho sú využité aj príznaky ako TCP možnosti (TCP options), IP príznaky (IP Flags), zdrojový port. Príznaky, ktoré boli zvolené sú nasledovné:

- IP Time to Live (TTL) - TTL je v IPv4 8 bitové pole v hlavičke IP datagramu. Hodnota určuje koľko najviac uzlov dokáže datagram spracovať, než bude zahodený. Počiatočná hodnota je nastavená operačným systémov vysielajúcej strany, a preto sa môže použiť pri identifikácií opreračného systému. Napríklad operačný systém Windows má často počiatočnú hodnotu TTL nastavenú na hodnotu 128 a operačný systém Linux 64.
- TCP veľkosť okna - veľkosť okna je poľe, ktoré sa nachádza v TCP hlavičke a určuje počet bajtov, ktoré je príjmač momentálne ochotný prijať. Počiatočná hodnota závisí od nastavení operačného systému.
- TCP veľkosť počiatočného paketu SYN - paket SYN je prvým paketom v TCP spojení, ktorý neprenáša žiadne skutočné užívateľské dáta, ale je nevyhnutnou súčasťou pri nadväzovaní TCP spojenia. Veľkosť tohto paketu sa líši medzi rôznymi operačnými systémami, a preto je použitý ako ďalší zdroj informácií pre klasifikáciu.
- Zdrojový port - automaticky pridelený v rámci preddefinovaného rozsahu čísel portov IP zásobníkom operačného systému. Rôzne operačné systémy využívajú rôzny preddefinovaný rozsah čísel portov.
- TCP možnosti (TCP options) - klasifikátor využíva veľkosť tohto príznaku, ktorá sa mení na základe rôznych nastavení TCP.
- IP príznaky (IP FLAGS) - tento príznak využíva informáciu o tom či je príznak DF (Don´t fragment) nastavený alebo nie.

Užívateľský agent

Užívateľský agent je reťazec od používateľa, ktorý pochádza z HTTP požiadavky kvôli poskytnutiu serveru informáciu o operačnom systéme a o prehliadači. Účelom takejto vedomosti je, aby webový server mohol poskytnúť obsah prispôsobený pre konkrétne zariadenie alebo softvér a poskytol tak lepšiu užívateľskú skúsenosť [22]. Výhoda metódy HTTP užívateľského agenta je, že užívateľský agent obsahuje podrobné informácie o operačnom systéme ako jeho hlavnú a vedľajšiu verziu a často aj s konkrétnym zostavením tejto verzie [22, 34]. Prístup identifikácie operačného systému z reťazca užívateľského agenta často dosahuje vysokú presnosť viz. sekcia 2.3 Analýza prechádzajúcich prác. Z obrázka 3.2 je možné vidieť, že z reťazca užívateľského agenta je možné identifikovať operačný systém zariadenia alebo použitý prehliadač.

OS	Prehliadač	User-Agent
Mac OS X	Safari	Mozilla/5.0 (Macintosh; Intel **Mac OS X** 10_11_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/11.1.2 Safari/605.1.17
Ubuntu	Firefox 87	Mozilla/5.0 (X11; **Ubuntu**; Linux x86_64; rv:87.0) Gecko/20100101 Firefox/87.0
Android (Froyo)	Android Browser 4	Mozilla/5.0 (Linux; U; **Android 2.2**) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1

Obr. 3.2: Získaný operačný systém a prehliadač z užívateľského agenta

3.2 Implementácia

Nástroje

Pre implementáciu klasifikačného modelu bola použitá knižnica programovacieho jazyka Python[1] Scikit-learn[2]. Scikit-learn je populárna knižnica strojového učenia, ktorá poskytuje konzistentné aplikačné rozhranie (API) na vysokej úrovni pre efektívnu prácu s algoritmami strojového učenia. Na prácu s dátovou sadou bola použitá knižnica Pandas[3]. Pandas je rýchly, výkonny, flexibilný a ľahko použiteľný nástroj na analýzu a prácu s dátami v jazyku Python. Dátová sada je v knižnici Pandas reprezentovaná objektom, ktorý sa nazýva DataFrame. DataFrame[4] je základný dátový typ knižnice Pandas, jedná sa o efektívne tabuľkové úložisko dát, ktoré je vhodné pre dátovú analýzu alebo pre prácu s algoritmami knižnice Scikit-learn.

[1] https://www.python.org/
[2] https://scikit-learn.org/stable/
[3] https://pandas.pydata.org/
[4] https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Implementácia klasifikátora

Metóda strojového učenia, ktorá bola zvolená v tejto práci pre klasifikáciu operačných systémov je metóda rozhodovacích stromov. Metóda rozhodovacích stromov (angl. Decision tree) vo väčšine publikáciách vykazovala najlepšie výsledky, a napríklad ako už bolo spomenuté v kapitole 2.3 v práci [23] si metóda rozhodovacích stromov najlepšie viedla a dosiahla najlepšie výsledky či už v presnosti, v čase klasifikácie a trénovania alebo v pamäťových požiadavkách a bola vyhodnotená ako najviac vhodná pre klasifikáciu operačných systémov. Preto je metóda rozhodovacích stromov použitá aj v tejto práci.

Anotácia dátovej sady bola získaná z reťazca užívateľského agenta, ktorý však často poskytuje detailné informácie o operačnom systéme nielen na úrovni názvu operačného systému, ale aj na úrovni jeho hlavnej (major) a vedľajšej (minor) verzie. Metóda využívajúca parametre z TCP a IP hlavičiek môže poskytnúť informácie o hlavnej a vedľajšej verzii, ale parametre sú často rovnaké pre viac verzii. Preto sa úrovne vynechajú a klasifikátor identifikuje operačné systémy len na úrovni názvov.

Dátová sada, ktorá bola použitá na trénovanie a vyhodnotenie klasifikátora bola vytvorená z údajov získaných technikou pasívneho monitorovania tokov. Získaná pravda (anotácia dátovej sady) sa získala pomocou metódy identifikácie operačného systému z užívateľského agenta. Operačný systém sa získal z užívateľského agenta pomocou modulu, ktorý bol navrhnutý spoločnosťou CESNET. Klasifikátor využíva príznaky z hlavičiek TCP a IP paketov, a preto je závislý na TCP komunikácií. Komunikácia, ktorá nebola TCP, ale bola iného typu (napríklad UDP, ICMP, a pod.) bola odfiltrovaná. Dátova sada je reprezentovaná Pandas Dataframe objektom. DataFrame bol pred učením klasifikátora rozdelený na anotovanú hodnotu - názov operačných systémov (labels) a na príznaky (features). Príznaky, ktoré boli vybraté boli: TTL (Time-To-Live), veľkosť TCP okna, veľkosť SYN paketu, zdrojový port, TCP možnosti, IP príznaky (IP flags). Hodnoty TTL sa zaokruhľujú na najbližšiu mocninu dvoch, aby sa vylúčil vplyv umiestnenia pozorovacieho bodu [23, 24]. V takejto forme boli dáta vložené do klasifikátora.

Rozdelenie dátovej sady pre trénovanie a testovanie

Dátová sada bola rozdelená na tri časti - trénovaciu, testovaciu, validačnú sadu. Trénovacia sada bola použitá na učenie modelu, validačná sada bola použitá pre ladenie hyperparametrov a testovacia sada pre finálne vyhodnotenie modelu. Rozdelenie dátovej sady prebehlo spôsobom, že najprv bola rozdelená dátova sada na sadu trénovaciu a na testovaciu a to v pomere 80:20, 80 % dát tvorí trénovaciu sadu čo je 263 435 záznamov a 20 % dát tvorí testovaciu sadu, a to je 82 324 záznamov.

Následne aby sa získala validačná sada bola trénovacia sada rozdelená opäť v pomere 80:20. Vo validačnej sade sa nachádza 65 859 záznamov. Rozdelenie sa vykonalo pomocou nástroja z knižnice Scikit-learn `train_test_split`. Ladenie hyperparametrov sa vykonalo na validačnej dátovej sade a pomocou metódy RandomizedSearchCV[5]. RandomizedSearchCV využíva pre ladenie hyperparametrov krížovú validáciu a náhodné vyhľadávanie. Pri náhodnom vyhľadávaní sa definuje vyhľadávací priestor ako ohraničená doména hodnôt hyperparametra a náhodne vzorkuje body v tejto doméne [5]. Po nájdení najlepších hyperparametrov sa natrénuje klasifikátor s danými nájdenými najlepšími hyperparametrami a vyhodnotí sa jeho presnosť. Klasifikátor na vstup berie parametre z TCP a IP hlavičiek a vracia klasifikovaný operačný systém.

[5] RandomizedSearchCV

Kapitola 4

Experimentálne vyhodnotenie

V tejto kapitole sa nachádza popis, analýza a spracovanie dátovej sady, ktorá bola použitá pre experimentálne vyhodnotenie klasifikátora. Okrem toho kapitola obsahuje vyhodnotenie klasifikátora, ktorý bol navrhnutý v predchádzajúcej kapitole, porovnanie klasifikátora s prístupom, ktorý využíva porovnávanie príznakov na základe tabuľky známych príznakov operačných systémov a porovnanie klasifikátora, ktorý neklasifikuje jednotlivé toky, ale klasifikuje IP adresy. Okrem toho sa v tejto kapitole nachádzajú rôzne experimenty s použitím rôznych kombinácii príznakov a experiment s použitím rôznych tried operačných systémov a ďalšie zaujímavé experimenty.

4.1 Dátová sada

V tejto sekcií sa nachádza popis, analýza a spracovanie dátovej sady. Dáta boli získané technikou pasívneho monitorovania tokov a boli poskytnuté spoločnosťou CESNET, takže práca sa nezaoberá monitorovaním dát, ale je potrébne získané dáta spracovať a vytvoriť dátovú sadu, ktorá sa použije na experimentálne vyhodnotenie klasifikátora.

Obsah dátovej sady

Poskytnuté dáta sú získané z dvoch časti, a to z pluginov, ktoré sú označené ako Basic plus a HTTP. Oba pluginy Basic plus aj HTTP exportujú spolu so záznamami, ktoré sú popísané v tabuľkách pre Basic plus viz. tabuľka 4.3 a pre HTTP viz. tabuľka 4.3 aj záznamy zo základného pluginu Basic, ktorý je popísaný viz. tabuľka 4.2. Záznamy z pluginu Basic plus obsahujú polia z hlavičiek TCP/IP paketu, ktoré sú v tejto práci použité pre identifikáciu operačného systému. Plugin HTTP exportuje reťazec užívateľského agenta (User-Agent), ktorý je potrebný pre identifikáciu operačného systému a v tejto práci pre anotáciu dátovej sady. V tabuľkách 4.2 a 4.3 polia s príponou _REV sú polia od cieľa k zdroju, polia bez _REV sú od zdroja k cieľu. Tabuľka 4.1 poskytuje dáta z HTTP protokolu. Zoznam polí a význam polí je získaný z Githubu[1] CESNET IPFIX sondy.

[1] https://github.com/CESNET/ipfixprobe

Tabuľka 4.1: Zoznam polí unirec exportovaných spolu so základnými poľami toku na rozhraní pomocou HTTP pluginu

Pole	**Popis**
HTTP_REQUEST_METHOD	metóda z požiadavky HTTP
HTTP_REQUEST_HOST	hostiteľ z požiadaky HTTP
HTTP_REQUEST_URL	url adresa z požiadavky HTTP
HTTP_REQUEST_AGENT	reťazec User-Agent z požiadavky HTTP
HTTP_REQUEST_REFERER	spostredkovateľ z požiadavky HTTP
HTTP_RESPONSE_STATUS_CODE	návratový kód z odpovede HTTP
HTTP_RESPONSE_CONTENT_TYPE	typ obsahu odpovede HTTP

Tabuľka 4.2: Základné polia exportované z pluginu Basic

Pole	**Popis**
DST_MAC	cieľová MAC adresa
SRC_MAC	zdrojová MAC adresa
DST_IP	cieľová IP adresa
SRC_IP	zdrojová IP adresa
BYTES	počet bajtov v dátovom toku
BYTES_REV	počet bajtov v dátovom toku
LINK_BIT_FIELD	identifikácia exportéru
TIME_FIRST	prvý výskyt toku
TIME_LAST	posledný výskyt toku
PACKETS	počet paketov v dátovom toku
PACKETS_REV	počet paketov v dátovom toku
DST_PORT	cieľový port
SRC_PORT	zdrojový port
DIR_BIT_FIELD	bitové pole, ktoré značí odchádzajúcu alebo prichádzajúcu komunikáciu
PROTOCOL	číslo transportného protokolu
TCP_FLAGS	príznaky TCP protokolu
TCP_FLAGS_REV	príznaky TCP protokolu

Tabuľka 4.3: Zoznam polí unirec exportovaných spolu so základnými poľami toku na rozhraní pomocou pluginu Basic plus.

Pole	**Popis**
IP_TTL	IP TTL
IP_TTL_REV	IP TTL
IP_FLG	príznaky získané z IP hlavičky paketu
IP_FLG_REV	príznaky získané z IP hlavičky paketu
TCP_WIN	TCP veľkosť okna
TCP_WIN_REV	TCP veľkosť okna
TCP_OPT	bitové pole TCP možností
TCP_OPT_REV	bitové pole TCP možností
TCP_MSS	TCP maximálna veľkosť segmentu
TCP_MSS_REV	TCP maximálna veľkosť segmentu
TCP_SYN_SIZE	veľkosť SYN paketu

Analýza dátovej sady a spracovanie

Namerané sieťové toky z oboch pluginov boli uložené do viacerých súborov CSV. Záznamy boli načítané pomocou knižnice Pandas a boli uložené do Pandas DataFramu. Pre získanie veľkej dátovej sady boli tieto záznamy spojené. Spojenie viacerých súborov typu CSV sa vykonalo pomocou vstavaného modulu knižnice Pandas `glob`. Pri takomto spôsobe `glob()` hľadá v zadanom dátovom adresári všetky súbory CSV. Takýmto spojením vznikla dátová sada, kde každý záznam obsahuje exportované údaje z pluginu Basic plus viz. tabuľka 4.3 doplnené o údaje z pluginu Basic viz. tabuľka 4.2. Po spojení dátova sada obsahuje:

- 3 407 970 sieťových tokov
- 314 734 jedinečných IP adries
- 406 zdrojových MAC adries

Z dátovej sady boli vynechané všetky stĺpce, ktoré sú nepodstatné pre identifikáciu operačného systému. Vynechané stĺpce teda boli všetky stĺpce, ktoré sú označené ako _REV a označujú spätnú komunikáciu (t. j. od cieľa k zdroju), cieľová IP adresa a všetky ostatné stĺpce, ktoré sú pre identifikáciu operačného systému nepodstatné. Toky v opačnom smere niesu pre identifikáciu operačného systému zaujímave, a preto boli vynechané.

Anotácia dátovej sady bola získaná pomocou metódy zisťovania operačného systému z reťazca užívateľského agenta. Pre vytvorenie takejto anotácie bolo potrebné spojiť všetky súbory CSV, ktoré sú získané z pluginu HTTP. Po spojení týchto CSV súborov vznikla dátová sada, ktorá obsahuje 152 670 záznamov, kde každý záznam obsahuje údaje, ktoré sú popísané v tabuľke 4.1 doplnené o údaje z tabuľky 4.2. Z druhého DataFramu sú ponechané len stĺpce:

- SRC_IP
- HTTP_REQUEST_AGENT

Následne DataFrame, ktorý obsahuje záznamy s hodnotami SRC_IP (zdrojová IP adresa) a UA (reťazec User-Agent) je uložený do CSV súboru a poslaný do modulu, ktorý z reťazca užívateľského agenta identifikuje operačný systém.

Modul, ktorý zisťuje operačný systém z reťazca užívateľského reťazca je implementovaný spoločnosťou CESNET a vracia záznamy, ktoré obsahujú reťazec užívateľského agenta a k tomu zistený operačný systém. Ak sa operačný systém nepodarilo identifikovať z nájdeného reťazca užívateľského agenta, operačný systém je označený ako „None“. V dátovej sade sa podarilo nájsť 667 jedinečných reťazcov užívateľského agenta a 35 operačných systémov. Následne sa s pomocou Pandas funkciou `merge()`, ktorá funguje ako databázový `join` vytvorí DataFrame, ktorý obsahuje stĺpce:

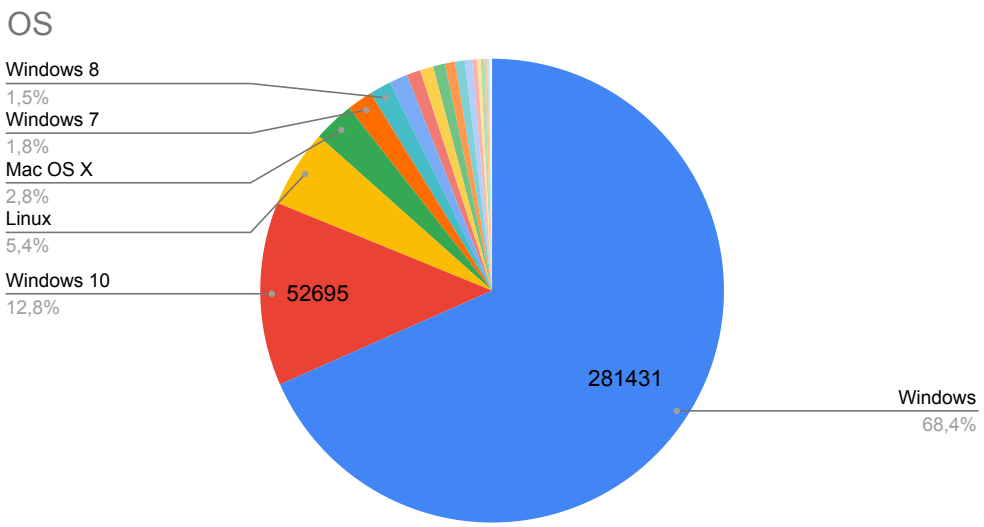
- IP - zdrojová IP adresa
- OS - operačný systém zistený z reťazca užívateľského agenta

Takýto Dataframe obsahuje 1873 záznamov a z toho je 1683 unikátnych IP adries. To znamená, že na niektorých IP adresách bolo detekovaných viac operačných systémov. To môže znamenať, že v sieti na danej IP adrese bolo detekované zariadenie na preklad adries (NAT) alebo na zariadení napríklad beží virtuálny stroj. Z analýzy sa zistilo, že takýto prípad nastáva v 18 % prípadoch a v 6 % sa na danej IP adrese zistilo viacej operačných systémov, ktoré neboli z rovnakej triedy operačných systémov. Spájanie takýchto tokov by bolo problematické a nie vždy by to bolo možné. Avšak z analýzy sa zistilo, že takýto problém nenastáva často, a preto takéto záznamy môžu byť vynechané, vymažú sa duplikované IP adresy. Vznikne tak DataFrame, ktorý ma 1683 záznamov a obsahuje 1683 unikátnych IP adries, kde každá IP adresa má zistený jeden operačný systém. Takýto Dataframe sa spojí na základe IP adresy s DataFramom, ktorý obsahuje TCP/IP parametre a vznikne tak dátová sada, ktorá má 3 407 970 záznamov. Každý záznam obsahuje informáciu o IP adrese, TCP/IP parametroch a k tomu zistený operačný systém. Dátová sada obsahuje nasledovné stĺpce:

- IP
- TCP OPT - TCP možnosti (TCP options)
- TCP MSS - maximálna veľkosť segmentu
- SRC PORT - zdrojový port
- TCP SYN SIZE - veľkosť paketu SYN
- TCP WIN - veľkosť okna TCP segmentu
- IP FLG - príznaky z hlavičky IP paketu.
- IP TTL - hodnota Time to Live z hlavičky IP paketu.
- PROTOCOL - číslo transportného protokolu.
- TCP FLAGS - príznaky z hlavičky TCP segmentu.
- OS - operačný systém zistený z užívateľského agenta.

Dátova sada obsahuje 65 % záznamov, ktoré sa nepodarilo identifikovať pomocou užívateľského agenta. Záznamy, ktoré sa nepodarilo identifikovať sú v stĺpci OS označené reťazcom „None“. Predtým ako sa dátová sada rozdelí na trénovaciu sadu a na testovaciu, záznamy, ktoré sú v stĺpci OS označené ako „None“ musia byť vynechané. Po vynechaní takýchto záznamov dátova sada obsahuje 1 181 739 záznamov.

Keďže metóda založená na identifikácii operačného systému z parametrov z hlavičiek TCP/IP paketu sa spolieha na spracovanie parametrov TCP/IP každého toku a dátova sada obsahuje reálny prenos, parametre TCP/IP paketu nemusia byť prítomné vo významnej časti tokov (t. j. prenos UDP, ICMP, a pod.). Takéto toky sú identifikované pomocou stĺpca PROTOCOL, ktorý udáva číslo transportného protokolu. Toky, ktoré nemali číslo transportného protokolu 6, čo reprezentuje TCP, boli vynechané, pretože sú nepoužiteľné pre metódu TCP/IP parametrov. Následne sa hodnoty TTL zaokrúhľujú na najbližšiu mocninu dvoch, aby sa vylúčil vplyv umiestnenia pozorovacieho bodu[23, 24]. Takýmto spôsobom sa zredukovala dátová sada na 411 618 záznamov a 1683 jedinečných IP adries, kde každý záznam obsahuje všetky potrebné parametre na idenfikáciu operačného systému a stanovenú základnú pravdu. V tabuľke 4.4 je možné vidieť obsah operačných systémov v dátovej sade. Na koláčovom grafe je možné vidieť, ktoré operačné systémy tvoria najväčšiu časť prenosu.



Obr. 4.1: Obsah stĺpca OS - operačné systémy nájdene v dátovej sade pomocou užívateľského agenta

Tabuľka 4.4: Zoznam operačných systémov v dátovej sade

Názov OS	**Počet**
Windows	281431
Windows 10	52695
Linux	22255
Mac OS X	11674
Windows 7	7371
Windows 8	6315
Ubuntu	5182
Android (Nougat)	3975
Windows XP	3839
Android 10	3451
Android (Pie)	2881
openSUSE Linux	2793
Windows 8.1	2059
CentOS	1416
Windows Vista	1060
Mac OS X (Lion)	1053
Windows 98	813
Android (Lollipop)	512
Android	343
Debian	298
Orbis OS	168
Android (Marshmallow)	13
Windows 2000	11
Mac OS X (El Capitan)	5
Android (Ice Cream Sandwich)	3
Fedora	2

Ako je možné vidieť v tabuľke 4.4 dátová sada obsahuje 26 tried operačných systémov. Anotácia dátovej sady bola vykonaná pomocou reťazca užívateľského agenta, preto je možné vidieť rozdelenie operačných systémov aj na úrovni ich hlavnej a vedľajšej verzie. Operačné systémy boli rozdelené na menšie triedy a identifikácia operačného systému sa vykonáva len na úrovni názvov operačných systémov. Klasifikátor rozlišoval v dátovej sade nasledovné triedy operačných systémov - Windows, Linux, Mac OS X, Android, Ubuntu, openSUSE Linux, CentOS, Debian, OrbisOS.

4.2 Meranie úspešnosti klasifikácie

Klasifikátor, ktorý bol navrhnutý spadá do kategórie viactriednych klasifikátorov s `k` triedami. Pre merania úspešnosti klasifikácie sa vypočítali metriky, ako presnosť (accuracy), precíznosť (precision), senzitivita (recall) a f-skóre podľa [36]. Distribúcia tried operačných systémov v dátovej sade nie je jednotná a sú v nej veľké rozdiely. Preto pre vyhodnotenie je použitá technika mikro priemerovania (micro-averaging), ktorá sa používa, keď je v dátovej sade nerovnováha tried. V takom prípade triedy s väčším počtom pozorovaní budú mať väčší vplyv na metriku. Všetky hodnotiace metriky pre viactriedne klasifikátory je možno

chápať v kontexte binárneho modelu klasifikácie, kde sú triedy jednoducho „pozitívne“ a „negatívne“. Tieto metriky sú odvodené z nasledujúcich štyroch kategórii:

- **True positive (TP)** - klasifikátor predikoval správne, že vzorka patrí do triedy, do ktorej skutočne patrí.
- **True negative (TN)** - klasifikátor predikoval správne, že vzorka nepatrí do triedy, do ktorej skutočne nepatrí.
- **False positive (FP)** - klasifikátor predikoval, že vzorka patrí do triedy, ale v skutočnosti do nej nepatrí.
- **False negative (FN)** - klasifikátor predikoval, že vzorka nepatrí do triedy ale v skutočnosti do nej patrí.

Jednotlivé metriky merania úspešnosti využívali nasledovné rovnice:

$$Precision_{\mu} = \frac{\sum_{i=1}^{k} TP_i}{\sum_{i=1}^{l}(TP_i + FP_i)} \tag{4.1}$$

$$Recall_{\mu} = \frac{\sum_{i=1}^{l} TP_i}{\sum_{i=1}^{k}(TP_i + FN_i)} \tag{4.2}$$

$$F - score_{\mu} = \frac{2 * Precision_{\mu} * Recall_{\mu}}{Precision_{\mu} + Recall_{\mu}} \tag{4.3}$$

$$AverageAccuracy = \frac{1}{k}\sum_{i=1}^{k} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \tag{4.4}$$

Presnosť (Precision) udáva schopnosť klasifikátora neoznačiť vzorku ako pozitívnu, keď je negatívna. Senzitivita (Recall) je schopnosť klasifikátora nájsť všetky pozitívne vzorky. F-skóre je kombinácia presnosti a senzitivity. F-skóre sa počíta pomocou priemeru, ale nie obvyklého aritmetického priemeru, ale používa sa harmonický priemer. Presnosť (accuracy) udáva aký podiel tried pozitívnych aj negatívnych bol správne klasifikovaný.

Matica zámeny

Ako ďalší spôsob merania úspešnosti je pomocou Matice zámeny (ang. confusion matrix). Matica zámeny sa používa pre zachytenie celkového prehľadu výsledkov klasifikačného modelu. Matica zámeny je dvojrozmerná matica, ktorá sa používa pri klasifikácii dvoch alebo viacerých tried. Jedným rozmerom matice zámeny sú skutočné hodnoty a druhým rozmerom sú predikované hodnoty klasifikátorom. Ideálny prípad je taký, keď sa na diagonále nachádzajú hodnoty a ostatné hodnoty sú nulové. V taktomto prípade sú predikované triedy rovnaké ako skutočné. Z matice zámeny je taktiež možné vyčítať, ktoré triedy sa často zamieňajú.

Klasifikačný report

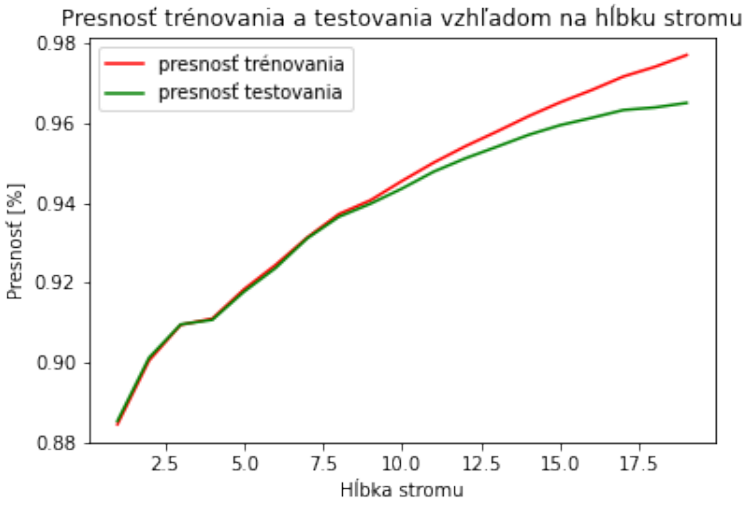
Ďalší spôsobom vyhodnotenia úspešnosti klasifikátora, ktorý bude v tejto práci často používaný je klasifikačný report. Výpis klasifikačného reportu je získaný pomocou Scikit-learn funkcie `classification_report`. Klasifikačný report poskytuje hlavné metriky klasifikácie ako precíznosť (precision), senzitivitu (recall), f-skóre. Vo výpise sa nachádza výpočet metrík pre každú triedu ale aj celkový výsledok. Celkový výsledok poskytuje priemery mikro priemerovania (micro-averaging) v tabuľké označené ako *presnosť* a v prípade mikro priemerovania platí: $mikro - F1 = mikro - precznos = mikro - senzitivita = presnos$, čo predstavuje celkovú presnosť klasifikátora: podiel správne klasifikovaných vzoriek zo všetkých vzoriek. Okrem toho klasifikačný výpis obsahuje aj makro priemer, ktorý vypočíta metriku nezávisle pre každú triedu a potom vypočíta priemer (zaobchádza so všetkými triedami rovnako) a vážený priemer (weighted-average), ktorý váži skóre každej triedy počtom vzoriek z tejto triedy.

4.3 Vyhodnotenie

Vyhodnotenie úspešnosti klasifikátora, ktorý bol navrhnutý v tejto práci prebiehalo na testovacej sade, ktorá obsahovala 82 324 záznamov. Príznaky, ktoré boli použité sú nasledovné : TCP OPT, SRC PORT, TCP WIN, TTL, IP FLG, TCP SYN SIZE.

Hĺbka stromu

Na obrázku 4.2 je možné vidieť, že so zvyšujúcou sa hĺbkou stromu sa zvyšuje aj presnosť klasifikátora.



Obr. 4.2: Grafické znázornenie presnosti klasifikátora na základe hĺbky stromu

5-násobná krížová validácia

Rozedelním dátovej sady do troch sád sa zníži počet vzoriek, ktoré je možné použiť na trénovanie modelu a výsledky môžu závisieť od konkrétnej náhodnej voľby pre dvojicu (trénovacia, validačna) množin. Riešením tohto problému je postup, ktorý sa nazýva K-násobná križová validácia. Pri K-násobnej krížovej validácii sa dátová sada najskôr rozdelí na k rovnako alebo takmer rovnako veľké časti. Väčšinou sa za k volí 5 alebo 10. V tom pripade sa bavíme o 5-násobej krížovej validácii alebo o 10-násobnej krížovej validácii. Krížová validácia následne funguje s princípom, že sa iteruje cez menšie časti a pri každej iterácii sa použije jeden z k menších častí ako validačná sada a ostatné k-1 časti sa použijú na učenie. Tento proces sa opakuje, pokým sa každá menšia časť nepoužije ako validačná sada. Natrénovaním a mnohonásobným testovaním modelu na k rôznych podmnožinách rovnakých tréningových údajov je možné získať presnejšiu predstavu o tom, ako dobre môže model fungovať na dátach, ktoré predtým nevidel. V K-násobnej križovej validácii sa hodnotí model po každej iterácii a na konci sa vypočíta priemer všetkých výsledkov. Krížová validácia sa často využíva pri hodnotení viacerých modelov alebo pri ladení hyperparametrov [33]. V tabuľke 4.5 je možno vidieť presnosť klasifikácie v jednotlivých iteráciach 5-násobnej krížovej validáce. Riadok Priemer vyjadruje priemernú presnosť 5-násobnej krížovej validácie.

Na obrázku 4.3 je možné vidieť maticu zámeny, z ktorej je možné vyčitať ďalšie informácie.

Tabuľka 4.5: Presnosť v [%] klasifikácie v 5-tich iteráciach krížovej validácie

Iterácia	**Presnosť [%]**
I1	93.3
I1	93.8
I1	94.3
I1	93.5
I1	92.8
Priemer	93.5

Klasifikačný report

Klasifikačný report, ktorý je v tabuľke 4.6 obsahuje výpočet precíznosti (Precision), senzitivity (Recall), F-skóre pre každú triedu. Okrem toho riadok `presnosť` vyjadruje celkovú presnosť pomocou techniky mikro priemerovania.

Tabuľka 4.6: Klasifikačný report

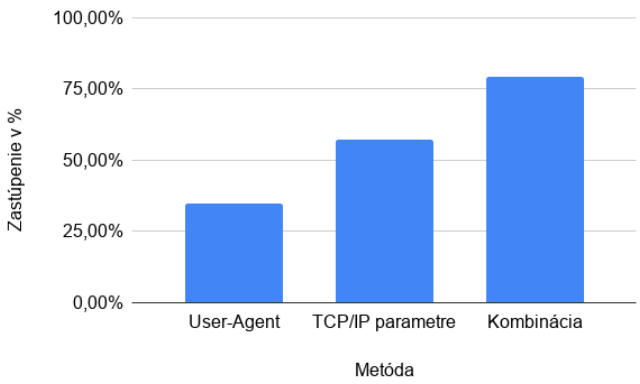
	Precíznosť [%]	**Senzitivita** [%]	**F-skóre** [%]]
Android	80	62	70
CentOS	80	77	78
Debian	56	30	39
Linux	80	88	84
Mac OS X	87	83	85
Orbis OS	78	62	69
Ubuntu	73	53	61
Windows	98	99	99
openSUSE Linux	64	56	60
presnosť	96	96	96
makro priemer	77	68	72
vážený priemer	96	96	96

Obr. 4.3: Matica zámeny pre jednotlivé triedy OS

Ako je možné vidieť na obrázku 4.3 je matica zámeny, kde na ose Y sa nachádzajú skutočné hodnoty a na ose X je možné vidieť ako boli tieto hodnoty predikované klasifikátorom. V idealnom prípade by boli hodnoty len na diagonále a ostatné hodnoty by boli nulové. Z matice zámeny je možné vyčítať, ktoré triedy operačných systémov sa často zamieňali. Ako je možné vidieť často dochádzalo k zámene medzi operačným systémom Debian a operačným systémom Linux. Andorid operačný systém sa celkovo 800 krát zamenil s operačným systémom Windows. Keď sa pozrieme na klasifikačný report, ktorý je v tabuľke 4.6 môžeme vidieť,že precíznosť triedy Debian je iba 56 %, senzitivita 30 % a f-skóre 39 %, to pretože sa Debian často zamieňal za Linux OS. Ako je ale možné vidieť klasifikátor dosiahol celkovú presnosť (accuracy) 96% s využitím techniky mikro priemerovania. Takáto presnosť je dosiahnutá kvôli vysokému počtu Windows operačných systémov, ktoré dosiahli vysokú presnosť, senzitivitu a aj f-skóre. Počet Windows operačných systémov je v dátovej sade naozaj veľký, oproti druhým operačným systémom. V testovacej sade sa nachádzalo 82 322 záznamov a z toho 71 119 patrilo operačnému systému Windows, ktorý sa dokázal 70 504 správne predikovať. V experimentálnej časti sa nachádza experiment, v ktorom sú Linux operačné systémy (CentOS, Debiam, Ubuntu, openSUSE Linux) zjednotené do jednej triedy Linux. Vďaka takémuto zjednoteniu sa zvýšila presnosť na 97 % viz. tabuľka 4.8.

Vyhodnotenie pokrytia jednotlivých metód

V tomto experimente je zmerané, akú veľkú časť prenosu sú schopné jednotlivé metódy na identifikáciu operačného systému identifikovať. Experiment bol vykonaný na experimentálnej dátovej sade, ktorá bola popísaná v sekcii 4.1. Dátová sada obsahuje 3 407 970 záznamov. Metóda, ktorá identifikuje operačný systém pomocou reťazca užívateľského agenta potrebuje požiadavku HTTP s užívateľským agentom, ktorý obsahuje informáciu o operačnom systéme. V dátovej sade sa nachádzalo až 65 % záznamov, ktoré neobsahovali takéto informácie a pokrytie tejto metódy je preto 34,60 %. Metóda, ktorá využíva na identifikáciu operačného systému parametre z hlavičiek TCP/IP paketu vyžaduje TCP spojenie. V dátovej sade sa nachádzala aj iná komunikácia ako len komunikácia TCP (napríklad ICMP, UDP) a teda len 57 % záznamov obsahovalo TCP spojenie. Kombinácia týchto metód je neúčinná v prípade ak záznam neobsahuje TCP spojenie a zároveň ani informáciu o User-agentovi (t. j. nieje možné ani z jednej metódy identifikovať operačný systém). Pokrytie kombinovanej metódy je 79 %. Súhrn pokrytia je možné vidieť aj na obrázku 4.4



Obr. 4.4: Pokrytie jednotlivý metód OS identifikácie

4.4 Experimenty

V tejto sekcii sa nachádza niekoľko experimentov, ktoré uverujú presnosť klasifikácie za rôznych podmienok.

Experiment č.1 - rôzny počet tried

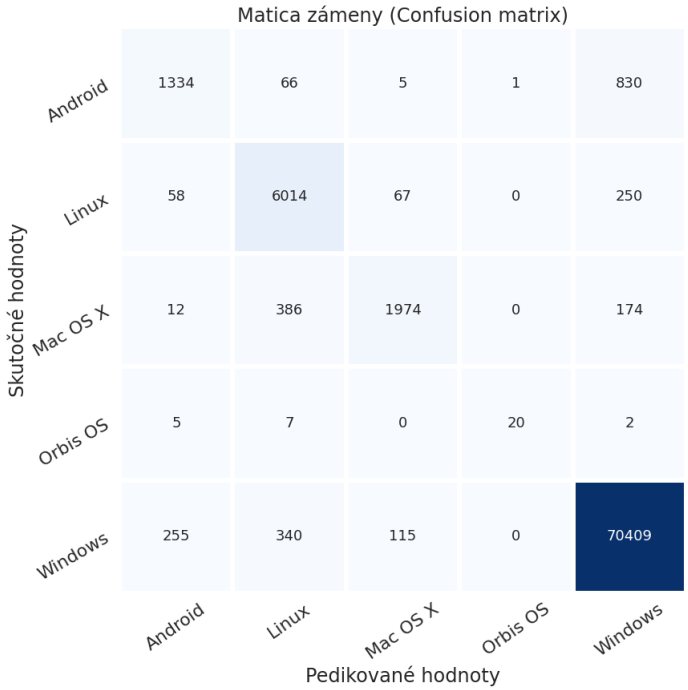
V tomto experimente sa vyhodnotí klasifikátor pri rôznom počte tried. V prvej časti experimentu, klasifikátor klasifikoval 26 tried operačných systémov. V tejto časti neboli operačné systémy rozdelené do menších tried a klasifikovali sa aj na úrovni ich hlavnej a vedľajšej verzii. O aké triedy operačných systémov išlo je možné vidieť v tabuľke 4.1. V tejto časti bola úspešnosť klasifikátora 81 %. Priemerný výsledok 5-násobnej krížovej validácie bol 74 %. V druhej časti sa spojili do jednej triedy Windows všetky operačné systémy Windows. Vzniklo tak 18 tried a priemerná úspešnosť 5-násobnej krížovej validácie bola 93,8 % a úspešnosť bola 96 %. V ďalšej časti sa operačné systémy rozdelili na 10 tried, a to podľa úrovni názov operačných systémov. Priemerný výsledok krížovej validácii bol opäť 93,9 % a úspešnosť 96 %. Najhoršie tu dopadli triedy OS Ubuntu a Debian CentOS, openSUSE Linux, ktoré sa často zamienali za Linux. Preto posledným spojením bolo spojenie Linuxových operačných systémov. Vzniklo tak už iba päť tried Windows, OrbisOS, Mac OS X, Android, Linux. Priemerná presnosť 5-násobnej krížovej validácie je 95 % a celková presnosť je 97 %. Klasifikačný report pre týchto päť tried sa nachádza v tabuľke 4.8 a matica zámeny na obrázku 4.5. Súhrn výsledkov pri rôznom počte tried je možné vidieť v tabuľke 4.7. Stĺpec Priemer 5-KV označuje priemerný výsledok 5-násobnej krížovej validácie.

Tabuľka 4.7: Súrh výsledkov pri rôznom počte tried

Počet tried	**Priemer 5-KV** [%]	**Presnosť** [%]
26	74	81
18	93,8	96
10	93.8	96
5	95	97

Tabuľka 4.8: Klasifikačný report

	Precíznosť [%]	**Senzitivita** [%]	**f1-skore** [%]
Android	80	60	68
Linux	88	94	91
Mac OS X	91	78	84
Orbis OS	95	59	73
Windows	98	99	99
presnosť	97	97	97
makro priemer	91	78	83
vážený priemer	97	97	97



Obr. 4.5: Matica zámeny pre 5 tried OS

Experiment č.2 - bez zmeny TTL hodnôt

Hodnoty TTL príznaku neboli zaokrúhlené na najbližšiu mocninu dvoch, ale hodnota bola ponechaná tak ako bola nameraná a interval si zvolil sám algoritmus rozhodovacích stromov. Priemerná presnosť 5-násobnej krížovej validácie je 94,6 % a celková presnosť je 97 %. Pri zmene TTL hodnôt na najbližšiu mocninu dvoch bol priemer 5-násobnej krížovej validácie 93,5 % a celková presnosť 96 %. Zhrnutie je možné vidieť v tabuľke 4.9.

Tabuľka 4.9: Tabuľka sumarizuje úspešnosť klasifkátora, bez zmeny TTL príznaku

Priemer 5-KV	94,6 %
celková presnosť	97 %

Experiment č.3 - rôzna kombinácia príznakov

Tento experiment je založený na experimentovaní s počtom príznakov a s rôznymi kombináciami príznakov. Príznaky, ktoré sú k dispozícií sú:

- TCP OPT (TCPOpt) - možnosti v hlavičke TCP segmentu
- TCP MSS - maximálna veľkosť segmentu
- SRC PORT (SrcPort) - zdrojový port
- TTL - Time To Live

- SYN SIZE (SynSize)- celková veľkosť SYN paketu
- TCP WIN -(TCP Win) veľkosť okna TCP
- IP FLG - príznaky (flags) z hlavičky IP paketu
- TCP FLAGS (TCPflags) - príznaky z hlavičky TCP

Tabuľka 4.10 obsahuje vyhodnotenie úspešnosti klasifikátora pri použití rôznych príznakov. Stĺpec Priemer 5-KV vyjadruje priemernú hodnotu 5-násobnej krížovej validácie, stĺpec celková presnosť vyjadruje celkovú presnosť technikou mikro priemerovania. Z tabuľky 4.10 je možné vidieť, že najlepšiu presnosť sa dá dosiahnúť použitím všetkých príznakov. Naopak najhoršie si klasifikátor viedol pri použití kombinácie IP príznakov (IPflg), TTL, TCP okna (TCP Win) a pri kombinácii TCP možnosti (TCPOpt), TTL a TCP príznakov (TCPflags). Pri použití malého počtu príznakov (napríklad TCP Win, TTL, SynSize) sa triedy medzi sebou často zamieňali. Pri použití väčšieho počtu príznakov sa triedy dokázali medzi sebou lepšie rozlíšiť a došlo k menšiemu počtu zamenení.

Tabuľka 4.10: Tabuľka znázorňujúca úspešnosť pri použití rôznych príznakov

	Priemer 5-KV [%]	**celková presnosť** [%]
TCP Win, TTL, SynSize	92	93
SrcPort ,TCPOpt, TCP Win	93	95
SrcPort, SynSize, TCP Win	93	95
TCP Win, TTL	90	91
SrcPort, TTL, TCP Win	91	94
TCOpt, TCPflags, TTL	89	90
IPflg, TTL, TCP Win	88	89
Všetky	93,7	97

Experiment č.4 - porovnanie tabuľky a klasifikátora

V tomto experimente sa porovnáva klasifikátor a prístup, ktorý nevyužíva žiadny algoritmus strojového učenia, ale identifikuje operačný systém na základe tabuľky známych príznakov operačných systémov. Tabuľka príznakov operančných systémov, ktorá bola použitá je odvodená z publikácie [22] a doplnená o vlastné experimenty a o informácie z iných publikácii. Tabuľka príznakov obsahuje 212 záznamov a využíva tri typy príznakov z TCP a IP hlavičiek: Veľkosť paketu SYN (Syn Size), veľkosť okna TCP paketu (Win Size) a TTL (Time to Live), kde každý záznam má pridelený operačný systém (OS). Záznamy sú znázornené v tabuľke 4.11. Keďže tabuľka obsahuje len tri príznaky, bolo potrebné klasifikátor natrénovať len na týchto troch príznakoch. Ako testovacia sada sa využila sada, ktorá obsahovala 43 512 záznamov. V tabuľke príznakov sa nachádzali operačné systémy aj s ich hlavnou verziou (napríklad Windows 10, Windows 8). Keďže klasifikátor bol natrénovaný na klasifikáciu operačných systémov len na úrovni názvov OS tak sa zmenili operačné systémy z tabuľky len na túto úroveň.

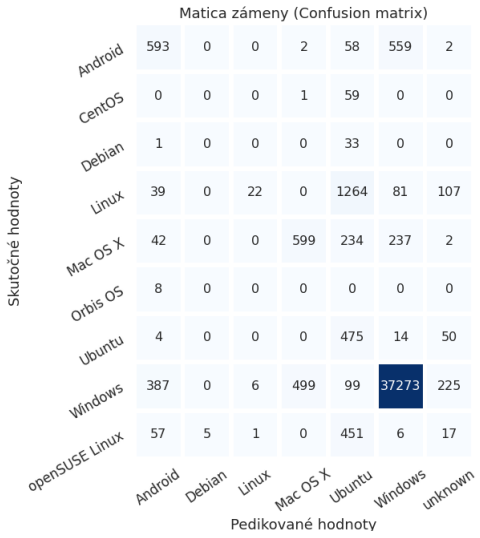
Tabuľka 4.11: Tabuľka známych príznakov TCP/IP parametrov operačných systémov

Syn Size	Win Size	TTL	OS
48	16384	128	Windows XP
48	17520	64	Windows 10
49	29200	64	Linux
60	14600	128	Android (Jelly Bean)
60	28640	64	Ubuntu
52	14600	64	Mac OS X (El Capitan)

Popis metódy využívajúcu tabuľku príznakov

Algoritmus metódy využívajúcu tabuľku známych príznakov funguje na princípe, že sa prechádza cez dátovú sadu a porovnávajú sa jednotlivé hodnoty príznakov s hodnotami v tabuľke. Aby došlo k identifikácii operačného systému je potrebné aby nastala zhoda vo všetkých troch príznakoch. Ak zhoda vo všetkých troch príznakoch nenastane operačný systém je označený za neznámy „unknown“.

V prvej časti experimentu sa porovnávala tabuľka so skutočnými hodnotami testovacej sady, ktoré boli získané z reťazca užívateľského agenta, opäť upravené len na úroveň názvov operačných systémov. Pri tomto experimente boli výsledky zhodné na 89 %. V druhej časti sa porovnávala tabuľka s predikovanými hodnotami klasifikátora a výsledky boli zhodné na 91 %. Z matice zámeny na obrázku 4.6 je možné vidieť, ktoré operačné systémy sa najčastejšie zamieňali. Napríklad operačný systém Linux sa často zamieňal s operačným systémom Ubuntu a operačný systém Android s operačným systémom Windows. V matice zámeny sa nachádza trieda unknown, ktorá vyjadruje, že operačný systém nebol rozpoznaný pomocou tabuľky (t. j. príznaky, ktoré sa porovnávali neboli nájdené v tabuľke).



Obr. 4.6: Matica zámeny reprezentujúca úspešnosť predikcie algoritmu využívajúceho tabuľku príznakov

Vyhodnotenie experimentu

Vyhodnotenie tohto experimentu je nasledovné - identifikácia operačného systému pomocou tabuľky známych príznakov je možná, avšak niektoré operačné systémy sa často zamieňali riešením by bolo zvoliť možno ďalšie príznaky, ktoré lepšie rozlíšia operačné systémy alebo niektoré operačné systémy spojiť do jednej triedy (napríklad Linux a Ubuntu). Takže z toho vyplýva, že vysokú presnosť je možné dosiahnuť len s použitím menšieho počtu tried. Na rovnaké zistenie prišili aj Lippmann a kol. [24]. Ďalšou nevýhodnou je, že ak nenastane zhoda vo všetkých troch príznakoch tak je operačný systém označený za „unknown“. Riešením tohto problému je aktualizovať tabuľku o nové záznamy, napríklad pomocou reťazca užívateľského agenta ako navrhli Matoušek a kol. v [27]. Klasifikátor využívajúci tabuľku dosiahol úspešnosť 89 % a klasifikátor využívajúci rozhodovacie stromy 93 %. Tieto dva klasifikátory sa zhodovali na 91 %.

Experiment č.5 - Multi Flow

V tomto experimente bol navrhnutý klasifikátor, ktorý namiesto klasifikovania každého toku (spojenia) klasifikuje skupinu tokov. Z každej zdrojovej IP adresy sa vezmú všetky toky, ktoré prináležia tejto zdrojovej IP adrese a namiesto príznakov (features) TCP OPT, SRC PORT, TCP WIN, TTL, IP FLG, TCP SYN SIZE sa vypočítajú štatistické funkcie z hodnôt týchto príznakov. Štatistické funkcie, ktoré boli navrhnuté sú minimum, maximum, priemer a 10 a 90 percentil.

Úspešnosť takéhoto klasifikátora pri 8 triedach operačných systémov (Windows, openSUSE Linux, Android, Linux, Ubuntu, CentOS, Debian, Mac OS X) je po vyladení hyperparametrov 72 %. Takáto menšia úspešnosť klasifikátora je dosiahnutá kvôli tomu, že operačné systémy Linux, Ubuntu, Debian, CentOS, openSUSE Linux sa medzi sebou zamieňali a boli klasifikátorom predikované ako Linux alebo openSUSE Linux. Naopak Andorid trieda dosiahla presnosť 97 %. Klasifikačný report tohto vyhodnotenia je možné vidieť v tabuľke 4.12, kde je možné vidieť, že niektoré triedy majú nulové vyhodnotenie, to preto lebo boli klasifikátorom predikované ako operačný systém Linux. V ďalšom kroku sú operačné systémy Ubuntu, Debian, CentOS, openSUSE Linux, Linux spojené do jednej triedy Linux.

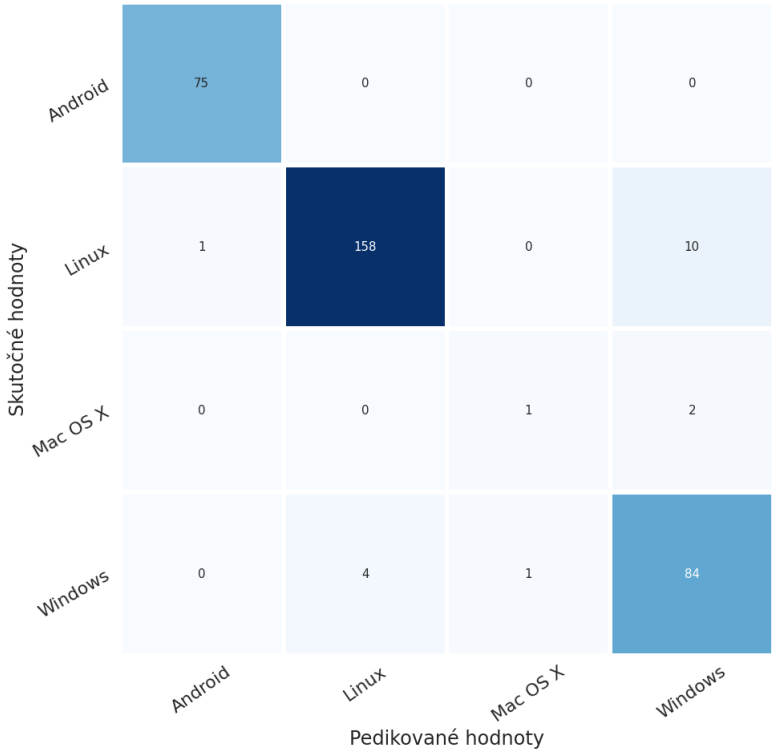
Tabuľka 4.12: Klasifikačný report Multi Flow - 8 tried

	Precíznosť [%]	**Senzitivita** [%]	**f1-skóre** [%]
Android	97	99	98
CentOS	0	0	0
Debian	0	0	0
Linux	49	40	44
Mac OS X	100	67	80
Ubuntu	0	0	0
Windows	88	87	89
openSUSE Linux	56	81	66
presnosť	73	73	73
makro priemer	49	47	47
vážený priemer	68	73	70

Vznikli tak 4 triedy (Android, Linux, Mac OS X, Windows) a úspešnosť takéhoto klasifikátora je 95 %. Andorid s úspešnosťou 99 %. Windows 88 %, Linux 98 % a Mac OS X 50 %. pretože z 3 záznamov sa 2 krát predikoval ako Windows a raz ako Mac OS X. Ako boli triedy klasifikátorom predikované je možné vidieť v matice zámeny na obrázku 4.7. Klasifikačný report je možné vidieť aj v tabuľke 4.13 a maticu zámeny na obrázku 4.7.

Tabuľka 4.13: Klasifikačný report Multi Flow - 4 triedy

	Precíznosť [%]	**Senzitivita** [%]	**f1-skóre** [%]
Android	99	100	99
Linux	98	93	95
Mac OS X	50	33	40
Windows	88	94	91
presnosť	95	95	95
makro priemer	84	80	81
vážený priemer	95	95	95



Obr. 4.7: Matica zámeny klasifikátora Multi-Flow pre 4 triedy

Porovnanie klasifikátorov

Aby bolo možné porovnať klasifikátor, ktorý klasifikoval každý tok zvlášť ďalej už len (Single-flow klasifikátor) s klasifikátorom, ktorý klasifikuje skupinu tokov ďalej už len (Multi-Flow klasifikátor) bolo potrebné zmeniť výstup, ktorý bol získaný klasifikátorom Single-Flow na výstup, ktorý poskytne informáciu aký operačný systém beží na danej IP adrese. Na testovanie bola použitá testovacia sada, ktorá mala 982 záznamov.

Porovnanie s 9 triedami OS (Windows, Linux, Mac OS X, Android, Ubuntu, openSUSE Linux, CentOS, Debian, Orbis) klasifikátor Multi-Flow po ladení hyperparametrov dosiahol úspešnosť 78 %. Výstup klasifikátora Multi-Flow a klasifikátora Single-Flow sa zhodoval na 74 %. Klasifikátor Single-Flow dosiahol presnosť 96 % pri klasifikácií tokov a pri zmenení výstupu na IP adresy sa zhodoval so skutočnou hodnotou len na 77 %. Čo je skoro rovnaká hodnota ako pri Multi-Flow klasifikátore.

Po zlúčení tried operančných systémov Linux (Linux, Ubuntu, CentOS, Debian, openSUSE Linux) vznikli 4 triedy. Presnosť klasifikátora Multi-Flow bola 93%. Presnosť klasifikátora Single-Flow bola 97 % po zmenení výstupu sa so skutočnou hodnotou zhodovala na 91 %. Pri porovnaní sa klasifikátory zhodovali na 91 %. Klasifikácia viacerých tokov je teda možná, ale dobrú úspešnosť vykazuje len pri použití menšieho počtu tried. Pri väčšom počte sa často medzi sebou triedy zamieňali.

Kapitola 5

Záver

V tejto práci bol navrhnutý klasifikátor strojového učenia využívajúci algoritmus rozhodovacích stromov. Metóda, ktorá bola navrhnutá v tejto práci dokáže pasívne identifikovať operačný systém z dát získaných pasívnou technikou monitorovania tokov, a tak zlepšiť situačné povedomie v sieti o vedomosť, aké operačné systémy sa v sieti nachádzajú. Z takejto vedomosti je možné zíkať informáciu nielen o operačnom systéme ale aj o tom aké je to zariadenie. Návrh metódy a klasifikátora je možné vidieť v kapitole 3.

Klasifikátor bol vyhodnotení na dátovej sade, ktorá je popísaná v kapitole 4 v sekcii 4.1. V tejto sekcii sa nachádza popis, analýza a spracovanie dát pre vytvorenie dátovej sady. Dátova sada obsahovala 3 407 970 záznamov a 26 rôznych operačných systémov. Navrhnutý klasifikátor identifikuje operačné systémy len na úrovni názvov, a preto dátova sada obsahovala 9 tried, ktoré klasifikátor rozpoznával. Klasifikátor využíval nasledujúce parametre z hlavičiek TCP/IP paketov TCP OPT, SRC PORT, TCP WIN, TTL, IP FLG, TCP SYN SIZE a skutočná pravda bola získaná z reťazca užívateľského agenta. Klasifikátor dosiahol presnosť (accuracy) 96 %. Klasifikátor najlepšie vedel predikovať triedu Windows, kde dosiahol precíznosť (precision) 98 %, senzitivitu (recall) 99 % a f-skóre 99 %. Najhoršie sa predikovala trieda Debian, ktorá sa často zamieňala za triedu Linux a dosiahla 56 % precíznosť, 30 % senzitivitu a f-skóre 39 %. Vylepšením bolo klasifikovať do menšieho počtu tried spojením tried Linux, Ubuntu, CentOS, Debian, openSUSE Linux do jednej triedy Linux.

V experimentálnej časti viz. sekcia 4.4 boli vykonané niekoľko experimentov a medzi nimi aj experiment, ktorý porovnával navrhnutý klasifikátor strojového učenia s klasifikátorom využívajúceho tabuľku známych príznakov parametrov TCP/IP paketov.

Klasifikátor využívajúci takúto tabuľku dosiahol presnosť 89 % a v porovnaní s klasifikátorom rozhodovacích stromov boli výsledky zhodné na 91 %.

V rámci vylepšenia by som navrhoval doplniť metódu ešte o ďalšie metódy napríklad o metódu, ktorá zisťuje operačný systém z doménových mien (napríklad z poľa SNI z TLS protokolu) alebo o metódu využívajúcu TLS protokol, kde je možné identifikovať opečaný systém zo šifrovacích balíčkov (cipher suite). Výhoda týchto metód je, že môžu byť použité v šifrovanom prenose a aj pri IPv6. Ďalšou zaujímavou činnosťou do budúcna je vplyv IPv6 protokolu na navrhnutú metódu, keďže IPv6 protokol obsahuje niektoré parametre v hlavičke paketu iné ako IPv4, napríklad TTL je v IPv6 označený ako Hop limit. Preto by príznaky pre IPv6 museli byť zmenené.

Metóda, ktorá vznikla v tejto práci by v budúcnosti mohla byť implementovaná ako modul do systému NEMEA v spoločnosti CESNET.

Literatúra

[1] Aksoy, A., Louis, S. a Gunes, M. H. Operating system fingerprinting via automated network traffic analysis. In: *2017 IEEE Congress on Evolutionary Computation (CEC)*. 2017, s. 2502–2509. DOI: 10.1109/CEC.2017.7969609.

[2] Al Shehari, T. a Shahzad, F. Improving Operating System Fingerprinting using Machine Learning Techniques. *International Journal of Computer Theory and Engineering*. Február 2014, zv. 6.

[3] Anderson, B. a McGrew, D. OS fingerprinting: New techniques and a study of information gain and obfuscation. In: *2017 IEEE Conference on Communications and Network Security (CNS)*. 2017, s. 1–9. DOI: 10.1109/CNS.2017.8228647.

[4] Beverly, R. A Robust Classifier for Passive TCP/IP Fingerprinting. In:. Apríl 2004, sv. 3015, s. 158–167. ISBN 978-3-540-21492-2.

[5] Browlee, J. *Hyperparameter Optimization With Random Search and Grid Search* [online]. Python Machine Learning, 14. septembra 2020 [cit. 2021-01-24]. Dostupné z: https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/.

[6] CCNA6RS. *TCP Three-Way Handshake* [online]. CCNA6 RS, august 2017 [cit. 2021-04-24]. Dostupné z: https://www.ccna6rs.com/secfnd/tcp-three-way-handshake/.

[7] Claise, B. *Cisco Systems Netflow Services Export Version 9* [online]. IETF RFC 3954, október 2004 [cit. 2021-01-24]. Dostupné z: https://tools.ietf.org/html/rfc3954.

[8] Eckstein, C. OS fingerprinting with IPv6. In:. 2011.

[9] Endsley, M. Design and Evaluation for Situation Awareness Enhancement. In:. Október 1988, sv. 32. DOI: 10.1177/154193128803200221.

[10] Endsley, M. Endsley, M.R.: Toward a Theory of Situation Awareness in Dynamic Systems. Human Factors Journal 37(1), 32-64. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. Marec 1995, zv. 37, s. 32–64. DOI: 10.1518/001872095779049543.

[11] Endsley, M. Theoretical underpinnings of situation awareness: A critical review. *Situation awareness analysis and measurement*. Január 2000, s. 3–32.

[12] Falch, P. B. *Investigating Passive Operating System Detection*. 2011. Diplomová práca. University of Oslo.

[13] Gagnon, F. a Esfandiari, B. A hybrid approach to operating system discovery based on diagnosis. *Int. Journal of Network Management*. Január 2011, zv. 21, s. 106–119. DOI: 10.1109/NOMS.2012.6212000.

[14] Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R. et al. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. *Communications Surveys & Tutorials, IEEE*. Apríl 2014, zv. 16, s. 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[15] Hope, C. *TCP/IP* [online]. Computer Hope, február 2020 [cit. 2021-04-24]. Dostupné z: https://www.computerhope.com/jargon/t/tcpip.htm.

[16] Husák, M., Cermák, M., Jirsík, T. a Celeda, P. Network-Based HTTPS Client Identification Using SSL/TLS Fingerprinting. In: *2015 10th International Conference on Availability, Reliability and Security*. 2015, s. 389–396. DOI: 10.1109/ARES.2015.35.

[17] Matoušek, P. *Monitorováni toku NetFlow* [online]. 2019 [cit. 2019-10-02]. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FISA-IT%2Flectures%2Fisa-netflow.pdf&cid=13349.

[18] Jirsik, T. a Celeda, P. Toward real-time network-wide cyber situational awareness. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, s. 1–7. DOI: 10.1109/NOMS.2018.8406166.

[19] JIRSÍK, T. *Cyber Situation Awareness via IP Flow Monitoring [online]*. 2019 [cit. 2021-01-17]. Disertační práce. Masarykova univerzita, Fakulta informatiky, Brno. Vedúci práce Čeleda, P. Dostupné z: DostupnézWWW<https://is.muni.cz/th/ejynv/>.

[20] J.M.Allen. OS and Application Fingerprinting Techniques. In:. September 2017.

[21] Kohno, T., Broido, A. a Claffy, K. C. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*. 2005, zv. 2, č. 2, s. 93–108. DOI: 10.1109/TDSC.2005.26.

[22] Lastovicka, M., Jirsik, T., Celeda, P., Spacek, S. a Filakovsky, D. Passive os fingerprinting methods in the jungle of wireless networks. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, s. 1–9. DOI: 10.1109/NOMS.2018.8406262.

[23] Laštovička, M., Dufka, A. a Komárková, J. Machine Learning Fingerprinting Methods in Cyber Security Domain: Which one to Use? In: *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. 2018, s. 542–547. DOI: 10.1109/IWCMC.2018.8450406.

[24] Lippmann, R., Fried, D., Piwowarski, K. a Streilein, W. Passive Operating System Identification From TCP / IP Packet Headers *. In:. 2003.

[25] Lyon, G. *Nmap reference guide* [online]. September 1981 [cit. 2021-01-24]. Dostupné z: https://nmap.org/.

[26] Matoušek, P. *Síťové služby a jejich architektura.* Publishing house of Brno University of Technology VUTIUM, 2014. 396 s. ISBN 978-80-214-3766-1. Dostupné z: https://www.fit.vut.cz/research/publication/10567.

[27] Matoušek, P., Ryšavý, O., Grégr, M. a Vymlátil, M. Towards identification of operating systems from the internet traffic: IPFIX monitoring with fingerprinting and clustering. In: *2014 5th International Conference on Data Communication Networking (DCNET).* 2014, s. 1–7.

[28] Matsunaka, T., Yamada, A. a Kubota, A. Passive OS Fingerprinting by DNS Traffic Analysis. In:. Marec 2013, s. 243–250. DOI: 10.1109/AINA.2013.119. ISBN 978-1-4673-5550-6.

[29] Mavrakis, C. Passive asset discovery and operating system fingerprinting in industrial control system networks. In:. 2015.

[30] Onwubiko, C. Functional requirements of situational awareness in computer network security. In:. Júl 2009, s. 209 – 213. DOI: 10.1109/ISI.2009.5137305.

[31] Postel, J. *Internet Protocol - RFC 791* [online]. Information Sciences Institute University of Southern California, september 1981 [cit. 2021-01-24]. Dostupné z: https://tools.ietf.org/html/rfc791.

[32] Postel, J. *Transmission Control Protocol - RFC 793* [online]. Information Sciences Institute University of Southern California, september 1981 [cit. 2021-01-24]. Dostupné z: https://tools.ietf.org/html/rfc793.

[33] Refaeilzadeh, P., Tang, L. a Liu, H. Cross-Validation. In: LIU, L. a ÖZSU, M. T., ed. *Encyclopedia of Database Systems.* Boston, MA: Springer US, 2009, s. 532–538. DOI: 10.1007/9780387399409_565. ISBN 978-0-387-39940-9. Dostupné z: https://doi.org/10.1007/978-0-387-39940-9_565.

[34] R.Fielding, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [online]. Internet Engineering Task Force (IETF), jún 2014 [cit. 2021-01-24]. Dostupné z: https://www.ietf.org/rfc/rfc7231.txt.

[35] Richardson, D., Gribble, S. a Kohno, T. The Limits of Automatic OS Fingerprint Generation. In:. Január 2010, s. 24–34. DOI: 10.1145/1866423.1866430.

[36] Sokolova, M. a Lapalme, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management.* 2009, zv. 45, č. 4, s. 427–437. ISSN 0306-4573. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0306457309000259.

[37] Tyagi, R., Paul, T., Manoj, B. S. a Thanudas, B. Packet Inspection for Unauthorized OS Detection in Enterprises. *IEEE Security Privacy.* 2015, zv. 13, č. 4, s. 60–65. DOI: 10.1109/MSP.2015.86.

Príloha A

Obsah priloženého pamäťového média

- **src**: zdrojové súbory pre spracovanie dát a klasifikáciu
- **dataset**: dátová sada
- **trained_models**: natrénované modely
- **xbolf00Profilovaniesietovychentit.pdf**: text bakalárkej práce
- **README.md**: popis spustenia klasifikátora