VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

FÚZE DAT PRO KLASIFIKACI SÍŤOVÝCH ZAŘÍZENÍ

INFORMATION FUSION FOR CLASSIFICATION OF NETWORK DEVICES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ SEDLÁČEK**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Sedláček Ondřej**
Program: Informační technologie
Název: **Fúze dat pro klasifikaci síťových zařízení**
Information Fusion for Classification of Network Devices
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s projektem ADiCT organizace CESNET zabývajícím se objevováním a klasifikací síťových zařízení. Podrobně nastudujte v současnosti implementované zdroje dat.
2. Nastudujte metody z oblasti fúze informací, jako např. Dempster-Shafer teorie důkazů či model přenositelného domnění (Transferable belief model).
3. Navrhněte metodu umožňující zpracovat informace o síťových zařízeních pocházející z různých zdrojů, přičemž tato data mohou být různého typu, míry detailu, spolehlivosti a mohou se navzájem podporovat či naopak vylučovat. Cílem je sloučit taková data do štítku či kategorie popisující dané zařízení. Zaměřte se především na odvození typu zařízení a jeho operačního systému.
4. Navrženou metodu implementujte a integrujte do systému ADiCT.
5. Metodu otestujte na datech z reálné sítě.
6. Shrňte dosažené výsledky a navrhněte další pokračování práce.

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 2021
Datum odevzdání: 11. května 2022
Datum schválení: 29. října 2021

Abstrakt

Tato práce se zabývá problémem fúze informací z několika zdrojů dat v kontextu monitorování počítačových sítí. Práce představuje řešení postavené na konceptu klasifikačních pravidel, konfigurovaných experty. Konfigurace je usnadněna vyhrazeným konfiguračním jazykem, který je interpretován v rámci řešení práce. Klasifikační pravidla umožňují pokrytí rozmanitých typů dat, přičemž výsledek poskytují přiřazením štítku z navržené taxonomie. Takto je zachována rozdílná úroveň podrobnosti mezi jednotlivými zdroji dat i ve sloučeném výsledku. Řešení zároveň využívá Dempster-Schaferovy teorie důkazů, pomocí které je provedeno slučování štítků z jednotlivých zdrojů dat pro získání štítků výsledných. Provedený výzkum ukázal, že fúze informací v tomto kontextu zvyšuje přesnost klasifikace zařízení. Na základě testování a experimentů s datovou sadou z reálné sítě byl stanoven postup optimalizace klasifikačních pravidel, kterým se navíc podařilo zvýšit přesnost řešení o 19 % oproti původnímu řešení.

Abstract

This work is focused on solving information fusion when dealing with multiple data sources in computer network monitoring. A solution built on the concept of classification rules configured by experts is presented. Configuration is simplified using a designated configuration language interpreted by the solution. The classification rules enable coverage of diverse types of data. The result is given as a label from specified taxonomy. Using a taxonomy maintains the different levels of detail between the data sources, even in the output label. The solution also uses the Dempster-Schafer theory for merging labels from different sources into a single output label. Results of experiments show that information fusion in this context does increase the accuracy of device classification. A process of rule optimization was developed based on testing and experiments with a dataset from a real network. The accuracy was increased by 19 % compared to the original solution using this process.

Klíčová slova

Dempster-Schaferova teorie důkazů, Model přenositelného domnění, síťová zařízení, monitorování sítě, klasifikace operačního systému, klasifikace typu zařízení, ADiCT

Keywords

Dempster-Schafer theory, Transferrable Belief Model, network devices, network assets, network monitoring, OS classification, device type classification, ADiCT

Citace

SEDLÁČEK, Ondřej. *Fúze dat pro klasifikaci síťových zařízení.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Martin Žádník, Ing, Ph.D.

Fúze dat pro klasifikaci síťových zařízení

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Žádníka, PhD. Další informace mi poskytl odborný konzultant Ing. Václav Bartoš, PhD. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

. . . . . .
Ondřej Sedláček
8. května 2022

Poděkování

Chtěl bych poděkovat Ing. Martinu Žádníkovi, PhD. za jeho trpělivost, čas a podněty, poskytnuté v rámci konzultací. Rovněž bych chtěl poděkovat Ing. Václavu Bartošovi, PhD. ze sdružení Cesnet, který častými konzultacemi a radami rovněž velmi pozitivně přispěl k vývoji.

Obsah

Kapitola 1

Úvod

Počítačové sítě jsou nezbytnou, ačkoliv pro mnohé uživatele neviditelnou, součástí moderní civilizace. Existují počítačové sítě, které jsou kritické pro infrastrukturu společností, univerzit či nemocnic, což z nich dělá cíl útočníků. Pro získání schopnosti reagovat na útoky je nutné sítě sledovat, což umožní probíhající útoky detekovat. Identifikace typů zařízení a jejich operačních systémů je nápomocná při identifikaci relevantních hrozeb ve vlastní síti. Součástí správy počítačových sítí, pokud má být účinná, je i monitorování spravované sítě a analýza provozu, včetně získávání informací o připojených zařízeních. Analýzou chování konkrétního zařízení lze mimo jiné získat lepší představu o jeho typu (zda se jedná o pracovní stanici, server či IoT zařízení) a běžícím operačním systému. Takový profil zařízení na základě jeho předchozí aktivity může být využit i při řešení bezpečnostních incidentů, například pokud u zařízení došlo ke změně chování.

Tato práce byla zadána a provedena v rámci výzkumného projektu ADiCT organizace CESNET, který se zmíněným monitorováním sítí zabývá. Aktivita v projektu ADiCT byla směřována zejména na podporu výzkumu metod pro analýzu síťového provozu, které pak slouží jako vstupy do jeho systému. Problém, který však nastává, je, že se zvyšujícím se počtem zdrojů dat systému se zároveň zvyšuje komplexita interpretace nasbíraných dat, zejména pro neznalého uživatele, protože každý nový zdroj dat se sebou přináší i nový formát a typ informace. Krom toho může stejný typ informace přicházet z různých zdrojů, které se nemusí vzájemně shodovat.

Cílem této práce je navrhnout a implementovat metodu pro sloučení těchto různorodých dat, týkajících se klasifikace síťových zařízení, která mohou být různě podrobná, být zcela odlišného typu, a přitom se vzájemně podporovat či vylučovat. Výstupem této metody jsou štítky, které pokrývají dva aspekty daného zařízení: typ zařízení a jeho operační systém.

Pro dosažení cíle jsou provedeny následující kroky, které odpovídají i jednotlivým kapitolám: Kapitola 2 se věnuje studiu projektu ADiCT a architektuře jeho systému včetně vstupních modulů, zároveň shrnuje současný stav na poli fúze informací a metod klasifikace síťových zařízení. Kapitola 3 obsahuje přesnější formulaci problému zpracování jednotlivých zdrojů dat, přičemž jsou stanoveny požadavky na konečné řešení. Následně je představen návrh řešení ve formě jazyka pro klasifikační pravidla, kterým bude možné konfigurovat způsob zahrnutí jednotlivých datových zdrojů do výsledných štítků, a navíc odvozovat nové znalosti ze surových dat systému. Toto řešení je v kapitole 4 následně implementováno. V kapitole 5 je provedeno testování řešení na datové sadě získané od organizace CESNET, které následuje provedení série experimentů s metodou fúze informací, na základě kterých byla metoda optimalizována.

Kapitola 2

Současný stav

Tato kapitola shrnuje současný stav, který bylo třeba nastudovat pro vypracování práce. První sekce, 2.1, je věnována projektu ADiCT. Je popsána architektura, tok dat v rámci platformy a použitá rozhraní. Zbylé sekce shrnují relevantní literaturu v okruhu témat spřažených s touto prací. Sekce 2.2 pojednává o metodách fúze informací, jmenovitě Dempster-Schaferově teorii a Modelu přenositelného domnění. Zároveň prochází příklady kombinace několika klasifikátorů za účelem zvýšení přesnosti a robustnosti řešení. Zbylé dvě sekce se zabývají tématikou klasifikace zařízení v síti, nejprve v aspektu operačního systému zařízení v sekci 2.3, následně typu hardware zařízení v sekci 2.4.

2.1 Projekt ADiCT – Asset Discovery, Classification and Tagging

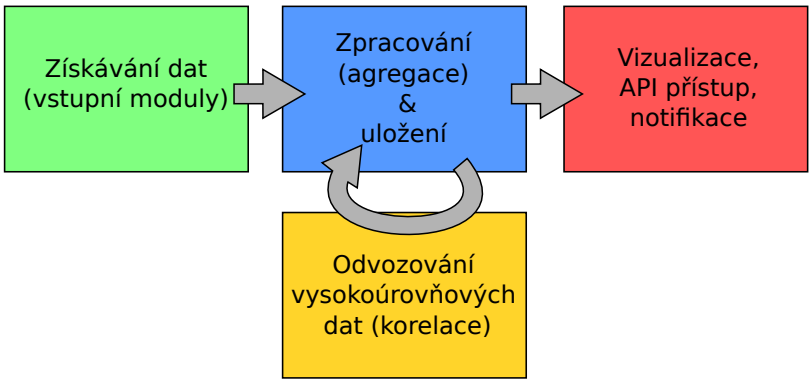
Informace v této sekci jsou čerpány z interních wiki stránek[1] dostupných v rámci systému Redmine projektu.

Projekt ADiCT (Asset Discovery, Classification and Tagging), byl založen na konci roku 2019 v rámci organizace CESNET a je stále ve vývoji i v době psaní této práce. Projekt byl navržen jakožto pokračování s jiným zaměřením od původního projektu NERD[2] (Network Entity Reputation Database). Zatímco NERD shromažďuje bezpečnostní informace o entitách z celého světa, které jsou pak veřejně dostupné[3], projekt ADiCT vznikl s cílem shromažďovat veškeré informace o entitách z vybrané sítě, přičemž přístup k nim by měl mít pouze její správce. Zároveň během vývoje vznikla obecná platforma, umožňující zpracovávání dat vázajících se k entitám, zvaná DP^3 (Dynamic Profile Processing Platform). Systém ADiCT je nad platformou DP^3 postaven, její architektuře se věnuje podsekce 2.1.1. Systém ADiCT, stejně jako platforma DP^3 a moduly běžící v rámci systému, je implementován v jazyce Python verze 3.

Hlavním cílem projektu ADiCT je automaticky tvořit bázi znalostí o sledované síti. Přesněji, jde o sběr nejrůznějších informací, které jsou vázány k IP nebo MAC adresám, či jiným entitám, včetně ukládání historie s nastavitelnou délkou a agregací. Tyto informace by následně mělo být možné vzájemně korelovat a odvozovat z nich další závěry. Praktickým využitím projektu je tedy poskytnutí kontextu pro správce sítě či jiné pracovníky vyšetřující nehody v síti. Dalším praktickým využitím je využití uložené historie pro detekci nečekaných

[1] https://redmine.liberouter.org/projects/adict/wiki
[2] https://github.com/CESNET/NERD
[3] https://nerd.cesnet.cz/nerd/ips/

či nežádoucích změn chování jednotlivých entit, na kterou lze následně navázat notifikace určené správcům sítě. Prioritou projektu nicméně v tuto chvíli zůstává podpora výzkumu ve zkoumané oblasti, nasazení systému jako produkční služby není v nejbližší době plánováno.



Obrázek 2.1: **Vysokoúrovňový tok dat v systému ADiCT.** Obrázek převzat z interních wiki stránek projektu.

Na obrázku 2.1 je znázorněn tok dat v projektu ADiCT. Data jsou do systému posílána primárními moduly prostřednictvím webového rozhraní API (Application Programming Interface). Primární moduly jsou takové, které na základě externího vstupu přináší do systému nová data. V současnosti se jedná převážně o zpracovávání dat z obohacených síťových toků získaných pasivním monitorováním. Jádrem systému je ukládání těchto zpracovaných dat, případně jejich agregace. Zároveň nad těmito daty běží další procedury prostřednictvím tzv. sekundárních modulů, které z nich mohou odvozovat další vysokoúrovňové informace. Sekundární moduly jsou definovány jako reagující na změnu dat či události v rámci systému, a až na základě toho doplňující další data. V neposlední řadě lze data ze systému opět získat, ať prostřednictvím vizualizace skrz webové rozhraní, či pomocí API.

2.1.1 Architektura platformy DP^3

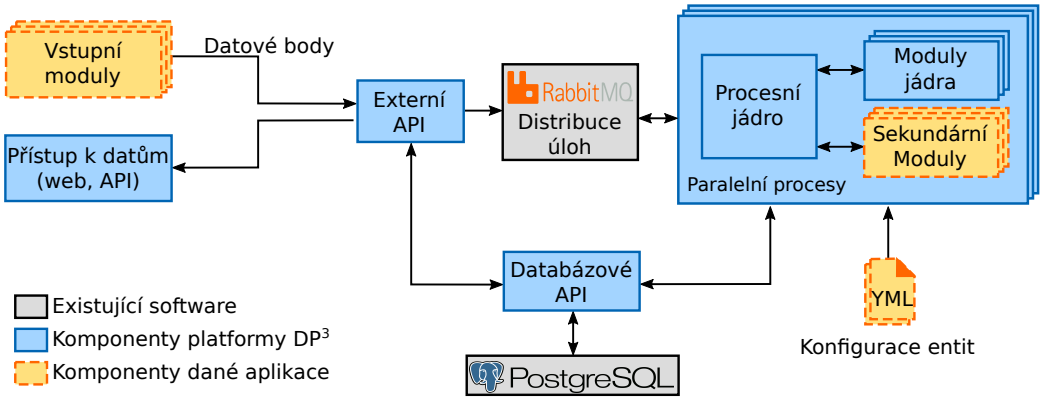
Tato podsekce popisuje architekturu platformy DP^3. Jedná se o obecnou platformu umožňující správu dat vázaných k entitám. Zpracovávané entity tak mohou reprezentovat cokoliv odpovídající následujícím podmínkám, ne pouze data o síťových zařízeních. Entita je identifikována svým primárním klíčem entity `eid` a váže se k ní množina hodnot jejích atributů. Entity i atributy jsou plně konfigurovatelné, takže každý typ entity může mít jinou sadu atributů a každý atribut může mít jiné nastavení. Krom názvu atributu a datového typu lze například nastavit, zda má být u daného atributu uchovávána historie (a jak dlouho), případně zda je povoleno mít u daného atributu více aktivních hodnot v jednom okamžiku. Konfigurace entit je prováděna pomocí souborů ve formátu YAML[4], přičemž na základě této konfigurace je platformou vytvořeno i databázové schéma. Výpis 2.1 ukazuje příklad takové konfigurace:

```
entity:
  id: ip
  name: IP address
  key_data_type: string
```

[4] https://yaml.org/

```
    auto_create_record: false
  attribs:
    os_by_tcpip:
      name: OS by TCP/IP data
      description: Operating system determined from values of ...
      data_type: category
      multi_value: true
      history: true
      history_params:
        pre_validity: 8h
        post_validity: 8h
        max_age: 14d
  ...
```

Výpis 2.1: **Ukázka konfigurace entity reprezentující data sbíraná k IP adresám.** Definice atributu `os_by_tcpip` obsahuje nastavení datového typu výčet, uchování historie a umožnění platnosti vícero aktivních hodnot v jednom okamžiku.



Obrázek 2.2: **Architektura a datový tok platformy DP^3.**

Zpracování dat v rámci platformy je podrobněji ukázáno na schématu 2.2 a je následující: Data nasbíraná primárními moduly do systému přichází skrze vstupní rozhraní API, kam putují ve formě tzv. **datových bodů** (data-point). Vstupní API hodnoty korektních datových bodů předává dále na zpracování procesnímu jádru systému. Je vhodné zmínit, že procesní jádro je navrženo tak, že ho lze spustit paralelně v několika procesech, což umožňuje vysokou škálovatelnost řešení. Distribuce úloh využívá službu na distribuci zpráv RabbitMQ[5], nad kterou je postaveno rozhraní pro zpracování a tvorbu nových úloh. Každý nový datový bod se tedy takto stává úlohou pro uložení záznamu procesním jádrem. Úlohy mohou být požadavky na změnu atributu konkrétní entity, jako v případě přijatých datových bodů, ale také například vyvolání události. Procesní jádro přijímá úlohy a při jejich zpracování může navíc volat funkce sekundárních modulů, nastavené jako reakce na změnu atributu či jinou událost.

Jako příklad lze uvést situaci při vytvoření nového záznamu entity IP, na kterou je v systému ADiCT vázán sekundární modul, který vykoná reverzní dotaz systému DNS a k entitě uloží případný získaný `hostname` do připraveného atributu. Procesní jádro má jako jediné

[5] https://www.rabbitmq.com/

přístup ke čtení i zápisu do databáze, ostatní moduly se dostávají k datům prostřednictvím objektů, které jsou připraveny jádrem. Přístup pro čtení z databáze umožňuje navíc výstupní webové rozhraní API, na které je navázána i webová prezentace.

Před popisem uložení hodnot v databázi je potřeba krátce definovat již zmíněné datové body. **Datový bod** definuje hodnotu konkrétního atributu konkrétní entity v určitém časovém rozmezí, přičemž může navíc obsahovat údaj o věrohodnosti, označovaný `c` (z anglického *confidence*). Volitelně může být navíc uveden řetězec identifikující zdroj datového bodu. Časové rozmezí, kdy je hodnota atributu považována za platnou, je určeno v datovém bodu časovými okamžiky `t1` a `t2`. Je povolena absence hodnoty `t2`, pak je předpokládáno `t2`=`t1`. Hodnota atributu může být platformou považována za platnou před započetím i po vypršení základní platnosti na základě nastavení `pre_validity` a `post_validity`, které lze vidět ve výpisu 2.1.

Pro integraci konečného řešení této práce do systému ADiCT bude potřeba využít rozhraní sekundárních modulů. Sekundární moduly systému ADiCT jsou definovány pomocí tříd, dědících z třídy `BaseModule`. Tato bázová třída slouží jako základ a implementuje metody pro spouštění a ukončení běhu modulu. V rámci konstruktoru třídy sekundárního modulu jsou registrovány funkce reagující na změny. Při registraci je vybrán typ entity, jaké změny jsou touto funkcí sledovány a seznam jmen atributů, které mohou být funkcí změněny. Signatura funkce reagující na změny je pevně definována, přičemž jí jsou jako parametry předány typ a klíč entity a objekt `Record` sloužící jako rozhraní modulu do databáze. Pomocí třídy `Record` je možné získat aktuální hodnoty atributů. Pro aktualizaci hodnot v databázi je nutno vznést požadavek procesnímu jádru prostřednictvím návratové hodnoty, která má podobu seznamu N-tic, které specifikují, jaké operace se mají vykonat s jakými atributy. V neposlední řadě, pro správnou funkcionalitu modulu musí být modul uveden v konfiguraci platformy.

2.1.2 Dostupná vstupní data

Tato podsekce popisuje vstupní moduly projektu ADiCT, na základě kterých lze odvodit operační systém či typ zařízení. Podrobněji jsou popsány atributy těmito moduly poskytované opět jen v případě, že budou v této práci využity k dalšímu zpracování. Ačkoliv systém ADiCT není omezen pouze na tento typ zdroje dat, všechny uvedené moduly analyzují data pocházející z monitorování sítě pomocí měření IP toků, v některých případech toků rozšířených o data z aplikační vrstvy. Pro začlenění do monitorovací infrastruktury implementují moduly rozhraní systému NEMEA [8].

Modul HTTP User-Agent

Tento modul, jehož autorem je Josef Koumar, má za cíl získat veškeré dostupné informace z HTTP pole User-Agent. Pro tento účel je použito dvou metod. První je vyhledávání v databázi známých hodnot pole User-Agent [40]. Druhou metodou je odhadnutí operačního systému zařízení pomocí detekce předdefinovaných řetězců, jenž specifikují operační systém či aplikaci pro konkrétní operační systém typickou. Tato klíčová slova se mohou a nemusí v řetězcích User-Agent objevit, detekce je prováděna pomocí regulárních výrazů.

Následuje výčet atributů, které budou využity v dalším zpracování.

- `operating_system_ua` obsahuje operační systém zařízení, získaný libovolnou ze zmíněných metod. Příklady hodnot jsou: Windows 10, Ubuntu, Fedora.

- `hardware_type_ua` obsahuje typ hardware zařízení, získáno z databáze. Ukázky hodnot jsou: computer, phone, server.
- `http_useragent` obsahuje původní řetězec User-Agent, tak jak se vyskytl v původním paketu.

Modul OS by TCP/IP

Tento modul rozpoznává OS na základě položek z TCP a IP protokolů (IP TTL, IP flags, TCP win size, TCP SYN size, TCP options, src port) pomocí natrénovaného klasifikátoru DecisionTree modelu. Model je založený na práci Reného Bolfa [6].

Výstupním atributem modulu je atribut `os_by_tcpip`. Modul rozeznává a poskytuje jako výstup následující systémy: Windows, Ubuntu, Android, Linux, Debian, Mac OS X, CentOS, openSUSE Linux a Orbis OS.

Modul Service Labels

Modul vyvinutý v rámci bakalářské práce Josefa Koumara [20], sloužící k odhalení služeb běžících na zařízení. V odchytávaném provozu kontroluje přítomnost navazování TCP spojení, přičemž u úspěšně navázaných si zaznamená cílový port jako otevřený. Uvažovány jsou pouze dobře známé a registrované porty, které jsou spravovány registrační autoritou IANA.

Každý zaznamenaný port je vyhledán v tabulce mapující číslo portu na štítek typu služby, respektive protokolu, který je běžně provozován na daném portu. Tyto štítky jsou odesílány v rámci atributu `tags_by_services`. Příklady hodnot štítků jsou: Mail Server, DHCP Client, Printer, či Windows.

Modul TLS Fingerprinting

Tento modul je založený na práci Lukáše Hejcmana [17]. Ten ve své práci představuje techniku JA3cury, tedy spojení dvou existujících přístupů tvoření otisků TLS spojení, JA3 [19] a Cisco Mercury [28]. Tímto nejen zachovává kompatibilitu s existujícími identifikačními metodami, ale přidanou hodnotou je i podrobnější databáze otisků, umožňující rigoróznější analýzu výsledků.

Modul samotný pak využívá databázi otisků, vůči které porovnává otisky vytvořené z odchyceného provozu. Pomocí stromovité struktury je ukládáno, jakým operačním systémům odpovídají vytvořené otisky. Strom má 3 úrovně: rodina systému, název systému a konkrétní číslo sestavení. Klasifikace na základě tohoto modulu pak spočívá ve výběru cesty stromem od kořenu po listové uzly tak, že na každém kroku je vybrán dceřiný uzel s největším zaznamenaným počtem výskytů. Jelikož na každé úrovni stromu lze vypočítat relativní „přesnost“ nejčastějšího záznamu, pro jednu kompletní klasifikaci jsou výstupem 3 hodnoty spolehlivosti.

Výstup pro systém ADiCT má pak podobu 3 atributů, které nesou tyto hodnoty přesnosti v rámci pole confidence. Hodnoty přesnosti jsou násobeny přesností z nadřazeného uzlu, čímž je získána přesnost vůči všem záznamům, nikoliv jen vůči těm v daném podstromu. Ukázka atributů a vývoje přesnosti v tabulce 2.1:

atribut	popis	hodnota	přesnost
`tls_os_family`	rodina	[Mac OS X]	0,54
`tls_os_name`	jméno	[Mac OS X, Catalina]	0,30
`tls_os_version`	verze	[Mac OS X, Catalina, 10.15.5]	0,25

Tabulka 2.1: Ukázka výstupních atributů modulu TLS Fingerprinting.

Příklady dalších hodnot (zahrnuty jsou pro stručnost pouze kompletní varianty) jsou: [WinNT, Windows 10 Enterprise, 10.0.17134], [Mac OS X, El Capitan, 10.11.6], [Linux, Ubuntu 19.04, 5.0.0-32-generic].

Modul SDP Analyzer

Tento modul vznikl v rámci mé projektové praxe [34]. Modul zpracovává data z lokální sítě ve formátu síťových toků, obohacených o informace z protokolů typu Service Discovery. Pole z vybraných protokolů (SSDP, DNS-SD a NET-BIOS) jsou agregována na základě zdrojové MAC adresy zařízení. Následně je provedena extrakce operačního systému, identifikátoru hostname a v některých případech i detailů o konkrétním modelu zařízení.

Výstup modulu je rozdělen do jednotlivých atributů, jejichž výčet následuje.

- `sdp_label` obsahuje extrahované informace o operačním systému, hostname, či typu zařízení. Příklady hodnot:
  `{'name': 'hostname', 'value': 'Milan-MacBook-Pro.local'}`
  `{'name': 'os', 'value': 'LINUX'}`
  `{'name': 'airplay_model', 'value': 'AppleTV5,3'}`
- `ssdp_query` obsahuje dotazy na konkrétní službu pomocí protokolu SSDP. Příklady hodnot:
  `'dial-multiscreen-org:service:dial:1',`
  `'schemas-upnp-org:device:MediaServer:1'`
- `ssdp_service` obsahuje oznámení o běžící službě pomocí protokolu SSDP. Příklady hodnot:
  `{'port': 4321, 'service': 'dial-multiscreen-org:service:dial:1'}`
  `{'port': 7654, 'service': 'Belkin:service:remoteaccess:1'}`
- `dnssd_query` obsahuje dotazy na konkrétní službu pomocí protokolu DNS-SD. Příklady hodnot: `'_nfs._tcp.local', '_webdav._tcp.local',`
  `'_afpovertcp._tcp.local', '_ftp._tcp.local'`
- `dnssd_service` obsahuje oznámení o běžící službě pomocí protokolu DNS-SD. Příklady hodnot:
  `{'port': 1234, 'service': 'Local-Pc._spotify-connect._tcp.local'}`
  `{'port': 22, 'service': 'pc-451-221._udisks-ssh._tcp.local'}`

2.2 Metody fúze informací

Tato sekce se zabývá metodami fúze informací, zejména Dempster-Schaferovou teorií důkazů a Modelem přenositelného domnění. V podsekci 2.2.1 naleznete představení zmíněných teorií, zejména použitých vztahů a terminologie. Podsekce 2.2.2 je věnována diskusi

vlastností Dempster-Schaferovy teorie a tomu, jaké jsou důsledky jejího použití. V poslední podsekci 2.2.3 je uveden přehled současného stavu z pohledu využití zmíněných teorií k fúzi informací se zaměřením na konkrétní způsoby jejich aplikace.

2.2.1 Dempster-Schaferova teorie důkazů a Model přenositelného domnění

Dempster-Schaferova teorie důkazů (D-S teorie) je teorie umožňující pracovat s neurčitostí a vyjadřovat nevědomost. Oproti pravděpodobnostnímu přístupu je zmírněn požadavek, aby součet měr důvěry v událost a její negace musel být roven 1.

Vznikla jako kombinace prací Dempstera [11] a Schafera [35]. V průběhu času vzniklo několik interpretací, nejvýznamnější z nich zvaná Model přenositelného domnění (TBM[6]). TBM byl představen Smetsem a Kennesem v [37].

D-S teorie začíná definicí takzvaného rámce domnění[7] Θ (frame of discernment). Tento rámec je konečnou množinou vzájemně vylučujících se základních hypotéz, popisující doménu. Potenční množina 2^Θ označuje množinu všech možných podmnožin Θ včetně prázdné množiny $\emptyset$.

Funkce důvěry m přiřazuje každé množině, náležící do potenční množiny 2^Θ, míru důvěry ležící mezi 0 a 1. Občas bývá tato funkce také nazývána základním přiřazením pravděpodobnosti (Basic Probability Assignment). Tato funkce musí splňovat dvě omezení: míra důvěry v prázdnou množinu musí být rovna nule a součet měr důvěry všech členů potenční množiny musí být 1.

$$\sum_{A \subseteq \Theta} m(A) = 1, \quad m(\emptyset) = 0 \tag{2.1}$$

Každé $A \in 2^\Theta$, pro které platí $m(A) > 0$ se nazývá fokální hypotézou. Rozdíl oproti pravděpodobnostním modelům je ten, že míry důvěry mohou být přiřazeny každé množině hypotéz z 2^Θ, nikoliv jen jednotlivým hypotézám z Θ.

Míra důvěry $m(A)$ vyjadřuje poměr důvěry pomyslného agenta, která podporuje množinu hypotéz A. Jinými slovy, poměr přesvědčení, že skutečný stav světa se nachází v A. Tato míra důvěry, z důvodu nedostatku informací, nevyjadřuje podporu žádné konkrétní podmnožině hypotéz z A. [37]

Pokud funkce důvěry splňuje požadavek, že $m(\emptyset) = 0$, označuje se za normalizovanou. V rámci teorie TBM není tato podmínka vyžadována, a místo toho míra důvěry přiřazená $\emptyset$ vyjadřuje možnost existence pravdivé hypotézy mimo rámec domnění. Smets v [36] argumentuje, že vyžadování $m(\emptyset) = 0$ odpovídá předpokladu uzavřeného světa (rámec domnění obsahuje všechny možné hypotézy), zatímco povolení $m(\emptyset) > 0$ odpovídá předpokladu otevřeného světa (pravdivá hypotéza může existovat i mimo rámec domnění). V této práci bude vždy uvažováno $m(\emptyset) = 0$.

Na základě přiřazených měr důvěry lze získat horní a dolní hranice pravděpodobnostních intervalů. Tyto intervaly jsou ohraničeny dvěma funkcemi, mírou domnění Bel a mírou přípustnosti Pl. Míra domnění pro hypotézu A je definována jako součet všech měr důvěry všech neprázdných podmnožin A.

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{2.2}$$

[6] Z anglického názvu Transferrable Belief Model
[7] Též označováno jako univerzum diskurzu či prostředí

Míra přípustnosti Pl pro hypotézu A je definována jako součet měr důvěry všech hypotéz, které mají s A neprázdný průnik. Jinými slovy, vyjadřuje míru důvěry všech příbuzných hypotéz.

$$Pl(A) = \sum_{B \cap A} m(B) \tag{2.3}$$

Míra domnění Bel(A) a míra přípustnosti Pl(A) vyjadřují pro hypotézu A dolní a horní hranici pravděpodobnostního intervalu. Interval (Bel(A), Pl(A)) pak reprezentuje pravděpodobnostní interval nejistoty.

Dempstrovo kombinační pravidlo (DRC[8]), značeno pomocí $\oplus$, umožňuje sloučit dvě oddělené funkce důvěry m, které existují nad stejným rámcem domnění Θ. Pro ukázku označíme tyto dvě funkce m_1 a m_2. Funkce důvěry, která bude reprezentovat jejich sloučení, budiž $m_{1,2}$, a míra důvěry přiřazená hypotéze A touto funkcí bude vyjádřena jako

$$m_{1,2}(A) = (m_1 \oplus m_2)(A), \tag{2.4}$$

přičemž

$$(m_1 \oplus m_2)(A) = \eta \sum_{B \cap C = A} m_1(B) m_2(C), \qquad \forall A \ \subseteq \Theta, A \ \neq \emptyset, \tag{2.5}$$

kde η je normalizační konstantou, zajišťující, že výsledná funkce důvěry bude normalizována, a je definována vůči míře konfliktu K jako

$$\begin{aligned} \eta &= \frac{1}{1-K} \\ K \ &= \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \end{aligned} \tag{2.6}$$

Dempstrovo kombinační pravidlo je komutativní, asociativní a jeho neutrální prvek je prázdná funkce důvěry[9]. V důsledku komutativity nezáleží na pořadí kombinace, pokud je potřeba zkombinovat více než dvě funkce důvěry.

Doposud popsaná terminologie je společná pro D-S teorii i pro teorii TBM. Hlavním rozdílem je, že teorie TBM zcela odmítá představu, že vyjádření míry důvěry má něco společného s jakýmkoliv pravděpodobnostním rozložením uvažované situace, které pro použití TBM může, ale nemusí existovat [37]. Toto nelze nutně říct o všech interpretacích D-S teorie.

Teorie TBM tedy definuje dvě úrovně, na kterých popisuje důvěru. První úroveň, zvanou kredální, kde jsou představy o situaci reprezentovány pomocí funkcí důvěry, a druhou úroveň, zvanou pignistická[10], kde představy o situaci mohou být použity pro tvorbu rozhodnutí a jsou reprezentovány pravděpodobnostními funkcemi.

Transformaci z kredální úrovně na pignistickou lze provést pomocí pignistické transformace. Nechť m je funkcí důvěry s rámcem domnění Θ a nechť $BetP_m$ označuje odpovídající pignistickou pravděpodobnostní funkci. Poté lze pignistickou transformaci z m na $BetP_m$ definovat jako

[8] DRC z anglického Dempster's Rule of Combination

[9] Prázdná funkce důvěry má jedinou fokální hypotézu, rámec domnění, a platí $m(\Theta) = 1$.

[10] pignistická z latinského pignus = sázka

$$BetP_m(a) = \sum_{A \ni a} \frac{m(A)}{|A|(1 - m(\emptyset))}, \quad \forall a \ \in \Theta. \tag{2.7}$$

Normalizační faktor $1 - m(\emptyset)$ může být vynechán, pokud je m již normalizovaná. Tato transformace v podstatě způsobí to, že pro všechny fokální množiny A, kde $|A| > 1$, bude míra důvěry přiřazená A rovnoměrně rozdělena mezi její prvky, základní hypotézy a.

2.2.2 Výhody a nevýhody D-S teorie

Z prezentovaných základů D-S teorie je zřejmé, že při tvorbě systému pro fúzi informací postavených nad touto teorií zůstává tvůrci velká míra volnosti. D-S teorie neříká například nic o tom, jakým způsobem mají být tvořeny funkce důvěry, a na základě čeho má být přiřazována míra důvěry. Tuto vlastnost lze brát jako výhodu, jelikož to dovoluje tvorbu vysoce specializovaných funkcí pro přiřazení míry důvěry, ale zároveň jako nevýhodu, jelikož tyto specializované funkce nelze snadno nasadit v jiných situacích. V následující podsekci na tento aspekt narazíme při porovnávání existujících použití D-S teorie pro fúzi informací.

Nepopiratelnou výhodou D-S teorie je její vyjadřovací schopnost neznalosti. Potenciálně by tedy mohla být vhodná pro klasifikaci předem neznámých dat. Oproti tomu Bayesovská inference vyžaduje předchozí znalost a neumožňuje přiřadit pravděpodobnost nevědomosti. Bayesovský přístup nemusí být vždy vhodný, pokud není možné zaručit předchozí znalost situace, například pro klasifikaci dříve neznámých útoků v počítačových sítích [9].

Existují dvě hlavní potenciální komplikace spojené s použitím D-S teorie. Prvním problémem je problém výpočetní složitosti, druhý problém je problém ztráty informací při spojování konfliktních funkcí důvěry.

Výpočetní složitost slučování dvou funkcí důvěry roste exponenciálně s velikostí rámce domnění Θ. Pokud má Θ n prvků, může funkce důvěry obsahovat až 2^{n-1} fokálních hypotéz a sloučení dvou funkcí důvěry bude vyžadovat výpočet až 2^n průniků [9]. Existují příklady aplikace aproximačních technik pro snížení výpočetní složitosti, například Reineking v [32] ukazuje přístup na základě metody Monte-Carlo. V této práci je použit rámec domnění o velikosti nízkých desítek, přičemž výchozí funkce důvěry mají pouze dvě fokální hypotézy, jak je popsáno v kapitole 3. Pro cílovou aplikaci v rámci projektu ADiCT navíc není výkonnost řešení kritická.

Na problém ztráty informace při spojování konfliktních funkcí důvěry poukázal jako první Zadeh v [42]. Jeho ukázka problému na hypotetické situaci je následující:

Uvažujme pacienta P, který je vyšetřen dvěma doktory, A a B. A diagnostikuje pacienta tak, že má buďto zápal mozkových blan, s pravděpodobností 0,99, anebo nádor na mozku, s pravděpodobností 0,01. B diagnostikuje pacienta tak, že má buďto otřes mozku, s pravděpodobností 0,99, anebo také připouští možnost nádoru na mozku, s pravděpodobností 0,01. Pokud diagnózy doktorů A a B budeme reprezentovat pomocí funkcí důvěry a aplikujeme na ně Dempstrovo kombinační pravidlo, výsledná funkce důvěry bude obsahovat pouze závěr, že P má nádor na mozku, s pravděpodobností 1,0.

Tento výsledek jde jasně proti intuici, jelikož oba doktoři přiřadili právě této možnosti velmi malou míru důvěry. Z pohledu fúze informací zde dochází ke ztrátě informace, konkrétně informace o nejvíce pravděpodobných diagnózách dle jednotlivých doktorů. Problém ztráty informace může vzniknout tehdy, když $|\Theta| > 2$, a když fokální hypotézy slučovaných funkcí obsahují rozdílné podmnožiny základních hypotéz, jelikož DRC v tomto případě vede k jejich průniku.

Tento aspekt použití D-S teorie v relevantní literatuře je jeden z klíčových bodů popsaných v podkapitole 2.2.3. Na základě zkoumaných článků je zároveň navrženo řešení použité v této práci, popsáno v kapitole 3.

2.2.3 Využití D-S teorie

Toto shrnutí relevantních výzkumných článků se soustředí na několik klíčových oblastí použití D-S teorie pro fúzi dat. Ukazuje používaný způsob návrhu funkcí důvěry, kterým autoři řeší problém ztráty informací při vysoké míře konfliktu, používané přiřazení konkrétních hodnot míry důvěry pro výstupy jednotlivých klasifikátorů a jejich případné optimalizace a prochází několik vybraných článků podrobněji pro lepší přiblížení prací čtenáři.

Prvním klíčovým aspektem zkoumaných článků je způsob návrhu funkcí důvěry. Několik ze zkoumaných článků se problému zcela vyhýbá tím, že provádí fúzi vedoucí k binární klasifikaci [43, 12, 30]. V těchto pracích je $|\Theta| = 2$, a tak k danému problému nemůže dojít. Lze poznamenat, že v případě, kdy $|\Theta| = 2$, je tedy návrh použití D-S teorie výrazně ulehčen. V této práci bohužel platí $|\Theta| > 2$, a tak tyto články nepřináší v tomto ohledu příliš mnoho inspirace.

Další skupina zkoumaných článků z pohledu návrhů funkcí důvěry jsou ty, které navrhují funkce důvěry ručně [7, 15, 9]. K transformaci zkoumaných hodnot na funkce důvěry zde dochází na základě velmi specifických vztahů, buďto vytvořených autory nebo převzatých z relevantních přístupů v oblasti řešeného problému. Je zde ukázáno, že použití D-S teorie tímto způsobem se vyrovnává jiným nejmodernějším přístupům v době vydání prací. Zároveň se ukazuje, že tento způsob tvorby je časově náročný, a výsledky v podobě navržených vztahů nelze znovu použít v jiné oblasti. Zároveň, alespoň v případě [15] a částečně i [9], nedochází k fúzi dat, ale spíše fúzi vlastností klasifikovaného vstupu. Toto není příliš srovnatelné s cílem této práce, navíc z důvodu zmíněných nevýhod nelze ruční tvorbu funkcí důvěry vyhodnotit jako vhodnou cestu.

Pro lepší rozdělení zbylých zkoumaných prací je třeba provést krátké rozdělení typů klasifikátorů na základě jejich výstupu, jak je uvedeno v [21]. Klasifikátory lze takto dělit do tří typů:

1. Štítky tříd. Klasifikátory prvního typu mají na výstupu klasifikovanou třídu z množiny všech možných.

2. Seřazené štítky tříd. Klasifikátory druhého typu mají výstup v podobě pořadí všech klasifikovatelných tříd, od nejpravděpodobnější po nejméně pravděpodobnou.

3. Číselné vyjádření podpory jednotlivých tříd. Výstup klasifikátorů třetího typu je v podobě vektoru, kdy hodnota na indexu i vyjadřuje podporu hypotézy, že klasifikovaný prvek patří do třídy s odpovídajícím indexem i. Po normalizaci lze také chápat jako pravděpodobnostní rozdělení mezi jednotlivé třídy.

V rešerši této práce se objevily práce řešící fúzi klasifikátorů prvního a třetího typu. V případě klasifikátorů třetího typu, jak lze vidět v [39], je použití D-S teorie provedeno přímým převedením výsledného pravděpodobnostního rozdělení klasifikátorů do podoby funkcí důvěry. Tento způsob použití vede k velmi dobrým výsledkům, kdy autoři překonávají nejen použité individuální klasifikátory, ale dokonce jiné z nejmodernějších přístupů. Pro účely této práce bohužel nelze s pravděpodobnostním rozdělením jakožto výstupem klasifikátorů počítat, jak je uvedeno v sekci 2.1.

Klasifikátory prvního typu, poskytující pouze štítek třídy, jsou nejbližší cílům této práce. V tomto případě je ve zkoumaných článcích [2, 10, 1] jasná shoda v postupu v tom, že je třeba pravděpodobnostní rozdělení nějakým způsobem nahradit, což bude podrobněji rozebráno níže.

Z pohledu řešení ztráty informace se objevuje několik přístupů. V [2, 10] je použit přístup transformace štítku do funkce důvěry s dvěma fokálními hypotézami. Pokud je klasifikátorem i klasifikován štítek C a experimenty získaná pravděpodobnost je značena c, vzniká následující funkce důvěry m_i:

$$\begin{aligned} m_i(C) &= c, \\ m_i(\Theta) &= 1 - c \end{aligned} \tag{2.8}$$

Tento jednoduchý přístup zamezuje ztrátě informace. Jelikož všechny hypotézy jsou podmnožinou Θ, je vyloučeno, že by některá z hypotéz neměla při sloučení s další funkci důvěry průnik s žádnou jinou hypotézou.

V [1] je prezentována varianta, kdy jsou výsledky jednotlivých klasifikátorů zaznamenány do matice záměn. Při fúzi dat z více klasifikátorů je řádek matice záměn, odpovídající klasifikované třídě, použit jako pravděpodobnostní rozdělení pro funkci důvěry. Tímto není problém konfliktu ošetřen, nicméně, jelikož v práci dochází k fúzi dat z pouze dvou klasifikátorů, tato skutečnost se neprojevuje negativně ve výsledcích.

Dalším klíčovým aspektem zkoumaných článků je způsob nastavení měr důvěry pro optimalizaci výsledků fúze dat. Jsou uvažovány pouze články soustředící se na fúzi dat z klasifikátorů prvního typu. Objevené přístupy jsou: na základě matice záměn, použité v [10, 1], na základě minimalizace střední kvadratické chyby v [2] a na základě optimalizací metodou gradientního sestupu v [30]. Následuje podrobnější shrnutí článků.

Dainotti a kolektiv v [10] porovnávají různé metody fúze informací v kontextu multiklasifikace. Článek při převodu na funkce důvěry využívá matici záměn, přičemž výsledná funkce má formát popsaný rovnicí 2.8. Je porovnáno 6 kombinačních algoritmů na zvýšení přesnosti oproti samostatným klasifikátorům. Přesnost kombinačních algoritmů je měřena s různými podmnožinami natrénovaných klasifikátorů. V tomto srovnání dominují kombinační algoritmy Behavior Knowledge Space (BKS) a Werneckova metoda, která je BKS příbuzná. D-S teorie za těmito metodami zaostává o několik desetin procenta až v nejhorším případě o 1,5 %. Interpretace výsledků je taková, že nejúspěšnější samostatný klasifikátor má přesnost 97,2 %, přičemž teoretická maximální přesnost při kombinaci výsledků by byla 98,8 %. Nevyšší dosažená přesnost kombinací množiny klasifikátorů je 97,7 %, dosažená současně BKS i Werneckovou metodou. DST dosahuje v této množině klasifikátorů přesnosti 97,0 %. DST v jedné konfiguraci klasifikátorů (ze zkoušených osmi) překonává BKS a Werneckeho metodu, a to o 0,6 %. Článek se zaměřuje na použití několika klasifikátorů na tytéž data, přičemž zvýšení přesnosti pak vychází z překonání omezení jednotlivých klasifikátorů. Toto není zcela srovnatelné se situací v projektu ADiCT, kde se jedná o oddělené klasifikátory, pracující každý nad jinými daty. S tím souvisí i ztráta záruky výstupu všech klasifikátorů, tj. některé klasifikátory, čili vstupní moduly, nemusí pro dané zařízení poskytovat žádná data, což vyřazuje použití v tomto článku úspěšných metod.

Abdollahpour a kol. v [1] využívají D-S teorie pro sloučení výsledků dvou oddělených klasifikátorů pracujících nad rozdílnými zdroji dat, jejichž výstupem je štítek třídy (5 tříd). Výstup je převeden do funkcí důvěry použitím řádku v matici záměn a následným spojením pomocí DRC. Výsledné spojení ukazuje spolehlivost vyšší o jednotky až desítky procent

oproti jednotlivým výstupům klasifikátorů. DST je v závěru označena za mocný nástroj pro fúzi dat.

Al-ani a Deriche v [2] ukazují obecný způsob kombinace výsledků N klasifikátorů pomocí D-S teorie, přičemž důvěra přiřazená jednotlivým klasifikátorům je stanovena na základě experimentů nad trénovací datovou sadou tak, aby se minimalizovala střední kvadratická chyba. V článku navržená architektura předpokládá klasifikaci použitím několika klasifikátorů nad stejnými daty. Navržená technika ukazuje lepší výkonnost v porovnání s dalšími kombinačními metodami.

Peñafiel a kol. v [30] volí trochu jiný přístup oproti doposud zmíněným článkům. Vytváří interpretovatelný binární klasifikátor, implementovaný pomocí D-S teorie. Klasifikátor je tvořen vektorem pravidel, přiřazujících funkci důvěry, pokud je splněna stanovená podmínka vůči konkrétním vlastnostem klasifikovaného záznamu. Míry důvěry jednotlivých pravidel jsou optimalizovány vůči tréninkové datové sadě pomocí metody gradientního sestupu. Přesnost klasifikátoru je o něco nižší než ta nejmodernějších klasifikátorů, ale stále akceptovatelná. Svou metodu dále ověřují v [31], kde ji používají k předpovědi rizika infarktu v následujícím roce na základě zdravotních záznamů. Metoda dosahuje ukazatelů přesnosti srovnatelných s ostatními používanými metodami, přičemž v případě chybějících dat porovnávané metody překonává. Zároveň je ukázáno, že definice pravidel prostřednictvím expertů a jejich následná optimalizace pomocí gradientního sestupu je mocnou kombinací, kterou si autoři vysvětlují část úspěchu své metody.

Poslední z vybraných článků se tématicky odklání od fúze výstupů klasifikátorů, nicméně je pro tuto práci neméně důležitý. Reineking v [32] implementuje algoritmy D-S teorie a TBM, včetně několika alternativních kombinačních pravidel a aproximačních technik pro kombinační pravidla ve formě knihovny PyDS, napsané v jazyce Python. Sám demonstruje využití TBM k řešení několika problémů souvisejících se senzorickými daty (filtrování částic, simultánní lokalizace a mapování (SLAM) a aktivní rozpoznávání důkazů), kde porovnává výsledky při použití několika kombinačních pravidel a Bayesovské aktualizace. V závěru uvádí, že „status outsidera“, který je přisuzován teoriím funkcí důvěry, lze odstranit pouze tvořením příkladů demonstrujících přidanou hodnotu jejich využití, o což se s pomocí jeho implementace tato práce pokouší.

2.3 Metody detekce operačního systému ze síťového provozu

Následující sekce se věnuje relevantním metodám pro detekci operačního systému zařízení ze síťového provozu. Jsou popsány volně dostupné nástroje, detekce na základě parametrů protokolu TCP/IP, na základě pole User-Agent protokolu HTTP, na základě údajů předávaných při vytváření TLS spojení, a krátce je pojednáno i o využití logů z pohledu detekce.

2.3.1 Nástroje pro získávání otisku zařízení

Problém klasifikace operačního systému zařízení je řešen značným množstvím veřejně dostupných nástrojů. Nejznámějším zástupcem aktivních skenerů sítí je NMap [25], který využívá ke klasifikaci 16 specificky navržených sond, pomocí kterých zkoumá vlastnosti implementace TCP, UDP a ICMP daného zařízení. Na základě výsledku sond je sestaven charakteristický otisk subjektu. Otisk je vyhledán v databázi otisků, která obsahuje informace o daném operačním systému [24].

NMap se jakožto populární nástroj stává cílem bezpečnostních opatření. Alternativou může být hybridní nástroj, například SinFP, který nevyužívá více sond, než je nezbytně nutné. SinFP je navržen se snahou o co nejmenší šanci odhalení. Použije nanejvýš tři aktivní sondy, přičemž obsahuje nastavení umožňující počet aktivních dotazů dále redukovat v případě přítomnosti IDS (Intrusion Detection System). Zároveň umožňuje klasifikovat operační systém čistě pomocí pasivního monitorování [4].

Jako zástupce pasivně monitorujících nástrojů pro získávání otisků operačního systému lze zmínit p0f [38]. p0f (zkratka z passive OS fingerprinting) opět vytváří otisk subjektu na základě vybraných parametrů TCP. Téměř všechny nástroje pro pasivní klasifikaci operačních systémů využívají p0f pro provedení analýzy na úrovni TCP [24].

Přesnost klasifikátoru na základě databáze otisků se do velké míry odvíjí od aktuálnosti databáze. Databáze otisků jsou často udržovány ručně (je tomu tak v případě všech zmíněných nástrojů), což je časově náročné pro zodpovědné vývojáře. Tuto slabinu se pokouší Schwartzenberg ošetřit v [33] použitím strojového učení pro klasifikaci nových verzí operačních systémů.

Jiný přístup volí Matoušek a kolektiv [27], kteří ve své práci ukazují průběžnou aktualizaci databáze TCP/IP otisků na základě HTTP hlavičky User-Agent. Příchozí paket je nejprve zkontrolován, zda se jedná o SYN paket. Pokud ano, vybraná pole jsou porovnána s databází otisků a OS je identifikován. Pokud se nejedná o SYN paket, je zkontrolována přítomnost HTTP hlavičky, konkrétně dotazů GET a POST, ze které je následně vyjmut OS z pole User-Agent. Následně je vygenerován otisk, který je uložen do DB. Výchozí databáze dle autorů vychází z DB nástroje p0f a jejich vlastních experimentů. Přesnost klasifikace je navzdory řádově nižšímu počtu otisků v DB srovnatelná s přesností nástroje p0f. Ztrácíme tak ale podrobnější klasifikaci, kterou nástroj p0f nabízí. V případě on-line monitorování toto není autory vnímáno jako kritické. V porovnání s p0f má výsledný plugin výrazně více falešně negativních klasifikací (neznámý OS). Toto je vysvětlováno porovnáváním na dokonalou shodu, které lze vylepšit použitím shlukové analýzy. Je použita metoda K-means, kde databáze otisků je transformována do k-rozměrného prostoru. V případě SYN paketu dochází ke klasifikaci nalezením nejbližšího centroidu, následně je přepočítán průměr shluku. V případě HTTP paketu dochází k aktualizaci existujícího shluku či vytvoření nového shluku pro dosud neexistující OS. Nejlepší dosažená přesnost tímto přístupem je 92 %.

2.3.2 Klasifikace na základě parametrů TCP/IP

Tento způsob klasifikace byl již zmíněn okrajově v podsekci o dostupných nástrojích. Principiálně jde o zneužívání volnosti některých aspektů specifikace TCP/IP protokolu. Různé implementace TCP, používané různými operačními systémy, produkují rozdílné výchozí hodnoty. Na základě těchto odlišností lze detekovat jednotlivé operační systémy.

Varianta používaná v dostupných nástrojích spočívá v seskupení klíčových hodnot spojení do tzv. otisku, a vyhledávání otisku v existující databázi. Další variantou, v různých obměnách přítomnou v akademické literatuře, je použití strojového učení.

Laštovička a kol. [22] porovnávají několik existujících metod strojového učení v prostředí reálné akademické sítě. Cílem je zjistit, která z metod je nejvhodnější z hlediska potřebného strojového času, operační paměti a přesnosti samotné klasifikace. Ze zkoumaných metod (Naive Bayes, Decision tree, SVM a K-NN) se ukazuje jako nejvhodnější metoda Decision tree, která dosahuje přesnosti 97,6 %, s nejrychlejším časem 0,0876 s na 1 000 000 klasifikovaných vzorků.

Hagos a kol. [16] ukazují, že modely strojového učení produkují lepší výsledky po rozšíření vstupního vektoru o predikovanou variantu algoritmu řízení zahlcení sítě protokolu TCP. Jsou provedeny čtyři experimenty pro porovnání různých scénářů, přičemž jsou zahrnuty metody SVM, Random Forest, K-NN, Naive Bayes, Mulitlayer Perceptron a Long Short-Term Memory. Výsledky ukazují 6 % zlepšení přesnosti, z 88 % bez algoritmu řízení zahlcení na 94 % při jeho zahrnutí jako vstupu, získaném z druhého modelu hlubokého učení. Z důvodu potřeby přítomnosti congestion window dat pro predikci implementace řízení zahlcení je pro porovnání použita pouze emulovaná datová sada, vytvořená autory.

Laštovička a kol. [23] rovněž ukazují kombinovaný přístup ke klasifikaci operačních systémů. Práce kombinuje tři známé metody – TCP/IP fingerprinting, detekci na základě HTTP pole User-Agent a detekci podle komunikace se specifickými doménami pro daný operační systém. Výsledky těchto jednotlivých metod jsou kombinovány na bázi většinového hlasování. Autoři řeší problém různé podrobnosti klasifikace jednotlivých metod pomocí hierarchie, která je strukturována do čtyř úrovní: výrobce, jméno OS, hlavní verze, vedlejší verze. Kombinace metod je možná jen do úrovně jména OS. Výsledná přesnost se ukazuje nejlepší pro izolovanou metodu HTTP User-Agent, 91,8 %, metoda TCP/IP dosáhla přesnosti 80,8 %, metoda detekce na základě domén 84 %. Kombinace pomocí většinového hlasování pak dosahuje přesnosti 85,8 %, tedy ukazuje se jako průměr pozitiv i negativ jednotlivých metod.

2.3.3 Klasifikace pomocí HTTP pole User-Agent

Hlavička User-Agent protokolu HTTP je definována v [13] jakožto řetězec od uživatele posílajícího HTTP dotaz, obsahující informace o používaném prohlížeči a operačním systému. Hlavička User-Agent je opodstatněna jako informace napomáhající serveru odhadnout známé problémy v kompatibilitě, aby mohl upravit odpovědi takovým způsobem, kterým se vyhne omezením implementace konkrétního klienta. Zároveň může být User-Agent použit pro tvorbu statistik ohledně používaných prohlížečů a operačních systémů. Je explicitně zmíněno, že klient by neměl vkládat do pole User-Agent další podrobnosti a měl by omezit rozšiřování pole třetími stranami, mimo jiné proto, aby se zabránilo identifikaci uživatele na základě otisku z tohoto pole.

V praxi je běžné, že aplikace do tohoto pole vyplní operační systém včetně hlavní a vedlejší verze, často dokonce s konkrétním číslem sestavení dané verze. Na základě řetězce User-Agent lze tedy s pomocí lexikálního analyzátoru (parseru) detekovat operační systém [23].

Ukázku použití hlavičky User-Agent pro detekci operačního systému lze vidět v [23], kde se metoda projevuje jako nejpřesnější (91,89 %), nicméně pokrývá pouze 64,3 % klasifikovaných sezení. Jak již bylo zmíněno, operační systém získaný z pole User-Agent je zároveň použit v [27] pro aktualizaci existující databáze TCP/IP otisků. Dalším příkladem využití HTTP User-Agent dat je k dispozici v [16], kde slouží k anotaci použité datové sady, původně vytvořené v [23].

2.3.4 Detekce na základě TLS provozu

Analýza otisků protokolu TLS je metoda umožňující identifikovat komunikující strany. Otisky jsou vytvořeny z parametrů TLS komunikace, které klient nabízí serveru při jejich spojení. Různé kombinace procesů, operačních systémů a kryptografických knihoven vytváří různé kombinace těchto parametrů. Je tedy možné extrapolovat informace o klientovi na základě vytvořeného otisku a databáze otisků různých klientů [17].

Metoda JA3 [19] je nejrozšířenější metodou pro vytváření otisků TLS spojení. Využívá zprávy ClientHello a ServerHello pro vytvoření otisku úvodního vyjednávání mezi klientem a serverem, přičemž využívá hashovací funkci MD5 pro vytvoření 32 znakového otisku, který lze snadno zpracovat [14].

Nástroj Mercury [28], vyvinutý v rámci výzkumu společnosti CISCO, produkuje otisky zahrnující navíc kontextuální informace z paketu obsahujícího zprávu ClientHello. Zahrnutím těchto informací je možné rozlišit různé komunikace, produkující identické JA3 hashe. Anderson a kol. v [3] využívají tento nástroj v kombinaci s klasifikačním algoritmem Naive Bayes, tento přístup dosahuje ukazatelů úspěšnosti preciznost 99,9 % a senzitivita 88,7 %.

Matoušek a kol. v [26] ověřují spolehlivost použití JA3 hashů pro detekci mobilních aplikací. Práce ukazuje, že kombinace s dalšími informacemi z protokolu TLS, jako je hash serveru JA3S a pole Server Name Indicator (SNI), výrazně zlepšuje všechny klíčové ukazatele úspěšnosti detekce.

2.3.5 Klasifikace z logů

V případě klasifikace operačních systémů z pozice administrátora sítě je možné detekovat operační systém zařízení pomocí zpracování dat dostupných z logů. Pro tento účel je možno použít logů z DHCP serveru, případně Radius serveru, zároveň v kombinaci s daty z polí User-Agent protokolu HTTP, jak již bylo zmíněno. Tento přístup je často používán pro anotaci nasbírané datové sady pro další klasifikační experimenty nad síťovým provozem, jak lze pozorovat v [23, 16, 29]. Nevýhodou tohoto přístupu je zřejmá nutnost přístupu do správy sítě.

Další metody detekce

Výše zmíněný výčet nelze považovat v žádném případě za úplný. Snahou této podsekce bylo pokrýt metody, které jsou relevantní k modulům projektu ADiCT, viz podsekce 2.1.2.

Pro úplnost lze uvést přístup, který zkoumají Millar a kol. v [29]. Detekce operačních systémů je zde provedena čistě na základě IP adres, se kterými zařízení komunikuje. Využívá afilačních grafů, které jsou předkládány v podobě kódování 1 z N klasifikátoru RandomForest. Problém klasifikace rozděluje do dvou fází, nejprve dochází ke klasifikaci rodiny, kde je množina tříd Apple, Windows a Android. Poté následuje klasifikace konkrétního operačního systému, opět do množiny tříd. Dosažená přesnost klasifikace rodiny je 99,3 %, konkrétního operačního systému 94,6 % za využití 200 nejdůležitějších komunit. Výhodou je nezávislost na obsahu komunikace, jeho šifrování apod. Slabinou metody je použití VPN uživateli, pokud by VPN byla využita pro všechnu komunikaci. Studie obsahuje porovnání po 3 a 6 měsících po natrénování původního modelu, které ukazuje zhoršení přesnosti v řádu jednotek procent u Apple a Android OS, ale až 29 % v případě Windows. Závěrem je, že pro uchování přesnosti v produkčním prostředí je třeba model každých několik měsíců znovu natrénovat.

2.4 Metody detekce typu zařízení ze síťového provozu

Detekce typu zařízení na základě síťového provozu je v akademické literatuře skloňována především v kontextu internetu věcí (IoT). Tato sekce se krátce věnuje shrnutí současného stavu výzkumu na tomto poli.

Yang a kol. v [41] ukazují generování otisků (fingerprint) k typu zařízení pomocí neuronové sítě. Pro získání relevantních štítků zařízení, jejich typu a výrobce je použit webcrawler. Anotace nasbírané datové sady, ve formě jednotlivých paketů, probíhá pomocí regulárních výrazů, hledajících konkrétní názvy výrobců či označení typu zařízení v obsahu paketů. Takto označená data tvoří tréninkovou datovou sadu. Neuronová síť následně generuje otisky zařízení na třech úrovních podrobnosti, kterými jsou typ zařízení, jeho výrobce a konkrétní produkt. Otisky obsahují informace ze síťové, transportní a aplikační vrstvy. Přesnost generovaných otisků je otestována, dosažená přesnost je pro úroveň typu zařízení 94,7 %, pro výrobce zařízení 93,3 % a pro konkrétní produkt 91,4 %.

Bai a kol. v [5] provádí klasifikaci sémantického typu zařízení na základě časové analýzy jeho komunikace. Jsou definovány 4 sémantické kategorie (zařízení IoT hub, domácí spotřebiče´, kamery a spínače). Je použita kombinace konvoluční a neuronové sítě. Tímto přístupem je dosažena přesnost 74,8 % při rozdělení datové sady na trénovací a testovací data v poměru 1:1.

Hsu a kol. v [18] ukazují klasifikaci zařízení pouze na základě bytů za sekundu, směru paketu a času od poslední komunikace. Cílem tohoto omezení je snaha klasifikovat bez závislosti na nešifrovaném obsahu přenášených dat. Jsou získávány otisky pro kategorii zařízení, přičemž byly vytyčeny 4 kategorie (chytré zásuvky, kamery, pohybová čidla a teplotní čidla). Metoda umožňuje klasifikaci předem neznámých zařízení. Součástí práce je i demonstrace v produkčním prostředí, zahrnujícím několik podsítí a překladů adres. Ukazuje se, že přesnost metody závisí na kategorii zařízení. Při aplikaci metody na veřejně dostupné datové sadě UNSW byla metoda schopna rozeznat jen některé typy zařízení, navíc tato datová sada obsahovala zařízení, jejichž typ se nevyskytoval v tréninkové datové sadě.

Tato podsekce byla velmi stručným shrnutím současného stavu na poli detekce a klasifikace typu zařízení na základě jeho síťové komunikace. Zmíněné přístupy se ukázaly jakožto nesrovnatelné s těmi vyskytujícími se v projektu ADiCT (viz podsekce 2.1.2). Zdroje projektu ADiCT obsahují typ zařízení v případě, že je tato informace přímo odvoditelná z obsahu aplikační či transportní vrstvy síťové komunikace zařízení. Zmíněné přístupy ukazují použití databáze otisků či aplikaci strojového učení nad konkrétními parametry komunikace. Jelikož cílem této práce je fúze existujících zdrojů dat, nikoliv rešerše za účelem vývoje zdrojů nových, rešerše tímto končí.

Kapitola 3

Návrh řešení

Tato kapitola obsahuje definici cílů práce a návrh řešení na nich založený. Cíle jsou definovány v sekci 3.1 a stanovují požadavky, které by řešení mělo splňovat. Na základě vznesených požadavků jsem řešenou úlohu dekomponoval do dvou částí.

První část je transformace hodnot vybraných atributů do jednotné reprezentace definované pro každý štítek (operační systém, typ zařízení), pokud je to možné. Pro formát jednotné reprezentace jsem určil taxonomii, protože tímto způsobem bude možné vyjádřit různou míru podrobnosti jednotlivých atributů, jak je popsáno v sekci 3.6. Pro specifikaci konverze hodnot atributů jsem navrhl podobu sady klasifikačních pravidel, které přiřadí hodnotu třídy z taxonomie na základě splněné podmínky. Koncept klasifikačních pravidel je popsán v sekci 3.3. Na základě výčtu typů podmínek, které je třeba podporovat, a uživatelské přívětivosti tvorby těchto podmínek jsem navrhl jednoduchý konfigurační jazyk, který bude zpracován interpretem. Návrh jazyka a interpretu je řešen v sekcích 3.4 a 3.5.

Druhou částí je využití přenositelného modelu domnění pro kombinaci již transformovaných hodnot a vybrání nejpravděpodobnějšího štítku jako výsledku. V rámci této části je rozhodnut formát použitých funkcí domnění. Z důvodu logické návaznosti je sekce 3.2, věnující se této části, předřazena té předchozí.

3.1 Cíle práce

Cílem práce je sloučení dat z jednotlivých atributů do sjednoceného štítku. Přesnost tohoto štítku by měla být lepší než ta z individuálních modulů či atributů. Slučování bude dle zadání zaměřeno na dva aspekty: typ operačního systému a typ zařízení. Způsob sloučení by měl být schopen pojmout veškeré typy dat, které jsou naznačeny v tabulkách 3.1, pro typ operačního systému, a 3.2, pro typ zařízení. Jak tabulky naznačují, logika pro rozpoznání konkrétních hodnot bude muset být přizpůsobitelná dle konkrétního atributu. Atributy přítomné v tabulkách byly dříve definovány v podsekci 2.1.2.

Dalším požadavkem na řešení je nutnost přizpůsobení se míře podrobnosti výstupu daného modulu. V případě operačního systému by například mělo být rozlišováno mezi názvem a konkrétní verzí.

V neposlední řadě je nutné, aby řešení bylo konfigurovatelné do stejné míry, jako je systém ADiCT. Při přidání nového atributu či změně datového typu existujícího lze po autorovi změny požadovat specifikaci konverze jeho atributu do sloučené reprezentace, ať je jakákoliv. Stejně tak sloučená reprezentace musí být konfigurovatelná, aby byla například schopna zahrnout nové verze operačních systémů. V případě rozšíření akceptovaných dato-

vých typů v systému ADiCT by mělo řešení nabídnout způsob rozšíření způsobu specifikace konverze.

atribut	očekávané hodnoty	typ zpracování
`operating_system_ua`	`'Fedora'`, `'Mac OS X 10.4 (Tiger)'`	vyhledání podřetězce
`http_useragent`	`'Windows-Update-Agent/10.0. ...'`	regulární výrazy
`os_by_tcpip`	`'Ubuntu'`	překladová tabulka
`tags_by_services`	`['Windows:3389']`	vyhledání podřetězce v poli
`tls_os_family`	`['Mac OS X']`	vyhledání řetězce v poli
`tls_os_name`	`['Mac OS X', 'Catalina']`	vyhledání podřetězce v poli
`tls_os_version`	`['Mac OS X', 'Catalina', '10.15.5']`	vyhledání podřetězce v poli
`sdp_label`	`{'name': 'hostname',` `'value': 'MacBook-Pro-3.local'}`	kontrola hodnot pod několika klíči slovníků uložených v poli

Tabulka 3.1: **Souhrn atributů využitelných pro odvození operačního systému.** Sloupec *typ zpracování* ukazuje typ operace potřebný pro převod do společné reprezentace.

atribut	očekávané hodnoty	typ zpracování
`hardware_type_ua`	`'computer'`, `'server'`, `'tablet'`, ...	překladová tabulka
`software_type_ua`	`'web-browser'`, `'in-app-browser'`, ...	překladová tabulka
`tls_categories`	`'browser'`, `'programming'`, `'gaming'`, ...	překladová tabulka
`tags_by_services`	`['WEB Server:443', 'WEB Server:80']`	vyhledání podřetězce v poli
`ssdp_service`	`{'port': 80,` `'service': 'schemas-upnp-org: ...'}`	kontrola hodnot pod několika klíči slovníků uložených v poli
`dnssd_service`	`{'service': '_ipp._tcp.local'}`	kontrola hodnoty ve slovníku

Tabulka 3.2: **Souhrn atributů využitelných pro odvození typu zařízení.**

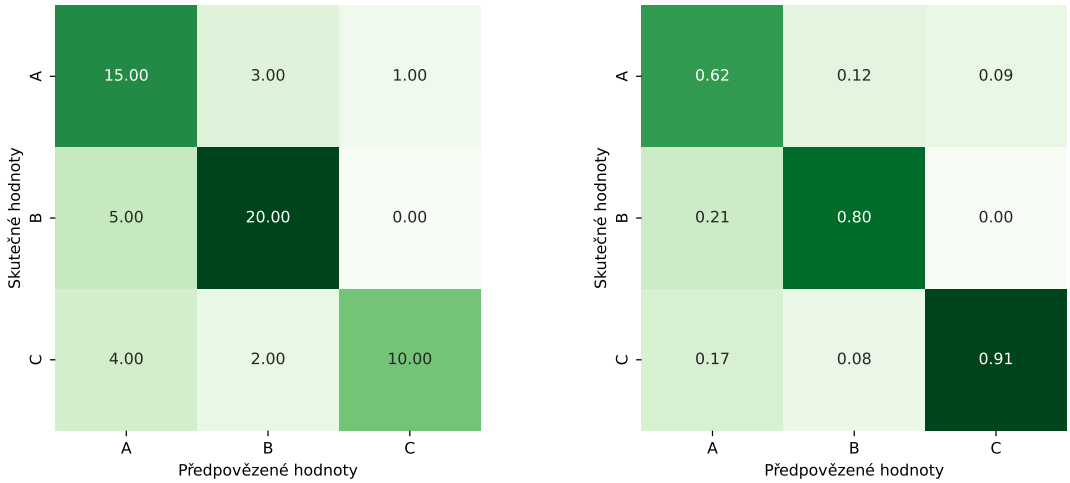
3.2 Použití TBM pro sloučení všech dostupných dat k dané entitě

Tato sekce uvažuje návrh pro zjednodušený problém slučování dat, kdy všechny vstupy již mají stejný formát. Pro řešení tohoto problému jsem zvolil použití funkcí důvěry v modelu přenositelného domnění pro reprezentaci jednotlivých vstupů. Nabízí se dvě možnosti tvorby těchto funkcí důvěry.

První způsob tvorby, použit v [2, 10] a podrobně popsán v sekci 2.2.3, produkuje funkce důvěry o dvou fokálních hypotézách. První je hypotéza obsahující hodnotu vstupu, druhá hypotéza obsahuje rámec domnění. K hodnotě vstupu je přiřazena stanovená míra důvěry, rámec domnění pak má míru důvěry, která je doplňkem do 1. Pro zjednodušení nazvěme hypotézu s hodnotou vstupu hlavní hypotézou a tento přístup metodou dvou fokálních hypotéz. Pokud by funkce důvěry měla vyjadřovat štítek A, kterému bude přiřazena míra důvěry 0,6, bude vypadat takto:

$$m_A = \{A : 0.6, \Theta : 0.4\} \tag{3.1}$$

Druhý způsob tvorby, použit v [1], produkuje funkce důvěry na základě matice záměn (confusion matrix) jednotlivých vstupů. Tuto metodu nazvěme metodou distribuce záměn. Pro lepší ukázku přístupu jsou na obrázku 3.1 vyobrazeny matice záměn pro třídy A, B a C.



Obrázek 3.1: **Ukázka matice záměn.** Matice vlevo obsahuje absolutní počty hodnot, matice vpravo má normalizované sloupce.

Matice vlevo obsahuje absolutní počty hodnot v klasifikované datové sadě, přičemž řádky ukazují skutečné hodnoty a sloupce hodnoty předpovězené. Matice vpravo ukazuje tytéž výsledky, ale má normalizované sloupce. Ve funkci důvěry vytvořené touto metodou jsou přiřazeny každé třídě jako individuální hypotéze míry důvěry, odpovídající sloupci dané predikované třídy v normalizované matici záměn. Touto metodou by pro představené třídy byly vytvořeny následující funkce důvěry:

$$\begin{aligned} m_A &= \{A : 0.62, B : 0.21, C : 0.17\} \\ m_B &= \{A : 0.12, B : 0.80, C : 0.08\} \\ m_C &= \{A : 0.09, B : 0.00, C : 0.91\} \end{aligned} \tag{3.2}$$

Ve fázi návrhu jsem zohlednil možnost nedostupnosti matice záměn, například v případě nedostatečné anotované datové sady pro validaci. Zvolil jsem proto metodu dvou fokálních hypotéz, které bude možno snadněji konfigurovat autorem daného vstupu do systému ADiCT. I tato metoda může těžit z přítomnosti anotované datové sady, pokud bude k dispozici. Namísto staticky nastavených měr důvěry lze využít pro míru důvěry hlavní hypotézy hodnoty na diagonále normalizované matice záměn.

Z výše uvedeného jsem odvodil, že řešení musí přijímat i dodatečnou konfiguraci, která každému vstupu přiřadí míru důvěry. Tyto vstupy jsou metodou dvou fokálních hypotéz transformovány do podoby funkcí důvěry. Pro N vstupů je vytvořeno N funkcí důvěry, které lze již kombinovat pomocí DRC. Výsledkem je jediná funkce důvěry, která odráží všechny vstupy. Výstupem řešení je pak nejpravděpodobnější třída, která bude získána z funkce důvěry po provedení pignistické transformace.

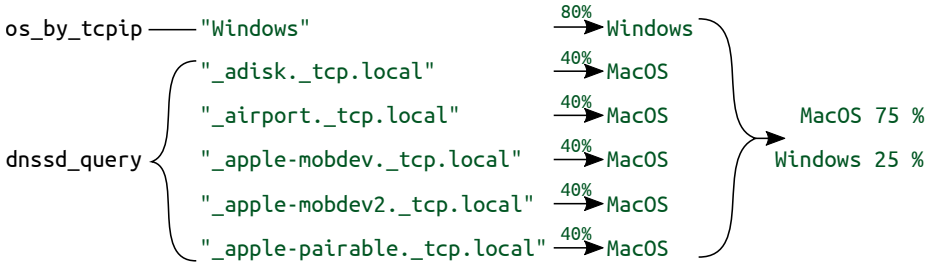
3.3 Klasifikační pravidla

Pro transformaci skutečného problému na ten popsaný v předchozí sekci jsem zvolil přístup klasifikačních pravidel. Klasifikační pravidlo se skládá z několika konfigurovatelných částí.

První je podmínka nad záznamem dané entity. Druhou částí je štítek ze sjednocené reprezentace zároveň se stanovenou mírou důvěry. Při splnění podmínky pravidlo vrací přiřazený štítek, při nesplnění nemá výstup.

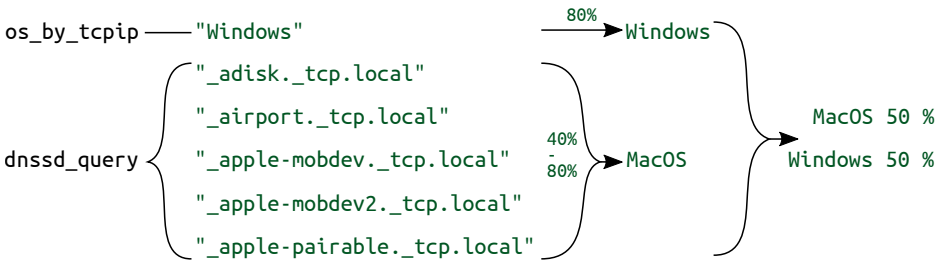
V rámci podmínky je přistupováno k hodnotám jednotlivých atributů entity. Na základě očekávaného zpracování dat, jak bylo uvedeno v tabulkách 3.1 a 3.2, jsem navrhl potřebné operátory pro porovnání elementárních hodnot, slovníků, zpracování atributu pomocí regulárních výrazů a vyhledávání podřetězce. Vzhledem k převažujícímu nastavení platnosti vícero hodnot atributů v jednom časovém úseku jsem navrhl způsob aplikování těchto operátorů na pole hodnot. Tyto jednoduché výrazy jazyka je možné spojovat pomocí logických spojek. Podrobnější specifikace syntaxe podmínek bude uvedena v rámci následující sekce 3.4.

V rámci prvotních experimentů se zpracováním atributů jsem objevil potřebu pro pravidla, která budou záviset na několika podmínkách. Toto je užitečné v případě modulů, kde je výstup bližší surovým extrahovaným hodnotám a klasifikační pravidla plní roli prvního převodu na štítek, například modul SDP Analyzer (vizte sekci 2.1.2). V těchto případech dochází k tomu, že větší množství hodnot identifikuje jeden výsledný štítek. Identifikoval jsem problém, kdy mohlo dojít k převážení systému při použití pravidel tak, že každá výstupní hodnota takového modulu produkovala vlastní štítek, zatímco moduly dávající na výstup jedinou klasifikaci produkovaly pouze jeden, jak je ilustrováno na obrázku 3.2. Jádrem problému je, že není v zájmu přesnosti klasifikace uměle snižovat míru důvěry v pravidla závislá na modulu s vyšší kadencí, konkrétně uvedený atribut `dnssd_query` predikuje systém Mac OS s vysokou přesností. Zároveň ale není možné vytvořit jediné pravidlo spojením jednotlivých podmínek disjunkcí, protože je rozdíl, zda je z daných podmínek splněna jedna, či vícero.



Obrázek 3.2: **Ilustrace problému převážení systému kadencí vstupu.**

Tento nepoměr v experimentech vedl k „přehlasování“ modulů s nižší kadencí výstupních hodnot. V rámci vyvážení jednotlivých zdrojů dat jsem navrhl řešení v podobě pravidel s rozsahem. Efekt pravidla s rozsahem je opět ilustrován, tentokrát na obrázku 3.3



Obrázek 3.3: **Ilustrace řešení problému pomocí pravidla s rozsahem.**

Rozdíl oproti základním pravidlům definovaným v začátku této sekce je, že pravidla s rozsahem závisí na seznamu podmínek. Zároveň je jim přiřazen rozsah míry důvěry namísto pevně stanovené hodnoty. Pravidlo přiřadí nastavený štítek při splnění jedné a více podmínek. Míra důvěry je interpolována ve stanoveném rozsahu na základě počtu splněných podmínek. Tímto způsobem lze přiřadit i relativně vysokou důvěru pro jakoukoliv z objevených hodnot (není pravidlem, že se v provozu objeví všechny), ale zároveň odlišit mezi objevením jedné hodnoty od objevení většího počtu hodnot.

3.4 Konfigurační jazyk pro specifikaci pravidel

Požadavky na konfigurační jazyk jsou následující. V rámci konfigurace musí být vyjádřeny podmínky, specifikující klíčové hodnoty v rámci atributu a štítek, který těmto klíčovým hodnotám bude přiřazen. Výstupem překladu tohoto konfiguračního jazyka bude sada pravidel, jak byla definována v předchozí sekci, spustitelná nad záznamem entity a převádějící všechny konfigurovatelné hodnoty do společné reprezentace. Použití konfiguračního jazyka by mělo být uživatelsky příjemnější než úprava kódu v jazyce Python, pomocí kterého bude jazyk interpretován. Tato sekce pokrývá návrh syntaxe a sémantiky jazyka.

Jazyk obsahuje dvě základní konstrukce, které slouží ke konfiguraci dvou typů pravidel, popsaných v předchozí sekci. Pro popis základního pravidla jsem navrhl tuto syntaxi:

```
<Třída>
    <MíraDůvěry>: <Podmínka>
```

`Třída` představuje štítek ze sjednocené taxonomie, který bude přiřazen při splnění podmínky `Podmínka`, spolu s mírou důvěry stanovenou `MíraDůvěry`. Míra důvěry je číselná hodnota v rozsahu $\langle 0, 1 \rangle$, která je zadávána s desetinnou tečkou. Dokud je zachováno odsazení po specifikaci třídy, lze na nových řádcích zadávat další dvojice `<MíraDůvěry>: <Podmínka>`, čímž budou definována další nezávislá pravidla, přidělující specifikovanou třídu.

V rámci deklarace třídy `Třída` lze uvést i několik štítků ze sjednocené taxonomie oddělených čárkou. Pak je míra důvěry přiřazena celé množině tříd. Rovněž je umožněno před seznam štítků uvést vykřičník, což je sémanticky interpretováno jako doplněk z Θ k množině (uvedenému seznamu).

Syntaxe pro konfiguraci pravidla s rozsahem je obdobná té základního pravidla:

```
<Třída>
    <MinimálníMíraDůvěry> - <MaximálníMíraDůvěry>: [
        <Podmínka1>
        <Podmínka2>
        ...
    ]
```

`MaximálníMíraDůvěry` a `MinimálníMíraDůvěry` jsou opět číselné hodnoty s desetinnou tečkou, které popisují rozsah důvěry přidělovaný pravidlem. Seznam podmínek uvedený v hranatých závorkách určuje podmínky, které budou pravidlem kontrolovány.

Syntaxe podmínek samotných je inspirovaná jazykem Python a umožňuje provádění základních dotazů nad entitou. Přístup k hodnotám atributů entity je vyjádřen notací začínající tečkou, následovanou jménem atributu, například `.os_by_tcpip`. Pro elementární typy jsou podporovány základní operátory porovnání (`<`, `>`, `==`). Spojování pravdivostních hodnot je umožněno logickými operátory `or` a `and`. Pro kontrolu přítomnosti hodnoty v poli či atributu mající vícero platných hodnot současně slouží operátor `in`. Konfiguraci lze do-

plnit komentáři, které začínají znakem # a respektují úroveň odsazení. Konkrétní příklad na základě doposud stanovených pravidel jazyka:

```
# Zde jsou definována dvě pravidla pro štítek OperatingSystem.Windows
OperatingSystem.Windows
    0.2: .x > 0 or .y < 0.6
    # Složená podmínka
    0.5: .a == 'str' and (.b == 1 or .c == 1)

# Zde je definováno jedno pravidlo s rozsahem
OperatingSystem.Linux
    0.4 - 0.7: [
        '_ftp._tcp.local' in .dnssd_query
        '_nfs._tcp.local' in .dnssd_query
        '_webdavs._tcp.local' in .dnssd_query
        '_webdav._tcp.local' in .dnssd_query
    ]
```

Pro rozšíření vyjadřovací schopnosti jazyka je k dispozici sada funkcí, která je rovněž rozšiřitelná. Syntaxe pro použití funkce je `<JménoFunkce>(argument1, argument2, ...)`, přičemž počet argumentů může být 0 až N. Jako argument lze předat jak konstantu, tak atribut entity. Funkce může mít větší množství signatur s odlišnou implementací. Jako příklad použití funkce je ukázána funkce `re`, která umožňuje zpracování pomocí regulárních výrazů.

```
OperatingSystem.Windows
    0.5: re('a+b?') in .y
    0.5: re('a+b?', .a)
```

V rámci použití operátoru `in` lze provádět kontrolu slovníků, což je umožněno pomocí slovníkové notace odpovídající jazyku Python či formátu JSON[1]. Funkcionalita bude vysvětlena na následujícím příkladu:

```
OperatingSystem.MacOS
    0.4 - 0.8: [
        {'name': 'os', 'value': 'Mac OS X'} in .sdp_label
        {'name': 'hostname', 'value': contains('MacBook')} in .sdp_label
        {'service':'_ssh._tcp.local'} in .dnssd_service
    ]
```

Atribut `sdp_label` je definován jako slovník s klíči `name` a `value`, přičemž oba mají datový typ `string`. První podmínka kontroluje přítomnost přesně takového slovníku, jaký je uveden, jde tedy o doslovnou kontrolu pro oba klíče.

Druhá podmínka ukazuje použití funkce `contains` v rámci slovníku. Jakákoliv kompatibilní funkce může být tímto způsobem použita namísto hodnoty v rámci slovníkové notace. Sémantika této podmínky je taková, že bude platná pro slovníky s hodnotou `'hostname'` pod klíčem `name`, a takovou hodnotou klíče `value`, která po zavolání funkce `contains` vrátí pravdivou hodnotu, tedy obsahuje podřetězec `'MacBook'`.

Třetí podmínka ukazuje, že ne všechny klíče slovníku musí být pro kontrolu specifikovány. Atribut `dnssd_service` je definován jako slovník s klíči `port` a `service`. Pokud, jak

[1] https://www.json.org/json-en.html

je ukázáno na příkladu, není pro danou podmínku klíč `port` důležitý, lze jej z porovnání zcela vynechat.

3.5 Interpret konfiguračního jazyka

Tato sekce pojednává o návrhu interpretu konfiguračního jazyka, představeného v předchozí sekci. Detailní podklady pro návrh interpretu byly v zájmu plynutí textu umístěny do přílohy B, která je dle potřeby odkazována.

Lexikální analýza je první částí interpretu, která pracuje se vstupním souborem. Jejím cílem je převést znaky ze vstupu do podoby tokenů a byla navržena s použitím regulárních výrazů. Jejich výčet, spolu s přiřazenými názvy tokenů, lze nalézt v tabulce B.1.

Na lexikální analýzu navazuje analýza syntaktická. Probíhá v rámci jednoho průchodu vstupem a je rozdělena na dvě části. První část řeší zpracování syntaxe pravidel a využívá metody rekurzivního sestupu. LL gramatika navržená pro tuto část řešení je v sekci B.2. Druhou částí je zpracování podmínek, tedy výrazů v rámci jazyka, a je řešena metodou precedenční syntaktické analýzy. Pro tuto část byla navržena gramatika ze sekce B.3, která je doplněna precedenční tabulkou B.2.

Na základě precedenční syntaktické analýzy je generován AST (abstraktní syntaktický strom). V podobě AST probíhá kontrola typů vůči specifikaci datového modelu v rámci platformy DP^3 a mezi výrazy na stejné úrovni. Na základě AST je následně generována specifikovaná podmínka, což opět umožňuje řadu sémantických kontrol.

3.6 Použití taxonomie pro sloučení klasifikací s různou mírou detailu

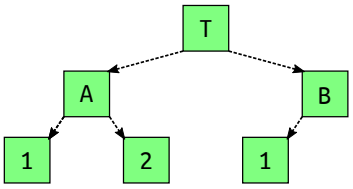
Dosavadní popis návrhu v rámci dekompozice nezmiňuje roli taxonomie v řešení. Vysvětlení použití taxonomie a způsob její integrace do navrženého řešení je předmětem této sekce. Jsou řešeny následující aspekty: použití taxonomie v rámci funkcí důvěry TBM a přechod mezi jednotlivými úrovněmi taxonomie, syntaxe pro specifikaci tříd taxonomie v rámci konfiguračního jazyka a způsob konfigurace a načítání taxonomie samotné.

Taxonomie tříd, také uchopitelná jako hierarchie tříd, je použita pro definici sdílené reprezentace namísto prosté množiny tříd. Použitím taxonomie je řešena situace, kdy jsou vstupní data na různé úrovni detailu. Z tohoto důvodu je výhodnější mít místo množiny tříd jejich víceúrovňovou hierarchii – taxonomii tříd.

V rámci konfiguračního jazyka je syntaxe pro třídy taxonomie specifikována jako spojení názvů tříd na cestě od kořene ke specifikované třídě znakem tečky. Pokud má pravidlo například přiřadit třídu `Windows`, bude specifikována jako `OperatingSystem.Windows`. V případě, když je pravidlo konkrétnější a přiřazuje třídu odpovídající `Windows 10`, specifikace bude `OperatingSystem.Windows.10`.

Jak již bylo naznačeno, kořenová třída taxonomie nemá z pohledu klasifikace žádný význam, nicméně slouží k identifikaci daného taxonomického stromu. Těchto stromů může být v rámci řešení specifikován libovolný počet, každý reprezentující jiný typ klasifikace. V rámci řešení této práce jsem navrhl dvě taxonomie, a to `OperatingSystem` pro typ operačního systému a `Device` pro typ zařízení. Kompletní navržená taxonomie je uvedena v příloze C.

Pro lepší ilustraci následujícího textu uvažujme tuto jednoduchou taxonomii, uvedenou na obrázku 3.4:



Obrázek 3.4: **Ilustrační příklad taxonomie tříd.**

Funkce důvěry jsou definovány nad rámcem domnění Θ. Pro využití taxonomie jsem tedy navrhl následující způsob interakce mezi funkcemi důvěry a konfigurovanými třídami taxonomie. V rámci funkcí důvěry, určených ke vzájemné kombinaci, se třídy taxonomie mohou vyskytovat pouze ve formě listových tříd taxonomického stromu. Tímto způsobem nedochází ke ztrátě detailu při kombinaci. V případě, kdy je konfigurací specifikována jiná než listová třída, je tato třída před kombinací interně převedena na množinu dceřinných tříd, mezi které je rovnoměrně rozdělena míra důvěry přiřazená původní třídě.

Pokud by tedy byla například specifikována třída `T.A`, bude převedena na na množinu `(T.A.1, T.A.2)`. Specifikovaná třída `T.A.1` by byla ponechána beze změny, třída `T.B` by byla převedena na `T.B.1`.

Výsledkem získatelným z výsledné funkce důvěry po provedení kombinace je nejpravděpodobnější třída na nejpodrobnější úrovni taxonomie. Poté lze pomocí slučování listových tříd do rodičovských tříd taxonomie získat výsledky i na obecnějších úrovních, kdy míra důvěry rodičovské třídě je určená součtem měr důvěry tříd dceřinných. Tento proces umožňuje získat výslednou třídu i v případě, kdy by na listové úrovni bylo větší množství tříd se stejnou pravděpodobností po pignistické transformaci. Slučování dále pokračuje až po sloučení do jediné kořenové třídy, kde končí.

Lze opět uvést příklad. Uvažujme následující výslednou funkci důvěry po provedení pignistické transformace:

$$\{\texttt{T.A.1} : 0.4, \quad \texttt{T.A.2} : 0.3, \quad \texttt{T.B.1} : 0.3\} \tag{3.3}$$

Nejpravděpodobnější třídou je zde `T.A.1`, která má pravděpodobnost 0,4. Získání výsledků na obecnějších úrovních lze provést sloučením dceřinných tříd:

$$\begin{aligned} &\{\texttt{T.A} : 0.7, \quad \texttt{T.B} : 0.3\} \\ &\{\texttt{T} : 1.0\} \end{aligned} \tag{3.4}$$

Takto získáme nejpravděpodobnější třídu pro první úroveň taxonomie, `T.A` s pravděpodobností 0,7. Sloučení je provedeno až ke kořeni taxonomie, kde proces končí.

Kapitola 4

Implementace

Pro implementaci byl použit jazyk Python verze 3, stejně jako v případě systému ADiCT, což umožňuje kompatibilitu vytvořeného řešení se systémem. Reprezentace funkcí důvěry je realizována s použitím existující knihovny `PyDS`[1] pro jazyk Python, jejímž autorem je Thomas Reineking.

Klasifikační pravidla, jak byla navržena v sekci 3.3, jsem implementoval pomocí tříd `Rule` a `RangeRule`. U obou tříd byla implementována speciální metoda `__call__()`, která umožňuje volat objekty těchto tříd stejně jako běžné funkce. Pro snazší použití je návratovou hodnotou z těchto volání již reprezentace v rámci funkce důvěry, implementované v knihovně PyDS jako třída `MassFunction`. Zároveň třídy využívají speciálního atributu `__slots__`, který výrazně snižuje množství alokované paměti a zrychluje přístup k jednotlivým atributům objektů.

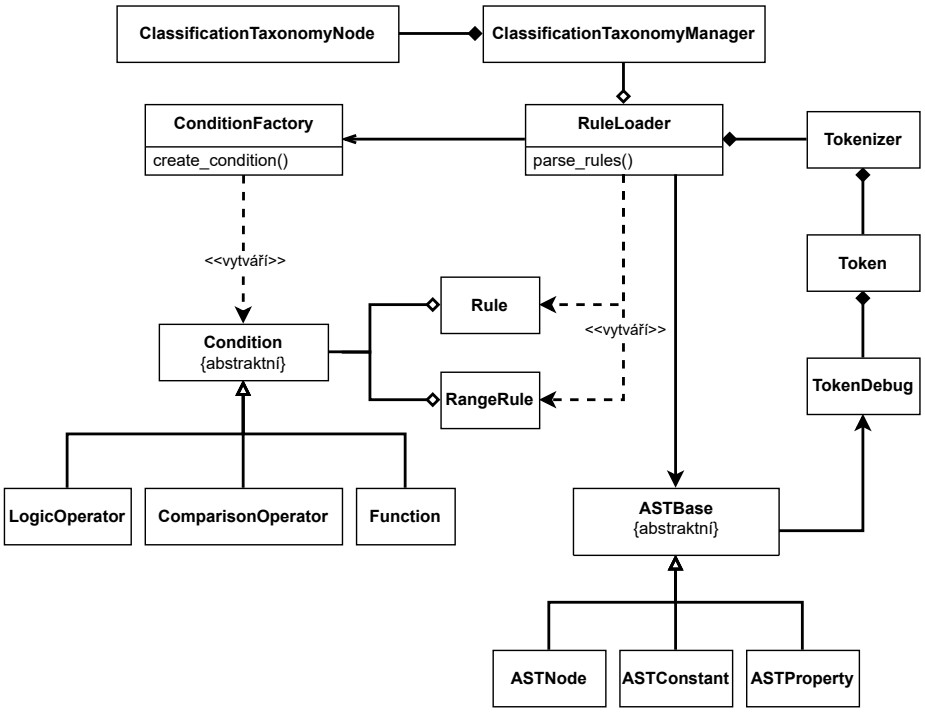
Načítání taxonomie jsem implementoval ve funkci `load_classification_taxonomy`. Pro ukládání taxonomie byl zvolen formát YAML, který je široce používán v systému ADiCT. Taxonomie má podobu zanořených slovníků, kde klíče jsou názvy tříd. Názvy listových třídy pak odkazují hodnotu `None`, což je ve formátu YAML reprezentováno prázdným záznamem. Celá taxonomie je uvedna v příloze C, zde je zkrácená ukázka z konfiguračního souboru:

```
OperatingSystem:
    Windows:
        '10':
        '8.1':
    Linux:
    ...
```

Způsob použití tříd z taxonomie v rámci funkcí důvěry, popsaný v sekci 3.6, vyžaduje stejnou hloubku taxonomie pro všechny listové třídy. Pro zaručení tohoto požadavku byl implementován algoritmus, který v případě potřeby vytvoří generické podtřídy pro každou větev taxonomie s nedostatečnou hloubkou. Správa taxonomie, zahrnující rozhraní pro získání rodičovské či kořenové třídy, dceřinných a listových tříd, ale i kontroly validity předkládané třídy, je implementována třídou `ClassificationTaxonomyManager`, která interně spravuje strom objektů třídy `ClassificationTaxonomyNode`. Z důvodu snadnější kontroly příslušnosti jsou třídy reprezentovány ve formátu výčtu, v jazyce Python implementovaným třídou `Enum`, který je dynamicky vytvořen po načtení správcovskou třídou.

[1] https://github.com/reineking/pyds

Interpret pro načítání konfiguračních pravidel jsem implementoval ve třídě `RuleLoader`, která navíc při překladu využívá řadu pomocných tříd. Diagram tříd řešení je na obrázku 4.1.



Obrázek 4.1: **Diagram tříd implementujících načítání konfiguračních pravidel.** Za účelem větší přehlednosti nejsou do diagramu zahrnuty všechny podtřídy ve stromech dědičnosti `Condition` a `ASTBase`.

První významnou třídou je třída `Tokenizer`, která je zodpovědná za lexikální analýzu zdrojového textu. Zpracování je provedeno pomocí regulárních výrazů, jak bylo popsáno v sekci 3.5. V rámci konstruktoru je vytvořen seznam objektů třídy `Token`, reprezentujících jednotlivé tokeny. Objekty `Token` se navíc odkazují na objekty `TokenDebug`, které obsahují informace o pozici tokenu v původním textu. Třída `Tokenizer` implementuje metodami `__iter__` a `__next__` protokol pro iteraci, čímž umožňuje přes seznam tokenů iterovat. Navíc je implementována metoda `seek`, umožňující získat tokeny na základě relativní pozice vůči aktuální pozici iterátoru.

Třída `RuleLoader` pak na základě tokenů provádí syntaktickou analýzu vstupu. Syntaktická analýza je implementovaná kombinací metody rekurzivního sestupu a precedenční syntaktické analýzy, které se dělí o zpracování jazyka způsobem popsaným v sekci 3.5. Rekurzivní sestup začíná metodou `parse_rules` a následně jsou rekurzivně volány metody na základě pravidel LL gramatiky, přičemž jedna metoda zpracovává jeden neterminál gramatiky. Precedenční syntaktická analýza je prováděna metodou `expr_precedence_syntax` a její základní operace jsou implementovány v metodách `precedence_begin_handle`, `precedence_shift` a `precedence_end_handle`. Během precedenční analýzy jsou na základě pravidel gramatiky výrazů vytvářeny uzly AST, které jsou reprezentovány objekty, jejichž třídy dědí z třídy `ASTBase`. Pomocí stromu AST jsou prováděny typové a jiné sémantické kontroly.

V případě úspěšného průchodu zpracování výrazu podmínky jsou uzly AST transformovány na uzly stromu podmínek. Tyto objekty dědí z třídy `Condition` a skutečně již po zavolání vykonávají sémantickou funkci podmínek. Konstrukce objektů tvořících podmínky je implementována na základě návrhového vzoru továrna (Factory method pattern) ve třídě `ConditionFactory`. Objekty jsou konstruovány na základě předaného řetězce, identifikujícího jejich třídu. Je zde využita introspekce umožněná jazykem Python tím způsobem, že slovník mapující identifikující řetězec na konkrétní třídu je vytvářen dynamicky při konstrukci objektu `ConditionFactory` na základě všech doposud definovaných podtříd abstraktní třídy `Condition`. Tento způsob implementace umožňuje volné rozšiřování souboru tříd podmínek bez nutnosti změn v procesu překladu jazyka i tvorby podmínek, čehož je využito pro snadnou rozšiřitelnost sady funkcí v konfiguračním jazyce.

Po dokončení zpracování podmínky je řízení předáno zpět metodám rekurzivního sestupu. Pokud se jedná o pravidlo s jedinou podmínkou, či poslední podmínku pravidla s rozsahem, je patřičný objekt vytvořen a přidán do seznamu všech načtených pravidel. Tento seznam je pak výstupem celého načítání pravidel.

Při implementaci načítání pravidel byla zohledněna i uživatelská přívětivost řešení. V případě chyby ve vstupu dochází k vyvolání výjimky, která díky informacím z třídy `TokenDebug` obsahuje přesnou lokaci chyby a podrobný popis, proč k chybě došlo. Příklad několika takových hlášení:

```
Syntax Error: Unexpected token <gt: '<'> while parsing expression,
top term on stack was <gt: '<'>
Error ocurred here:
>    0.2: 1 < .x < 2
>               ^
```

```
Semantic Error: No function called unknown.
Error ocurred here:
> 0.1: unknown() in .a
>      ^^^^^^^
```

```
Semantic Error: No way to convert 'name' key's type {<class 'int'>,
<class 'float'>} to <class 'str'>, defined by sdp_label.
Error ocurred here:
>    0.5: {"name": 2.0, "value":""} in .sdp_label
>          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Doposud popsané řešení klasifikačních pravidel předpokládá hodnotu confidence 1 u všech hodnot. Pro umožnění integrace do systému ADiCT byla proto rozšířena implementace pro podporu práce s hodnotami confidence[2]. Při volání pravidel je krom záznamu obsahujícího hodnotě entity v konkrétním časovém okamžiku umožněno předat i druhý parametr, který obsahuje záznam s danými hodnotami společně s hodnotami confidence. Po splnění podmínky je zjištěno, které konkrétní uzly přistupující k hodnotám atributů splnění způsobily, a je vypočítán koeficient confidence součinem hodnot confidence z těchto atributů. Koeficient je aplikován na míru důvěry přisouzené hlavní hypotéze, navrácené ve funkci důvěry.

[2]Data ze vstupních modulů mají hodnotu confidence v rozsahu $(0, 1\rangle$, navíc při extrapolaci hodnot v čase dochází ke snižování této hodnoty.

Pokud má tedy pravidlo základní nastavenou míru důvěry 0,8 a závisí na hodnotě atributu `os_by_tcpip`, která měla v okamžiku klasifikace hodnotu confidence 0,5, bude míra důvěry přiřazená hlavní hypotéze 0,4. Pokud pravidlo závisí na dvou hodnotách atributů, spojených logickou spojkou `and`, bude koeficient součinem confidence obou hodnot. Pokud se jedná o logickou spojku `or`, zahrnuje koeficient pouze využitou část výrazu dle zkráceného vyhodnocování.

Řešení bylo verifikováno proti sadě jednotkových testů, které byly napsány s pomocí testovací platformy `unittest` jazyka Python. Sada čítá celkem 85 testovacích případů, které dohromady pokrývají 94 % řádků kódu ze všech 8 souborů obsahujících implementaci.

4.1 Sada klasifikačních pravidel

Sada klasifikačních pravidel specifikuje konverzi z jednotlivých atributů do společné reprezentace. Pravidla jsou vnímána jako součást konfigurace a jsou ve většině případů psána ručně. Některá pravidla obsahují i „expertní znalosti“, umožňující odvozovat i nové informace ze surových dat. Příkladem může být detekce typu zařízení na základě určitých otevřených portů, odvozování operačního systému na základě nabízení či dotazování se po konkrétních službách pomocí SDP protokolů daným zařízením či zúžení možného typu zařízení na základě znalosti operačního systému, např. Android. V této sekci bude krátce popsán způsob tvorby pravidel pro jednotlivé atributy.

V případě jednoduchých atributů, které reprezentují výstup klasifikátoru či jiný jednoznačný štítek, je tvorba pravidla zjednodušena na získání všech možných hodnot daného atributu a nalezení odpovídající hodnoty v taxonomii. Jako příklady takovýchto pravidel lze uvést pravidla k atributům `os_by_tcpip`, `operating_system_ua` či atributy modulu TLS Fingerprinting:

```
OperatingSystem.Linux.Debian
    0.7: re('(?i)Debian') in .operating_system_ua
    0.4: re('(?i)Debian') in .os_by_tcpip
    0.65: contains('Debian') in .tls_os_name
    0.65: contains('Debian') in .tls_os_version
```

Data modulu Service Labels byla krom přispění ke klasifikaci operačního systému využita více pro klasifikaci typu zařízení. Lze si povšimnout toho, že i když nelze na základě hodnoty atributu přiřadit konkrétní klasifikaci, lze definovat co nejpřesněji odpovídající množinu štítků:

```
Device.Workstation, Device.SmartPhone
    0.4: contains_any_of('DHCP Client', 'End Device') in .tags_by_services

Device.Appliance, Device.Hub, Device.VoiceAssistant
    0.5: contains('IoT') in .tags_by_services
```

U dat z modulu SDP Analyzer bylo při tvorbě pravidel vycházeno z mých předchozích experimentů s metodami dolování dat. Příklad pravidel:

```
OperatingSystem.MacOS
    0.7: {'name': 'os', 'value': contains('Mac OS')} in .sdp_label
    0.4 - 0.8: [
```

```
        '_airport._tcp.local' in .dnssd_query
        contains('_apple-mobdev2._tcp.local') in .dnssd_query
        '_apple-pairable._tcp.local' in .dnssd_query
        ... zkráceno ...
        '_ptp._tcp.local' in .dnssd_query
        {'service':'_sftp-ssh._tcp.local'} in .dnssd_service
        {'service':'_ssh._tcp.local'} in .dnssd_service
    ]
```

Míry důvěry, nastavené u jednotlivých pravidel, vychází z „expertních znalostí“. Tímto je míněno, že na základě pozorování výstupů, znalosti vnitřního principu modulů a konzultaci s pracovníky projektu ADiCT byla stanovena míra důvěry k danému modulu, která byla následně aplikována na všechna pravidla k němu vztažená. Jedinou výjimkou je modul SDP Analyzer, kde, jak již bylo zmíněno, se hodnoty měr důvěry odvíjí z předchozích experimentů s dolováním dat.

4.2 Integrace řešení do systému ADiCT

Integrace do systému ADiCT byla provedena v podobě sekundárního modulu. Rozhraní sekundárních modulů bylo popsáno v závěru sekce 2.1.1.

Výstup implementovaného řešení je ukládán do dvou atributů pro každou existující taxonomii, tedy každý typ štítku. První atribut obsahuje nejpravděpodobnější třídu na každé úrovni taxonomie včetně daných pravděpodobností. Druhý ukládá seznam funkcí důvěry, který byl výsledkem po klasifikaci jednotlivých pravidel. Oba atributy jsou ukládány v datovém formátu JSON, názvy atributů jsou nastaveny v konfiguraci modulu. Zde je příklad hodnot pro taxonomii `OperatingSystem`:

```
atribut os_taxonomic:
[
  ["OperatingSystem", 1.0],
  ["OperatingSystem.Windows", 0.917],
  ["OperatingSystem.Windows.10", 0.566]
]

atribut os_evaluations:
[
  [
    {"key": ["OperatingSystem.Windows"], "belief": 0.8},
    {"key": ["OperatingSystem"], "belief": 0.2}
  ],
  [
    {"key": ["OperatingSystem.Windows.10"], "belief": 0.448},
    {"key": ["OperatingSystem"], "belief": 0.552}
  ]
]
```

V rámci konstruktoru třídy `LabelFusionModule`, která implementuje funkcionalitu řešení, je provedena inicializace objektů `ClassificationTaxonomyManager` a `RuleLoader`, pomocí kterých jsou následně načtena klasifikační pravidla ze souborů specifikovaných

v konfiguraci modulu. Pro provádění aktualizace je registrována funkce `classify_record`. Seznam atributů měněných klasifikační funkcí je stanoven na všechny konfigurované výstupní atributy. Seznam sledovaných změn zahrnuje události o aktualizaci hodnoty confidence i vypršení platnosti atributu. Zbytek seznamu je získán sjednocením množin atributů, na kterých závisí jednotlivá pravidla.

Zde jsem narazil na problém ze strany platformy DP[3], která u sekundárních modulů neumožňuje získávání dat z vícero entit zároveň. I když jsou tedy u daného zařízení k dispozici data vázaná k IP i MAC adrese, které jsou v systému ADiCT reprezentovány jako separátní entity, neexistuje v rámci poskytnutého rozhraní možnost získat data obou entit. Z toho důvodu jsem byl v rámci integrace nucen omezit sadu pravidel tak, aby závisela pouze na atributech vázaných k IP adrese. Rozšíření rozhraní sekundárních modulů pro umožnění takovýchto operací je do budoucna plánováno a z pohledu modulu by rozšíření mělo představovat minimální změny.

Klasifikační funkce `classify_record` provede aplikaci všech klasifikačních pravidel na konkrétní záznam, načež výsledné funkce důvěry rozdělí do seznamů podle taxonomie, do které patří. Funkce důvěry jsou v oddělených seznamech sjednoceny pomocí DRC, čímž je získána celková funkce důvěry všech pravidel. Výsledek klasifikace pro jednotlivé úrovně taxonomie je získán pomocí převodu popsaného v sekci 3.6, přičemž na každé úrovni je jako výsledná brána třída s pravděpodobností nad 50 % po provedení pignistické transformace. V závěru funkce jsou vypočítané hodnoty uloženy do připravených atributů.

Algoritmus výběru výsledné třídy pro jednotlivé úrovně taxonomie, implementovaný ve funkci `get_taxonomy_from_evaluation`, provádí dva průchody stromem taxonomie, přičemž při prvním průchodu jde od listové úrovně po kořenovou, v druhém se vrací z kořene směrem k listům. V druhém průchodu je množina uvažovaných tříd redukována na podtřídy výsledné třídy z předchozí iterace. Tímto způsobem je vyloučena situace, kdy by došlo k výběru výsledných tříd nenavazujících v taxonomickém stromu.

Kapitola 5

Testování a experimenty na datech z reálné sítě

Tato kapitola obsahuje popis testování řešení na datech z reálné sítě. Po provedení úvodního testování jednotlivých datových zdrojů a implementované metody kombinace jsou provedeny další experimenty za účelem optimalizace měřených ukazatelů úspěšnosti řešení. Z důvodu nedostatku dat pro typ zařízení bylo testováno pouze slučování dat typu operačního systému.

Nejprve je v sekci 5.1 uveden způsob tvorby a anotace datové sady, použitého způsobu vyvažování tříd a výsledné složení datové sady. V sekci 5.2 je provedeno testování přesnosti a dalších ukazatelů úspěšnosti jednotlivých modulů, přičemž je tímto způsobem stanovena minimální hranice pro úspěšnost jakékoliv kombinační metody. Sekce 5.3 obsahuje popis testování implementované kombinační metody s ručně stanovenými mírami důvěry spolu s dalšími metodami pro porovnání. V následující sekci 5.4 je popsán experiment, kdy je provedena optimalizace klasifikačních pravidel na základě doposud známých statistických dat. Na to navazuje sekce 5.5, ve které je sledován vliv omezení datové sady vyžadováním určitého počtu datových zdrojů u konkrétního sezení. Poslední sekcí s experimentem je sekce 5.6, kde je zkoumána další optimalizace metody dvou fokálních hypotéz na základě změny distribuce přiřazovaných měr důvěry. Provedené experimenty jsou následně shrnuty v sekci 5.7.

V rámci této kapitoly jsou pro vyhodnocení úspěšnosti testovaných metod či datových zdrojů použity ukazatele úspěšnosti založené na matici záměn (confusion matrix). Matice záměn ukazuje výsledky klasifikátoru tím způsobem, že řádky obsahují hodnoty skutečné a sloupce předpovězené. Lze dále definovat čtyři kategorie, do kterých pole matice pro konkrétní třídu spadá.

- **True Positive** (TP) – třída byla vybrána klasifikátorem a odpovídá skutečnosti.
- **True Negative** (TN) – třída nebyla vybrána a skutečně se jedná o jinou třídu.
- **False Positive** (FP) – třída byla vybrána, ale jedná se o jinou třídu.
- **False Negative** (FN) – třída nebyla vybrána, i když se jedná o danou třídu.

Pomocí součtů polí v těchto kategoriích lze dále vypočítat následující ukazetele přesnosti klasifikace: **Preciznost** udává, jaký podíl ze všech klasifikací dané třídy odpovídal skutečnosti. **Senzitivita** udává, jaký podíl z případů, kdy šlo o danou třídu, byla třída správně

klasifikována. S pomocí těchto dvou ukazatelů lze vyjádřit ukazatel přesnosti klasifikace označovaný **F1 skóre**, které je jejich harmonickým průměrem.

$$Preciznost = \frac{TP}{TP + FP} \quad Senzitivita = \frac{TP}{TP + FN} \quad F1score = \frac{2 \cdot Preciznost \cdot Senzitivita}{Preciznost + Senzitivita}$$

Poslední použitým ukazatelem je **Přesnost**, která udává celkovou přesnost klasifikátoru pro danou třídu podílem správně klasifikovaných vůči všem záznamům.

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

Jak již bylo uvedeno, ukazatele jsou vypočítávány pro jednotlivé třídy odděleně. Pro získání celkového ukazatele klasifikátoru je třeba provést průměr ukazatelů všech klasifikovaných tříd. Vzhledem k nevyváženosti datové sady, jak bude ukázáno v následující sekci 5.1, je v rámci experimentů použit vážený průměr, kdy počet vzorků dané třídy určuje váhu ukazatele ve výsledném průměru.

5.1 Datová sada

Pro vyhodnocení navržené metody je třeba získat datovou sadu obsahující hodnoty různých atributů k množství IP adres a k nim anotaci, tedy informaci o operačním systému, který na zařízení skutečně běží. Datová sada pro evaluaci vychází z dat sbíraných v rámci projektu ADiCT. Datové body zasílané na centrální server jsou krom zpracování a uložení do databáze uloženy i do souborů obsahující logy těchto datových bodů. Tyto logovací soubory slouží jako základ evaluační datové sady. Byly vybrány soubory s logy z 64 dní z časového rozmezí od 1. února 2022 do 5. dubna 2022, které dohromady čítají 67 121 583 datových bodů, 66 875 881 k IP adresám, 245 702 k MAC adresám.

Pro anotaci datové sady bylo použito dvou referenčních zdrojů. Prvním zdrojem jsou data z modulu HTTP User-Agent. Použití HTTP User-Agent dat pro anotaci datové sady lze vidět například v [16]. Druhým zdrojem referenčních dat je vyhledávač internetových zařízení Shodan[1]. Shodan skenuje veřejný prostor internetu a získaná data o otevřených portech a dalších aspektech včetně operačního systému ukládá do databáze. Přístup k datům je umožněn prostřednictvím webového rozhraní a rozhraní API. U vyhledávače Shodan nelze očekávat pokrytí dynamických sítí, jak je tomu v případě dat z pole HTTP User-Agent, nicméně poskytuje informace o běžících serverech, které naopak HTTP provoz s hlavičkou User-Agent neprodukují. Kombinace těchto dvou zdrojů umožňuje anotovat širší spektrum zařízení.

Z logů jsou tyto datové body načteny a transformovány do podoby „profilů“ jednotlivých zařízení, podle adresy, ke které je každý datový bod určen. Tímto způsobem je vytvořeno 7967 profilů, každý se svou vlastní IP adresou. Bohužel ne všechny vytvořené profily lze využít pro evaluaci. První z podmínek, která je vynucená potřebou přítomnosti referenčních dat, je přítomnost dat HTTP User-Agent v profilu nebo úspěšná identifikace IP adresy vyhledávačem Shodan. Na základě této podmínky se počet použitelných profilů snižuje na 1303. Další podmínkou je odstranění takových profilů, kde dochází k překladu adres (NAT). Překlad adres porušuje adresování na síťové vrstvě, čímž je monitorování sítí výrazně stíženo. Při vnějším pohledu generuje IP adresa, za kterou se skrývá překlad adres, provoz všech zařízení na dané podsíti. Odvozování informací o jednotlivých zařízeních

[1] https://www.shodan.io/

v dané podsíti pomocí pasivního monitorování je proto problematické. Profily, které vykazovaly známky přítomnosti překladu adres byly proto vyřazeny z evaluační datové sady.

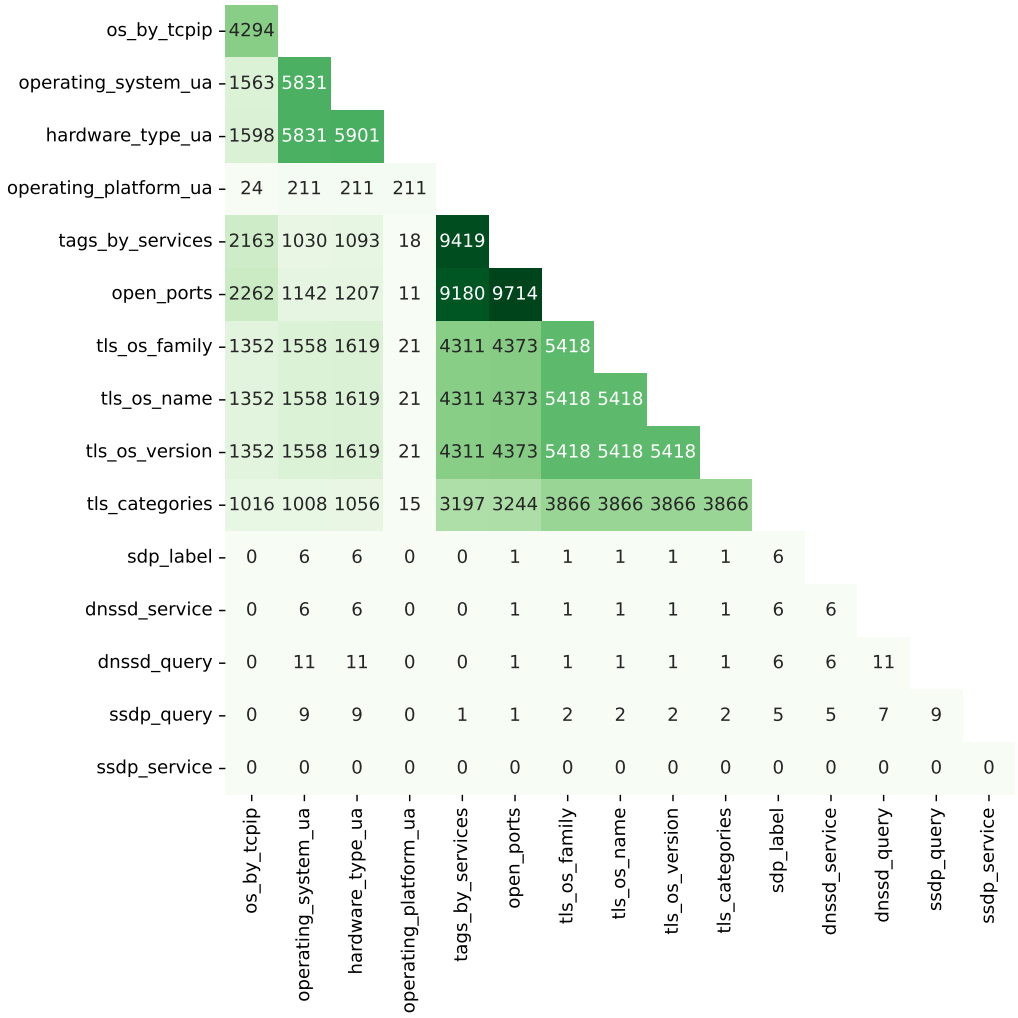
Metoda detekce překladu adres použitá pro přípravu evaluační datové sady je následující. V rámci dat z modulu HTTP User-Agent byla jako indikátor překladu adres použita přítomnost několika datových bodů s identickým časovým rozsahem, ale rozdílnou hodnotou. Profily obsahující tento indikátor byly odstraněny. Další skupinou nejednoznačných profilů jsou ty, kde dochází k postupnému vyměňování detekovaného operačního systému na základě HTTP User-Agentu. Tyto profily nemůžeme zcela vyřadit, protože očekáváme přítomnost sítí s dynamicky přidělovanými veřejnými adresami (např. síť Eduroam). Uvedený způsob není dokonalý a nedetekuje všechny profily s překladem adres. Pokrývá nicméně ty případy, kdy se chování adresy často mění, respektive se tváří jako více zařízení zároveň, což je z pohledu práce klíčové.

Protože použitá datová sada pochází z dlouhého časového období, přiřazení jediné anotace pro jednu adresu nemusí být vhodné, jelikož se na dané adrese mohlo během zkoumaného období vystřídat více zařízení. Tento problém je řešen detekcí „sezení“, což je metoda inspirovaná [23]. Ve zmíněném článku je anotace dynamických sítí prováděna na základě kombinace obohacení dat z DHCP a logů Radius serveru. Pomocí kombinace těchto dat jsou vytvořena sezení obsahující informace o tom, kdy se daný uživatel připojil a odpojil od sítě, a jaká IP adresa mu byla přiřazena.

Použitá datová sada neobsahuje informace z autentizačních serverů, navzdory tomu je koncept anotace jednotlivých sezení uživatelů použitelný. Profily k jednotlivým adresám jsou dále děleny na časové intervaly, sezení, kdy je souvisle po celou dobu intervalu detekován operační systém z protokolu HTTP. Minimální počet navazujících datových bodů z HTTP User-Agent modulu byla experimentálně stanovena na 2. Hodnoty všech atributů jsou do sezení uložena na základě časového intervalu z protokolu HTTP. Při anotaci na základě vyhledávače Shodan je předpokládána statická síť, do sezení jsou proto zahrnuty všechny informace k danému zařízení. Zpracování hodnot sezení je identické pro obě metody anotace. Pokud v rámci omezených intervalů sezení dochází k oscilaci hodnot vybraných atributů, není sezení využito.

Výsledkem tohoto zpracování je získání 16 571 sezení pro 1208 unikátních IP adres a 7 unikátních MAC adres, přičemž je detekováno 7 unikátních párů IP adresa, MAC adresa[2]. Datová sada obsahuje hodnoty atributů všech popsaných modulů v podsekci 2.1.2, kde jsou popsány i jednotlivé atributy včetně ukázky dat. Zastoupení jednotlivých atributů v datové sadě je ukázáno na obrázku 5.1, který navíc ukazuje počty současně přítomných hodnot u dvojic atributů. Na diagonále jsou zobrazeny celkové počty daného atributu, pod diagonálou se nachází počty současně přítomných hodnot pro dané dva atributy v jednom sezení.

[2]Informace o MAC adresách jsou k dispozici jen u malé části dat, pocházející z monitorované lokální Wi-Fi sítě. Většina dat pochází z rozsáhlé sítě, kde MAC adresy koncových zařízení nejsou k dispozici.



Obrázek 5.1: **Matice znázorňující zastoupení jednotlivých atributů v datové sadě.**

Analýza datové sady po dokončení anotace ukázala, že převažují třídy Linux a Windows, přičemž třída Linux má 4,3násobek výskytů oproti třídě Windows. Rozhodl jsem se proto pro vyvážení datové sady. Počet sezení anotovaných třídou Linux byl proto uměle snížen na stejný počet, jako třída Windows. Kompletní počty před a po provedení vyvážení jsou uvedeny v tabulce 5.1.

Třída	Počet před vyvážením	Počet po vyvážení
Linux	7876	1834
Windows	1834	1834
MacOS	56	56
iOS	28	28
Android	12	12
Celkem	9811	3764

Tabulka 5.1: **Zastoupení jednotlivých tříd v datové sadě**

Další podstatnou statistikou pro další experimenty jsou počty sezení, které mají větší počet datových zdrojů současně. Toto je naznačeno v tabulce 5.2.

Modul	Samotný	+ 1 další	+ 2 další	Celkem
OS by TCP/IP	1058	660	165	1883
TLS Fingerprinting	988	855	165	2008
SDP Analyzer	4	0	0	4
Service Labels	496	591	165	1252
Celkem	2546	1053	165	3764

Tabulka 5.2: **Zastoupení modulů v datové sadě.**

5.2 Získání přesnosti individuálních zdrojů

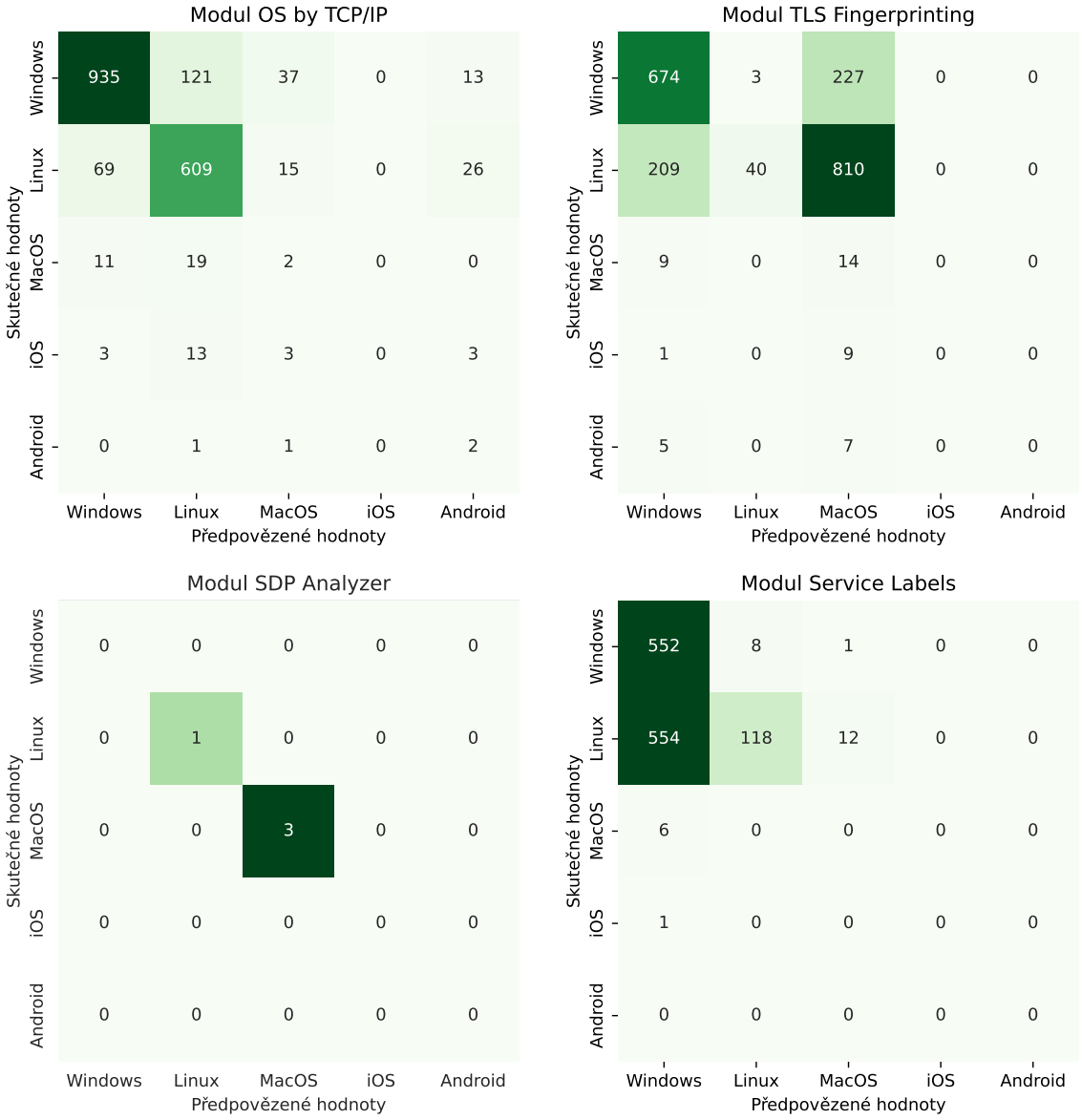
Prvním provedeným experimentem je jakási kalibrace, či přesněji, získání přesnosti a dalších ukazatelů individuálních zdrojů v rámci anotované datové sady. Důvodem pro tento experiment je získání lepší představy o vstupních datech z jednotlivých modulů. Zároveň je po zohlednění pokrytí datové sady možno uvažovat ukazatele jednotlivých modulů jako akceptovatelné minimum pro jakoukoliv metodu slučování.

Experiment byl proveden následovně. Sada pravidel, popsaná v sekci 4.1, byla rozdělena do několika podmnožin tak, aby v každé vzniklé sadě všechna pravidla používala pouze atributy vybraného modulu. S každou takto vzniklou sadou byla provedena klasifikace datové sady a výsledky byly porovnány vůči referenčním hodnotám, přičemž byly ignorovány prázdné výstupy, jelikož v daném sezení nejsou k dispozici data modulu.

Výsledky jsou popsány v následující formě: pro jednotlivé moduly byly vytvořeny matice záměn, ukazující výsledky jejich klasifikace, zobrazené na obrázku 5.2. Ukazatele jednotlivých modulů byly zároveň zaznačeny do tabulky 5.3. Matice záměn, stejně jako měřené ukazatele, obsahují pouze takové výsledky, kdy daný modul poskytoval data. Sezení neobsahující data modulu nejsou započítána.

Ze všech uvedených matic záměn je patrná dominance systémů Windows a Linux v datové sadě. Na základě výsledků lze jako nejpřesnější vstup označit modul OS by TCP/IP, který ukazuje velmi obstojné výsledky pro oba dominantní systémy. Modul SDP Analyzer dosahuje výsledku 100 % u jednotlivých ukazatelů, což je vidět i obsazením diagonály matice záměn modulu, bohužel je tato skutečnost pouze důsledkem velmi nízkého pokrytí datové sady. Modul TLS Fingerprinting provádí záměnu operačního systému Mac OS za systém Linux i Windows, což vede k nízkým skóre senzitivity, přesnosti a skóre F1. Na základě dat modulu Service Labels jsou klasifikovány především operační systémy Windows a Linux, přičemž dosažená přesnost se pohybuje okolo 54 %. Z matice záměn lze vyčíst, že pravidlo pro klasifikaci operačního systému Linux je spolehlivé, zatímco pravidlo indikující Windows klasifikuje i operační systém Linux v poměru 1:1.

Z představených modulů ukazuje nejlepší výsledky vzhledem k pokrytí datové sady modul OS by TCP/IP. Tento modul lze proto použít jako minimální akceptovatelnou hranici pro jakoukoliv metodu kombinace výsledků. Při klasifikaci celé datové sady se započítáním chybějících hodnot produkují data modulu OS by TCP/IP ukazatele uvedené v tabulce 5.4.



Obrázek 5.2: **Matice záměn jednotlivých modulů.**

ukazatel	OS by TCP/IP	TLS Fingerprinting	SDP Analyzer	Service Labels
Preciznost	0,845	0,829	1,000	0,734
Senzitivita	0,822	0,363	1,000	0,535
Skóre F1	0,832	0,375	1,000	0,455
Přesnost	0,822	0,363	1,000	0,535

Tabulka 5.3: **Ukazatele jednotlivých modulů.**

	Preciznost	Senzitivita	F1 skóre	Přesnost
OS by TCP/IP	0,824	0,405	0,540	0,405

Tabulka 5.4: **Ukazatele modulu OS by TCP/IP nad celou datovou sadou.** Tyto ukazatele lze považovat za minimální akceptovatelnou hranici pro kombinační metody.

5.3 Slučování dat na základě expertního ohodnocení

Druhým experimentem je provedení slučování výsledků individuálních modulů z prvního experimentu několika metodami a jejich vzájemné porovnání.

Hlavní zkoumanou metodou slučování datových zdrojů je pomocí D-S teorie. V tomto experimentu bude využito měr důvěry stanovených na základě expertních znalostí, tedy uvedených v konfiguraci pravidel. Všechny dostupné výsledky v podobě funkcí důvěry jsou sloučeny pomocí DRC a výsledkem se stává třída s nejvyšší pravděpodobností po provedení pignistické transformace.

Další porovnávanou metodou slučování je triviální metoda pomocí prioritního výběru zdroje dat. Moduly jsou na základě předchozího experimentu seřazeny dle přesnosti. Metoda pomocí tohoto řazení vybírá moduly, přičemž první na řadě mající výstup je určen jako výstup metody.

Jako teoretické maximum je stanoven způsob slučování, jež bude označován jako orákulum, po vzoru [10]. Orákulum má k dispozici anotaci datové sady. Pokud alespoň jeden z modulů klasifikuje dané sezení správně, orákulum předloží tento správný výstup. V opačném případě je na výstupu náhodný z dostupných výsledků.

Míry důvěry u pravidel pro slučování pomocí D-S teorie, nastavené expertem[3], jsou ve skutečnosti velmi odlišné od přesností jednotlivých modulů zjištěných v předchozí kapitole. Pro přesnější srovnání byly nastavené míry důvěry metody D-S teorie aktualizovány na dosažené přesnosti modulů v prvním experimentu, uvedené v tabulce 5.3. Výsledek je označen jakožto informovaná D-S teorie.

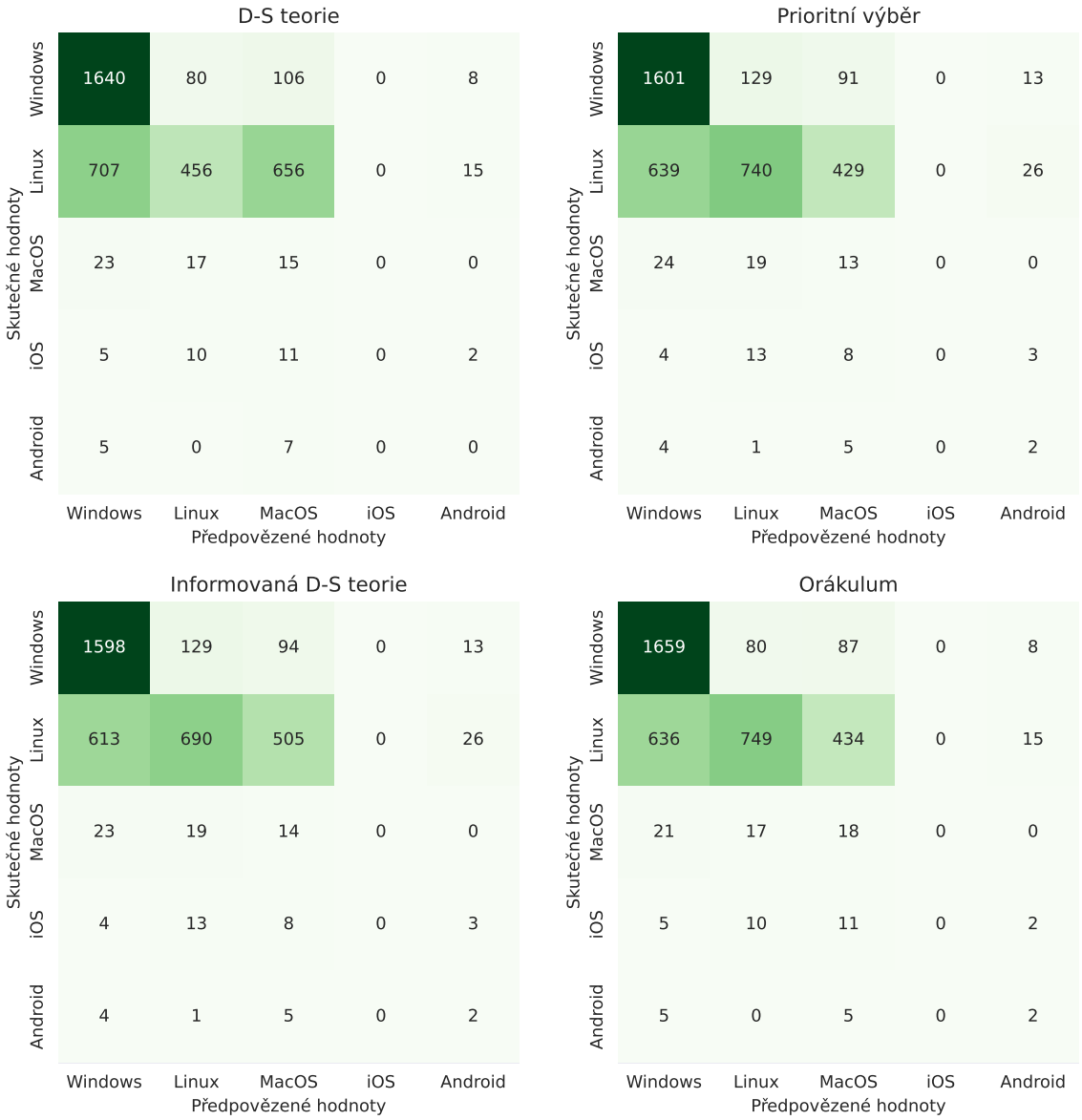
ukazatel	D-S teorie	Prioritní výběr	Informovaná D-S teorie	Orákulum
Preciznost	0,731	0,744	0,742	0,775
Senzitivita	0,561	0,626	0,612	0,645
Skóre F1	0,565	0,644	0,633	0,661
Přesnost	0,561	0,626	0,612	0,645

Tabulka 5.5: **Ukazatele kombinačních metod.**

Výsledky byly zaznačeny v podobě matice záměn na obrázku 5.3 a tabulky s dosaženými ukazateli 5.5. Většina klasifikovaných sezení se pohybuje v prvních dvou řádcích zobrazených matic záměn. Největší odlišnosti lze pozorovat mezi maticí D-S teorie a ostatními, jelikož při spojování upřednostní data z modulu TLS Fingerprinting před modulem OS by TCP/IP a častěji tak zaměňuje Linux za Mac OS. Matice orákula, prioritního výběru i informované D-S teorie pak obsahují menší rozdíly, ale lze stále pozorovat větší vliv modulu TLS Fingerprinting na matici informované D-S teorie oproti ostatním maticím. Ani jedna z metod neukazuje slibnou klasifikaci operačních systémů Mac OS, iOS a Android, což lze vysvětlit nízkou četností i nízkou přesností vstupních dat.

[3]Původně byly nastaveny hodnoty míry důvěry 0,4 pro modul OS by TCP/IP, 0,65 pro TLS Fingerprinting, rozmezí 0,4–0,7 pro SDP Analyzer a 0,6 pro Service Labels.

Tyto rozdíly lze stejně pozorovat i v tabulce s ukazateli 5.5, kde vidíme, že nejblíže ukazatelům orákula je metoda prioritního výběru, za ní následuje informovaná D-S teorie a nejhorší ukazatele ukazuje D-S teorie na základě expertního ohodnocení.



Obrázek 5.3: **Matice záměn kombinačních metod.**

Na základě výsledků byly provedeny následující závěry: Metoda prioritního výběru zdroje dat, ačkoliv se zdá triviální, produkuje lepší výsledky než metoda D-S teorie. Toto může být způsobeno několika faktory, například tím, že pokrytí dat více zdroji dat zároveň není časté, a proto se ve výsledcích neprojeví ignorance výsledků zbylých modulů touto metodou. Zároveň lze říct, že výsledky kombinace pomocí D-S teorie jsou odrazem nastavených hodnot měr důvěry, které byly nastaveny zřejmě špatně a neodráží nutně přesnosti modulů z prvního experimentu. Toto je napraveno metodou informované D-S teorie. Informovaná metoda D-S teorie má již podobnou pozici jako metoda prioritního výběru co se týká dat o přesnosti modulů, navzdory tomu za ní zaostává přibližně o jedno procento ve

všech ukazatelích. Dle výsledků spojování D-S teorií s využitím statistických dat překonává základní nastavení D-S teorie, zejména v senzitivitě, přesnosti a skóre F1.

5.4 Optimalizace klasifikačních pravidel na základě dat

Třetí provedený experiment se již blíží více do prostoru dolování dat či strojového učení. V rámci experimentu je provedena optimalizace pravidel na datovou sadu následujícím způsobem: Pravidla jsou aplikována na datovou sadu a v případech, kdy je splněna podmínka, je k danému pravidlu zaznačena skutečná třída dle referenčních dat. Vzniklý vektor ukazující distribuci tříd pro dané pravidlo je normalizován, převeden na funkci domnění a nastaven namísto původně přidělované funkce domnění pravidla. Proces lze také popsat tak, že pro každé pravidlo je vytvořena matice záměn s normalizovanými sloupci, přičemž jediný vyplněný sloupec je použit pro tvorbu funkce domnění pravidla.

Převod sloupce záměn na funkci domnění je možno provést oběma způsoby popsanými v sekci 3.2. V rámci experimentu jsou tedy provedeny obě možnosti a výsledky jsou porovnány.

Původně nečekaný důsledek tohoto experimentu je možnost změny původní hlavní hypotézy pravidla, což znamená, že výsledky těchto experimentů jsou neporovnatelné s předchozím experimentem. Proto je pro obě metody převodu na funkce domnění krom kombinace pomocí DRC navíc opět provedeno sloučení orákulem a na základě prioritního výběru. Priorita je v tomto experimentu stanovena pro jednotlivá pravidla namísto na úrovni modulů.

K metodě je přistupováno jako k metodě strojového učení, která vyžaduje oddělenou trénovací a testovací datovou sadu pro ověření, že bude fungovat i na doposud nespatřených datech. Pro tento účel je použita pětinásobná křížová validace s náhodným rozložením vzorků. Tato varianta K-násobné křížové validace náhodně rozděluje datovou sadu do pěti množin a následně provede pět iterací, kdy 4 množiny jsou předloženy společně jako tréninková data a 1 jakožto testovací. Model je hodnocen v každé provedené iteraci a výsledkem je průměr hodnocených ukazatelů.

	D-S teorie			
ukazatel	Distribuce záměn	Dvě fokální hypotézy	Prioritní výběr+	Orákulum
Preciznost	0,767	0,761	0,789	0,851
Senzitivita	0,748	0,734	0,795	0,860
Skóre F1	0,733	0,717	0,785	0,850
Přesnost	0,748	0,734	0,795	0,860

Tabulka 5.6: **Ukazatele kombinačních metod s použitím trénovaných pravidel.**

Výsledky experimentu jsou uvedeny v tabulce 5.6. Lze vidět celkové zlepšení ukazatelů u všech metod, nejvýrazněji u orákula, jehož přesnost se zvýšila o 22 % a skóre F1 o 19 %. U metod D-S teorie došlo k podobným přírůstkům, u přesnosti až 14 % a u skóre F1 až 10 %. Prioritní výběr vykazuje u přesnosti zlepšení 17 % a u skóre F1 15 %. Metoda prioritního výběru byla označena „Prioritní výběr+“, protože pracuje s výrazně přesnějšími daty o přesnostech jednotlivých pravidel, nikoliv modulů jako celku. Metoda překonává oba způsoby aplikace D-S teorie. Výsledek porovnání tvorby funkcí důvěry je, že metoda distribuce záměn vykazuje lepší senzitivitu, přesnost a skóre F1 než metoda dvou fokálních hypotéz, a to zhruba o 2 %. Obě metody překonávají informovanou D-S teorii a prioritní výběr z předchozího experimentu.

5.5 Omezení datové sady dle množství současných datových zdrojů

Účelem čtvrtého provedeného experimentu je získání lepší představy o vlivu počtu zdrojů dat k danému sezení na jednotlivé metody kombinace. Na základě předešlých experimentů jsem formuloval hypotézu, že metoda prioritního výběru dosahuje lepších výsledků než kombinace pomocí D-S teorie z toho důvodu, že 68 % sezení v datové sadě má pouze jeden zdroj dat. Díky tomuto složení dat se neprojeví ignorance méně prioritních zdrojů, která by měla být hlavní nevýhodou metody.

Pro ověření této hypotézy jsem se rozhodl omezit datovou sadu tak, že budou použita pouze taková sezení, kdy jsou data k dispozici minimálně z N různých modulů, pro $N \in 2, 3$. Nad takto omezenými datovými sadami byl opakován předchozí experiment.

Omezení pro $N = 2$ produkovalo datovou sadu o 1218 sezeních a 322 unikátních IP adresách, přičemž naměřené ukazatele jsou uvedeny v tabulce 5.7. Omezení pro $N = 3$ dále redukovalo datovou sadu na 165 sezení 51 unikátních IP adres, naměřené ukazatele jsou v tabulce 5.8.

	D-S teorie			
ukazatel	Distribuce záměn	Dvě fokální hypotézy	Prioritní výběr+	Orákulum
Preciznost	0,807	0,812	0,833	0,956
Senzitivita	0,799	0,794	0,838	0,968
Skóre F1	0,792	0,786	0,832	0,961
Přesnost	0,799	0,794	0,838	0,968

Tabulka 5.7: **Ukazatele kombinačních metod při omezení na minimálně dva současné zdroje dat.**

	D-S teorie			
ukazatel	Distribuce záměn	Dvě fokální hypotézy	Prioritní výběr+	Orákulum
Preciznost	0,844	0,864	0,965	0,994
Senzitivita	0,818	0,836	0,958	0,994
Skóre F1	0,808	0,809	0,959	0,994
Přesnost	0,818	0,836	0,958	0,994

Tabulka 5.8: **Ukazatele kombinačních metod při omezení na minimálně tři současné zdroje dat.**

Výsledky tohoto experimentu nepotvrdily původní hypotézu. Jak lze vidět, metoda prioritního výběru pro $N = 2$ v metrice F1 skóre posílila o 4 % na 83 %, pro $N = 3$ pak o dalších 13 % na 96 %. Mezitím obě metody D-S teorie se pro $N = 2$ blíží k hranici F1 skóre 80 %, kterou přesahují až pro $N = 3$. Vzhledem k tomu, že skóre F1 orákula se zvýšilo o přírůstky 10 a 5 % až na 99 %, lze usoudit, že čím větší je počet zdrojů dat, tím pravděpodobnější je správnost alespoň jednoho z nich. Zároveň se ukazuje, že větší množství dat znamená i zvýšený podíl šumu, který je u D-S teorie vždy zahrnut, zatímco v případě prioritního výběru dochází častěji k jeho odfiltrování.

Bylo experimentováno i s redukcí celé datové sady bez provedení vyvážení tříd, kde bylo pro $N = 2$ získáno 3063 sezení 435 IP adres a pro $N = 3$ 283 sezení 66 IP adres. Použitím těchto datových sad byly zvýšeny ukazatele prioritního výběru i distribuce záměn v rozmezí

3–5 %, metoda dvou fokálních hypotéz oslabila při $N = 3$ na F1 skóre 63 %. Dochází tedy k mírnému posunu výsledků, nicméně závěr z experimentu zůstává stejný.

5.6 Optimalizace rozložení měr důvěry

Tento pátý experiment si klade za cíl prozkoumat další možnou optimalizaci metody D-S teorie. V dosavadních experimentech zahrnujících optimalizace pravidel byly míry důvěry přiřazovány na základě distribuce záměn bez dalších úprav. V tomto experimentu bude provedena optimalizace pomocí dvou transformačních funkcí a bude otestováno, zda jejich použití při přiřazování míry důvěry může přinést lepší výsledky pro metodu D-S teorie.

Z důvodu snadnější úpravy distribuce je v experimentu optimalizována metoda dvou fokálních hypotéz. Míra důvěry, přiřazená hlavní hypotéze této metody, bude transformována s použitím optimalizovaných funkcí.

Pro transformaci měr důvěry jsem navrhl dvě parametrizované funkce s definičním oborem $D_f = (0, 1)$. První funkce, f_1, je definována rovnicí 5.1. f_1 realizuje přechod takovým způsobem, že do vstupní hodnoty $x < a$ zůstává ve funkční hodnotě 0, následně lineárně stoupá k hodnotě 1 až do $x = b$, od které pokračuje ve funkční hodnotě 1.

$$y = \begin{cases} 0 & \text{pro } x < a \\ \frac{x}{b-a} - \frac{1}{\frac{b}{a}-1} & \text{pro } a \leq x < b; a \neq b, a \neq 0 \\ 1 & \text{pro } b \leq x \end{cases} \tag{5.1}$$

Druhá funkce, f_2, definována rovnicí 5.2, odpovídá dvěma úsečkám, které mají společný bod $A(a, b)$, ke a a b jsou opět parametry. První úsečka vede z bodu $(0, 0)$ do A, druhá z A do $(1, 1)$.

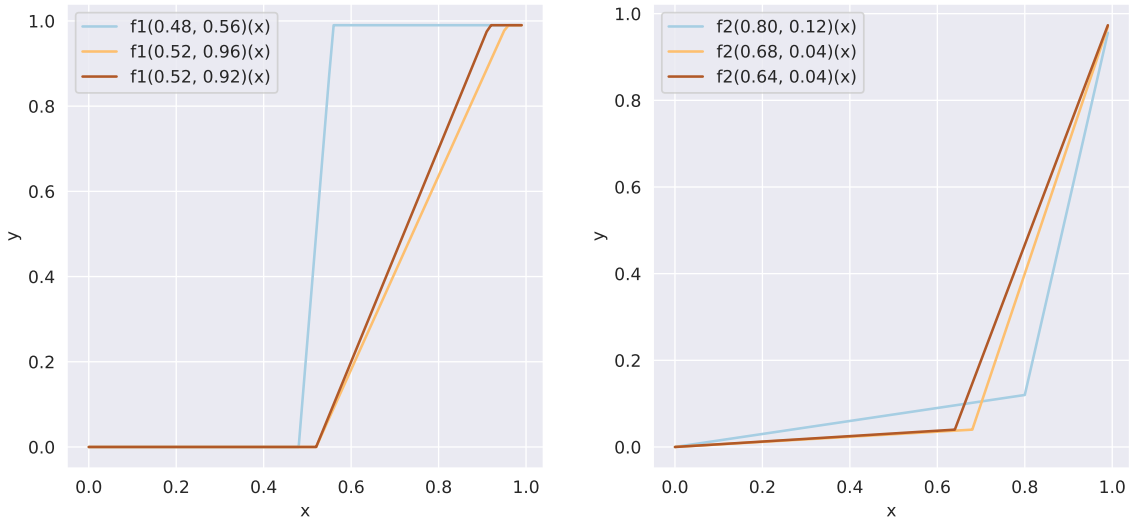
$$y = \begin{cases} \frac{b}{a} \cdot x & \text{pro } x < a \\ \frac{b-1}{a-1} \cdot x + \frac{a-b}{a-1} & \text{pro } x \geq a \end{cases} \tag{5.2}$$

Parametry funkcí a a b byly optimalizovány prohledáváním celého stavového prostoru dvojic (a, b), kde a i b mohou nabývat hodnot z intervalu $(0, 1)$. Pro každou zvažovanou dvojici (a, b) byla vytvořena funkce odpovídající doplnění těchto parametrů. Funkce byla následně dosazena do metody dvou fokálních hypotéz tak, že je aplikována na míru důvěry přiřazenou hlavní hypotéze. S takto upravenou metodou je provedena optimalizace sady pravidel nad datovou sadou. Optimalizovaná pravidla jsou následně nad datovou sadou otestována, přičemž jako hodnotící funkce pro optimalizaci bylo vybráno F1 skóre. Je uchováváno dosavadní nejlepší nastavení parametrů spolu s nejvyšší hodnotou F1 skóre. Nejlepší nastavení po dokončení průchodu stavovým prostorem je výstupem algoritmu.

Optimalizace byla provedena nad omezenou datovou sadou na základě současných zdrojů dat na sezení, podobně jako v předešlém experimentu, následně i na kompletní datové sadě. Získané parametry jsou uvedeny v tabulce 5.9. Vzniklé funkce jsou vykresleny na obrázku 5.4.

funkce	f_1		f_2	
datová sada	a	b	a	b
Kompletní	0,48	0,56	0,80	0,12
$N = 2$	0,52	0,96	0,68	0,04
$N = 3$	0,52	0,92	0,64	0,04

Tabulka 5.9: **Parametry získané optimalizací**



Obrázek 5.4: **Průběhy funkcí f_1 a f_2.**

Takto optimalizované funkce byly otestovány způsobem definovaným v sekci 5.4, navíc včetně omezení datové sady provedené v předchozím experimentu 5.5. Výsledky jsou uvedeny v tabulkách 5.10 pro kompletní datovou sadu, 5.11 pro $N = 2$ a 5.12 pro $N = 3$.

Funkce	Preciznost	Senzitivita	F1 skóre	Přesnost
$f_1(0{,}48, 0{,}56)$	**0,763**	**0,734**	**0,717**	**0,734**
$f_1(0{,}52, 0{,}96)$	0,762	0,717	0,696	0,717
$f_1(0{,}52, 0{,}92)$	0,770	0,734	0,716	0,734
$f_2(0{,}80, 0{,}12)$	**0,774**	**0,753**	**0,738**	**0,753**
$f_2(0{,}68, 0{,}04)$	0,770	0,738	0,720	0,738
$f_2(0{,}64, 0{,}04)$	0,765	0,734	0,716	0,734

Tabulka 5.10: **Ukazatele pro optimalizovanou metodu dvou fokálních hypotéz nad kompletní datovou sadou.** Zvýrazněny byly řádky s parametry optimalizovanými na kompletní datovou sadu.

Funkce	Preciznost	Senzitivita	F1 skóre	Přesnost
$f_1(0{,}48, 0{,}56)$	0,798	0,774	0,765	0,774
$f_1(0{,}52, 0{,}96)$	**0,856**	**0,851**	**0,845**	**0,851**
$f_1(0{,}52, 0{,}92)$	0,855	0,850	0,843	0,850
$f_2(0{,}80, 0{,}12)$	0,835	0,833	0,827	0,833
$f_2(0{,}68, 0{,}04)$	**0,839**	**0,837**	**0,830**	**0,837**
$f_2(0{,}64, 0{,}04)$	0,838	0,838	0,832	0,838

Tabulka 5.11: **Ukazatele pro optimalizovanou metodu dvou fokálních hypotéz nad omezenou datovou sadou.** $N = 2$. Zvýrazněny byly řádky s parametry optimalizovanými na redukovanou datovou sadu $N = 2$.

Funkce	Preciznost	Senzitivita	F1 skóre	Přesnost
$f_1(0{,}48, 0{,}56)$	0,852	0,812	0,772	0,812
$f_1(0{,}52, 0{,}96)$	0,911	0,897	0,891	0,897
$f_1(0{,}52, 0{,}92)$	**0,961**	**0,958**	**0,957**	**0,958**
$f_2(0{,}80, 0{,}12)$	0,958	0,952	0,953	0,952
$f_2(0{,}68, 0{,}04)$	0,938	0,933	0,932	0,933
$f_2(0{,}64, 0{,}04)$	**0,926**	**0,921**	**0,920**	**0,921**

Tabulka 5.12: **Ukazatele pro optimalizovanou metodu dvou fokálních hypotéz nad omezenou datovou sadou.** $N = 3$. Zvýrazněny byly řádky s parametry optimalizovanými na redukovanou datovou sadu $N = 3$.

Mezi tabulkami lze pozorovat rostoucí tendenci se zvyšujícím se N bez ohledu na konkrétní funkci či parametry. Funkce s parametry optimalizovanými pro redukovanou datovou sadu mají horší ukazatele pro celou datovou sadu, což naznačuje špatnou přenositelnost parametrů. Funkce $f_2(0{,}68, 0{,}04)$ vykazuje anomálii, protože konkurující dvojice parametrů získaná nad $N = 3$ produkuje pro $N = 2$ lepší ukazatele F1 skóre i přesnosti, ačkoliv byly parametry $(0{,}68, 0{,}04)$ získány nad omezenou sadou $N = 2$. Toto je vysvětlováno vlivem rozdělení datové sady v průběhu 5násobné křížové validace a podobné anomálie lze pozorovat i u dalších kombinací parametrů.

Všechny testované kombinace funkcí a parametrů vedly ke zlepšení výsledků oproti původní metodě dvou fokálních hypotéz. V případě kompletní datové sady došlo ke zlepšení přesnosti až o 2 %, pro $N = 2$ až o 6 % a v případě $N = 3$ dokonce o 12 %. Funkce f_1 produkovala nejlepší ukazatele s parametry $(0{,}52, 0{,}92)$, přičemž kombinace $(0{,}52, 0{,}96)$ dle očekávání podává velmi podobné výsledky, překonávající první zmíněnou kombinaci pouze pro $N = 2$. U funkce f_2 vedly k nejlepším ukazatelům parametry $(0{,}80, 0{,}12)$. Pro zvolení výsledné nejúspěšnější kombinace funkce s parametry je uvažován jako klíčový výkon nad celou datovou sadou. Z tohoto důvodu je za optimální považována funkce $f_2(0{,}80, 0{,}12)$ a bude použita v souhrnném vyhodnocení.

5.7 Souhrnné vyhodnocení experimentů

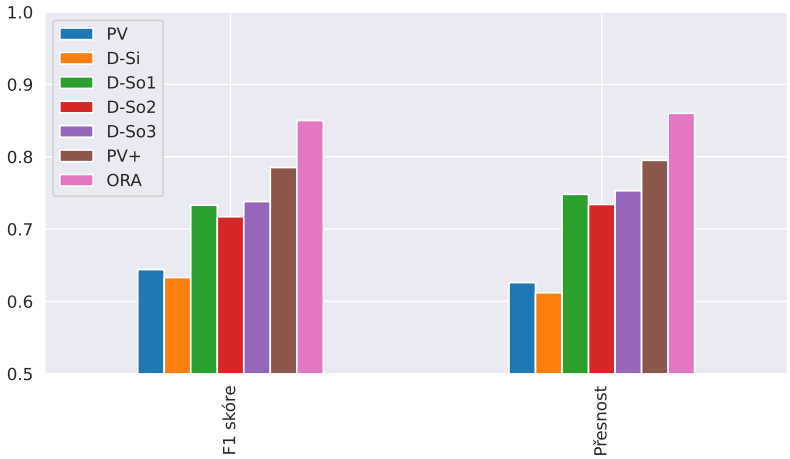
Tato sekce shrnuje všechny provedené experimenty do souhrnného vyhodnocení. Byla provedena série experimentů, které si kladly za cíl otestovat korektní funkčnost řešení a druhotně se zaměřovaly na zkoumání přesnosti implementované metody, její optimalizaci v několika fázích a porovnání s dalšími navrženými metodami. Součástí zkoumání bylo i sledování vlivu omezení datové sady dle kritéria počtu zdrojů dat ke konkrétnímu sezení.

Řešení se ukázalo jako funkční, nicméně v základní verzi konfigurované experty či informované přesností modulů je jeho přesnost srovnatelná s triviálním přístupem, který byl v rámci experimentů 5.3 označován jako prioritní výběr.

Za účelem vylepšení přesnosti řešení bylo v sekci 5.4 experimentováno s optimalizací použití D-S teorie dvěma metodami vycházejícími z relevantní literatury, přičemž bylo využito matic záměn získaných z předešlých experimentů. Zároveň byla implementována vylepšená verze metody prioritního výběru, označovaná prioritní výběr+. Následně byl v sekci 5.6 proveden další pokus o optimalizaci metody dvou fokálních hypotéz.

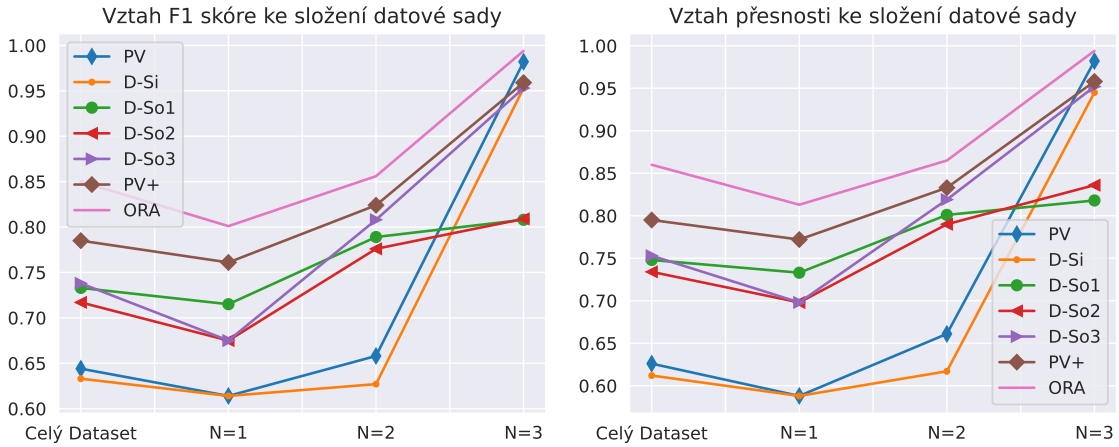
Výsledkem těchto optimalizací je, že metodami využívajícími D-S teorie jsou překonány nejen ukazatele osamoceného nejpřesnějšího modulu, ale i metody prioritního výběru. Porovnání ukazatelů metod založených na D-S teorii s metodou prioritního výběru jakožto

minimem, orákulem jakožto maximem a metodou prioritní výběr+ lze vidět na obrázku 5.5. Obrázek dále ukazuje, že pokud jsou k dispozici data o přesnosti datových zdrojů, což umožňuje použití metod prioritního výběru a informované D-S teorie, prioritní výběr produkuje lepší ukazatele s rozdílem 1 %. Podobně je ukázáno, že optimalizace na úrovni jednotlivých pravidel vede k výraznějšímu zlepšení ukazatelů. Z pokusů o optimalizaci metody D-S teorie se ukazuje jako nejlepší metoda distribuce záměn, druhá metoda, metoda dvou fokálních hypotéz, produkuje lepší výsledky při optimalizaci distribuce přidělování měr důvěry. Při porovnání výsledků proti orákulu je možno vidět, že prostor pro optimalizaci ukazatelů kombinačních metod byl prozkoumán přibližně do poloviny rozsahu, nicméně experimenty lze prohlásit za úspěšné při porovnání oproti informované D-S teorii. Metoda prioritního výběru+ opět mírně překonává metody D-S teorie, tentokrát o 4 %.



Obrázek 5.5: **Porovnání ukazatelů testovaných metod.** Prioritní výběr a orákulum zde slouží jako hranice, kde by se metoda kombinace měla pohybovat. Legenda: *PV* – Prioritní výběr, *D-Si* – Informovaná D-S teorie, *D-So1* – D-S metoda distribuce záměn, *D-So2* – D-S metoda dvou fokálních hypotéz, *D-So3* – D-S metoda dvou fokálních hypotéz s transformační funkcí, *PV+* – Prioritní výběr+, *ORA* – Orákulum.

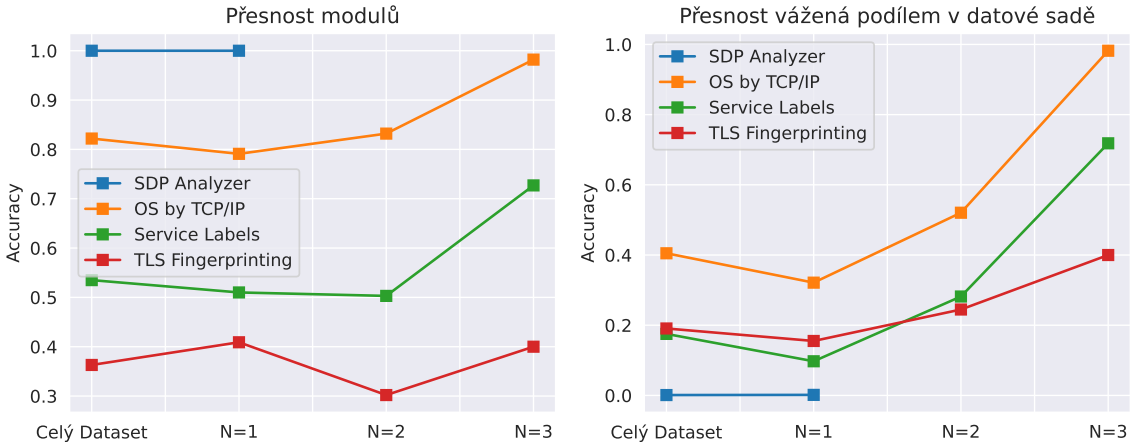
Pro lepší pochopení výkonnosti metod bylo provedeno rozložení datové sady dle kritéria současných zdrojů dat a naměření ukazatelů. Výsledky lze vidět na grafech na obrázku 5.6. Omezení datové sady promítnuté na těchto grafech označuje počet současných zdrojů dat rovný N, nikoliv větší nebo rovno N, jak tomu bylo v sekci 5.5.



Obrázek 5.6: **Výsledky metod kombinace z pohledu omezení datové sady.** Legenda: *PV* – Prioritní výběr, *D-Si* – Informovaná D-S teorie, *D-So1* – D-S metoda distribuce záměn, *D-So2* – D-S metoda dvou fokálních hypotéz, *D-So3* – D-S metoda dvou fokálních hypotéz s transformační funkcí, *PV+* – Prioritní výběr+, *ORA* – Orákulum.

Grafy ukazují výkonnost metod pro jednotlivé segmenty datové sady. Pro $N = 1$ jsou ukazatele všech metod nejnižší z celého průběhu, zároveň produkují relativně malé rozdíly, což je očekáváno, protože nedochází k žádné kombinaci. Pro $N = 2$ lze vidět rostoucí vývoj ukazatelů u všech metod krom informované D-S teorie. Při třech současných zdrojích dat lze pozorovat několik vývojů ukazatelů. Ukazatele metody D-S dvou fokálních hypotéz s transformační funkcí se vyrovnávají metodě prioritního výběru+, zatímco ukazatele ostatních optimalizovaných metod D-S teorie zůstávají na úrovních z $N = 2$. Toto může být zapříčiněno použitou transformační funkcí, která uvažuje pravidla s přesností nad hodnotu 0,8, a zbylým přiřazuje velmi malou míru důvěry, viz obrázek 5.4. Takto získává vlastnosti podobné prioritnímu výběru, který jako výsledek uvažuje první dostupné statisticky nejpřesnější pravidlo. Překvapující je skvělá výkonnost metod majících informace o přesnosti pouze na úrovni modulů.

Pro lepší pochopení tohoto fenoménu bylo provedeno měření přesnosti modulů pro identický rozklad datové sady, jehož výsledky jsou uvedeny na grafech v obrázku 5.7. Graf vlevo ukazuje přesnost modulů pro jednotlivá omezení datové sady. Graf vpravo ukazuje stejná data, ale násobená podílem množství dat z modulu v datové sadě, jak bylo dříve uvedeno v tabulce 5.2. Tato data jasně ukazují, že se zvyšujícím se počtem současných zdrojů dat v použité datové sadě také roste míra, do které se lze spolehnout na modul OS by TCP/IP. Tímto lze vysvětlit vynikající ukazatele metod mající informace o přesnosti pouze na úrovni modulů, které modulu OS by TCP/IP přisuzovaly po modulu SDP Analyzer největší důvěru. Stejně tak lze vysvětlit vynikající ukazatele metody prioritní výběr+. Tento graf zároveň opodstatňuje rostoucí tendenci grafů 5.6, protože přesnost vážená podílem v datové sadě má rostoucí tendenci pro všechny moduly krom modulu SDP Analyzer.



Obrázek 5.7: **Vztah přesnosti modulů ke složení datové sady.** Data modulu SDP Analyzer se v datové sadě vyskytují pouze pro $N = 1$, proto nejsou výsledky dále značeny.

Závěrem lze říct, že použité metody kombinace byly více či méně optimalizovány na tyto vlastnosti datové sady. Metody prioritního výběru dosahovaly vynikajících ukazatelů úspěšnosti z toho důvodu, že při takovémto složení vstupních dat může být postup prioritního výběru nejlepší strategií. Otázkou, kterou lze zodpovědět pouze otestováním metod na datové sadě s jiným rozložením přesnosti datových vstupů, je, zda by metody založené na D-S teorii obstály lépe v takovém kontextu. Věřím, že ano, a můj další výzkum se zaměří především na sběr dalších dat a objasnění této otázky.

Důležitým úspěchem experimentální části zůstává implementace optimalizace na úrovni jednotlivých pravidel. Je nutné poznamenat, že prováděním optimalizací pravidel dochází ke ztrátě původního rozlišení z pohledu taxonomie. Optimalizované řešení, ať se jedná o metody D-S teorie či optimalizovaný prioritní výběr, má na výstupu rozlišení dané zdroji dat pro anotaci. Ačkoliv lze argumentovat, že přenositelnost takto optimalizované sady pravidel do sítě s jinou distribucí zařízení je diskutabilní, optimalizovaná sada klasifikačních pravidel může stále mít svá použití. Z toho důvodu byl implementován způsob ukládání vzniklých pravidel do souboru.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat metodu fúze informací, pomocí které bude možné spojit vstupní zdroje dat v projektu ADiCT takovým způsobem, že bude zachována jejich různá míra podrobnosti. Výstupem metody jsou štítky zaměřující se na operační systém a typ daného zařízení.

V práci bylo vytvořeno řešení v podobě interpretu klasifikačních pravidel, která jsou vyjádřena navrženým konfiguračním jazykem. Pro zachování rozdílné míry podrobnosti byla využita taxonomie, vytvořená pro každý typ štítku. Výstupy klasifikačních pravidel jsou následně spojeny pomocí metody Dempster-Schaferovy teorie, která je spojí s pomocí vah udávaných v klasifikačních pravidlech do jediného štítku, vypovídajícího o všech vstupech. Všechny zmíněné součásti řešení byly navrženy a implementovány tak, aby bylo možné je konfigurovat a snadno rozšířit pro nové zdroje dat či měnící se potřeby projektu ADiCT.

Implementace řešení, interpretu jazyka i správy taxonomie byla verifikována pomocí sady jednotkových testů. Řešení bylo následně integrováno do systému ADiCT v podobě sekundárního modulu a dále bylo otestováno na datech z reálné sítě.

Součástí práce bylo i vyhodnocení přesnosti použité metody a její optimalizace. Byla použita datová sada vycházející z dat nasbíraných v projektu ADiCT, která po vyvážení čítala 3764 záznamů anotovaných sezení vázaných k IP adresám. Výstupem experimentů byla krom statistik přesnosti modulů i optimalizovaná metoda trénování pravidel na konkrétní datovou sadu, která zvýšila přesnost kombinace o 14 % vůči statistikami informované metodě a o 19 % vůči metodě založené na expertních ohodnoceních datových zdrojů. Byl implementován i způsob uložení takto optimalizovaných pravidel, což umožňuje využití výsledků experimentů i v rámci integrovaného řešení.

Výsledek této práce bude využit v rámci projektu ADiCT, kde by se měl stát součástí systému a připravovaného webového rozhraní, kde uživatelům může umožňovat snazší orientaci ve sbíraných datech. V rámci provedených experimentů byly vyčerpány možnosti získané datové sady. Ačkoliv se ve výsledcích experimentů ukázala triviální metoda Prioritní výběr+ jako přesnější než optimalizované metody D-S teorie, neznamená to, že jsou tyto metody obecně horší. Pro ověření je třeba použití kvalitnější datové sady, na což bych se chtěl zaměřit v navazující práci. V projektu ADiCT je sada vstupních modulů nadále vylepšována, takže lze očekávat zlepšení datové sady, díky kterému by se měly projevit výhody D-S teorie. Experimenty v této práci se v souladu se zadáním zaměřovaly na analýzu a vylepšení metody založené na Dempster-Schaferově teorii, další možnou navazující prací je porovnání s dalšími metodami fúze informací s jiným formálním základem.

Literatura

[1] Abdollahpour, M., Rezaii, T. Y., Farzamnia, A. a Meshgini, S. Sleep stage classification using dempster-shafer theory for classifier fusion. In: IEEE. *2018 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. 2018, s. 1–4.

[2] Al Ani, A. a Deriche, M. A new technique for combining multiple classifiers using the Dempster-Shafer theory of evidence. *Journal of Artificial Intelligence Research*. 2002, sv. 17, s. 333–361.

[3] Anderson, B. a McGrew, D. Accurate TLS Fingerprinting using Destination Context and Knowledge Bases. *ArXiv preprint arXiv:2009.01939*. 2020.

[4] Auffret, P. SinFP, unification of active and passive operating system fingerprinting. *Journal in computer virology*. Springer. 2010, sv. 6, č. 3, s. 197–205.

[5] Bai, L., Yao, L., Kanhere, S. S., Wang, X. a Yang, Z. Automatic device classification from network traffic streams of internet of things. In: IEEE. *2018 IEEE 43rd conference on local computer networks (LCN)*. 2018, s. 1–9.

[6] Bolf, R. *Profilování síťových entit pro zlepšení situačního povědomí*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/24028/.

[7] Burger, T., Aran, O., Urankar, A., Caplier, A. a Akarun, L. A Dempster-Shafer theory based combination of classifiers for hand gesture recognition. In: Springer. *International Conference on Computer Vision and Computer Graphics*. 2007, s. 137–150.

[8] Cejka, T., Bartos, V., Svepes, M., Rosa, Z. a Kubatova, H. NEMEA: A Framework for Network Traffic Analysis. In: *12th International Conference on Network and Service Management (CNSM 2016)*. 2016.

[9] Chen, Q., Whitbrook, A., Aickelin, U. a Roadknight, C. Data classification using the Dempster–Shafer method. *Journal of Experimental & Theoretical Artificial Intelligence*. Taylor & Francis. 2014, sv. 26, č. 4, s. 493–517.

[10] Dainotti, A., Pescapé, A. a Sansone, C. Early classification of network traffic through multi-classification. In: Springer. *International Workshop on Traffic Monitoring and Analysis*. 2011, s. 122–135.

[11] Dempster, A. P. Upper and Lower Probabilities Induced by a Multivalued Mapping. *Institute of Mathematical Statistics*. duben 1967, sv. 38, č. 2, s. 325–339.

[12] Dhakal, A. a Ramakrishnan, K. Machine learning at the network edge for automated home intrusion monitoring. In: IEEE. *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. 2017, s. 1–6.

[13] Fielding, R. a Reschke, J. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [Internet Requests for Comments]. RFC 7231. RFC Editor, June 2014. http://www.rfc-editor.org/rfc/rfc7231.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc7231.txt.

[14] Gancheva, Z., Sattler, P. a Wüstrich, L. TLS Fingerprinting Techniques. *Network*. 2020, sv. 15.

[15] Gao, H., Shen, X., Jiang, Z., Yang, H. a Yan, L. Image subcategory classification based on Dempster-Shafer evidence theory. In: IEEE. *2012 International Conference on Computer Science and Service System*. 2012, s. 2289–2292.

[16] Hagos, D. H., Yazidi, A., Kure, Ø. a Engelstad, P. E. A Machine-Learning-Based Tool for Passive OS Fingerprinting With TCP Variant as a Novel Feature. *IEEE Internet of Things Journal*. IEEE. 2020, sv. 8, č. 5, s. 3534–3553.

[17] Hejcman, L. *Fingerprinting and Identification of TLS Connections*. Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/23922/.

[18] Hsu, A., Tront, J., Raymond, D., Wang, G. a Butt, A. Automatic iot device classification using traffic behavioral characteristics. In: IEEE. *2019 SoutheastCon*. 2019, s. 1–7.

[19] John, A., Jeff, A. a Josh, A. *JA3 - A method for profiling SSL/TLS Clients* [online]. 2017 [cit. 2022-03-04]. Dostupné z: https://github.com/salesforce/ja3.

[20] Koumar, J. *Automatické rozpoznávání sít'ových zařízení a jejich závislostí*. Praha, CZ, 2020. Bakalářská práce. České vysoké učení technické v Praze. Vypočetní a informační centrum. Dostupné z: https://dspace.cvut.cz/handle/10467/88277.

[21] Kuncheva, L. I. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014.

[22] Laštovička, M., Dufka, A. a Komárková, J. Machine learning fingerprinting methods in cyber security domain: Which one to use? In: IEEE. *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 2018, s. 542–547.

[23] Laštovička, M., Jiršík, T., Čeleda, P., Špaček, S. a Filakovský, D. Passive os fingerprinting methods in the jungle of wireless networks. In: IEEE. *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. 2018, s. 1–9.

[24] Li, R., Sosnowski, M. a Sattler, P. An Overview of OS Fingerprinting Tools on the Internet. *Network*. 2020, sv. 73.

[25] Lyon, G. F. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure. Com LLC (US), 2008.

[26] Matoušek, P., Burgetová, I., Ryšavý, O. a Victor, M. On Reliability of JA3 Hashes for Fingerprinting Mobile Applications. In: *Digital Forensics and Cyber Crime. ICDF2C 2020. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.* Springer International Publishing, 2021, sv. 351, s. 1–22. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. DOI: 10.1007/978-3-030-68734-2_1. ISBN 978-3-030-68733-5. Dostupné z: https://www.fit.vut.cz/research/publication/12307.

[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: *DCNET2014. Proceedings of the 5th International Conference on Data Communication Networking.* SciTePress - Science and Technology Publications, 2014, s. 21–27. DOI: 10.5220/0005099500210027. ISBN 978-989-758-042-0. Dostupné z: https://www.fit.vut.cz/research/publication/10642.

[28] McGrew, D., Enright, B. a Anderson, B. *Mercury: Fast TLS, TCP, and IP Fingerprinting.* [online]. 2020 [cit. 2022-03-04]. Dostupné z: https://github.com/cisco/mercury.

[29] Millar, K., Cheng, A., Chew, H. G. a Lim, C.-C. Operating system classification: A minimalist approach. In: IEEE. *2020 International Conference on Machine Learning and Cybernetics (ICMLC).* 2020, s. 143–150.

[30] Peñafiel, S., Baloian, N., Sanson, H. a Pino, J. A. Applying Dempster–Shafer theory for developing a flexible, accurate and interpretable classifier. *Expert Systems with Applications.* Elsevier. 2020, sv. 148, s. 113262.

[31] Penafiel, S., Baloian, N., Sanson, H. a Pino, J. A. Predicting stroke risk with an interpretable classifier. *IEEE Access.* IEEE. 2020, sv. 9, s. 1154–1166.

[32] Reineking, T. *Belief functions: theory and algorithms.* 2014. Disertační práce. Universität Bremen.

[33] Schwartzenberg, J. *Using machine learning techniques for advanced passive operating system fingerprinting.* 2010. Diplomová práce. University of Twente.

[34] Sedláček, O. *Gathering knowledge about devices in local networks by analyzing service discovery protocols.* Brno, CZ, 2021. Projektová praxe. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://excel.fit.vutbr.cz/submissions/2021/002/2.pdf.

[35] Shafer, G. *A mathematical theory of evidence.* Princeton university press, 1976.

[36] Smets, P. The nature of the unnormalized beliefs encountered in the transferable belief model. In: Elsevier. *Uncertainty in artificial intelligence.* 1992, s. 292–297.

[37] Smets, P. a Kennes, R. The transferable belief model. *Artificial intelligence.* Elsevier. 1994, sv. 66, č. 2, s. 191–234.

[38] Zalewski, M. *P0f v3 (3.09b)* [online]. 2014 [cit. 2022-02-23]. Dostupné z: https://lcamtuf.coredump.cx/p0f3/.

[39] Wang, T., Liu, R. a Qi, G. Multi-classification assessment of bank personal credit risk based on multi-source information fusion. *Expert Systems with Applications*. Elsevier. 2022, sv. 191, s. 116236.

[40] WhatIsMyBrowser. *User Agents Database* [online]. 2022 [cit. 2022-03-08]. Dostupné z: https://developers.whatismybrowser.com/useragents/database/.

[41] Yang, K., Li, Q. a Sun, L. Towards automatic fingerprinting of IoT devices in the cyberspace. *Computer Networks*. Elsevier. 2019, sv. 148, s. 318–327.

[42] Zadeh, L. A. Review of a mathematical theory of evidence. *AI magazine*. 1984, sv. 5, č. 3, s. 81–81.

[43] Zhang, L., Zhang, L., Teng, W. a Chen, Y. Based on information fusion technique with data mining in the application of finance early-warning. *Procedia Computer Science*. Elsevier. 2013, sv. 17, s. 695–703.

Příloha A

Obsah přiloženého paměťového média

Kořenový adresář přiloženého paměťového média obsahuje následující adresáře:

- **src** – obsahuje zdrojové kódy řešení, rozdělené do dvou podadresářů:
  experiments – implementace všech dokumentovaných experimentů s řešením
  adict – integrace řešení do systému ADiCT
- **documentation** – zdrojové soubory této technické zprávy a její finální podoba ve formátu PDF

Příloha B

Podklady pro návrh interpretu

Tato příloha obsahuje podklady použité při návrhu interpretu. Jedná se o soubor regulárních výrazů, použitých pro lexikální analýzu, LL gramatiku popisující konfigurační jazyk, a gramatiku spolu s precedenční tabulkou, které popisují syntaxi podmínek.

B.1 Regulární výrazy pro lexikální analýzu

Terminál	Regulární výraz
WORD	[a-zA-Z_][a-zA-Z0-9_]*
PROPERTY	\.[a-zA-Z0-9_]+
FLOAT	-?\d+(.\d+)?
STRING1	\'([^\n\']*)\'
STRING2	\"([^\n\"]*)\"
OPEN_BRACK	(
CLOSE_BRACK	)
OPEN_CURLY	{
CLOSE_CURLY	}
OPEN_SQUARE	[
CLOSE_SQUARE	]
COLON	:
COMMA	,
LT	>
GT	<
EQ	==
DASH	-
NEWLINE	\n
NEG	\!
COMMENT	[\t]*\#[^\n]*\n?
WHITESPACE	[\t]+

Tabulka B.1: **Regulární výrazy použité pro lexikální analýzu.**

B.2 LL gramatika

Terminály jsou značeny velkými písmeny, případně znakem, který reprezentují (v případě `,`, `:`, `-`, `[` a `]`). Terminál `EXPR` zastupuje výrazy v jazyce a naznačuje předání řízení syntaktické analýzy precedenční analýze. Neterminály jsou značeny malými písmeny.

1. `config` → `blank_lines classification_seq EOF`
2. `classification_seq` → `classification NEWLINE blank_lines classification_seq`
3. `classification_seq` → ε
4. `classification` → `opt_neg opt_whitespace class_name class_list NEWLINE WHITESPACE rule rules`
5. `optional_neg` → `NEG`
6. `optional_neg` → ε
7. `class_list` → `, opt_whitespace class_name class_list`
8. `class_list` → ε
9. `class_name` → `WORD property_list`
10. `property_list` → `PROPERTY property_list`
11. `property_list` → ε
12. `rule` → `FLOAT opt_whitespace rule_development`
13. `rule_development` → `basic_rule`
14. `rule_development` → `range_rule`
15. `basic_rule` → `: EXPR NEWLINE`
16. `range_rule` → `- WHITESPACE FLOAT : opt_newline`
    `opt_whitespace [ opt_newline`
    `opt_whitespace expr_list opt_newline`
    `opt_whitespace ] NEWLINE`
17. `rules` → `WHITESPACE rule rules`
18. `rules` → ε
19. `expr_list` → `EXPR NEWLINE WHITESPACE expr_list`
20. `expr_list` → ε
21. `opt_newline` → `NEWLINE`
22. `opt_newline` → ε
23. `opt_whitespace` → `WHITESPACE`
24. `opt_whitespace` → ε
25. `blank_lines` → `opt_whitespace NEWLINE blank_lines`
26. `blank_lines` → ε

B.3 Gramatika pro zpracování výrazů

Terminály jsou značeny velkými písmeny, případně znakem, který reprezentují (v případě `,`, `(`, `)`, `:`, `{` a `}`). Neterminály jsou značeny malými písmeny.

1. `val` → `val AND val`
2. `val` → `val OR val`
3. `val` → `val IN val`
4. `val` → `val LT val`
5. `val` → `val GT val`
6. `val` → `val EQ val`
7. val → `val ,`
8. `val` → `PROPERTY`
9. `val` → `STRING`
10. `val` → `FLOAT`
11. `val` → `WORD empty_brack`
12. `empty_brack` → `( )`
13. `val` → `WORD val`
14. `val` → `( val )`
15. `val` → `val , val`
16. `val` → `{ key_val }`
17. `key_val` → `key_val , key_val`
18. `key_val` → `val : val`

B.4 Precedenční tabulka

Tato tabulka určuje precedenci operátorů, uvedených v gramatice z předchozí sekce. Označení terminálu `PROPERTY` bylo zkráceno na `PROP`, označení `STRING` na `STR`.

	WORD	PROP	STR	FLOAT	(	)	{	}	,	:	LT	GT	EQ	IN	OR	AND	EOF
WORD					<	>		>	>		>	>	>	>	>	>	>
PROP						>			>		>	>	>	>	>	>	>
STR						>		>	>	>	>	>	>	>	>	>	>
FLOAT						>		>	>	>	>	>	>	>	>	>	>
(	<	<	<	<	<	=	<	<	<	<	<	<	<	<	<	<	
)						>	>	>	>	>	>	>	>	>	>	>	>
{	<		<	<	<	>	<	=	<	<	<	<	<	<	<	<	
}					<	>		>	>	>	>	>	>	>	>	>	>
,	<	<	<	<	<	>	<	>	>	<	>	>	>	>	>	>	>
:	<		<	<	<	>	<	>	>		>	>	>	>	>	>	>
LT	<	<	<	<	<	>	<	>	>	>				>	>	>	>
GT	<	<	<	<	<	>	<	>	>	>				>	>	>	>
EQ	<	<	<	<	<	>	<	>	>	>				>	>	>	>
IN		<			<	>	<	>	>	>					>	>	>
OR	<	<	<	<	<	>	<	>	>	>	<	<	<	<	>	<	>
AND	<	<	<	<	<	>	<	>	>	>	<	<	<	<	>	>	>
EOF	<	<	<	<	<		<		<	<	<	<	<	<	<	<	

Tabulka B.2: **Precedenční tabulka pro zpracování podmínek.**

Příloha C

Navržená taxonomie tříd

Tato příloha ukazuje navrženou taxonomii tříd, která je použita pro sloučenou reprezentaci dat. Taxonomie je uvedena ve formátu YAML.

```
OperatingSystem:
  Windows:
    '7':
    '8':
    '8.1':
    '10':
  Linux:
    'Ubuntu':
    'Fedora':
    'CentOS':
    'Debian':
    'openSUSE':
  MacOS:
    '10.11_ElCapitan':
    '10.12_Sierra':
    '10.13_HighSierra':
    '10.14_Mojave':
    '10.15_Catalina':
  iOS:
  Android:

Device:
  Workstation:
  SmartPhone:
  SmartTV:
  VoiceAssistant:
  Hub:
  Appliance:
  Router:
  NAS:
  Server:
```