UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

IDENTIFIKÁCIA RELEVANTNÝCH DIGITÁLNYCH STÔP PRI FORENZNOM VYŠETROVANÍ
DIPLOMOVÁ PRÁCA

Študijný program:	Informatika
Pracovisko:	Ústav informatiky
Vedúci práce:	doc. RNDr. JUDr. Pavol Sokol, PhD.
Konzultant práce:	Mgr. Eva Marková

Košice 2024

Bc. František KURIMSKÝ

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. František Kurimský
Študijný program:	informatika (jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor:	Informatika
Typ záverečnej práce:	Diplomová práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Identifikácia relevantných digitálnych stôp pri forenznom vyšetrovaní

Názov EN: Identification of relevant digital evidence in forensic investigation

Cieľ: 1) Analyzovať forenzné artefakty v operačnom systéme Windows.
2) Porovnať existujúce prístupy k identifikácii relevantných digitálnych stôp pri forenznom vyšetrovaní operačného systému Windows.
3) Navrhnúť model pre identifikáciu relevantných digitálnych stôp pri forenznom vyšetrovaní operačného systému Windows, implementovať model a zhodnotiť efektívnosť tohto modelu.

Literatúra:
1) Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. Computer networks, 51(12), 3448-3470.
2) Alabadi, M., & Celik, Y. (2020, June). Anomaly detection for cyber-security based on convolution neural network: A survey. In 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-14). IEEE.
3) Mohammad, R. M. A., & Alqahtani, M. (2019). A comparison of machine learning techniques for file system forensics analysis. Journal of Information Security and Applications, 46, 53-61.
4) Grajeda, C., Breitinger, F., & Baggili, I. (2017). Availability of datasets for digital forensics–and what is missing. Digital Investigation, 22, S94-S105.

Vedúci:	doc. RNDr. JUDr. Pavol Sokol, PhD.
Konzultant:	RNDr. Eva Marková
Oponent:	doc. RNDr. Csaba Török, CSc.
Ústav :	ÚINF - Ústav informatiky
Riaditeľ ústavu:	doc. RNDr. Ondrej Krídlo, PhD.

Dátum schválenia: 30.04.2024

Abstrakt v štátnom jazyku

V dnešnej dobe neustále narastá počet kybernetických útokov. Adekvátna reakcia na bezpečnostné útoky a incidenty si vyžaduje vo viacerých prípadoch vyžaduje použitie digitálnej forenznej analýza. Pri samotnom forenznom vyšetrovaní je potrebné, čo najrýchlejšie sa dopracovať k relevantným digitálnym stopám (dátam), ktoré obsahujú informácie o bezpečnostnom incidente a postupe útočníka. Základným problémom digitálnej forenznej analýza je veľké množstvo forenzných artefaktov získaných z napadnutých systémov. Tie sa z veľkej časti skladajú z artefaktov nerelevantných pre vyšetrovanie daného prípadu, resp. bezpečnostného incidentu. Cieľom predloženej práce je ušetriť čas forenzného analytika pri hľadaní relevantných digitálnych stôp (dát) pre účely forenznej analýzy. To je možné dosiahnuť automatizáciou tohto procesu pomocou metód strojového učenia, najmä metód primárne určených na hľadanie anomálii. Nájdené digitálne artefakty poskytujú forenznému analytikovi nadhľad nad bezpečnostným incidentom a umožňujú rýchlejšie stanovenie a následne potvrdenie alebo vyvrátenie forenzných hypotéz o bezpečnostnom incidente a činnosti útočníka.

Kľúčové slová: digitálna stopa, digitálna forenzná analýza, detekcia anomálie, forenzný artefakt

Abstrakt v cudzom jazyku

Nowadays, the number of cyber-attacks is constantly increasing. Adequate response to security attacks and incidents requires the use of digital forensics in several cases. In the actual forensic investigation, it is necessary to get to the relevant digital traces (data) as quickly as possible, which contain information about the security incident and the attacker's actions. A fundamental problem of digital forensics is the large amount of forensic artifacts recovered from compromised systems. These largely consist of artifacts not relevant to the investigation of the case or security incident. The aim of the present work is to save the time for the forensic analyst in finding relevant digital traces (data) for forensic analysis purposes. This can be achieved by automating this process using machine learning methods, especially methods primarily designed for anomaly search. The digital artefacts found provide the forensic analyst with insight into the security incident and allow for more rapid determination and subsequent confirmation or refutation of forensic hypotheses about the security incident and the attacker's actions.

Keywords: digital evidence, digital forensics, anomaly detection, forensics artefact

Obsah

Úvod **9**
1 Digitálna forenzná analýza **11**
1.1 Digitálna stopa 13
1.2 Forenzné artefakty 15
1.2.1 Master File Table 15
1.2.2 Metadáta 17
1.2.3 Záznamy udalostí 19
1.2.4 Exchangeable image file format (EXIF) dáta 20
1.2.5 Prefetche 22
1.2.6 LNK súbory 23
1.2.7 Thumbcache 23
1.2.8 Shellbagy 23
1.2.9 System Resource Usage Monitor 24
2 Detekcia anomálií **25**
2.1 Povaha vstupných dát 26
2.2 Typy anomálií 26
2.3 Režimy techník detekcie anomálií 27
2.4 Prístupy k detekcii anomálií 28
2.4.1 Štatistická detekcia anomálií 28
2.4.2 Detekcia anomálií na základe klasifikácie 28
2.4.3 Detekcia anomálií technikou najbližšieho suseda 29
2.4.4 Detekcia anomálií na základe zhlukovania 30
2.5 Algoritmy a metódy detekcie anomálií 30
2.6 Podobné práce 35
3 Dataset a predspracovanie dát **38**
3.1 Popis datasetov 39
3.2 Vytvorenie časovej osi 39
3.3 Binarizácia dát 41
4 Automatizácia detekcie anomálií **43**
4.1 Agregácia dát 43
4.2 Metódy 46
4.3 Príprava funkcie 47

4.4 Autoencoder neurónová sieť 53
5 Výsledky 56
5.1 Metriky modelov pre agregačné funkcie 58
5.2 Neurónová sieť 64
5.3 Aplikácia 66
Záver 69

Zoznam ilustrácií

Obr. 1 Porovnanie modelov digitálnej forenznej analýzy [5] 13
Obr. 2 Príklad anomálií v dvojdimenzionálnom datasete [24] 25
Obr. 3 Príklad kolektívnej anomálie [24] 27
Obr. 4 Detekcia anomálií na základe klasifikácie [24] 29
Obr. 5 Architektúra LSTM bunky [36] 34
Obr. 6 Architektúra neurónovej siete autoencoder [38] 34
Obr. 7 Princíp fungovania funkcie automatizácie 49
Obr. 8 Proces získania výsledkov 51
Obr. 9 Agregované dáta zobrazené v aplikácii 67
Obr. 10 Definovanie modelov v aplikácii 67
Obr. 11 Výsledky modelov zobrazené v applikácii 68

Zoznam tabuliek

Tab. 1 Príklady aktualizovania MACB pečiatok pri konkrétnych aktivitách v atribúte SIA 19
Tab. 2 Príklady aktualizovania MACB pečiatok pri konkrétnych aktivitách v atribúte FNA 19
Tab. 3 Príklad EXIF metadát [18] 22
Tab. 4 Tabuľka popisujúca atribúty dát po binarizácii 42
Tab. 5 Porovnanie veľkosti, počtu udalostí a počtu záznamov po agregácii 46
Tab. 6 Výber metód pre detekciu anomálií 46
Tab. 7 Tabuľka popisujúca parametre pre metódy detekcie anomálií 47
Tab. 8 Náčrt hárku vo výstupnom exceli 52
Tab. 9 Náčrt hárku pre metriky úspešnosti modelov 53
Tab. 10 Štruktúra navrhnutej neurónovej siete autoencodera 54
Tab. 11 Pozície inodov po usporiadaní na základe sumárnej hodnoty 56
Tab. 12 Príslušnosť inodov k skupinám naprieč skóre 57
Tab. 13 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie sum. 59
Tab. 14 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie max. 59
Tab. 15 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie mean 60
Tab. 16 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie freq1 61
Tab. 17 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie freq2 62
Tab. 18 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z1 62
Tab. 19 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z2 63
Tab. 20 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z3 64
Tab. 21 Výsledky neurónovej siete 66

Úvod

V dnešnej dobe neustále narastá počet kybernetických útokov. Pri forenznom vyšetrovaní je potrebné sa čo najrýchlejšie dopracovať k relevantným dátam, ktoré obsahujú informácie a naznačujú postup útočníka. Artefakty získané z napadnutého systému sa z veľkej časti skladajú z nerelevantných artefaktov pre forenzné vyšetrovanie konkrétneho prípadu, resp. bezpečnostného incidentu. Cieľom práce je ušetriť čas forenzného analytika (ďalej aj analytika) pri hľadaní významných dát pre forenznú analýzu. Inými slovami, ide o automatizáciu tohto procesu. Nájdené artefakty poskytujú forenznému analytikovi nadhľad nad bezpečnostným incidentom a umožňujú rýchlejšie stanovenie a následne potvrdenie alebo vyvrátenie forenzných hypotéz o bezpečnostnom incidente a činnosti útočníka.

Prvým cieľom práce je analýza forenzných artefaktov v operačnom systéme Windows. Operačný systém si interne zaznamenáva dôležité udalosti vykonané v určitom čase. Sú to udalosti ako prihlásenie používateľa alebo odosielanie emailovej správy. Príklady forenzných artefaktov sú Recycle Bin, Browsers, Windows Error Reporting Forensics, Remote Desktop Protocol (RDP) Cache, LNK Files, Jump Lists, Prefetch Files a Shellbag. Výsledkom prvého cieľa je analýza týchto artefaktov, analýza ich štruktúry a obsiahnutých informácií. Súčasťou tohto cieľa je aj identifikácia vhodnej reprezentácie týchto informácií v dátových rámcoch (dataframeoch) a transformácia dát do podoby vhodnej pre navrhovaný model v treťom cieli.

Prácou forenzného analytika je identifikácia artefaktov, ktoré mu umožnia odhaliť kroky vykonané útočníkom. Tento krok analýzy je časovo náročný. Druhým cieľom práce je naučiť sa postup práce pri identifikácií relevantných digitálnych stôp pri forenznom vyšetrovaní v operačnom systéme Windows a takisto porovnať existujúce prístupy a nástroje používané pri tomto úkone. Výsledkom druhého cieľa je zistiť, ktoré údaje sú dôležité pre identifikáciu stôp, a na základe získaných informácií hľadať možnosti automatizácie.

Tretím cieľom je príprava modelu, ktorý bude automatizovať identifikáciu relevantných digitálnych stôp. V princípe ide o hľadanie neštandardných udalostí vykonávaných v operačnom systéme. Vstupom pre tento model je obraz disku operačného systému Windows so súborovým systémom New Technology File System (NTFS). K danej problematike existuje málo datasetov, ktoré obsahujú obrazy diskov s definovanými digitálnymi stopami útočníka. Možnosťou je trénovať model metódou

bez učiteľa a overenie jeho efektívnosti nad nami pripravenými obrazmi disku operačného systému, v ktorom bol vykonaný útok.

V prvej kapitole práce popisujeme pojem digitálnej forenznej analýzy, kde ďalej vysvetlíme dôležitosť digitálnej stopy, a následne analyzujeme dôležité forenzné artefakty pre expertov v oblasti. V druhej časti je ukázaný vzťah medzi hľadaním relevantných digitálnych stôp vo forenznom vyšetrovaní a detekciou anomálií. Ďalej analyzujeme rôzne techniky a algoritmy detekcie anomálií. Pre prácu je dôležitá časť predspracovanie datasetov. V tretej kapitole sa venujeme povahe dát, procesu ich získania, spracovania, binarizácie a následne agregácie. V poslednej časti ukážeme implementáciu automatizácie riešenia hľadania relevantných digitálnych stôp vo forenznom vyšetrovaní použitím programovacieho jazyka Python a knižníc pyod a Tensorflow.

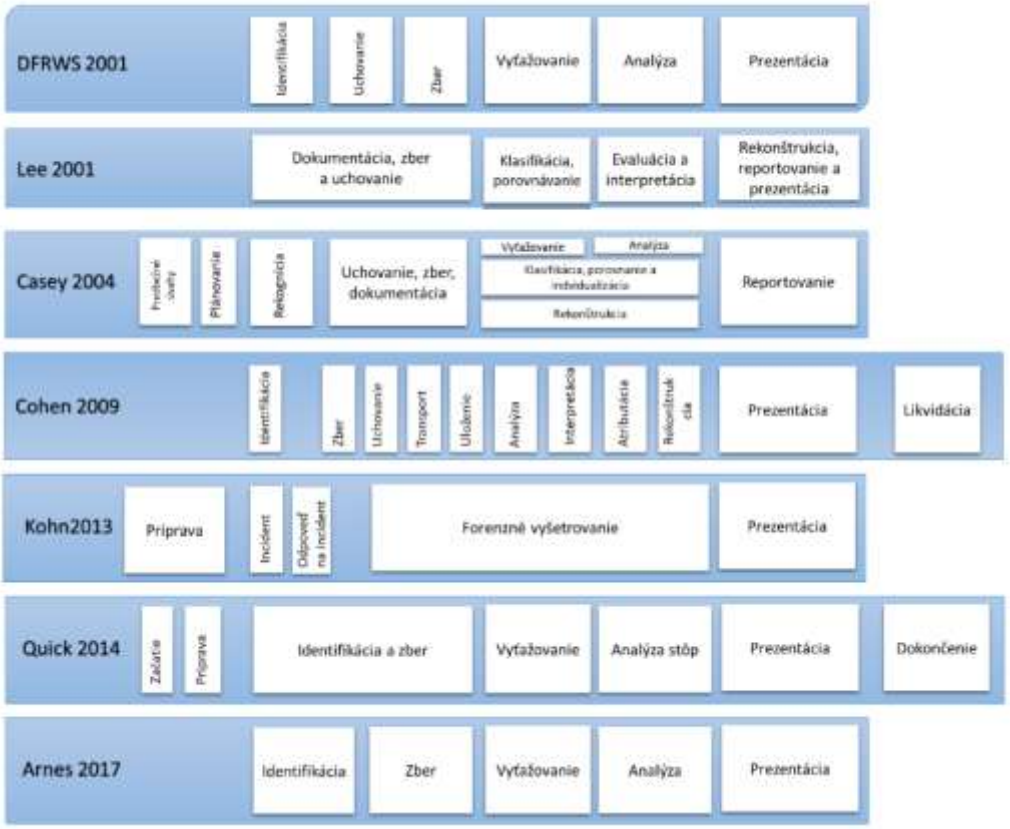
1 Digitálna forenzná analýza

Digitálna forenzná analýza je odvetvie forenzných vied, ktoré na základe vedeckých poznatkov zhromažďuje, analyzuje, dokumentuje a prezentuje digitálne stopy súvisiace s počítačovou trestnou činnosťou s účelom použitia týchto stôp na súde [1]. Podľa [2] je digitálna forenzná analýza takisto metódou systematického vyšetrovania zariadení, systémov, sieťovej komunikácie a pamäťových obrazov. V oblasti riešenia kybernetických bezpečnostných incidentov je hlavným cieľom poskytnúť odpovede na otázky čo sa stalo, kto vykonal útok a kedy bol útok vykonaný. Odvetvie digitálnej forenznej analýzy sa považuje za relatívne nové a stáva sa veľmi dôležitým s narastajúcim počtom trestných činov v kybernetickom priestore. V porovnaní s tradičnejšími forenznými vedami digitálna forenzná analýza nie je vyspelou vedou. Musí sa vysporiadať s častými zmenami vo svete informatiky a operačných systémov. Náročnosť zvyšuje široký záber odvetví, ktorým sa musí digitálna forenzná analýza venovať a prispôsobovať. Sú nimi napríklad právny systém, podnikanie, povaha internetu a presadzovanie práva [1]. Proces digitálnej forenznej analýzy prebieha vo viacerých fázach. Tie definujú postup práce s digitálnymi dôkazmi od ich prvej identifikácie a zaistenia, až po predloženie vedeniu alebo súdnemu konaniu. Existuje viacero modelov, ktoré definujú tento proces, a spravidla fungujú na podobných princípoch. Príkladom je schéma Digital Investigate Framework [3], ktorá obsahuje šesť fáz:

- Identifikácia – Počas incidentu útočník vykonáva aktivitu, ktorá zanecháva stopy. Úlohou je identifikovať, ktoré zdroje môžu obsahovať digitálne stopy po útočníkovi [4].
- Zaistenie – Po identifikácii zdrojov je dôležité zachovať ich stav a súčasne ochrániť ich pred zmenami. Napríklad pri útoku na operačný systém stolového počítača sa vytvorí obraz operačného systému. Pri zaistení digitálnych stôp je potrebné zabezpečiť:
  - korektnosť – získané stopy sú totožné s dátami z pôvodného média,
  - autentickosť – získané stopy pochádzajú z analyzovaného zariadenia/systému/zdroja v danom čase,
  - integrita – získané stopy nesmú byť v čase pozmenené, resp. je ich zmenu možné detegovať,
  - dôvernosť a dostupnosť [2].

- Zber – Pri skúmaní digitálnych dôkazov je dôležité brať ohľad na nestálosť respektíve volatilnosť niektorých dôkazov. V prípade stolových počítačov medzi nestále dôkazy patria napríklad registre, Random-access memory (RAM) a vyrovnávacia pamäť. Je podstatné, aby forenzní analytici brali ohľad na nestálosť niektorých zdrojov pri procese zberu zdrojov. Riešením je zhromažďovať údaje z volatilných zdrojov a presúvať na nevolatilné médium, externý pevný disk.
- Vyťažovanie – V tejto časti analytici používajú konkrétne nástroje a techniky na extrahovanie údajov zo zdrojov. Pri podozrení, že malware napadol operačný systém, analytik vykoná extrakciu určitých informácií z obrazu pamäte operačného systému. V tejto fáze je dôležité dbať na ochranu dôkazov pri manipulácii s nimi. Inak by mohli byť nepoužiteľné ako dôkazový materiál pri súdnom konaní.
- Analýza – Po fáze vyťažovania forenzný analytik analyzuje získané dáta a vzťahy medzi nimi navzájom. Jeho cieľom je nájsť relevantné digitálne stopy zanechané po útoku.
- Prezentácia – Správa digitálnej forenznej analýzy musí byť jasná, stručná a objektívna. Analytik musí spracovať podrobnú dokumentáciu, ktorá zaznamenáva dôležité údaje a postupnosť krokov forenzej analýzy. Dokumentácia je často dôležitá pri súdnom konaní, na ktorom forenzný analytik môže vystupovať ako expert. Pri jednaní musí prezentovať výsledky digitálneho vyšetrovania objektívne, a nemal by vyjadriť svoj názor, pokiaľ to nebude žiadané. Závisí to od expertízy daného analytika [4].

Vo vyššie uvedenom texte sme popísali model Digital Investigate Framework. Obr. 1 ukazuje porovnanie ďalších modelov digitálnej forenznej analýzy.



Obr. 1 Porovnanie modelov digitálnej forenznej analýzy [5]

1.1 Digitálna stopa

Podľa [5] je digitálna stopa ústredným pojmom digitálnej forenznej analýzy. Vo všeobecnosti ide o akékoľvek dáta, ktoré obsahujú spoľahlivé informácie o bezpečnostnom incidente a podporujú alebo vyvracajú stanovené hypotézy o incidente, respektíve o trestnom čine [6]. V nasledujúcich riadkoch je niekoľko pojmov, ktoré pomáhajú zadefinovať digitálnu stopu. Podľa [5] ide o informáciu:

- s výpovednou hodnotou uloženou alebo prenášanou v digitálnej forme,
- uloženú alebo prenášanú v binárnej forme, ktorá môže byť predložená pri súdnom konaní ako vecný dôkaz,
- uloženú alebo prenášanú pomocou počítača, ktorá podporuje alebo vyvracia teóriu o tom, ako došlo k trestnému činu.

V priebehu digitálnej forenznej analýzy je potrebné spracovávať a uchovávať údaje tak, aby sa dodržali zásady integrity stôp a reťazca dôvery [7]. Reťazec dôvery

(chain of custody) je neoddeliteľnou súčasťou každého procesu digitálnej forenznej analýzy. Jeho úlohou je jasne deklarovať, ako boli digitálne stopy získané, objavené, uchovávané, prepravované medzi stranami podieľajúcimi sa na vyšetrovaní a analyzované, teda aké techniky boli použité. Takisto vypovedá, ako sa s dátami zaobchádzalo medzi viacerými stranami zapojenými do vyšetrovania [1]. Existujú dva spôsoby, ako môžeme zaznamenávať a udržiavať reťazec dôvery. Jedným z riešení je využitie služieb od poskytovateľov softvérovo-hardvérových riešení, ktoré automatizujú proces uchovávania dôkazov. Druhou metódou je použitie papierových formulárov. Aj keď sa tento prístup zdá zastaraný a vyžaduje ochranu pred zničením a manipuláciou týchto formulárov, tak pre malé tímy, ktoré nemajú dostatočné finančné prostriedky na automatizáciu tohto procesu, je toto riešenie efektívnejšie z pohľadu nákladov a jednoduchšej implementácie [4]. Správny reťazec dôvery odpovedá na uvedené otázky [1]:

- Čo sú digitálne stopy? (reťazec opisuje získané digitálne dôkazy)
- Kde sa digitálne stopy našli? (tablet, počítač, takisto uvádza stav zariadenia pri získaní stôp, napríklad či bolo zariadenie vypnuté alebo zapnuté)
- Ako boli stopy získané? (popisuje použité nástroje a opatrenia pre zachovanie integrity dát)
- Ako sa digitálne stopy uchovávali, prenášali, čo sa s nimi robilo?
- Ako boli digitálne dôkazy preskúmané? (použité nástroje a techniky pri analýze digitálnych stôp)
- Kedy bol získaný prístup k digitálnym stopám, kým, a na aký účel?
- Ako boli digitálne stopy použité pri vyšetrovaní?

V tejto práci sledujeme digitálne stopy z obrazu operačného systému Windows. Integritu dát zachovávame práve vytvorením obrazu disku, a následne s ním manipulujeme len vo fáze prípravy časového radu udalostí súborového systému a binarizácie týchto údajov. Tieto fázy sú popísané nižšie v kapitole 3. Dataset a predspracovanie dát. Našou úlohou je nájsť medzi získanou evidenciou tie digitálne stopy, ktoré sú potencionálne súčasťou incidentu, a tak pomôcť expertom forenznej analýzy pri vyšetrovaní, stanovení hypotéz a ich následného potvrdenia alebo vyvrátenia.

1.2 Forenzné artefakty

Digitálne forenzné artefakty sú ovplyvnené fyzickým médiom, operačným systémom, súborovým systémom a používateľskými aplikáciami [8]. Každý z týchto faktorov má vplyv na vytváranie a zanechávanie digitálnych stop. Forenzná analýza sa zaoberá skúmaním týchto artefaktov a snaží sa porozumieť predchádzajúcemu správaniu na danom zariadení. Podľa [9] sú forenzné artefakty informácie, ktoré majú hodnotu pre forenzné vyšetrovanie. Často sú forenznými artefaktmi obrázky, slová, dokumenty, ktorých význam a obsah je pomerne jednoducho interpretovateľný. Operačný systém Windows si však uchováva informácie o používaní a aktivitách spôsobom, ktorý je často zaujímavý pre forenzného analytika. Problémom je, že spoločnosť Microsoft poskytuje málo informácií o tom, ako tieto artefakty fungujú. Vedomosti o funkcionalite týchto artefaktov najčastejšie pochádzajú zo skúseností pri ich využívaní na analýzu bezpečnostných incidentov.

1.2.1 Master File Table

Operačný systém Windows používa súborový systém New Technology File System (NTFS). Pri tomto súborovom systéme je dôležité uvedomiť si ideu, že všetko je súbor. NTFS bol vytvorený a cieľom ľahkej škálovateľnosti, bezpečnosti a spoľahlivosti na všetkých úrovniach [10]. Prekonáva obmedzenia svojho predchodcu, a to súborového systému File Alocation Table (FAT). Obmedzenia, s ktorými sa tvorcovia NTFS vysporiadali spočívajú v absencii zoznamov riadenia prístupu k objektom súborového systému, šifrovania, žurnálovania, kompresie, bohatých metadát a ďalších. Ďalšou výhodou, ktorá je prínosná aj pre forenznú analýzu je, že časy súborov v systéme NTFS sú ukladané vo formáte Universal Coordinated Time (UTC) pričom FAT používa lokálny čas operačného systému. NTFS prináša forenznej analýze dva hlavné artefakty, Master File Table (MFT) a alternatívny dátový tok (ADS) [11].

MFT je najdôležitejším súborom v súborovom systéme NTFS. Súbory a adresáre majú svoje záznamy v MFT. Veľkosť takého záznamu je 1024 bajtov. Prvých 16 záznamov v MFT patrí systémovým súborom. Medzi ne patrí aj samotný MFT súbor. Z pohľadu forenznej analýzy je MFT veľmi dôležitý, pretože poskytuje podrobné informácie o súboroch a o ich histórii, vrátane zmazaných súborov a ich metadát. Taktiež umožňuje obnoviť informácie o súboroch a ich aktivitách na disku, čo môže byť kľúčové

pri riešení bezpečnostných incidentov. Záznamy súborov a adresárov v MFT sa skladajú z informácií o [8]:

- názve súboru a prípony,
- veľkosti súboru,
- časových pečiatok (vytvorenia, prístupu, zmeny),
- atribútoch súborov (skrytý, systémový, komprimovaný atď.),
- adresáre, v ktorom je súbor uložený,
- fyzickej polohy súboru na disku (klastre, ofsety).

Každý záznam v MFT má svoju vlastnú štruktúru dát. Záznam zahŕňa aj medzeru, ktorá sa vyskytuje medzi koncom posledného atribútu v zázname a začiatkom nasledujúceho MFT záznamu. Dekódovanie dát v týchto záznamoch je možné vykonať manuálne, ale tento proces je náročný. Zvyčajne sa na získanie týchto dát používajú forenzné nástroje. Informácie v záznamoch MFT sú udržiavané vo forme atribútov, z ktorých každý má svoju vlastnú funkciu a štruktúru. Každý atribút má vlastnú hlavičku, ktorá identifikuje atribút a udáva jeho veľkosť. Atribúty môžu byť

- rezidentné - existujú v rámci daného MFT záznamu,
- nerezidentné - existujú mimo daného MFT záznamu a sú odkazované v rámci záznamu.

Z pohľadu forenznej analýzy sú atribúty Standard Information Attribute (SIA), Filename Attribute (FNA) a Data Attribute najviac prínosné.

SIA je rezidentný atribút identifikovaný pomocou hexadecimálnej sekvencie "\x10\x00\x00\x00". Tento atribút je zdrojom dátumových a časových pečiatok. Informácie o dátumových pečiatkach začínajú na ofsete 24 v atribútovom prúde bajtov, a nasledujúcich 32 bajtov predstavuje dátumové a časové pečiatky vytvorenia súboru, poslednej modifikácie súboru, modifikácie MFT záznamu a posledného prístupu k súboru. Tieto pečiatky sú vo formáte FILETIME. Začínajúc na ofsete 24 v rámci atribútového prúdu (t.j. 23 bajtov po \x10 v identifikátore atribútu), nasledujúcich 32 bajtov tvorí dátumové a časové pečiatky vytvorenia súboru, poslednej modifikácie, modifikácie záznamu MFT a posledného prístupu vo formáte FILETIME. Pečiatky

sú zobrazované systémom Windows a väčšinou forenzných nástrojov v súvislosti s konkrétnym súborom [12].

FNA je rezidentný atribút identifikovaný pomocou hexadecimálnej sekvencie "\x30\x00\x00\x00". Atribút obsahuje informácie o konkrétnom súbore ako je odkaz na jeho nadradený priečinok (podľa čísla záznamu MFT), fyzickú a logickú veľkosť súboru, jeho názov súboru v Unicode formáte, a rovnako ako pri SIA, obsahuje štyri dátumové a časové pečiatky v 64-bitovom formáte. Tieto informácie začínajú na ofsete 32 v atribútovom prúde bajtov. Posledných 32 bajtov tvoria dátumové a časové pečiatky vytvorenia súboru, poslednej modifikácie, modifikácie záznamu MFT a posledného prístupu k súboru. Rozdiel medzi dátumovými a časovými pečiatkami uloženými v SIA a FNA je, že pečiatky uložené v SIA sa aktualizujú pri prístupe a modifikácii súboru používateľom, zatiaľ čo pri FNA sa pečiatky nastavujú pri prvom vytvorení súboru, a zvyčajne nie sú aktualizované pri bežnom používaní systému [12].

Data Attribute je pre vyšetrovateľov dôležitý z dôvodu, že obsahuje buď samotné dáta súboru, ak je jeho veľkosť menšia ako 600 bajtov, alebo obsahuje ukazovateľ na miesto, kde sa obsah súboru nachádza. V prvom prípade ide o rezidentný atribút a v druhom o nerezidentný. Ak je súbor väčší ako 600 bajtov, atribút obsahuje zoznam klastrov pridelených danému súboru. Bez ohľadu na to, či ide o rezidentný alebo nerezidentný atribút, v zázname MFT je identifikovaný hexadecimálnou postupnosťou "\x80\x00\x00\x00" [12].

1.2.2 Metadáta

Jedným z najdôležitejších artefaktov pre forenznú analýzu sú metadáta. Sú to základné informácie o objektoch v operačnom systéme. Pri operačnom systéme Windows so súborovým systémom NTFS sa pre súbory zaznamenávajú metadáta, ktoré obsahujú informácie o tom, kedy bol súbor vytvorený, kedy bol upravený kto súbor vytvoril. Niektoré typy súborov ukladajú metadáta naviac, ako napríklad meno autora pri súboroch balíka Microsoft Office [9].

Metadáta súborového systému zaznamenávajú časové pečiatky aktivít so súbormi. Ide o pečiatky vytvorenia súboru, modifikácie súboru, prístupu k súboru a zmeny súboru. V závislosti od súborového systému ide buď o MAC (Modification, Access, Creation) pečiatku, alebo MACB(Modification, Access, Creation, Birth) [13]. MACB časové

pečiatky sú rozšírením časových pečiaoky MAC. Pri časových pečiatkách MAC jednotlivé písmená predstavujú nasledujúce :

- M – čas poslednej zmeny,
- A – čas posledného prístupu,
- C – čas vytvorenia.

Pri časových pečiatkách MACB, ktorá sú štandardom v súborovom systéme NTFS, sa tieto časové pečiatky nachádzajú v dvoch atribútoch MFT, a to v SIA a FNA, ktoré sme popísali vyššie. Vysvetlenie konkrétnych písmen je [14]:

- M – čas modifikácie. Táto pečiatka sa aktualizuje, ak bol zmenený Data attribute v zázname pre daný súbor. Ak používateľ upraví dokument a zavrie ho, čas modifikácie sa zmení, ak ho len otvorí a zavrie, tak čas modifikácie sa nezmení.
- A – čas prístupu. Táto pečiatka sa môže zmeniť nielen pri otvorení súboru, ale aj keď používateľ umiestni myš na súbor v prieskumníkovi. Takisto sa môže aktualizovať pri systémových činnostiach.
- C – čas zmeny. Pečiatka sa aktualizuje pri zmene metadát súboru, teda napríklad pri zmene záznamu MFT.
- B – čas vytvorenia. Pečiatka sa vzťahuje na čas vytvorenia položky MFT, čas vytvorenia súboru v súborovom systéme..

Pri manipulácii, vytváraní, mazaní a ďalších aktivitách so súbormi, sa atribúty SIA a FNA aktualizujú rôzne. Na základe zdroja [15] v Tab. 1 vidíme, ako sa aktualizujú MACB časové pečiatky pre SIA, a v tabuľke vidíme, ako sa aktualizujú časové pečiatky MACB pre FNA.

Pečiatka	M	A	C	B
Aktivita so súborom	-	-	-	-
File Rename	No change	No change	No change	Change
Local File Move	No change	No change	No change	Change
Volume File Move	No change	Change	Change	No change
File Copy	No change	Change	Change	No change

File Access	No change	No change	No change	No change
File Modify	Change	Change	No change	Change
File Creation	Change	Change	Change	Change
File Deletion	No change	No change	No change	No change

Tab. 1 Príklady aktualizovania MACB pečiatok pri konkrétnych aktivitách v atribúte SIA

Pečiatka	M	A	C	B
Aktivita so súborom	-	-	-	-
File Rename	No change	No change	No change	No change
Local File Move	No change	No change	No change	No change
Volume File Move	Change	Change	Change	Change
File Copy	Change	Change	Change	Change
File Access	No change	No change	No change	No change
File Modify	Change	Change	No change	Change
File Creation	Change	Change	Change	Change
File Deletion	No change	No change	No change	No change

Tab. 2 Príklady aktualizovania MACB pečiatok pri konkrétnych aktivitách v atribúte FNA

Časové pečiatky sa pri digitálnej forenznej analýze využívajú pri sledovaní aktivít vykonávaných v súboroch. Umožňujú vytváranie časových radov, ktoré obsahujú informácie o zmenách nad súbormi.

1.2.3 Záznamy udalostí

V operačnom systéme je dôležité zaznamenávať zmeny v tomto systéme a činnosť vykonávanú v systéme. Akúkoľvek pozorovateľnú udalosť, ktorá sa odohrala v určitom čase v systéme, najmä ak je dôležitá, označujeme ako udalosť. Príkladom udalosti je odoslanie emailovej správy, či prihlásenie používateľa do systému. Na druhej strane bezpečnostná udalosť je akákoľvek viditeľná udalosť v prostredí informačných a komunikačných technológií, ktorá je relevantná pre bezpečnosť. Príkladom bezpečnostnej udalosti je aktualizácia operačného systému, vytvorenie procesu,

resp. vytvorenie nového používateľa. Bezpečnostné udalosti zvyčajne vytvárajú stopy, ktoré sa ukladajú v záznamoch operačného systému. Tieto záznamy systému vieme ďalej analyzovať [14].

Windows event logy sú súbory, do ktorých sa ukladajú udalosti významného charakteru v operačnom systéme Windows. Tento logovací systém zaznamenáva napr. chybové hlásenia z aplikácií, ale aj samotného operačného systému. Podľa [16] delíme typ event logov do piatich kategórií:

- **Error** - Tento typ udalosti naznačuje významný problém, ako je strata dát alebo strata funkčnosti. Napríklad, ak sa služba nedokáže načítať počas spustenia, zaznamená sa udalosť typu Error.
- **Warning –** Táto udalosť nie je nevyhnutne významná, ale môže naznačovať možný budúci problém. Napríklad, ak je nízky diskový priestor, zaznamená sa udalosť typu Warning. Ak sa aplikácia dokáže zotaviť z udalosti bez straty funkčnosti alebo dát, označujeme ju ako udalosť typu Warning.
- **Information -** Tento typ udalosti popisuje úspešnú prevádzku aplikácie, ovládača alebo služby. Príkladom je úspešne načítanie sieťového ovládača a následne zaznamenanie udalosti typu Information. Avšak v prípade desktopových aplikácií nie je vhodné, aby aplikácia zaznamenávala udalosť pri každom svojom spustení.
- **Success Audit** - Tento druh udalosti zaznamenáva úspešný pokus o auditovaný prístup. Napríklad, úspešný pokus používateľa o prihlásenie do systému sa zaznamenáva ako udalosť Success Audit.
- **Failure Audit –** Udalosť Failure Audit zaznamenáva neúspešný pokus o auditovaný prístup k zabezpečeniu. Ak sa používateľ pokúša o prístup k sieťovému disku a zlyhá, pokus sa zaznamenáva ako udalosť Failure Audit.

1.2.4 Exchangeable image file format (EXIF) dáta

EXIF dáta sú metadáta uložené v obrázkoch. Forenzné vyšetrovanie často hľadá relevantné obrázky súvisiace s incidentom, a po ich nájdení je dôležité určiť, kde, kedy a kým boli obrázky vytvorené. Tieto informácie sa získajú analýzou EXIF údajov [9]. Prvý štandardný formát Joint Photographic Experts Group (JPEG) bol definovaný v roku 1992 s cieľom umožniť zdieľanie bitových tokov JPEG medzi rôznymi aplikáciami a platformami. V roku 1998 bola technika vylepšená na nový štandard EXIF. Technika

umožnila vkladať metadáta do JPEG a TIFF obrázkov [17]. V Tab. je uvedený typický príklad EXIF metadát v čitateľnej forme [18].

File name	0805-153933.jpg
File size	463023 bytes
File date	2001:08:12 21:02:04
Camera make	Canon
Camera model	Canon PowerShot S100
Date/Time	2001:08:05 15:39:33
Resolution	1600 x 1200
Flash used	No
Focal length	5.4mm (35mm equivalent: 36mm)
CCD Width	5.23mm
Exposure time	0.100 s (1/10)
Aperture	f/2.8
Focus Dist.	1.18m
Metering Mode	center weight
Jpeg process	Baseline

Tab. 3 Príklad EXIF metadát [18]

1.2.5 Prefetche

Prefetching v rámci terminológie operačného systému Windows je proces, pri ktorom ide o sledovanie bežného používania aplikácie a načítanie údajov, ktoré aplikácia zvyčajne potrebuje počas behu alebo keď sa aplikácia načítava. Implementácia tohto procesu zabezpečila zvýšenie výkonu aplikácií, ktoré fungujú podobným spôsobom pri každom použití. Údaje sa ukladajú v súboroch prefetch. Ich umiestnenie je v priečinku „Prefetch“ v koreňovom adresári systému, najčastejšie v „C:\Windows“. Z forenzného hľadiska sa súbory prefetch používajú na analýzu informácií o tom, koľkokrát bol súbor spustený a kedy bol spustený naposledy [9].

1.2.6 LNK súbory

LNK súbory sú v podstate skratky v operačnom systéme Windows. Tieto súbory sa vytvárajú aj v prípade, keď používateľ otvorí lokálny alebo vzdialený súbor [19]. Ak používateľ otvoril súbor s názvom example.txt, názov vytvoreného .LNK súboru bude example.txt.lnk [9]. Tieto súbory sú vynikajúcimi artefaktmi pre forenznú analýzu, vďaka tomu, že aj keď hľadané súbory nemusia v systéme existovať, tak .LNK súbory spojené s pôvodným súborom budú stále existovať a pomôžu odhaliť informácie o tom, čo sa v systéme vykonalo [19]. Poskytujú informácie o predchádzajúcich aktivitách používateľa, veľkosti, pôvodnej ceste, sériovom čísle a názve zväzku prepojeného súboru, a o atribútoch času MAC (Modified, Accessed, Created), ktoré hovoria o tom, kedy bol súbor naposledy modifikovaný, pristupovaný alebo vytvorený [1].

1.2.7 Thumbcache

Vďaka funkcionalite Thumbcache sa znižuje potreba čítania všetkých obrázkov a ich následnej premeny na miniatúry pri každom zobrazení priečinka. Miniatúry obrázkov sa tvoria pri prvom vytvorení originálnych súborov, a ukladajú sa do databátových súborov s názvom thumbcaches. Z pohľadu fornenznej analýzy sú tieto miniatúry prínosné vďaka tomu, že aj keď pôvodný súbor bol odstránený zo systému, tak miniatúry ostávajú uložené v databáze [9].

1.2.8 Shellbagy

Shellbagy sú k dispozícii už od operačného systému Windows XP. Shellbagy sa používajú na ukladanie informácií o nastaveniach grafického rozhrania Prieskumníka, ktorý sa používa na prehľadávanie súborov a priečinkov v počítači so systémom Windows. Len nedávno sa stali populárnym artefaktom. Význam týchto artefaktov spočíva v tom, že shellbag sa vytvára pre určitý priečinok vtedy, ak ho používateľ skutočne prezeral. Pri ukladaní týchto vlastností sa vytvárajú ďalšie artefakty, ktoré obsahujú informácie o prehľade priečinku, histórii prehliadania priečinku ale aj podrobnosti o priečinkoch, ktoré boli zo systému odstránené [20]. Shellbagy sa teda môžu použiť na rekonštrukciu súborového systému chýbajúcich zariadení alebo vymazaných priečinkov umožňujú vytvorenie časovej osi miest, ktoré používateľ navštívil v systéme [21].

1.2.9 System Resource Usage Monitor

Informácie súvisiace s procesmi a využívanie siete spojené s procesmi sa priebežne zaznamenávajú pomocou mechanizmu System Resource Usage Monitor (SRUM). Informácie, ktoré SRUM priebežne zaznamenáva sú napríklad:

- podrobnosti o procese, ako je napríklad celá cesta procesu,
- informácie o vlastníkovi procesu,
- metriky pre I/O dáta (bajty čítaných a zapisovaných v popredí a na pozadí),
- využitie cyklov CPU (popredie/pozadie),
- kontextové prepínače,
- využitie siete procesom, sledovanie odoslaných a prijatých dát,
- Windows Push notifikácie [22].

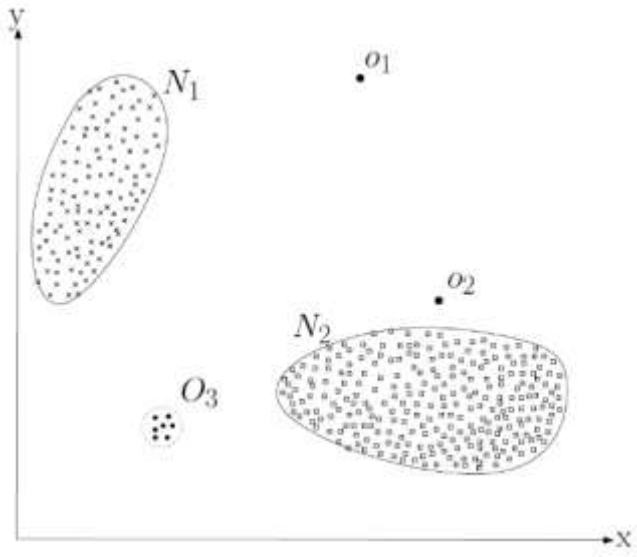
2 Detekcia anomálií

V našej práci pristupujeme k analýze forenzných artefaktov a následne hľadaniu relevantných digitálnych stôp pomocou detekcie anomálií. V prípade útoku na operačný systém je s veľkou pravdepodobnosťou aktivita útočníka anomáliou, teda neštandardnou aktivitou. V ideálnom prípade je množina aktivít, respektíve udalostí, detegovaných v systéme ako anomália rovnaká, ako množina aktivít útočníka. Teda predpokladáme, že väčšina anomálnych udalostí nájdených v systéme boli vykonané útočníkom [23].

Prístup k detekcii anomálií zvyčajne pozostáva z dvoch fáz: fázy trénovania a fázy testovania. V prvom kroku je definovaný normálny profil prevádzky, zatiaľ čo v druhom kroku sa naučený profil aplikuje na nové dáta [23].

Detekcia anomálií sa týka problému hľadania vzorov v dátach, ktoré nesplňujú očakávané správanie. Tieto nezvyčajné vzory sa často označujú ako anomálie, odchýlky, rozporuplné pozorovania, výnimky, prekvapenia alebo kontaminanty v dátach naprieč rôznymi odvetviami. Detekcia anomálií nachádza široké uplatnenie, napríklad pri detekcii podvodov s kreditnými kartami, poistení, zdravotnej starostlivosti, detekcii prienikov v kybernetickej bezpečnosti, detekcii chýb v systémoch kritických pre bezpečnosť a vojenský dohľad nad nepriateľskými aktivitami [24].

Postupne bolo vyvinutých množstvo techník detekcie anomálií v rôznych výskumných oblastiach. Mnohé z týchto techník boli špecificky vyvinuté pre určité aplikačné oblasti, zatiaľ čo iné sú univerzálnejšie [24]. Obr. 2 ukazuje jednoduchý príklad anomálií v dvojdimenzionálnom datasete.



Obr. 2 Príklad anomálií v dvojdimenzionálnom datasete [24]

2.1 Povaha vstupných dát

Kľúčovým aspektom akejkoľvek techniky detekcie anomálií je povaha vstupných dát. Vstupom je zvyčajne kolekcia dátových inštancií (nazývaných tiež objekt, záznam, bod, vektor, vzor, udalosť, prípad, vzorka, pozorovanie alebo entita). Každá dátová inštancia môže byť opísaná pomocou sady atribútov (tiež nazývaných premenná, charakteristika, vlastnosť, pole alebo dimenzia). Atribúty delíme na:

- binárne,
- kategorické a
- spojité.

Každá dátová inštancia môže pozostávať z jedného alebo viacerých druhov atribútov. Povaha atribútov určuje vhodnosť techník detekcie anomálií. Napríklad pre štatistické techniky sa musia použiť rôzne štatistické modely pre spojité a kategorické dáta. Podobne, pre techniky založené na najbližších susedoch, povaha atribútov by určovala mieru vzdialenosti, ktorá sa má použiť.

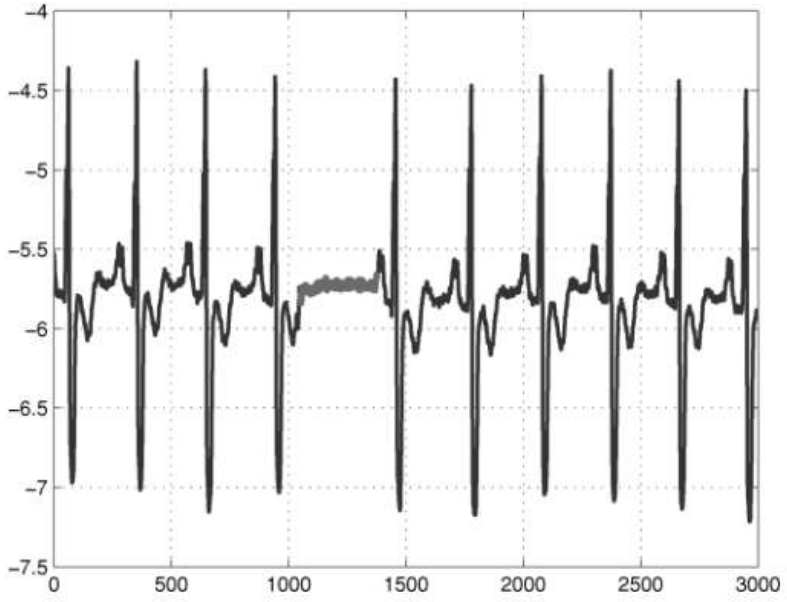
Všeobecne platí, že dátové inštancie môžu byť navzájom prepojené. Niektoré príklady sú sekvencie dát, priestorové dáta a grafy. V sekvenciách dát sú dátové inštancie lineárne usporiadané, napríklad časové rady [24].

2.2 Typy anomálií

Dôležitým aspektom pri výbere techniky detekcie anomálií je povaha a vlastnosti hľadanej anomálie. Anomálie delíme na nasledujúce kategórie.

- **Bodové anomálie** sú anomálie, kedy jednotlivú inštanciu údajov môžeme považovať za anomálnu vzhľadom na zvyšok dát. Príklad vidíme na Obr. 2, kde body o_1, o_2 a body z oblasti O_3 ležia mimo hranice normálnych oblastí a teda ide o bodové anomálie.
- **Kontextové anomálie** sú tie inštancie údajov, kde je daná inštancia údajov anomálna v konkrétnom kontexte, ale nie inak.
- **Kolektívne anomálie** - ak je zbierka súvisiacich údajových inštancií anomálna vzhľadom na celý súbor údajov, tak táto zbierka sa nazýva kolektívna anomália. Jednotlivé údaje v kolektívnej anomálii nemusia byť samy o sebe anomáliou,

ale ich výskyt spolu, ako zbierka, je anomálny. Obr. 3 ukazuje príklad kolektívnej anomálie. Úsek medzi hodnotami 1000 a 1500 na x-ovej osi je definovaný ako kolektívna anomália, keďže sa v rámci periodickej pravidelnosti úplne vymyká štandardnému správaniu [24].



Obr. 3 Príklad kolektívnej anomálie [24]

2.3 Režimy techník detekcie anomálií

Ohodnotenie datasetu a všetkých jeho inštancií na anomálne alebo štandardné je časovo náročný proces. Ohodnotenie digitálnych stôp je vykonávané odborníkom manuálne, a vyžaduje značné úsilie. Techniky detekcie anomálií môžu pracovať v jednom z nasledujúcich troch režimov:

1. **Detekcia anomálií s učiteľom** – trénovanie prebieha v režime s dozorom, kde predpokladáme existenciu trénovacej dátovej sady, ktorá obsahuje označené inštancie pre normálnu aj anomálnu triedu.
2. **Detekcia anomálií bez učiteľa** - techniky, ktoré nepotrebujú trénovacie dáta a sú tak najširšie použiteľné. Techniky v tejto kategórii počítajú s implicitným predpokladom, že normálne inštancie sú v testovacích dátach prítomnejšie oveľa častejšie ako anomálie. Ak tento predpoklad nie je splnený, tak tieto techniky majú vysokú mieru falošných označení anomálnych vzoriek.

3. **Kombinovaná detekcia** anomálií – techniky, ktoré pracujú v polovičnom režime s dozorom a bez dozoru a predpokladajú, že trénovacia dátová sada obsahuje označené inštancie len pre normálnu triedu. Vzhľadom k tomu, že nepotrebujú označenia pre anomálnu triedu, sú širšie použiteľné ako techniky s dozorom [24].

2.4 Prístupy k detekcii anomálií

V tejto podkapitole preskúmame rôzne architektúry a metódy, ktoré boli navrhnuté na detekciu anomálií. Patria tu štatistická detekcia anomálií, detekcia anomálií na základe klasifikácie, detekcia anomálií na základe techník najbližšieho suseda a detekcia na základe zhlukovania.

2.4.1 Štatistická detekcia anomálií

Štatistická detekcia anomálií sa používa na identifikáciu nezvyčajných vzorov v dátach. Pri tejto metóde pozorujeme správanie subjektov a pri tom vytvárame takzvaný profil subjektu. Profil zahŕňa rôzne štatistické merania, ako je intenzita aktivity, distribúcia záznamov auditu, kategorické merania (rozloženie aktivity do kategórií) a ordinálne merania (napríklad využitie CPU).

Ako systém spracúva udalosti, aktuálny profil sa aktualizuje a pravidelne sa vypočíta skóre anomálie. Toto skóre vyjadruje mieru nepravidelnosti konkrétnej udalosti a porovnáva sa s uloženým profilom pomocou danej funkcie, ktorá vyjadruje abnormalitu. Toto môžeme vyjadriť jednoducho, napríklad euklidovskou vzdialenosťou. Ak je skóre anomálie vyššie ako stanovená hranica, generuje sa upozornenie, že bola zaznamenaná anomálna aktivita.

Štatistická detekcia anomálií je teda založená na porovnávaní aktuálnych dát so štatistickými profilmi a identifikuje nezvyčajné vzory, ktoré môžu naznačovať prítomnosť anomálií v systéme alebo sieti [23].

2.4.2 Detekcia anomálií na základe klasifikácie

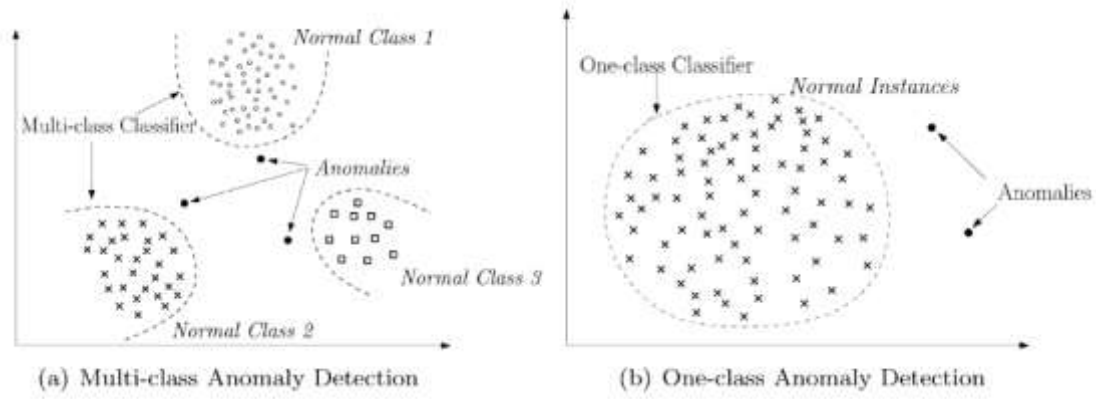
Techniky založené na klasifikácii na detekciu anomálií fungujú v dvojfázovom režime. Trénovacia fáza naučí klasifikátor pomocou dostupných označených trénovacích

dát. Testovacia fáza potom klasifikuje testovaciu inštanciu ako normálnu alebo anomálnu pomocou klasifikátora.

Techniky založené na klasifikácii na detekciu anomálií pracujú na základe nasledujúceho všeobecného predpokladu. V danom priestore príznakov je možné naučiť klasifikátor rozlíšovať medzi normálnymi a anomálnymi triedami.

Na základe dostupných označení pre trénovaciu fázu môžeme techniky detekcie anomálií založené na klasifikácii rozdeliť do dvoch širokých kategórií: viac-triedne a jedno-triedne techniky detekcie anomálií.

Techniky detekcie anomálií založené na viac-triednej klasifikácii predpokladajú, že trénovacie dáta obsahujú označené inštancie patriace do viacerých normálnych tried. Takéto techniky detekcie anomálií naučia klasifikátor rozlíšiť medzi každou normálnou triedou a zvyškom tried. Techniky detekcie anomálií založené na jedno-triednej klasifikácii predpokladajú, že všetky trénovacie inštancie majú iba jedno označenie triedy [24]. Príkladmi takejto detekcie sú neurónové siete, Bayesovské siete, SVM a prístup založený na pravidlách.



Obr. 4 Detekcia anomálií na základe klasifikácie [24]

2.4.3 Detekcia anomálií technikou najbližšieho suseda

Techniky detekcie anomálií založené na najbližších susedoch vyžadujú definovanie vzdialenosti alebo podobnosti medzi dvoma dátovými inštanciami. Vzdialenosť (alebo podobnosť) medzi dvomi dátovými inštanciami môže byť vypočítaná rôznymi spôsobmi. Pre spojité atribúty je populárnou voľbou Euklidovská vzdialenosť, ale je možné použiť aj iné metriky. Pre kategorické atribúty sa často používa jednoduchý koeficient zhody.

Väčšina techník nevyžaduje, aby vzdialenosť bola prísne metrická. Metriky obvykle musia byť symetrické, ale nie je potrebné, aby splňovali trojuholníkovú nerovnosť.

Techniky detekcie anomálií založené na najbližších susedoch možno široko rozdeliť do dvoch kategórií:

- Techniky, ktoré používajú vzdialenosť dátovej inštancie k jej *k*-tej najbližšej inštancii ako skóre anomálie.
- Techniky, ktoré vypočítajú relatívnu hustotu každej dátovej inštancie, na základe ktorej vypočítajú skóre anomálie [24].

2.4.4 Detekcia anomálií na základe zhlukovania

Zhlukovanie sa používa na triedenie podobných inštancií dát do takzvaných zhlukov. Jedná sa techniku bez dozoru. V poslednej dobe sa skúmal aj semi- unsupervised prístup zhlukovania. Aj napriek tomu, že zhlukovanie a detekcia anomálií sa zdajú byť zásadne odlišné, bolo vyvinutých niekoľko techník detekcie anomálií založených na zhlukovaní. Tieto techniky detekcie anomálií na základe zhlukovania možno rozdeliť do troch kategórií.

- Normálne inštancie v dátach patria do nejakého zhluku, zatiaľ čo anomálie nepatria do žiadneho zhluku.
- Normálne dátové inštancie ležia blízko ich najbližšieho ťažiska zhluku, zatiaľ čo anomálie sú ďaleko od ich najbližšieho ťažiska klastra.
- Normálne inštancie údajov patria do veľkých a hustých klastrov, zatiaľ čo anomálie patria buď do malých alebo riedkych klastrov [24].

2.5 Algoritmy a metódy detekcie anomálií

V našej práci sa zaoberáme viacerými metódami detekcie anomálii. V priebehu rokov boli navrhnuté viaceré algoritmy hľadania anomálií bez učiteľa. V tejto časti bližšie popíšeme dôležité typy algoritmov na detekciu anomálií nad tabuľkovými dátami.

Algoritmy založené na základe blízkosti pracujú na základe informácií o lokálnom okolí každého bodu. Metódy založené na blízkosti sú väčšinou neparametrické, teda nepredpokladajú parametrické rozdelenie údajov. Ich výhodou je,

že pre ich použitie nie je potrebná dôkladná znalosť pravdepodobnostného rozdelenia datasetu. Metódy tohto typu sú výpočtovo náročné, a je pri nich náročné určiť správnu funkciu vzdialenosti a počet susedov [25]. Tento typ detekcie anomálií delíme na algoritmy založené na vzdialenosti a hustoty.

Algoritmy založené na princípe blízkosti porovnávajú body v dátach s ich susedmi. Body, ktoré sú veľmi vzdialené od svojich susedov sú označené ako anomálie [25]. Skóre anomality bodu je, v tomto prípade, najčastejšie určené vzdialenosťou ku *k*-tému najbližšiemu susedovi. Používajú sa aj iné variácie tohto prístupu, a to priemerná vzdialenosť *k*-najbližších susedov k bodu [26].

Pri algoritmoch založených na blízkosti rozlišujeme viaceré metriky vzdialenosti. Vzdialenosť bodov x, y bodu označme ako $D(x, y)$.

- City-block (Manhattan) metrika

$$D_C(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

- Euklidovská metrika

$$D_E(x, y) = \sum_{i=1}^{n} \sqrt{(x_i - y_i)^2}$$

- Chebyshevova metrika

$$D_{Ch}(x, y) = \max_{i=1}^{n}(|x_i - y_i|)$$

- Canberrova metrika [27]

$$D_{Can}(x, y) = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}$$

- Jaccardova metrika [28]

$$D_{Jacc}(x, y) = \frac{|x \cap y|}{|x \cup y|}$$

- Cosínusová metrika

$$\mathrm{D}_{cos} = 1 - \frac{\sum_{i=1}^{n} \frac{x_i y_i}{|x_i| + |y_i|}}{\sqrt{\sum_{i=1}^{n} x_i^2}\ \sqrt{\sum_{i=1}^{n} y_i^2}}$$

- Minkovského metrika [27]

$$D_{M,p} = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$$

- Sqeuklidovská metrika [29]

$$D_{SqE}(x, y) = \sum_{i=1}^{n} (x_i - y_i)^2$$

- Braycurtisova metrika [30]

$$D_{SqE}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \ / \sum_{i=1}^{n} |x_i + y_i|$$

Medzi metódy založené na princípe blízkosti patria:

- **Local outlier factor** (LOF) metóda je založená na hustote v okolí objektu a na vzdialenostiach k susedom. Na začiatku metóda počíta lokálne odľahlé hodnoty pre každý dátový bod, ktoré vyjadrujú stupeň odľahlosti každého bodu. Na základe týchto hodôt určuje anomalitu jednotlivých dátových bodov Metódu je možné použiť v režime trénovania bez učiteľa [31].
- **Subspace Outlier Detection** (SOD) pre každý dátový bod v datasete skúma osovo-paralelný podpriestor obsiahnutý jeho susedmi a určuje, ako veľmi sa objekt odchyľuje od susedov v tomto podpriestore. Metóda funguje aj v režime bez učiteľa, a dobre sa škáluje na datasetoch s vysokým počtom parametrov [32].
- **Rotation-based Outlier Detection** (ROD) je založená na rozklade celého atribútového priestoru na rôzne kombinácie atribútov. Trojdimenzionálne vektory reprezentujú dátové body v každom trojdimenzionálnom podpriestore. Tieto vektory sa otáčajú okolo geometrického mediánu pomocou Rodriguesovho rotačného vzorca, aby sa vytvorilo celkové skóre odľahlých hodnôt. Metóda na vstupe neočakáva žiadne parametre a ľahko sa implementuje [33].

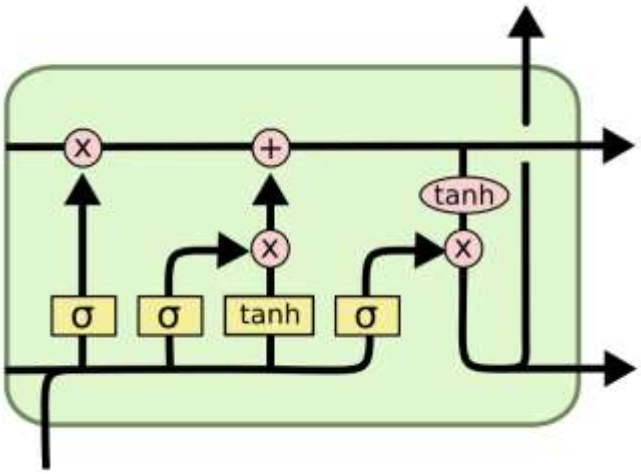
Myšlienka pri algoritmoch založených na **hustote** je, že anomálie vieme nájsť v regiónoch s nízkou hustotou. Podobné objekty so štandardným správaním sa nachádzajú v oblastiach s vysokou hustotou [26].

Pravdepodobnostné algoritmy sa snažia zovšeobecniť metódy analýzy extrémnych hodnôt v mnohorozmernej analýze. Metóda analýzy viacrozmerných extrémnych hodnôt založená na vzdialenosti Mahalanobis, môže byť chápaná ako model Gaussovej zmesi s jedinou zložkou v zmesi. Ak tento model zovšeobecníme na viaceré zložky zmesi, môžeme identifikovať odľahlé hodnoty, ktoré sú všeobecnejšie než extrémne hodnoty v rámci viacerých premenných [26] Medzi metódy založené na pravdepodobnostných algoritmoch patria napríklad:

- **Copula Based Outlier Detector** (COPOD) patrí k režimu detekcie anomálií bez učiteľa. Metóda je inšpirovaná kopulami na modelovanie viacrozmerného rozdelenia údajov. Na začiatok metóda skonštruuje empirickú kupolu, a následne ju použije na predpovedanie pravdepodobnosti chvosta pre každý dátový bod s cieľom určiť jeho úroveň abnormality. Je deterministická a založená na empirickej kumulatívnej distribučnej funkcii. [34].
- **Empirical-Cumulative-distribution-based Outlier Detection** (ECOD) najprv odhaduje základné rozdelenie vstupných údajov neparametrickým spôsobom na základe výpočtu empirického kumulatívneho rozdelenia pre dimenzie. ECOD potom používa tieto empirické rozdelenia na odhad pravdepodobností chvostov naprieč dimenziami pre každý dátový bod. Nakoniec vypočíta skóre abnormality každého dátového bodu agregovaním odhadnutých pravdepodobností chvostu naprieč dimenziami. Určenie anomálií je bez učiteľa [25].

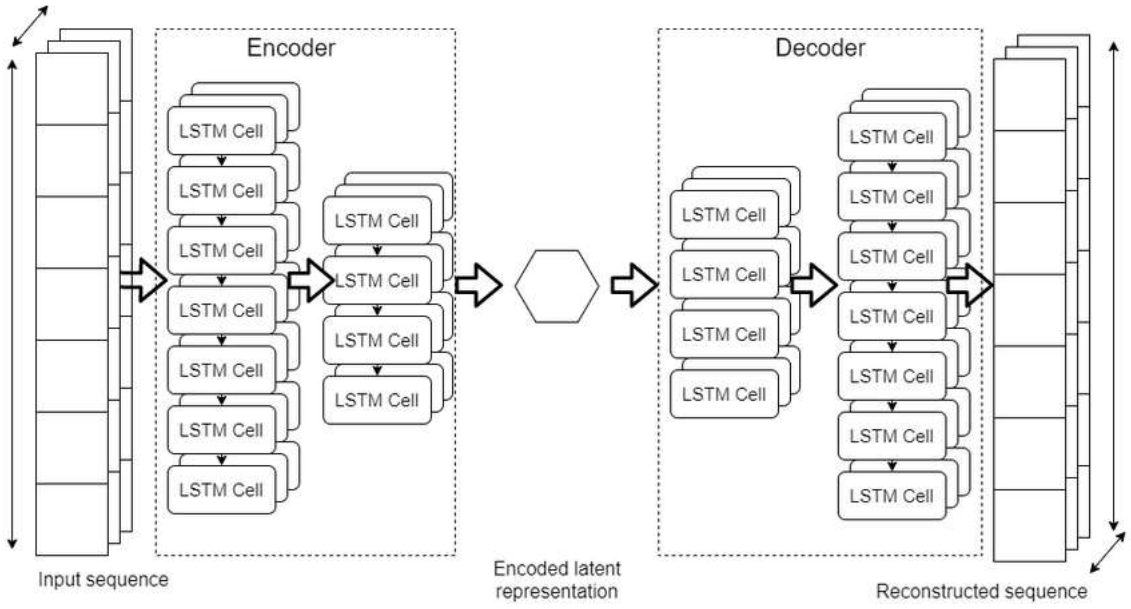
Metódy založené na základe **neurónových sietí** sa v poslednej dobe tešia veľkej obľube. Jedným z príkladov je použitie autoencodera. Neurónová sieť sa skladá z dvoch hlavných častí, encodera a decodera. Použitie tohto konceptu je postavené na myšlienke, že encoder a decoder sú trénované spoločne. Porovnaním výstupu modelu voči očakávanému objektu vieme určiť jeho anomalitu. Čím menej je daný objekt podobný výstupu, tým viac je anomálny [35].

LSTM (Long Short-Term Memory) je typ rekurentnej neurónovej siete, ktorá efektívne pracuje s dlhodobými závislosťami v dátach a rieši tak problém miznúceho gradientu pri trénovaní hlbokých neurónových sietí. LSTM vrstva obsahuje bunky pamäte, ktoré jej umožňujú rozhodovať, ktoré informácie, z predchádzajúcich časových krokov, si má zapamätať a ktoré zabudnúť. Tým sa zlepšuje schopnosť modelu pracovať s dlhodobými závislosťami v časových radách.



Obr. 5 Architektúra LSTM bunky [36]

Prínosom encodera a decodera s LSTM vrstvami je schopnosť efektívne pracovať s dátami časových radov a generovať predikcie na základe zachytených vzťahov medzi vstupnými a výstupnými sekvenciami. Tieto modely sú často využívané v oblasti strojového prekladu, generovania textu, predikcii časových radov, ale aj detekcie anomálií [37]. Na **Chyba! Nenašiel sa žiaden zdroj odkazov.** je ukázaná architektúra autoencodera s použitím LSTM vrstiev.



Obr. 6 Architektúra neurónovej siete autoencoder [38]

Algoritmy na detekciu anomálií založené na **ensemblových technikách** kombinujú viaceré detektory anomálií. Kombinácia rôznych detektorov je výhodná, ak nemajú rovnakú chybu. Tieto algoritmy majú viaceré výhody vďaka mnohým detektorom. Môžu byť robustnejšie voči rôznorodej povahe dát [39], [40]. Príklady metód založené na ensemblových technikách sú nasledujúce:

- **Isolation Forest** (IForest) konštruuje stromovú štruktúru z dát a izoluje objekty v dátach tak, že anomálie sú izolované bližšie ku koreňu stromu, a štandardné objekty sú izolované v hlbšej časti stromu [41].
- **Isolation-based Anomaly Detection Using Nearest-Neighbor Ensembles** (INNE) metóda sa snaží vylepšiť nedostatky Iforestu pri odhaľovaní lokálnych anomálií, anomálií s vysokým percentom nerelevantných atribútov, maskovaných anomáliách osovo paralelnými zhlukmi a anomáliách v multimodálnych datasetoch. Metóda je rýchla, jej časová zložitosť je lineárna a priestorová zložitosť je konštantná [42].
- **Lightweight on-line detector of anomalies** (LODA) dokáže sledovať atribúty, v ktorých sa objekt odlišuje od ostatných objektov. Táto vlastnosť je užitočná, ak potrebujeme zistiť, čo anomáliu spôsobuje, respektíve ktoré atribúty sú neštandardné [43].

2.6 Podobné práce

V rámci tejto časti práce sa bližšie zameriame na odborné a vedecké práce, ktoré sa venujú automatizácii digitálnej forenznej analýzy, resp. aplikovania dátovej analýzy alebo strojového učenia do digitálnej forenznej analýzy.

Automatizáciu je v digitálnej forenznej analýze môžeme považovať za nevyhnutnosť, keďže stále narastá väčší počet bezpečnostných incident, ktoré je nutné riešiť. Výskumníci a výskumné skupiny sa dlhodobo snažia urýchliť tento proces. Autori Du a Scanlon vo svojom článku [44] navrhujú metodológiu na automatické uprednostňovanie podozrivých forenzných artefaktov súborového systému, aby sa týmto znížila potreba manuálnej analýzy. Autori vo svojom článku predstavili aj sadu nástrojov na extrahovanie údajov z obrazov diskov. Iná výskumná skupina pod vedením Skopika študovala metódy analýzy údajov protokolu pomocou algoritmov strojového učenia na detekciu online anomálií [45]. Beebe a kol. zistili, že škodlivú aktivitu útočníkov

z prostredia organizácie (insiderov) v rámci zariadení možno identifikovať analýzou údajov a forenzných artefaktov súborového systému. Ich výskum poskytol výpočtový prístup na nájdenie digitálnych forenzných stôp. Následná manuálna analýza potvrdila väčšinu zistených anomálií [46].

Vyššie sme uviedli niekoľko vedeckých prístupov k automatizácii digitálnej forenznej analýzy ako celku. Nižšie sa budeme venovať konkrétnym odborným a vedeckým prácam podľa typu spracovaných údajov, a to artefaktom súborového systému a registru operačného systému Windows.

Prvú skupinu odborných a výskumných prác tvoria práce zamerané na zisťovanie anomálií v súborových systémoch. Tieto práce vychádzajú z predpokladu, že väčšina digitálnych stôp je uložená v súborovom systéme operačného systému. V roku 2005 Carrier a spol. preskúmali možnosť automatizácie vyhľadávania súborov a adresárov vytvorených počas bezpečnostného incidentu [47]. Iným príkladom je práca Pirkera a kol., ktorá sa zamerala na odhaľovanie nových útokov porovnaním správania sa operačného systému. Použili kombináciu zberu udalostí a analýzy procesov a ich správania sa pri prístupe k súborom [48]. Iným príkladom je výskumná skupina autora Du, ktorá zaviedla metódu hodnotenia relevantnosti forenzných artefaktov súborového systému. Tá sa opiera o centralizovaný model DFaaS. Využitím relevantných súborov, s ktorými sa vyšetrovateľ stretol pri predchádzajúcich svojich vyšetrovaniach, dokáže ich metóda klasifikovať novoobjavené súbory [49].

Druhá skupina odborných a výskumných prác sa zameriava na detekciu anomálií vregistri operačného systému Windows. Stolfo a kol. sa pokúsili zistiť nezvyčajnú aktivitu v registri operačného systému Windows pomocou metódy Support Vector Machines [50]. Ďalším príkladom je práca [51], v ktorej autori navrhli prístup nazvaný RAMD, ktorý používa súborový klasifikátor na detekciu škodlivého softvéru, ktorý zneužíva kľúče registra operačného systému Windows. Cieľom výskumnej skupiny pod vedením Chouhan bolo tiež odhaliť anomálne správanie v rámci procesov skúmaním anomálií v registroch operačného systému Windows, DLL knižniciach a prístupoch k súborom procesu [52].

Okrem vyššie uvedených výstupov, sú k dispozícii aj odborné a vedecké články, ktorých obsahom je detekcia anomálií v operačnom systéme Windows. Xu a kol. sa stretli s výzvou analyzovať protokoly konzoly a aplikovať metódy detekcie anomálií, pričom metóda PCA priniesla dobré výsledky [53]. Iná výskumná skupina pod vedením

Studiawana zdôraznila význam protokolov udalostí ako cenného zdroja digitálnych stôp pre forenzné vyšetrovania, keďže zaznamenávajú základné systémové aktivity [54]. Studiawan a kol. navrhli metódu klastrovania protokolov riadenia prístupu pomocou algoritmu MajorClust a identifikácie anomálií [55]. V roku 2020 tá istá výskumná skupina [56] vyhodnotili výkon siedmich detektorov anomálií. Vo svojom výskume z roku 2021 sa títo výskumníci zamerali na detekciu anomálií pomocou hlbokých automatických kódovačov [57]. Iní výskumníci tiež použili hlboké automatické kódovače na detekciu anomálií, vrátane výskumnej skupiny pod vedením Liua. [58]. Yuan a spol. použili automatické kódovače na detekciu anomálnych používateľov [59]. V inom výskume He a kol. preskúmali a diskutovali o troch metódach detekcie anomálií bez a s učiteľom [60]. Iným príkladom je článok [61] od Hirakawa a kol., ktorí navrhli metódu detekcie anomálií založenú na riedkych vlastnostiach a vnútornom stave modelu. Autori v rámci článku poukazujú na skutočnosť, že daný prístup funguje dobre, aj keď je množstvo trénovacích údajov malé.

3 Dataset a predspracovanie dát

Na výskum v oblasti digitálnej forenznej analýzy existuje málo datasetov, ktoré sú ohodnotené. Pri hľadaní datasetov vhodných na hľadanie a overovanie nášho riešenia sme našli len jeden, ku ktorému boli publikované relevantné digitálne stopy, v konkrétnom prípade názvy súborov, ktoré boli súčasťou bezpečnostného incidentu. Ide o dataset The Stolen Szechuan Sauce z portálu DFIRmadness [62]. V kapitole sa ďalej venujeme analýze dát, ktoré predstavujú obraz disku operačného systému Windows so súborovým systémom NTFS. Predspracovanie dát pozostáva z vytvorenia časového radu udalostí zaznamenaných v súborovom systéme. Na túto časť bol použitý nástroj Log2timeline plaso. Následne sa pokračuje binarizáciou dát. Pre potreby detekcie anomálií je vhodné agregovať dáta. Agregáciu dát vykonávame naprieč atribútom inode. Dátové vektory po agregácií predstavujú informáciu o správaní inodu naprieč jeho životným cyklom.

V tradičných unixových súborových systémoch, ako napríklad ext2, ext3, ext4 a podobne, "inode" je skratka pre “index node“ alebo “information node“. Ide o štruktúru dátového bloku, ktorá obsahuje metadáta o súbore alebo adresári. Každý súbor a adresár v unixovom súborovom systéme má priradený svoj vlastný inode, ktorý obsahuje informácie ako:

- identifikačné číslo súboru,
- typ súboru (súbor, adresár, symbolický odkaz atď.),
- oprávnenia prístupu,
- veľkosť súboru,
- časové značky (čas vytvorenia, modifikácie, prístupu),
- ukazovatele na dátové bloky, kde je uložený obsah súboru.

Na rozdiel od tradičných unixových súborových systémov, v súborovom systéme NTFS sa priamo koncept inode nepoužíva. Namiesto toho NTFS používa podobnú štruktúru vo forme MFT, ktorá je bližšie vysvetlená v časti 1.2.1. Inode v našich dátach predstavuje identifikačné číslo záznamu v MFT [63].

3.1 Popis datasetov

Pri výbere datasetov sme sa rozhodli použiť obrazy diskov z CTF súťaží zameraných na digitálnu forenznú analýzu, na reakcie voči bezpečnostným incidentom a na vyhľadávanie hrozieb. V rámci práce sme spracovali nasledujúce datasety:

- **Magnet Forensics CTF 2020** [64] - Obraz disku je súčasťou súťaže, kde cieľom bolo nájsť vlajku v určitom formáte použitím techník digitálnej forenznej analýzy. Nejde o klasický proces digitálnej forenznej analýzy, ale účastníci súťaže hľadali odpovede na otázky typu „kedy bol získaný obraz disku“, „kedy bol nainštalovaný softvér“ a ďalšie.
- **NIST Data Leakage Case** [65] a **NIST Hacking Case** [66] - Účelom týchto prípadov je naučiť sa používať, respektíve precvičiť si rôzne techniky forenzného vyšetrovania. V prípade Nist Data Leakage Case ide o vyšetrovanie porušenia ochrany údajov. Cieľom je nájsť relevantné digitálne stopy vykonané útočníkom a dôkazy o protiprávnom konaní.
- **The Stolen Szechuan Sauce** [62] - Cieľom tohto prípadu je zistiť, ako sa tajný recept na sečuánsku omáčku spoločnosti CITADEL dostal na temnú webovú stránku. Spoločnosť požiadala o forenznú analýzu svojho radiča domény a hostiteľa siete s cieľom identifikovať súbory súvisiace s bezpečnostným incidentom a škodlivé aplikácie. Pracujeme s artefaktmi zo servera radiča domény spoločnosti (DC server) a z pracovnej plochy (hostiteľ siete).

3.2 Vytvorenie časovej osi

Vo fáze predspracovania dát vytvárame časovú os a vyberáme zdroje dát, ktoré sú relevantné pre ďalšiu analýzu. Na vytvorenie časovej osi využívame nástroj Log2timeline plaso. Všeobecné použitie nástroja je podľa definície nasledujúce [67]:

```
log2timeline.py [OPTIONS] [-f FORMAT] [-z TIMEZONE] [-o OUTPUT MODULE] [-w BODYFILE] LOG_FILE/LOG_DIR [--] [FORMAT FILE OPTIONS].
```

Vzhľadom na to, že pracujeme s obrazom disku zariadenia s operačným systémom Windows, zvolili sme parser win7_slow, ktorý obsahuje ďalšie podparsery: win_gen, webhist a win7. Táto voľba je postačujúca na získanie potrebných dát, pretože tieto podparsery zahŕňajú ďalšie relevantné možnosti. Napríklad, win_gen obsahuje podparsery ako bencode, czip/oxml, filestat, gdrive_synclog, lnk, mcafee_protection,

olecf, pe, prefetch, setupapi, sccm, skydrive_log, skydrive_log_old, sqlite/google_drive, sqlite/skype, symantec_scanlog, usnjrnl, webhist, winfirewall, winjob a winreg. Viac informácií o dostupných parseroch respektíve pluginoch sú uvedené v dokumentácii nástroja Log2timeline [68]. Nižšie je uvedený príklad spustenia nástroja s výberom určených parserov:

log2timeline.py --parsers=win7_slow --status_view window –storage <file>.E01 --partitions "all".

Na súbore, ktorý sme dostali na výstupe, sme použili nástroj psort.py. Ten prevedie získané dáta do čitateľnej podoby. Pre naše účely je formát v tabuľkovej podobe najviac vyhovujúci, preto sme použili formát l2tcsv.

psort.py --status-view window --output_time_zone utc -o l2tcsv -w timeline.csv timeline.dump

Takto získaná časová os pozostáva zo 17 atribútov:

- date,
- time,
- timezone,
- MACB,
- source,
- sourcetype,
- type,
- user,
- host,
- short,
- desc,
- version,
- filename,
- inode,
- notes,
- format and extra.

3.3 Binarizácia dát

V tejto práci sa venujeme identifikácii relevantných digitálnych stôp naprieč zaznamenanými udalosťami v súborovom systéme. Preto sme na získanej časovej osi vybrali dáta zo zdroja file. Z kategorických premenných sme vytvorili nové atribúty v binárnej podobe a z atribútu extra sme získali ďalšie informácie. Tak sme získali dáta s atribútmi popísanými v Tab. 4.

Atribút	**Popis**
Column1	Pôvodné poradové číslo záznamu v súbore obsahujúcom časovú os
filename	Názov súboru
inode	Pôvodný stĺpec inode, identifikácia inode v súborovom systéme NTFS
extra	Stĺpec s informáciami - z tohto stĺpca boli extrahované rôzne atribúty
id	Stĺpec s vygenerovaným jedinečným id (pre získanie záznamu)
datetime	Časová pečiatka, spojenie stĺpcov s dátumom a časom, časové pásmo UTC
M	Modifikovaný
A	Prístupovaný
C	Zmenený, modifikovaný $MFT
B	Vytvorený, čas vytvorenia súboru
file_stat	Stĺpec na základe hodnôt zo stĺpca sourcetype
NTFS_file_stat	Stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype
file_entry_shell_item	Stĺpec na základe hodnôt zo stĺpca sourcetype
NTFS_USN_change	Stĺpec na základe hodnôt zo stĺpca sourcetype
file_path	Súbor s úplnou cestou
name	Názov stĺpca extrahovaný z desc
typef	Typ súboru ('None','file','directory','link')
filef	1 ak je to súbor, 0 ak nie je
directory	1 ak je to priečinok, 0 ak nie je
link	1 ak je to súbor odkazu, 0 ak nie je
dir_type	Typ priečinka - umiestnenie súboru ('Windows', 'AppData','Other','Users')
dir_appdata	Popis cesty - extrahovaný stĺpec - na základe hodnôt z desc, 1 ak je AppData v ceste
dir_win	Popis cesty - extrahovaný stĺpec - na základe hodnôt z desc, 1 ak je Windows v ceste
dir_user	Popis cesty - extrahovaný stĺpec - na základe hodnôt z desc, 1 ak je User v ceste
dir_other	Popis cesty - extrahovaný stĺpec - na základe hodnôt z desc, 1 ak nie je nič z vyššie uvedeného
file_type	Typ súboru podľa prípony

file_executable	Typ súboru podľa prípony - stĺpec extrahovaný na základe hodnôt typu súboru z názvu súboru (spustiteľné súbory)
file_graphic	Typ súboru podľa prípony - stĺpec extrahovaný na základe hodnôt typu súboru z názvu súboru (grafické súbory)
file_documents	Typ súboru podľa prípony - stĺpec extrahovaný na základe hodnôt typu súboru z názvu súboru (dokumenty)
file_ps	Typ súboru podľa prípony - stĺpec extrahovaný na základe hodnôt typu súboru z názvu súboru (powershell súbory)
file_other	Typ súboru podľa prípony - stĺpec extrahovaný na základe hodnôt typu súboru z názvu súboru (ostatné)
mft	Stĺpec vytvorený na základe hodnôt zo stĺpca formátu
lnk_shell_items	Stĺpec vytvorený na základe hodnôt zo stĺpca formátu
olecf_olecf_automatic_destinations /lnk/shell_items	Stĺpec vytvorený na základe hodnôt zo stĺpca formátu
winreg_bagmru/shell_items	Stĺpec vytvorený na základe hodnôt zo stĺpca formátu
usnjrnl	Stĺpec vytvorený na základe hodnôt zo stĺpca formátu
is_allocated	Stĺpec z extrahovanej hodnoty zo stĺpca extra (1 ak je informácia nájdená)
is_allocated0	1 ak hodnota v stĺpci is_allocated bola jedna (súbor je umiestnený v súborovom systéme)
is_allocated1	1 ak hodnota v stĺpci is_allocated bola nula (súbor bol odstránený)
file_size	Veľkosť súboru
sha_256	Hash extrahovaný zo stĺpca extra
timestamp	Časová pečiatka úpravy (dátum a čas)
epochtime	Epoch time (od 1.1.1970)
hour	Hodina (extrakcia z časovej pečiatky)
minute	Minúta (extrakcia z časovej pečiatky)

Tab. 4 Tabuľka popisujúca atribúty dát po binarizácii

Dáta následne ukladáme do .csv súboru. Binárna forma atribútov umožňuje ďalšie spracovanie dát, agregovanie týchto dát a sledovanie trendov a vzťahov medzi atribútmi. Takisto je možné použiť binárne atribúty na zobrazenie vzťahov do grafov alebo ako vstup pre neurónové siete.

4 Automatizácia detekcie anomálií

V tejto časti ukážeme návrh riešenia na automatizáciu hľadania anomálií, respektíve na automatizáciu identifikácie relevantných digitálnych stôp. Agregačné funkcie implementujeme v programovacom jazyku Python. Pri detekcii anomálií použijeme viaceré metódy, ktoré boli popísané v časti 2.5. Pri spracovávaní dát pracujeme s knižnicou pandas [69], ktorá nám umožňuje implementovať spracovanie dát časového radu a prípravu agregovaných vektory. Knižnica PyOD [70] nám umožnila rýchlu implementáciu metód hľadania anomálií. Prípravu neurónovej siete je naprogramovaná a natrénovaná pomocou knižnice Tensorflow [71].

Výstupom automatizovaného riešenia je excelovský súbor, ktorý obsahuje agregované dáta, výsledky konkrétnych modelov, a sumárne skóre predstavujúce anomalitu jedného vektora. Pre prípad datasetu The Stolen Szechuan Sauce, nakoľko je ohodnotený, boli vyhodnotené viaceré metriky úspešnosti jednotlivých modelov.

Časové rady vytvorené pomocou nástroja plaso a následnej binarizácie záznamov zo zdroja file máme uložené v .csv súbore. Ich načítanie do Dataframu prebieha jednoducho pomocou príkazu pandas.read_csv(‘filename.csv).

4.1 Agregácia dát

Pri agregácii používame dáta, ktorých generovanie bolo popísané v predošlej časti. Stĺpce reprezentujú atribúty objektu, a riadky konkrétne udalosti, teda v našom prípade indexom záznamu, inodom. V rámci agregácií používame štandardné agregačné funkcie maximum (max), súčet (sum) a priemer (mean). Navrhli sme aj ďalšie agregácie pre lepšiu výpovednú hodnotu dát.

Uvažujme pevný atribút A z binárnych atribútov. Označme

a_i^{inode} – binárna hodnota atribútu A i-tej udalosti s inodom $inode$,

n_{inode} – počet udalostí s inodom $inode$,

N – počet unikátnych inodov v datasete.

Ďalej zadefinujeme počet pozitívnych výskytov atribútu A v udalostiach s inodom $inode$

$$a^{inode} = \sum_{i=1}^{n_{inode}} a_i^{inode},$$

priemerný počet pozitívnych výskytov atribútu A

$$\bar{a} = \sum_{inode=1}^{N} a^{inode},$$

medián

$$\tilde{a} = medián(a^1, \dots, a^N),$$

výberový rozptyl

$$s^2 = \frac{1}{N-1} \sum_{inode=1}^{N} (a^{inode} - \bar{a})^2,$$

rozdiel horného a dolného kvartilu datasetu (IQR)

$$IQR = Q_{0,75} - Q_{0,25},$$

a median absolute deviation (MAD)

$$MAD = med(|a^1 - \tilde{a}|, \dots, |a^N - \tilde{a}|).$$

Agregačné funkcie pre atribút A a pre $inode$ sú

- frekvencie v rámci dní (freq1)

$$\frac{a^{inode}}{\bar{a}}$$

- frekvencie v rámci dní 2 (freq2)

$$\frac{a^{inode}}{n_{inode}}$$

- z-score (z1)

$$\frac{a^{inode} - \bar{a}}{s}$$

- modifikované z-skóre (z2)

$$\frac{a^{inode} - \tilde{a}}{IQR}$$

- robusné z-skóre (z3)

$$\frac{a^{inode} - \tilde{a}}{MAD}$$

V tejto časti popíšeme implementáciu agregačných funkcií popísaných vyššie s použitím konštruktora DataFrame z knižnice pandas. Pomocou funkcie groupby môžeme efektívne spracovať a zoskupiť dáta podľa vybraného atribútu. Jej funkcionalita umožňuje vytvoriť z veľkého množstva dát významne zredukovaný počet vektorov, po aplikovaní určitých funkcií [72]. Vytvorili sme funkciu get_aggregated_data(), v ktorej sme implementovali všetky agregačné funkcie. Na vstupe tejto funkcie sa nachádzajú dáta v binárnej podobe, názov agregačnej funkcie, z ktorej chceme získať upravené dáta, a názov atribútu, podľa ktorého vytvárame skupiny. V našom prípade ide o atribút inode. Možnosti parametra agregačnej funkcie sú: 'sum', 'max', 'mean', 'z1', 'z2', 'z3', 'freq1' a 'freq2'. Agregované dáta z funkcií max, mean a sum sa v knižnici pandas dajú veľmi jednoducho získať:

- `sum - data.groupby([group_by]).sum(),`
- `max - data.groupby([group_by]).max(),`
- `mean - data.groupby([group_by]).mean().`

Pri funkciách freq1 a freq2 sa počet jednotiek v danom atribúte pre konkrétny inode delí počtom všetkých jednotiek alebo počtom záznamov pre konkrétny inode pomocou funkcie div() z knižnice pandas.

Funkcie z1, z2 a z3 potrebujú viacero medzivýsledkov, pri ktorých sa používajú funkcie na odčítanie (sub), umocnenie (pow), výpočet kvantilov (quantile), násobenie (mul) a výpočet absolútnych hodnôt (abs). Funkcia get_aggregated_data() so všetkými implementovanými agregačnými funkciami je v **Prílohe A**. V Tab. 5 vidíme porovnania veľkostí súborov, počet udalostí, a počet záznamov po agregácii pre časové rady zdroja file z datasetov.

	Veľkosť v bajtoch	Počet udalostí	Počet záznamov po agregácii
The Stolen Szechuan Sauce	357339747	843863	85975
NIST Hacking Case	34120024	95920	17987
NIST Data Leakage Case	782668090	1907945	77870
Magnet Forensics CTF 2020	684470272	1654919	110148

Tab. 5 Porovnanie veľkosti, počtu udalostí a počtu záznamov po agregácii

4.2 Metódy

Pri postupe riešenia sme vybrali metódy, ktoré vieme použiť pri rôznych datasetoch. Vďaka automatizácii budeme schopní prejsť rôzne reprezentované dáta v datasetoch a rýchlejšie overiť úspešnosť týchto metód. V Tab. 6 vidíme zoznam metód použitých pri automatizácii hľadania relevantných digitálnych stôp.

METÓDA	NÁZOV	TYP
COPOD	Copula Based Outlier Detector	Unsupervised
ECOD	Empirical-Cumulative-distribution-based Outlier Detection	Unsupervised
INNE	Isolation-based Anomaly Detection Using Nearest-Neighbor Ensembles	Unsupervised
IForest	Isolation Forest	Unsupervised
LODA	Lightweight on-line detector of anomalies	Unsupervised
LOF	Local Outlier Factor	Unsupervised
PCA	Principal Component Analysis	Unsupervised

Tab. 6 Výber metód pre detekciu anomálií

Pre každú metódu z Tab. 6 je možné vybrať viaceré parametre na vstupe, a tým meniť správanie sa pri detekcii. Tab. 7 popisuje rôzne parametre týchto metód.

PARAMETER	VSTUP	VYSVETLENIE
contamination	float in (0., 0.5), optional (default=0.1)	Množstvo kontaminácie dátovej sady, teda podiel anomálií v dátovej sade. Používa sa pri prispôsobovaní modelu na definovanie prahu na rozhodovaciu funkciu.
n_neighbors	int, optional (default=20)	Počet susedov, ktorý sa predvolene použije pre dotazy k-neighbors. Ak je n_neighbors väčšie ako počet poskytnutých vzoriek, použijú sa všetky vzorky.
metrics	string or callable, default 'minkowski'	Metrika použitá pre výpočet vzdialeností.
n_bins	int or string, optional (default=10)	Počet binov pre histogram. Ak je nastavené na "auto", použije sa metóda Birge-Rozenblac na automatické určenie optimálneho počtu binov.
n_estimators	int, default=200	Počet základných odhadcov v súbore.

Tab. 7 Tabuľka popisujúca parametre pre metódy detekcie anomálií

4.3 Príprava funkcie

V nasledujúcom kroku sme pripravili funkciu outlier_detection(), ktorá nám umožňuje spustiť detekciu anomálií na rôznych metódach a porovnať výsledky na danom datasete. Výhodou takéhoto prístupu je možnosť overenia, respektíve hľadania, najlepšej metódy pre daný dataset. Funkcia zahŕňa metódy z predošlej kapitoly, a umožňuje používateľovi nastaviť rôzne vstupné parameter pre každú funkciu.

Funkcia akceptuje na vstupe list slovníkov, ktoré obsahujú podrobnosti o vstupných metódach, z ktorých sa pripraví viacero modelov. Takisto konkrétny slovník obsahuje hodnotu, ktorá označuje, či je inštancia zdrojom anomálie alebo normálnych údajov. Tento list funguje ako konfiguračný nástroj, ktorý nám umožňuje nastaviť niekoľko rôznych modelov, teda parametre pre všetky modely, ktoré chceme zahrnúť do analýzy. Z takto nakonfigurovaných modelov vieme ďalej spracovať výsledky a vypočítať sumárnu hodnotu anomality daného záznamu.

Konfigurovateľný prístup umožňuje nastaviť rôzne kombinácie metód a parametrov, a tým prispôsobiť funkciu svojim špecifickým potrebám a požiadavkám. Umožňuje nám to analyzovať riešenia v rôznych kontextoch, a aktualizovať modely pre ďalšie výpočty. Každý slovník v liste pozostáva z parametrov:

- name – názov funkcie

- function – konkrétná funkcia na detekciu anomálií
- voliteľné parametre špecifické pre každú metódu osobitne
    - contamination_values – zoznam kontaminácií použitých pre vytváranie modelov, je použitý všetkými metódami,
    - n_estimators – zoznam použitých hodnôt n_estimator parametra pri generovaní modelov, tento parameter je využívaný pre metódy INNE a Iforest,
    - n_bins – hodnoty počtov binov pre histogram, používa sa funkciou LODA pri vytváraní modelov,
    - n_neighbors – zoznam počtu susedov, je použitý metódou LOF
    - metrics – metriky, ktoré sa používajú na výpočet vzdialeností pri modeloch generovaných metódou LOF
- outlier_value – hodnota, ktorú používa daná metóda na označenie anomálnych záznamov.

Na tomto mieste ukazujeme, ako vyzerá konfiguračný slovník pre jednotlivé metódy:

pre PCA, ECOD a COPOD:

```
{'name': 'PCA'/'ECOD'/'COPOD',
 'function': PCA/ ECOD/ COPOD,
 'contamination_values': contamination_values,
 'outlier_value': 1},
```

pre INNE a IForest:

```
{'name': 'INNE'/'IForest',
 'function': INNE/IForest,
 'contamination_values': contamination_values,
 'n_estimators': n_estimators,
 'outlier_value': 1},
```

pre metódu LOF:

```
{'name': 'LOF',
 'function': LOF,
 'contamination_values': contamination_values,
 'neighbors_values': neighbors_values,
 'metrics': metrics,
 'outlier_value': 1}
```

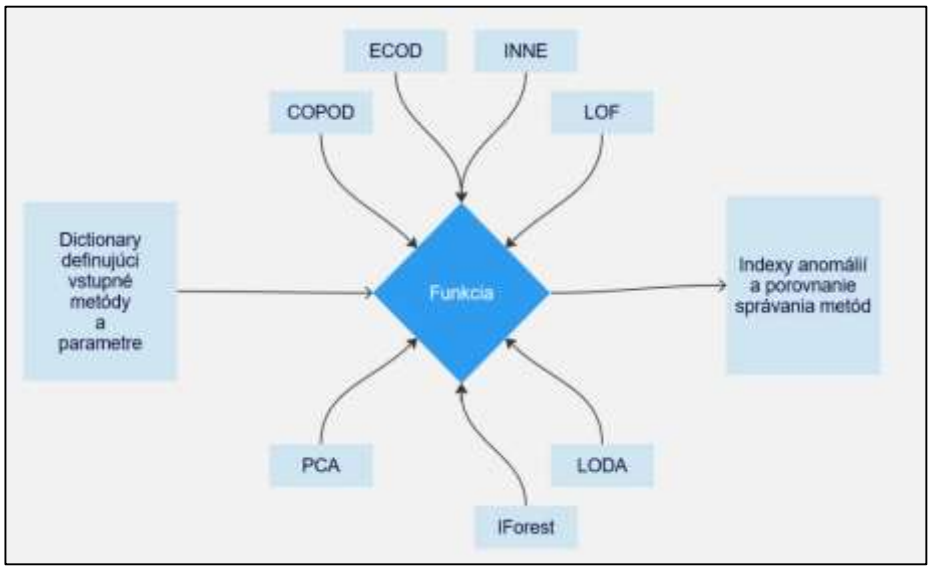
a pre metódu LODA:

```
{'name': 'LODA',
```

```
'function': LODA,
'contamination_values': contamination_values,
'n_bins': n_bins,
'outlier_value': 1}.
```

Celý konfiguračný list použitý pri automatizovanej detekcii anomálií je v **Prílohe B**, v ktorom sme zadefinovali 76 modelov.

Myšlienka tejto funkcie je založená na jednoduchom princípe spojenia viacerých modelov do jednej súhrnej automatizovnej detekcie anomálií. Na Obr. 7 môžeme vidieť schému spracovania dát funkciou, zapojenie metód a vygenerovanie výsledkov.



Obr. 7 Princíp fungovania funkcie automatizácie

Funkcia outlier_detection() postupne prechádza konfiguračným listom, a generuje modely, ktoré následne trénuje. Každý model vyhodnocuje anomalitu jednotlivých záznamov po natrénovaní. Model uchováva informácie ako názov metódy, skóre, označenie, indexy a počet anomálnych záznamov a parametre modelu v slovníku data. Názov modelu je taktiež obsiahnutý v tomto slovníku, a vytvára sa z názvu metódy, názvu agregácie, nad ktorou aktuálne trénujeme, a parametrov. Príkladom `'name': f'{name}-{agg}-cont-{contamination_value}'.`

Pre metódy COPOD, ECOD a PCA vyzerá generovanie modelov, ich trénovanie a následne vyhodnotenie takto:

```
if name == 'COPOD' or name == 'ECOD' or name == 'PCA':
  for contamination_value in method['contamination_values']:
    model1 = method['function'](contamination=contamination_value)
    model1.fit(agg_data)
    outlier_index=np.where(model1.labels_==method['outlier_value'])
    labels = model1.labels_
    if method['outlier_value'] == 1:
      labels = [item * -1 for item in labels]
    data = {
      'name': f'{name}-{agg}-cont-{contamination_value}',
      'method': name,
      'labels': labels,
      'score': model1.decision_scores_,
      'outlier_index': outlier_index[0],
      'outlier_count': len(outlier_index[0]),
      'contamination_value': contamination_value
    }
    results.append(data)
```

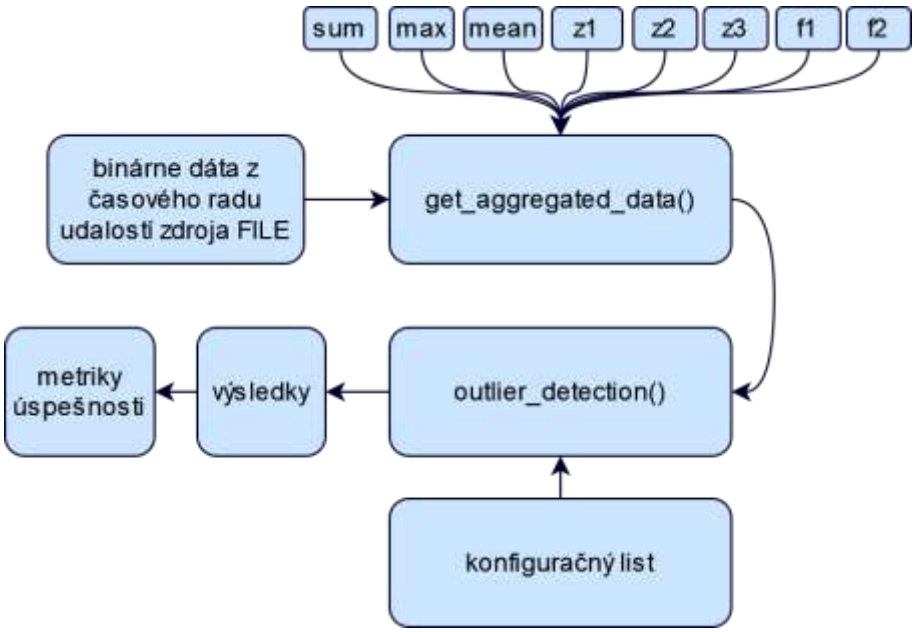
Implemetácia celej funkcie outlier_detection je v **Prílohe C**.

Ak je dataset, s ktorým pracujeme, ohodnotený, vieme si dopočítať hodnoty úspešnosti pre konkrétne modely. Funkcia calculate_acc() z **Prílohy D** nám vypočíta nasledujúce metriky úspešnosti:

- True Positive (TP) – počet správne identifikovaných anomálií
- True Negative (TN) - počet správne identifikovaných normálnych záznamov
- False Positive (FP) - počet nesprávne identifikovaných anomálií
- False Negative (FN) - počet nesprávne identifikovaných normálnych záznamov
- Recall - pomer správne identifikovaných anomálií k celkovému počtu anomálií
- Precision - pomer správne identifikovaných anomálií k celkovému počtu identifikovaných anomálií
- F1 skóre (F1) - harmonický priemer medzi Recall a Precision
- True Positive Rate (TPR) - pomer správne identifikovaných anomálií k celkovému počtu anomálií
- False Positive Rate (FPR) - pomer nesprávne identifikovaných normálnych záznamov k celkovému počtu normálnych hodnôt
- Accuracy – presnosť.

V tomto kroku máme pripravené funkcie pre získanie výsledkov, vypočítanie sumárnej hodnoty anomality pre jednotlivé záznamy, a máme ohodnotený dataset, takže vieme vypočítať aj metriky úspešností. Binárne dáta vygenerované z časového radu udalostí zdroja FILE dávame na vstup metódy get_aggregated_data(). Následne agregované dáta dávame na vstup funkcii outlier_detection() spolu s konfiguračným listom. Takto generujeme výsledky pre všetky agregačné funkcie. Ďalej vypočítame metriky úspešnosti pomocou metódy calculate_acc(). Obr. 8 ukazuje postup popísaný vyššie.

Následne potrebujeme výsledky reprezentovať. Najvhodnejšou formou je zobraziť agregované dáta každej funkcie spolu s výsledkami modelov do excelovských hárkov, pre rýchly prehľad. Pri počítaní metrík úspešnosti pridáme pre každú agregačnú funkciu hárok, kde ukážeme úspešnosť jednotlivých modelov pri detekcii anomálií.



Obr. 8 Proces získania výsledkov

V rámci reprezentácie dát počítame aj sumárnu hodnotu anomality jednotlivých záznamov a výstup do excelu usporadúvame podľa tejto hodnoty. Zdrojový kód generovania excelu je v **Prílohe E**. Výpočet sumáru robíme pomocou nasledujúcej časti kódu.

```
final_score = [ 0 for i in range(len(all_results[0]['labels']))]
```

```
for i in range(len(result['labels'])):
     if result['labels'][i] == -1:
          final_score[i] += 1
```

To následne pridávame do Dataframu, ktorý exportujeme do hárku excelovského súboru, takto:

```
agg_data_excel['final_score'] = final_score.
```

Hárok excelu pred uložením zafarbíme, a to tak, že agregované dáta, ktorých hodnota je rôzna od nuly, sú zafarbené zelenou farbou. Každý model označil daný atribút ako anomálny alebo normálny. Tieto hodnoty sa ukladajú do stĺpcov <názov_modelu>-labels. Ak ide o anomalitu, tak hodnota v týchto stĺpcoch je rovná -1, a daná bunka je zafarbená vo výslednom excelovskom hárku červenou farbou. Naopak ak ide o normálny záznam, tak hodnota je rovná 0. Skóre z každého modelu je uložené v stĺpcoch pomenovaných <názov_modelu>-score. Štruktúra jedného excelovského hárku je ukázaná v Tab. 8. Ak máme vypočítané metriky úspešnosti, tie ukladáme do hárku s názvom <názov agregácie>-acc. V Tab. 9 môžeme vidieť náčrt hárku pre metriky modelov.

inode	Attribute1	Attribute2	Model1-score	Model1-labels	Model1-score	Model1-labels	Final_score
Inode1	2	0	90	-1	90	-1	2
Inode2	0	1	80	-1	80	0	1
Inode3	10	4	70	0	70	0	0

Tab. 8 Náčrt hárku vo výstupnom exceli

mo	para	TP	FP	FN	TN	Rec	Pre cision	F1	TP R	FP R	Acc uracy
1	..		781		2179	,6	,002	,004 7	,6	0 ,0439	0 ,9559
2	..	3	151		1809	,86	,003	,006 2	,86	0 ,04828 9902	0 ,95169 526
3	..	5	262		1698		,003 5	,006 9		0 ,04958 1201	0 ,95042 745

Tab. 9 Náčrt hárku pre metriky úspešnosti modelov

4.4 Autoencoder neurónová sieť

Jedná sa o typ neurónovej siete, ktorej hlavným cieľom je naučiť sa zachytiť dôležité vlastnosti na vstupných dátach. Funkciou encodera je zachytiť kľúčové vlastnosti. Decoder na druhej strane obnovuje dáta po tom, čo encoder znížil ich dimenziu. Túto neurónovú sieť sme implementovali pomocou tensorflow frameworku [71]. Implementovaná sieť má 16884 parametrov na trénovanie. Pozostáva zo 4 LSTM vsrtiev, pričom prvá vrstva berie ako vstup časové okno 70 nasledujúcich udalostí z časového radu zdroja FILE s počtom atribútov 27. Nasledujúca vrstva zmenšuje dimenziu, aby sa naučila doležité atribúty zo vstupu. Jedná sa o časť encodingu v neurónovej sieti autoencoder. Nasledujúce 2 vrstvy patria do časti decodingu. Na výstupe neurónovej siete dostávame jeden vektor s 27 atribútmi, čo predstavuje predikciu nasledujúceho záznamu zo vstupných dát. V Tab. 10 máme popísané typy vrstiev navrhnutej neurónovej siete autoencoder spolu s hodnotami výstupných dimenzií. Aby sme vedeli počítať skóre anomality z neurónovej siete, zvolili sme loss-funkciu mean square error (MSE).

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(Y_i = \widehat{Y_1})^2$$

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 70, 27)	5940
lstm_5 (LSTM)	(None, 70, 18)	3312
lstm_6 (LSTM)	(None, 70, 18)	2664
lstm_7 (LSTM)	(None, 27)	4968
Total params: 16884 (65.95 KB) Trainable params: 16884 (65.95 KB) Non-trainable params: 0 (0.00 Byte)		

Tab. 10 Štruktúra navrhnutej neurónovej siete autoencodera

Záznamy z časového radu je potrebné pripraviť pre neurónovú sieť do takzvaného rolling-window formátu. Vstupné dáta predstavujú 70 za sebou idúcich udalostí. Ďalšia takáto sekvencia je vytvorená posunutím o jeden index. Cieľové dáta predstavuje nasledujúci prvý vektor po sekvencii 70 vstupných dát. Pre účely neurónovej sme pripravili generátor, ktorý posiela neurónovej sieti dáta v tejto forme po dávkach. Časť kódu generátora, ktorý pripravuje tieto sekvencie je vypísaná nižšie. Celá implementácia generátora a neurónovej siete je v **Prílohe F**.

```
def __getitem__(self, idx):
    start_idx = idx * self.batch_size
    end_idx = min(start_idx + self.batch_size, self.num_samples)
    batch_x = []
    batch_y = []
    for i in range(start_idx, end_idx):
        window = self.data[i:i+self.window_size]
        batch_x.append(window[:-1])
        batch_y.append(window[-1]))
    return np.array(batch_x), np.array(batch_y)
```

Neurónovú sieť trénujeme na jednej epoche. Na základe kontextu o predošlých udalostiach sa neurónová sieť učí, aká udalosť bude nasledovať. Takto jej správaanie kopíruje štandardné správanie operačného systému. Po natrénovaní tejto neurónovej siete sledujeme, nakoľko je predikovaná udalosť v zhode s očakávanou udalosťou. Skóre anomality vieme vypočítať ako zrekonštruovanú hodnotu loss-funkcie, teda pomocou

```
MSE = tf.keras.losses.MSE(
```

```
    target, trainPredict
).
```

Takto získané hodnoty priradzujeme naspať k inodom, a vytvoríme dataframe. Čím vyššia je hodnota zrekonštruovanej loss-funkcie, tým je udalosť viac anomálna. Dataframe usporiadame na základe zrekonštruovanej loss-funkcie a nakoniec ho exportujeme do .xlsx súboru. Pri datasete The Stolen Szechuan Sauce môžeme uložiť podmnožinu dataframu, ktorá obsahuje len napadnuté inody, a tak si skontrolovať ich výsledné poradie a overiť účinnosť neurónovej siete.

5 Analýza výsledkov

V tejto časti sa bližšie pozrieme na vygenerované dáta nášho riešenia automatizácie detekcie anomálií z .xlsx súboru. Ten obsahuje hárky pre každú agregačnú funkciu. V jednotlivých hárkoch sa nachádzajú agregované dáta a výsledky modelov definovaných konfiguračným listom, ako bolo ukázané v Tab. 8. V prvom stĺpci sa nachádza atribút *inode*. Rozoberme príklad The Stolen Szechuan Sauce. Na základe informácií zo stránok portálu DFIR madness [73], odkiaľ dataset pochádza, a manuálnej forenznej analýzy bolo v tomto datasete identifikovaných 15 inodov súvisiacich s bezpečnostným incidentom: 84630, 84880, 84987, 86966, 86967, 86968, 86970, 86971, 86975, 87059, 87060, 87064, 87111, 87112, 87137 [74]. Pre jednotlivé agregačné funkcie boli inody po usporiadaní na základe sumárnej hodnoty na pozíciach ukázaných v Tab. 11.

aggregácia	sum	max	mean	freq1	freq2	z1	z2	z3
inode	-	-	-	-	-	-	-	-
84630	14	64	17	39	51	36	10	18
84880	659	190	505	774	518	781	564	265
84987	19	59	18	52	4	8	18	24
86966	1467	586	1221	1290	1185	1368	1044	1055
86967	84	46	28	105	6	169	52	21
86968	17	47	29	55	7	3	5	8
86970	28	48	7	12	8	12	25	10
86971	18	49	30	56	40	27	12	13
86975	22	50	31	43	9	40	13	12
87059	5334	1638	2316	3765	1636	2545	1626	2350
87060	8	44	32	42	65	38	14	15
87064	16	51	33	37	53	37	15	16
87111	89	55	36	104	14	193	51	26
87112	23	56	37	54	41	14	21	17
87137	3475	4204	2921	3011	3254	4545	1931	2422

Tab. 11 Pozície inodov po usporiadaní na základe sumárnej hodnoty

Na základe poradia vieme určiť, ktoré inody sú problémové pri určovaní ich anomality použitím nášho navrhovaného riešenia. Pri inodoch 84880, 86966, 87059 a 87137 sú poradia výrazne vyššie ako pri ostatných inodoch. Keď sa pozrieme na hodnoty v rámci stĺpcov, pre každú agregačnú funkciu, vidíme, že freq1, max, a z1 majú vyššie ohodnotené poradia pre dané inody, teda ich použiteľnosť je horšia ako

pri ostatných agregačných funkciách. Agregačné funkcie sum, mean, freq2, z2 a z3 potvrdzujú, že navrhované riešenie priradilo väčšine inodov, okrem vyššie spomínaných, veľmi dobré poradie. Pre forenzného analytika je dôležité čo najskôr nájsť relevantnú digitálnu stopu po bezpečnostnom incidente, a preto, takto zoradené inody umožňujú expertovi digitálnej forenznej analýzy rýchly začiatok vyšetrovania.

Ohodnotenie sumárneho skóre reprezentuje počet modelov, ktoré určili daný inode ako anomálny. Tieto skóre sa opakujú, a preto každému inodu môžeme priradiť, ku ktorej skupine v rámci skóre patril. Napríklad, ak *final_score* pre inode je 67, a maximálna hodnota *final_score* je 72, tak konkrétny inode patrí do 72 – 67 = 5-tej skupiny. Tab. 12 ukazuje príslušnosť inodov relevantných k bezpečnostnému incidentu do skupín.

aggregácia	sum	max	mean	freq1	freq2	z1	z2	z3
inode	-	-	-	-	-	-	-	-
84630	3	2	3	3	3	2	3	2
84880	20	3	19	27	20	22	19	14
84987	4	2	3	4	1	1	4	3
86966	52	42	49	54	49	46	32	34
86967	8	2	3	8	1	6	5	3
86968	3	2	3	4	1	1	3	1
86970	4	2	2	2	1	2	5	2
86971	3	2	3	4	2	2	3	2
86975	4	2	3	3	1	2	3	2
87059	67	54	58	66	55	56	49	49
87060	3	2	3	3	3	2	3	2
87064	3	2	3	3	3	2	3	2
87111	8	2	3	8	1	6	5	3
87112	4	2	3	4	2	2	4	2
87137	63	55	59	63	58	61	50	49

Tab. 12 Príslušnosť inodov k skupinám naprieč skóre

Znova sa potvrdzuje, že inody 84880, 86966, 87059 a 87137 sú ťažšie identifikovateľné ako anomálie. Avšak hodnota skupiny, do ktorej daný inode patrí hovorí o tom, že ďalšie inody sú medzi top anomáliami. Sumárna hodnota teda nemusí byť najlepším merateľom anomality konkretného záznamu, a stojí za zváženie nájsť lepšie celkové ohodnotenie, ktoré by odlíšilo anomalitu jednotlivých záznamov.

5.1 Metriky modelov pre agregačné funkcie

V prípade The Stolen Szechuan Sauce obsahuje výstupný .xlsx súbor aj hárky s metrikami úspešností pre konkrétne modely naprieč agregačnými funkciami. Analýzou týchto metrík vieme určiť úspešnejšie modely, a pri ďalších spusteniach aktualizovať konfiguračný list pre metódu hľadania anomálií. Pri analýze týchto metrík je dôležité uvedomiť si, že naše agregované dáta pozostávajú z desiatok tisíc záznamov, avšak počet záznamov, o ktorých vieme že sú anomálne, je 15. Preto sa zameriame na metriky True Positive, False Negative a Recall. Pri hodnote False Positive je počet nesprávne určených anomálií vždy vysoký, keďže naše modely pracujú s parametrom kontaminácie rovným 0.05, teda 5 percent celého datasetu. Sledovaním hodnoty Recall vieme určiť, že modely, pri ktorých sa táto hodnota blíži k jednotke, určili väčšinu z 15 anomálií správne, a teda hodnota False Negative je blízka 0.

Pri agregačnej funkcii sum boli modely metódy INNE s najvyššou hodnotou Recall. Ďalej nasledovali modeli metódy IForest a LOF. Na Tab. 13 vidíme najlepšie modely pre agregačnú funkciu sum.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
INNE	0,05	200				15	4147	0	81813	1
INNE	0,05	300				15	4012	0	81948	1
INNE	0,05	350				15	4273	0	81687	1
INNE	0,05	400				15	4229	0	81731	1
INNE	0,05	450				15	4167	0	81793	1
IForest	0,05	100				13	4227	2	81733	0,866667
IForest	0,05	150				13	4034	2	81926	0,866667
IForest	0,05	200				13	4073	2	81887	0,866667
INNE	0,05	50				13	4051	2	81909	0,866667
INNE	0,05	100				13	4247	2	81713	0,866667
INNE	0,05	150				13	4053	2	81907	0,866667
INNE	0,05	250				13	4285	2	81675	0,866667
LOF	0,05			150	braycurtis	13	1203	2	84757	0,866667
LOF	0,05			200	braycurtis	13	1680	2	84280	0,866667
LOF	0,05			150	canberra	13	1203	2	84757	0,866667
LOF	0,05			150	canberra	13	1203	2	84757	0,866667
LOF	0,05			200	canberra	13	1442	2	84518	0,866667

LOF	0,05			200	canberra	13	1442	2	84518	0,866667
LOF	0,05			200	chebyshev	13	1680	2	84280	0,866667
LOF	0,05			200	chebyshev	13	1680	2	84280	0,866667

Tab. 13 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie sum

Recall hodnota pre modely trénované nad dátami agregačnej funkcie max bola viackrát rovná nule, čo predstavuje dobrý ukazovateľ nájdenia skutočných anomálií, ak neberieme do úvahy hodnotu False Positive. Tento jav sa vyskytol pri modeloch metód COPOD, ECOD, IForest, INNE a PCA. Tab. 14 ukazuje konkrétne modely s hodnotou Recall rovnou 0 pri agregačnej funkcii max.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
COPOD	0,05					15	4246	0	81714	1
ECOD	0,05					15	4246	0	81714	1
IForest	0,05	50				15	4236	0	81724	1
IForest	0,05	100				15	4233	0	81727	1
IForest	0,05	150				15	4233	0	81727	1
IForest	0,05	200				15	4233	0	81727	1
IForest	0,05	250				15	4233	0	81727	1
IForest	0,05	300				15	4233	0	81727	1
IForest	0,05	350				15	4233	0	81727	1
IForest	0,05	400				15	4233	0	81727	1
IForest	0,05	450				15	4233	0	81727	1
INNE	0,05	50				15	4246	0	81714	1
INNE	0,05	100				15	4246	0	81714	1
INNE	0,05	150				15	4246	0	81714	1
INNE	0,05	200				15	4246	0	81714	1
INNE	0,05	250				15	4246	0	81714	1
INNE	0,05	300				15	4246	0	81714	1
INNE	0,05	350				15	4246	0	81714	1
INNE	0,05	400				15	4246	0	81714	1
INNE	0,05	450				15	4246	0	81714	1
PCA	0,05					15	4246	0	81714	1

Tab. 14 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie max

Pri agregačnej funkcii mean mali modeli metód PCA, COPOD a ECOD najvyššiu hodnotu Recall. Niektoré modely metód IForest, INNE a LOF netrafili jednu, prípadne dve skutočné anomálie. V Tab. 15 vidíme usporiadanie top 21 modelov podľa metriky Recall.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
COPOD	0,05					15	3258	0	82702	1
ECOD	0,05					15	3759	0	82201	1
PCA	0,05					15	4038	0	81922	1
IForest	0,05	50				14	4284	1	81676	0,933333
IForest	0,05	100				14	3982	1	81978	0,933333
IForest	0,05	150				14	3489	1	82471	0,933333
IForest	0,05	200				14	3867	1	82093	0,933333
IForest	0,05	250				14	3925	1	82035	0,933333
IForest	0,05	300				14	3939	1	82021	0,933333
IForest	0,05	350				14	3940	1	82020	0,933333
IForest	0,05	400				14	3434	1	82526	0,933333
IForest	0,05	450				14	4240	1	81720	0,933333
INNE	0,05	50				14	3939	1	82021	0,933333
INNE	0,05	100				14	4283	1	81677	0,933333
INNE	0,05	300				13	4234	2	81726	0,866667
INNE	0,05	400				13	3599	2	82361	0,866667
INNE	0,05	450				13	4174	2	81786	0,866667
LOF	0,05			200	braycurtis	13	1342	2	84618	0,866667
LOF	0,05			200	canberra	13	1304	2	84656	0,866667
LOF	0,05			200	canberra	13	1304	2	84656	0,866667
LOF	0,05			150	chebyshev	13	877	2	85083	0,866667

Tab. 15 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie mean

Z modelov, ktoré dostali na vstup dáta z agregácie freq1, nedosiahol ani jeden hodnotu metriky Recall rovnú 1. Najvyššie skóre dosiahli modely metódy INNE, a ani tie nedokázali určiť jeden záznam, ktorý bol označený ako anomálny. Najúspešnejšie modely trénované nad dáatami z agregačnej funkcie freq1 zobrazuje Tab. 16.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
INNE	0,05	100				14	4272	1	81688	0,933333
INNE	0,05	150				14	4226	1	81734	0,933333
INNE	0,05	200				14	3831	1	82129	0,933333
INNE	0,05	250				14	4255	1	81705	0,933333
INNE	0,05	300				14	4236	1	81724	0,933333
INNE	0,05	350				14	4249	1	81711	0,933333
INNE	0,05	450				14	4280	1	81680	0,933333
IForest	0,05	150				13	3906	2	82054	0,866667
IForest	0,05	200				13	4240	2	81720	0,866667
IForest	0,05	450				13	4277	2	81683	0,866667
INNE	0,05	50				13	3887	2	82073	0,866667
INNE	0,05	400				13	3973	2	81987	0,866667
LOF	0,05			150	braycurtis	13	1203	2	84757	0,866667
LOF	0,05			200	braycurtis	13	1282	2	84678	0,866667
LOF	0,05			150	canberra	13	1203	2	84757	0,866667
LOF	0,05			150	canberra	13	1203	2	84757	0,866667
LOF	0,05			200	canberra	13	1442	2	84518	0,866667
LOF	0,05			200	canberra	13	1442	2	84518	0,866667
LOF	0,05			150	chebyshev	13	1203	2	84757	0,866667

Tab. 16 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie freq1

Podobne ako pri agregačnej funkcii max, aj pri agregačnej funkcii freq2 boli modely s najvyššou hodnotou Recall metriky z metód COPOD, PCA a ECOD. INNE, IForest a LOF modely nedokázali identifikovať jeden alebo dva, forenznou analýzou označené, anomálne záznamy. Zoradenie top modelov z pohľadu metriky Recall je v Tab. 17.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
COPOD	0,05					15	3258	0	82702	1
ECOD	0,05					15	3759	0	82201	1
PCA	0,05					15	4038	0	81922	1
IForest	0,05	50				14	4247	1	81713	0,933333
IForest	0,05	100				14	4268	1	81692	0,933333
IForest	0,05	200				14	4267	1	81693	0,933333

IForest	0,05	250				14	4163	1	81797	0,933333
IForest	0,05	300				14	3444	1	82516	0,933333
IForest	0,05	350				14	3957	1	82003	0,933333
IForest	0,05	400				14	3489	1	82471	0,933333
IForest	0,05	450				14	3449	1	82511	0,933333
INNE	0,05	150				14	3684	1	82276	0,933333
IForest	0,05	150				13	4178	2	81782	0,866667
INNE	0,05	50				13	3562	2	82398	0,866667
INNE	0,05	200				13	3956	2	82004	0,866667
INNE	0,05	300				13	3547	2	82413	0,866667
INNE	0,05	350				13	4196	2	81764	0,866667
INNE	0,05	450				13	3673	2	82287	0,866667
LOF	0,05			200	braycurtis	13	1342	2	84618	0,866667
LOF	0,05			200	canberra	13	1304	2	84656	0,866667

Tab. 17 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie freq2

Modely trénované nad agregačnou funkciou z1 dosiahli najvyššie skóre metriky Recall 0,933. Ani jeden z modelov neidentifikoval všetky anomálie. Najvyššiu hodnotu Recall dosiahli modely metód INNE, IForest. Prekvapením pri tejto agregačnej funkcii je model LODA, ktorý doteraz dosahoval najnižšie hodnoty metriky Recall. V Tab. 18 sú modely, ktoré odhalili najvyšší počet anomálií.

method	contaminatio n_value	n_estimator	n_bin	neighbors_va lue	metric_value	TP	FP	FN	TN	Recall
INNE	0,05	100				14	3935	1	82025	0,933333
INNE	0,05	150				14	3512	1	82448	0,933333
INNE	0,05	250				14	4167	1	81793	0,933333
INNE	0,05	300				14	4202	1	81758	0,933333
IForest	0,05	50				13	4141	2	81819	0,866667
IForest	0,05	150				13	4278	2	81682	0,866667
IForest	0,05	400				13	4278	2	81682	0,866667
INNE	0,05	50				13	3860	2	82100	0,866667
INNE	0,05	200				13	3926	2	82034	0,866667
INNE	0,05	350				13	3924	2	82036	0,866667
INNE	0,05	400				13	3316	2	82644	0,866667
INNE	0,05	450				13	3745	2	82215	0,866667
LODA	0,05		40			13	4152	2	81808	0,866667
LOF	0,05			150	braycurtis	13	1203	2	84757	0,866667

Tab. 18 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z1

Modely metódy INNE trénované nad dátami z agregačnej funkcie z2 dosiahli hodnotu Recall metriky rovnú 1. LOF, PCA a IForest modely neodhalili jednu prípadne dve anomálie. Podrobné informácie o parametroch týchto funkcií a ich dosiahnutých Recall hodnôt vidíme v Tab. 19.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
INNE	0,05	50				15	4189	0	81771	1
INNE	0,05	100				15	3926	0	82034	1
INNE	0,05	150				15	3559	0	82401	1
INNE	0,05	200				15	4259	0	81701	1
INNE	0,05	250				15	3578	0	82382	1
INNE	0,05	300				15	4259	0	81701	1
INNE	0,05	350				15	3793	0	82167	1
INNE	0,05	400				15	3978	0	81982	1
INNE	0,05	450				15	3788	0	82172	1
IForest	0,05	100				14	4062	1	81898	0,933333
IForest	0,05	150				14	4071	1	81889	0,933333
IForest	0,05	250				14	4015	1	81945	0,933333
IForest	0,05	400				14	4282	1	81678	0,933333
IForest	0,05	450				14	4240	1	81720	0,933333
PCA	0,05					14	4158	1	81802	0,933333
IForest	0,05	50				13	4105	2	81855	0,866667
IForest	0,05	200				13	3929	2	82031	0,866667
IForest	0,05	300				13	4171	2	81789	0,866667
IForest	0,05	350				13	3907	2	82053	0,866667
LOF	0,05			150	canberra	13	1113	2	84847	0,866667

Tab. 19 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z2

Pri agregačnej funkcii z3 dosiahli dosiahli modely podobné skóre Recall metriky ako pri agregácii z2. INNE identifikoval všetkých 15 anomálií, a modeli PCA, LOF, IForest identifikovali 14 prípadne 13 anomálií správne. Modely v Tab. 20 sú usporiadané podľa metriky Recall.

method	contamination_value	n_estimator	n_bin	neighbors_value	metric_value	TP	FP	FN	TN	Recall
INNE	0,05	50				15	4262	0	81698	1
INNE	0,05	100				15	3987	0	81973	1
INNE	0,05	150				15	3846	0	82114	1
INNE	0,05	200				15	4247	0	81713	1
INNE	0,05	250				15	3483	0	82477	1
INNE	0,05	300				15	4108	0	81852	1
INNE	0,05	350				15	3880	0	82080	1
INNE	0,05	400				15	4036	0	81924	1
INNE	0,05	450				15	3880	0	82080	1
IForest	0,05	250				14	4060	1	81900	0,933333
IForest	0,05	300				14	4252	1	81708	0,933333
IForest	0,05	450				14	4281	1	81679	0,933333
PCA	0,05					14	4158	1	81802	0,933333
IForest	0,05	50				13	4151	2	81809	0,866667
IForest	0,05	100				13	4122	2	81838	0,866667
IForest	0,05	150				13	3959	2	82001	0,866667
IForest	0,05	200				13	3942	2	82018	0,866667
IForest	0,05	350				13	3907	2	82053	0,866667
IForest	0,05	400				13	3951	2	82009	0,866667
LOF	0,05			150	canberra	13	1113	2	84847	0,866667

Tab. 20 Modely s najvyššou hodnotu Recall trénované nad dátami funkcie z3

Po preskúmaní úspešnosti modelov cez všetky agregačné funkcie môžeme vyvodiť nasledujúce fakty:

- Modely metódy INNE mali najvyššie hodnoty metriky Recall pri agregačných funkciách sum, max, freq1, z1, z2 a z3.
- COPOD, ECOD a PCA modely mali najvyššie hodnoty Recall pri agregačných funkciách max, mean a freq2.
- LODA modely sa naprieč agregačnými funkciami umiestňovali v rámci hodnoty Recall pravidelne nízko, až na prípad agregačnej funkcie max.

Podrobnejšie výsledky sú v prílohe vo forme súboru s názvom *The Stolen Szechuan Sauce.xlsx*.

5.2 Neurónová sieť

Pre prípad The Stolen Szechuan Sauce si vieme overiť funkčnosť neurónovej siete pomocou rekonštruovanej hodnoty loss funkcie. Ako bolo popísané v časti 534.4, ide

o MSE. Výstupom neurónovej siete bol Dataframe, ktorý obsahoval inody a k nim hodnoty MSE. Nakoľko poznáme inody, ktoré sú forenznou analýzou označené ako anomálne, zobrazíme len hodnoty týchto inodov a ich umiestnenie v pôvodnom DataFrame. V Tab. 21 Výsledky neurónovej siete máme výsledky predikcií pre 15 identifikovaných inodov. V prvom stĺpci sa nachádza hodnota poradia v pôvodnom DataFrame, ktorý bol usporiadaný od najvyššej hodnoty podľa *score*. Žltou farbou je vyznačený prvý výskyt inodu v dataframe. Vidíme, že 10 inodov malo hodnoty score z neurónovej siete také, že ich umiestnenie bolo do tisíceho miesta. Keď si to porovnáme s veľkosťou vstupného časového radu, ktorý mal cez osemsto tísíc záznamov, tak neurónová sieť úspešne ohodnotila anomalitu inodov. Avšak inody 86966, 87059, 87111, 86967 a 84880 boli neurónovou sieťou ohodnotené o dosť nižšie ako zvyšné atribúty.

	inode	score
0	86968	0,444297582
2	87137	0,434488237
23	86971	0,40722248
57	87137	0,38467443
85	86975	0,370207608
86	84630	0,37020725
87	87060	0,370207012
89	86971	0,370206416
91	86971	0,370206028
95	84630	0,370202094
108	86968	0,370110512
111	86968	0,369972736
113	87112	0,369875163
114	87112	0,369868815
116	86968	0,369782418
122	86971	0,36962378
606	86970	0,346918434
917	84630	0,333216906
918	87112	0,333206445
921	86971	0,333195657
923	86968	0,333185047
924	87064	0,33317697
926	86975	0,333171308
930	87060	0,333167881
935	84987	0,33316505
...	...	...
5143	86966	0,222227171
5201	86966	0,222211927

5232	86968	0,222111091
5259	87112	0,221909419
5260	87059	0,221906736
5294	86971	0,221594557
5910	87059	0,215447351
7007	84987	0,197358161
7021	84987	0,197183535
7870	87111	0,186220959
7974	86967	0,185098723
8829	87111	0,176733211
12429	84987	0,160241678
12726	84987	0,159349516
20316	84880	0,141152412

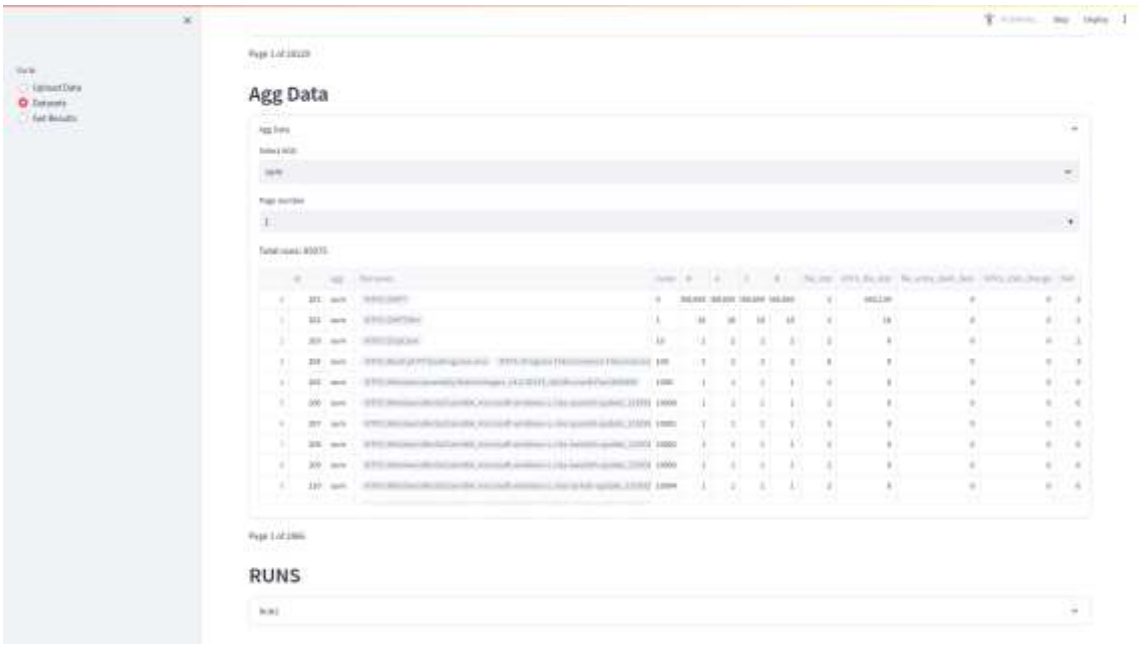
Tab. 21 Výsledky neurónovej siete

5.3 Aplikácia

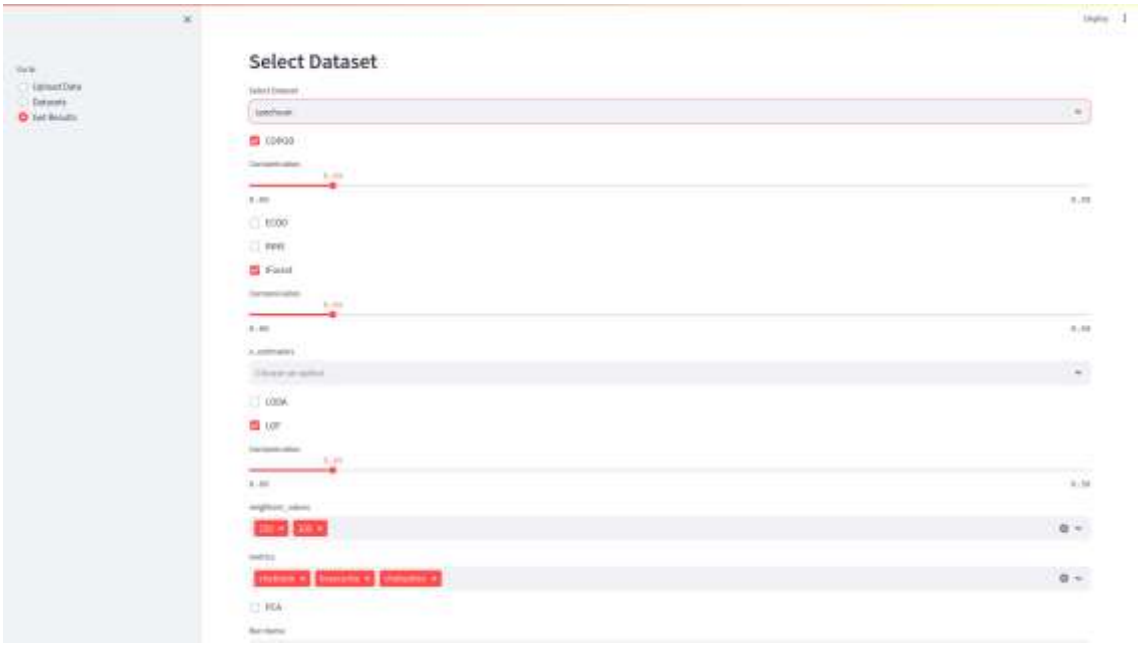
Vďaka získaným poznatkom z tejto práce sme pripravili webovú aplikáciu. Jej cieľom je automatizovane identifikovať relevantné digitálne stopy. Aplikácia bola vyvynutá pomocou dvoch python frameworkov.

Django [75] framework je postavený na jazyku Python a umožňuje prípravu rest servera. Dáta, s ktorými backend pracuje, sa ukladajú v PostgreSQL[76] databáze. S frontendom komunikuje pomocou http requestov. Umožňuje nahrať dáta do databázy, pre rýchlu prácu s nimi. Následne dokáže generovať agregované dáta. Pomocou konfiguračného listu generuje výsledky pre jednotlivé modely, a počíta pre nich sumárne finálne skóre. Oproti práci navyše k výstupu pridáva k inodom aj názvy súborov, ktoré sú s daným inodom prepojené.

Streamlit [77] je framework, ktorý umožňuje pomocou jazyka python programovať frontend webové stránky. Je tvorený komponentmi, ktoré ľahko zobrazia napríklad Dataframy a ploty. Jeho jednoduchý prístup oceňujú často dátoví analytici.



Obr. 9 Agregované dáta zobrazené v aplikácii



Obr. 10 Definovanie modelov v aplikácii

Obr. 11 Výsledky modelov zobrazené v applikácii

Záver

V rámci tejto diplomovej práce sme sa venovali otázke automatizácie digitálnej forenznej analýzy. S nárastom bezpečnostných hrozieb, narastá aj množstvo a sofistikovanosť bezpečnostných incidentov. To si vyžaduje nemalé technické a personálne zdroje na riešenie týchto incidentov. Automatizácia procesu digitálnej forenznej analýza ako aj samotnej reakcie na bezpečnostný incident predstavuje výrazný krok k zvýšeniu bezpečnosti organizácií.

Predložená práca mala tri ciele, ktoré sme splnili a výsledky uvádzame v rámci jednotlivých kapitol. Prvým cieľom práce bolo analyzovať forenzné artefakty v operačnom systéme Windows. Tomuto cieľu sme sa venovali bližšie v podkapitole 1.2. Popísali sme viacero forenzných artefaktov, najmä tie, ktoré využívame v rámci práce na detekciu anomálii, a to Master File Table a záznamy udalostí.

Druhým cieľom tejto záverečnej práce bolo porovnať existujúce prístupy k identifikácii relevantných digitálnych stôp pri forenznom vyšetrovaní operačného systému Windows. Tomuto cieľu sme sa bližšie venovali podkapitole 2.6. Podrobnejšie sa venuje viacerým odborným a vedeckým článkom z oblastí detekcie anomálií vo forenzných artefaktov operačného systému Windows, ako aj samotnej automatizácii digitálnej forenznej analýzy pomocou metód strojového učenia.

Napokon tretím cieľom bolo navrhnúť model pre identifikáciu relevantných digitálnych stôp pri forenznom vyšetrovaní operačného systému Windows, implementovať model a zhodnotiť efektívnosť tohto modelu. Tomuto cieľu sme venovali väčšinu predloženej práce a tvorí jadro samotnej práce. Postupne je tento cieľ rozobraný v kapitolách 3 až 5. V kapitole 3 a 4 sa podrobnejšie venuje návrhu a implementácii systému. Kapitola 5 predstavuje vyhodnotenie tohto systému na prípade The Stolen Szechuan Sauce Case.

Samotné riešenie identifikácie relevantných digitálnych stôp vychádza z predpokladu, že pri určitom správaní v rámci operačného systému, dochádza k vytváraniu digitálnych stôp, ktoré majú určité znaky. V prípade bezpečnostného incidentu je do týchto údajov zanesená anomália, ktorá sa vymyká vzoru bežného správania. Na základe tohto predpokladu sme postavili návrh systému na identifikáciu relevantných digitálnych stôp pre vyšetrovaný prípad, resp. bezpečnostný incident. Teda vychádzame z toho, že hľadaním anomálie (outliera) identifikujeme potenciálne relevantnú digitálnu stopu.

Vzhľadom na rozsah a množstvo forenzných artefaktov v operačnom systéme Windows sme sa zamerali len na forenzné artefakty súborového systému. Niektoré metódy sme skúšali na záznamoch udalostí, ale vzhľadom na rozsah a komplexnosť daného problému sme ostali len pri tomto forenznom artefakte.

Pri samotnom návrhu a implementácii sme sa streli s nasledujúcimi problémami a výzvami. Modelovanie neurónovej siete vyžadovalo viaceré pokusy pre identifikovanie správnej architektúry siete. Častým javom bolo pretrénovanie alebo prílišné zníženie dimenzií. Pri spájaní modelov do súhrnej automatizovanej detekcie anomálií bolo dôležité navrhnúť finálne skóre, a v tejto časti vidíme priestor na zlepšenie.

Ako sme už vyššie uviedli, automatizácia digitálnej forenznej analýzy predstavuje obrovskú výskumnú a vývojovú oblasť, ktorá sa nedá obsiahnuť do jednej záverečnej práce. V rámci práce sme vedeli pokryť určité aspekty skúmanej problematiky. V rámci danej oblasti je možné ďalej pokračovať, a to najmä pri identifikácii relevantných digitálnych stôp u iných forenzných artefaktov (záznamy udalostí, register operačného systému Windows, prefetch a pod.). V práci sme primárne vychádzali z tzv. tabuľkových dát. Ďalším možným smerovaním je reprezentácia dát v podobe grafu alebo časového radu. To by umožnilo aplikovať ďalšie modely, ktoré pracujú s týmito reprezentáciami dát.

Zoznam použitej literatúry

[1] *Digital Forensics Basics*. Accessed: Apr. 25, 2024. [Online]. Available: https://link.springer.com/book/10.1007/978-1-4842-3838-7

[2] "Digitálna forenzná analýza." Accessed: Jun. 27, 2023. [Online]. Available: https://istrosec.com/sk/service/digital-forensics/

[3] "A Framework for Digital Forensic Science," DFRWS. Accessed: Apr. 26, 2024. [Online]. Available: https://dfrws.org/presentation/a-framework-for-digital-forensic-science/

[4] G. Johansen, *Digital Forensics and Incident Response*. Packt Publishing Ltd, 2017.

[5] P. Sokol, L. Bačo, and T. Bajtoš, "Digitálna forenzná analýza I.".

[6] "2004_USA_paper-an_event-based_digital_forensic_investigation_framework.pdf." Accessed: Apr. 26, 2024. [Online]. Available: https://dfrws.org/wp-content/uploads/2019/06/2004_USA_paper-an_event-based_digital_forensic_investigation_framework.pdf

[7] A. Årnes, *Digital Forensics*. John Wiley & Sons, 2017.

[8] B. Carrier, *File System Forensic Analysis*. Addison-Wesley Professional, 2005.

[9] *Fundamentals of Digital Forensics*. Accessed: Apr. 25, 2024. [Online]. Available: https://link.springer.com/book/10.1007/978-3-319-96319-8

[10] O. Skulkin and S. de Courcier, *Windows Forensics Cookbook*. Packt Publishing Ltd, 2017.

[11] C. Gurkok, "Chapter 41 - Cyber Forensics and Incidence Response," in *Computer and Information Security Handbook (Third Edition)*, J. R. Vacca, Ed., Boston: Morgan Kaufmann, 2017, pp. 603–628. doi: 10.1016/B978-0-12-803843-7.00041-7.

[12] R. D. Pittman and D. Shaver, "Chapter 5 - Windows Forensic Analysis," in *Handbook of Digital Forensics and Investigation*, E. Casey, C. Altheide, C. Daywalt, A. de Donno, D. Forte, J. O. Holley, A. Johnston, R. van der Knijff, A. Kokocinski, P. H. Luehr, T. Maguire, R. D. Pittman, C. W. Rose, J. J. Schwerha, D. Shaver, and J. R. Smith, Eds., San Diego: Academic Press, 2010, pp. 209–300. doi: 10.1016/B978-0-12-374267-4.00005-7.

[13] C. Hargreaves and J. Patterson, “An automated timeline reconstruction approach for digital forensic investigations,” *Digital Investigation*, vol. 9, pp. S69–S79, Aug. 2012, doi: 10.1016/j.diin.2012.05.006.

[14] “CRZP - detail kniha.” Accessed: Jun. 27, 2023. [Online]. Available: https://opac.crzp.sk/?fn=detailBiblioFormChildA21P&sid=A68E927B0776E61A8D5994111187&seo=CRZP-detail-kniha

[15] SENTUREAN, “NTFS Timestamp changes on Windows 10.” Accessed: Apr. 29, 2024. [Online]. Available: https://www.senturean.com/posts/19_04_22_win10_ntfs_time_rules/

[16] Karl-Bridge-Microsoft, “Event Types - Win32 apps.” Accessed: Jun. 27, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/eventlog/event-types

[17] M. Ölvecký and M. Host’ovecký, “Digital image forensics using EXIF data of digital evidence,” in *2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Nov. 2021, pp. 282–286. doi: 10.1109/ICETA54173.2021.9726649.

[18] P. Alvarez, “Metering Mode: center weight,” vol. 2, no. 3, 2004.

[19] P. Froklage, “Analyze LNK Files - LNK Are Valuable Artifacts,” Magnet Forensics. Accessed: Apr. 26, 2024. [Online]. Available: https://www.magnetforensics.com/blog/forensic-analysis-of-lnk-files/

[20] P. Froklage, “Forensic Analysis of Windows Shellbags,” Magnet Forensics. Accessed: Apr. 26, 2024. [Online]. Available: https://www.magnetforensics.com/blog/forensic-analysis-of-windows-shellbags/

[21] R. Mize, “Behavior of Shellbags in Windows 10,” M.S., Utica College, United States -- New York. Accessed: Apr. 29, 2024. [Online]. Available: https://www.proquest.com/docview/2108990957/abstract/79DCEF9383DB4384PQ/1

[22] Y. Khatri, “Forensic implications of System Resource Usage Monitor (SRUM) data in Windows 8,” *Digital Investigation*, vol. 12, pp. 53–65, Mar. 2015, doi: 10.1016/j.diin.2015.01.002.

[23] A. Patcha and J.-M. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007, doi: 10.1016/j.comnet.2007.02.001.

[24] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[25] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen, “ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12181–12193, Dec. 2023, doi: 10.1109/TKDE.2022.3159580.

[26] C. C. Aggarwal, “Outlier Analysis,” in *Data Mining: The Textbook*, C. C. Aggarwal, Ed., Cham: Springer International Publishing, 2015, pp. 237–263. doi: 10.1007/978-3-319-14142-8_8.

[27] “Distance Metrics.” Accessed: Jan. 25, 2024. [Online]. Available: https://numerics.mathdotnet.com/Distance

[28] Tim, “Jaccard Index / Similarity Coefficient,” Statistics How To. Accessed: Apr. 14, 2024. [Online]. Available: https://www.statisticshowto.com/jaccard-index/

[29] “scipy.spatial.distance.sqeuclidean — SciPy v1.13.0 Manual.” Accessed: Apr. 14, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.sqeuclidean.html

[30] “scipy.spatial.distance.braycurtis — SciPy v1.13.0 Manual.” Accessed: Apr. 14, 2024. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.braycurtis.html

[31] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “LOF: identifying density-based local outliers,” in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, in SIGMOD ’00. New York, NY, USA: Association for Computing Machinery, May 2000, pp. 93–104. doi: 10.1145/342009.335388.

[32] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, “Outlier Detection in Axis-Parallel Subspaces of High Dimensional Data,” in *Advances in Knowledge Discovery and Data Mining*, T. Theeramunkong, B. Kijsirikul, N. Cercone, and T.-B. Ho, Eds.,

Berlin, Heidelberg: Springer, 2009, pp. 831–838. doi: 10.1007/978-3-642-01307-2_86.

[33] Y. Almardeny, N. Boujnah, and F. Cleary, "A Novel Outlier Detection Method for Multivariate Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4052–4062, Sep. 2022, doi: 10.1109/TKDE.2020.3036524.

[34] Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu, "COPOD: Copula-Based Outlier Detection," in *2020 IEEE International Conference on Data Mining (ICDM)*, Sorrento, Italy: IEEE, Nov. 2020, pp. 1118–1123. doi: 10.1109/ICDM50108.2020.00135.

[35] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Comput*, vol. 22, no. 1, pp. 949–961, Jan. 2019, doi: 10.1007/s10586-017-1117-8.

[36] R. Jain, "LSTM go_backwards() — Unravelling its 'hidden' secrets," Medium. Accessed: Apr. 30, 2024. [Online]. Available: https://towardsdatascience.com/lstm-go-backwards-unravelling-its-hidden-secrets-ed094952b5cc

[37] B.-S. Kwon, R.-J. Park, and K.-B. Song, "Short-Term Load Forecasting Based on Deep Neural Networks Using LSTM Layer," *J. Electr. Eng. Technol.*, vol. 15, no. 4, pp. 1501–1509, Jul. 2020, doi: 10.1007/s42835-020-00424-7.

[38] "Figure 1. Architecture of LSTM autoencoder.," ResearchGate. Accessed: Apr. 30, 2024. [Online]. Available: https://www.researchgate.net/figure/Architecture-of-LSTM-autoencoder_fig1_352898971

[39] D. Samariya and A. Thakkar, "A Comprehensive Survey of Anomaly Detection Algorithms," *Ann. Data. Sci.*, vol. 10, no. 3, pp. 829–850, Jun. 2023, doi: 10.1007/s40745-021-00362-9.

[40] C. C. Aggarwal, "Outlier ensembles: position paper," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, pp. 49–58, Apr. 2013, doi: 10.1145/2481244.2481252.

[41] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, Pisa, Italy: IEEE, Dec. 2008, pp. 413–422. doi: 10.1109/ICDM.2008.17.

[42] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells, "Isolation-based anomaly detection using nearest-neighbor ensembles,"

Computational Intelligence, vol. 34, no. 4, pp. 968–998, Nov. 2018, doi: 10.1111/coin.12156.

[43] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Mach Learn*, vol. 102, no. 2, pp. 275–304, Feb. 2016, doi: 10.1007/s10994-015-5521-0.

[44] X. Du and M. Scanlon, "Methodology for the Automated Metadata-Based Classification of Incriminating Digital Forensic Artefacts," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, in ARES '19. New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 1–8. doi: 10.1145/3339252.3340517.

[45] F. Skopik, M. Wurzenberger, and M. Landauer, "Detecting Unknown Cyber Security Attacks Through System Behavior Analysis," in *Cybersecurity of Digital Service Chains: Challenges, Methodologies, and Tools*, J. Kołodziej, M. Repetto, and A. Duzha, Eds., Cham: Springer International Publishing, 2022, pp. 103–119. doi: 10.1007/978-3-031-04036-8_5.

[46] N. Beebe, L. Liu, and Z. Ye, "Insider Threat Detection Using Time-Series-Based Raw Disk Forensic Analysis," in *Advances in Digital Forensics XIII*, vol. 511, G. Peterson and S. Shenoi, Eds., in IFIP Advances in Information and Communication Technology, vol. 511. , Cham: Springer International Publishing, 2017, pp. 149–167. doi: 10.1007/978-3-319-67208-3_9.

[47] B. D. Carrier and E. H. Spafford, "Automated Digital Evidence Target Definition Using Outlier Analysis and Existing Evidence," 2005.

[48] M. Pirker, P. Kochberger, and S. Schwandter, "Behavioural Comparison of Systems for Anomaly Detection," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, in ARES '18. New York, NY, USA: Association for Computing Machinery, Aug. 2018, pp. 1–10. doi: 10.1145/3230833.3230852.

[49] X. Du, Q. Le, and M. Scanlon, "Automated Artefact Relevancy Determination from Artefact Metadata and Associated Timeline Events," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Jun. 2020, pp. 1–8. doi: 10.1109/CyberSecurity49315.2020.9138874.

[50] "A comparative evaluation of two algorithms for Windows Registry Anomaly Detection - IOS Press." Accessed: Apr. 30, 2024. [Online]. Available: https://content.iospress.com/articles/journal-of-computer-security/jcs242

[51] A. Tajoddin and M. Abadi, "RAMD: registry-based anomaly malware detection using one-class ensemble classifiers," *Appl Intell*, vol. 49, no. 7, pp. 2641–2658, Jul. 2019, doi: 10.1007/s10489-018-01405-0.

[52] A. S. Chouhan *et al.*, "An Ensemble Approach for Modeling Process Behavior and Anomaly Detection," in *Proceedings of Emerging Trends and Technologies on Intelligent Systems*, A. Noor, A. Sen, and G. Trivedi, Eds., Singapore: Springer, 2022, pp. 165–177. doi: 10.1007/978-981-16-3097-2_14.

[53] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, in SOSP '09. New York, NY, USA: Association for Computing Machinery, Oct. 2009, pp. 117–132. doi: 10.1145/1629575.1629587.

[54] H. Studiawan, F. Sohel, and C. Payne, "A survey on forensic investigation of operating system logs," *Digital Investigation*, vol. 29, pp. 1–20, Jun. 2019, doi: 10.1016/j.diin.2019.02.005.

[55] H. Studiawan, C. Payne, and F. Sohel, "Graph clustering and anomaly detection of access control log for forensic purposes," *Digital Investigation*, vol. 21, pp. 76–87, Jun. 2017, doi: 10.1016/j.diin.2017.05.001.

[56] H. Studiawan and F. Sohel, "Performance Evaluation of Anomaly Detection in Imbalanced System Log Data," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, Jul. 2020, pp. 239–246. doi: 10.1109/WorldS450073.2020.9210329.

[57] "Anomaly detection in a forensic timeline with deep autoencoders - ScienceDirect." Accessed: Apr. 30, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212621002076?casa_token=WuO0jKb1wOcAAAAA:sNJ6hig-KzOhrNmkFePMLq2p2eRTnJgnvt2346PJ8abgyce_rs9YQ3BYSixXKwrCTnB9-6KXLiI

[58] “Anomaly-Based Insider Threat Detection Using Deep Autoencoders | IEEE Conference Publication | IEEE Xplore.” Accessed: Apr. 30, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8637390?casa_token=iFfKHFcmjDMAAAAA:jBPjpuoj1uhJoaa91Kp6sr1Lxvle4FkZWR1_9UX7AC-CVaLXyZKGzwHIp7eJ7RMp4s_dv85KJPoIFw

[59] L.-P. Yuan, E. Choo, T. Yu, I. Khalil, and S. Zhu, “Time-Window Based Group-Behavior Supported Method for Accurate Detection of Anomalous Users,” in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Jun. 2021, pp. 250–262. doi: 10.1109/DSN48987.2021.00038.

[60] “Experience Report: System Log Analysis for Anomaly Detection | IEEE Conference Publication | IEEE Xplore.” Accessed: Apr. 30, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7774521?casa_token=BaSZhA9Bq0wAAAAA:PhlnrQWaznBe376cCRKn4eXNae60yCAyHFSkT5NjaXdzzdyLPgggvm0kW1Y7M_7lLKnuftquPwf4EA

[61] R. Hirakawa, H. Uchida, A. Nakano, K. Tominaga, and Y. Nakatoh, “Large scale log anomaly detection via spatial pooling,” *Cognitive Robotics*, vol. 1, pp. 188–196, Jan. 2021, doi: 10.1016/j.cogr.2021.10.001.

[62] James, “Case 001 - The Stolen Szechuan Sauce,” DFIR Madness. Accessed: Jan. 25, 2024. [Online]. Available: https://dfirmadness.com/the-stolen-szechuan-sauce/

[63] T. ÖSTERHOLM and T. SUNDMARK, “A comparison of the file systems used in RTLinux and Windows CE.”, [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1b7f0506240ac19ef62d8c9cc5c2573455c1eba1

[64] “Magnet CTF 2020 Windows Desktop, 2020.” [Online]. Available: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2020 CTF - Windows.zip

[65] “CFReDS - Data Leakage Case.” Accessed: Apr. 14, 2024. [Online]. Available: https://cfreds-archive.nist.gov/data_leakage_case/data-leakage-case.html

[66] “CFReDS Portal.” Accessed: Apr. 14, 2024. [Online]. Available: https://cfreds.nist.gov/all/NIST/HackingCase

[67] “log2timeline/plaso.” log2timeline, Apr. 12, 2024. Accessed: Apr. 14, 2024. [Online]. Available: https://github.com/log2timeline/plaso

[68] “Welcome to the Plaso documentation — Plaso (log2timeline) 20240409 documentation.” Accessed: Apr. 14, 2024. [Online]. Available: https://plaso.readthedocs.io/en/latest/

[69] “pandas - Python Data Analysis Library.” Accessed: Jan. 25, 2024. [Online]. Available: https://pandas.pydata.org/

[70] “pyod 1.1.0 documentation.” Accessed: Jun. 27, 2023. [Online]. Available: https://pyod.readthedocs.io/en/latest/index.html

[71] “TensorFlow.” Accessed: Jan. 25, 2024. [Online]. Available: https://www.tensorflow.org/

[72] “pandas.DataFrame.groupby — pandas 2.2.0 documentation.” Accessed: Jan. 25, 2024. [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html

[73] James, “Answers to the Case of the Stolen Szechuan Sauce (Case001),” DFIR Madness. Accessed: Apr. 30, 2024. [Online]. Available: https://dfirmadness.com/answers-to-szechuan-case-001/

[74] E. Marková, P. Sokol, and K. Kováćová, “Detection of relevant digital evidence in the forensic timelines,” in *2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Jun. 2022, pp. 1–7. doi: 10.1109/ECAI54874.2022.9847438.

[75] “Django,” Django Project. Accessed: Apr. 30, 2024. [Online]. Available: https://www.djangoproject.com/

[76] P. G. D. Group, “PostgreSQL,” PostgreSQL. Accessed: Apr. 30, 2024. [Online]. Available: https://www.postgresql.org/

[77] “Streamlit • A faster way to build and share data apps.” Accessed: Apr. 30, 2024. [Online]. Available: https://streamlit.io/

Prílohy

Príloha A: Funkcia get_aggregated_data()

Príloha B: Konfiguračný zoznam pre funkciu automatizácie detekcie anomálií

Príloha C: Implementácia metódy detekcie anomálií naprieč modelmi

Príloha D: Fukcia calculate_acc() pre výpočet metrík úspešností

Príloha E: Získanie výsledkov a export do .xlsx súboru

Príloha F: Implementácia generátora a neurónovej siete

Príloha A

```
def get_aggregated_data(data, agg_function, group_by='inode'):

    if agg_function == 'sum':
        result = data.groupby([group_by]).sum()

    if agg_function == 'max':
        result = data.groupby([group_by]).max()

    if agg_function == 'mean':
        result = data.groupby([group_by]).mean()

    if agg_function == 'freq1':
        a_power_k = data.groupby(group_by).sum(numeric_only=True)
        result = a_power_k.div(a_power_k.sum()).fillna(0)

    if agg_function == 'freq2':
        a_power_k = data.groupby(group_by).sum(numeric_only=True)
        result = a_power_k.div(data.groupby(group_by).count()).fillna(0)

    if agg_function == 'z1':
        a_power_k = data.groupby(group_by).sum(numeric_only=True)
        bar_a_2 = a_power_k.sum() / len(a_power_k)
        s_pow_2 = \
            a_power_k.sub(bar_a_2).pow(2).sum().mul(1 / (len(a_power_k)-1))
        s = s_pow_2.pow(1/2)
        result = (a_power_k - bar_a_2).div(s).fillna(0)

    if agg_function == 'z2':
        a_power_k = data.groupby(group_by).sum(numeric_only=True)
        tilde_a_2 = a_power_k.median()
        quartiles__a_power_k = a_power_k.quantile([0.25, 0.5, 0.75])
        quartiles__a_power_k.loc['iqr'] =\
            quartiles__a_power_k.loc[0.75] - quartiles__a_power_k.loc[0.25]
        iqr = quartiles__a_power_k.loc['iqr']
        result = (a_power_k - tilde_a_2).div(iqr).fillna(0)

    if agg_function == 'z3':
        a_power_k = data.groupby(group_by).sum(numeric_only=True)
        tilde_a_2 = a_power_k.median()
        MAD = (a_power_k - tilde_a_2).abs().median()
        result = (a_power_k - tilde_a_2).div(MAD).fillna(0)

    return result
```

Príloha B

```
models = [
        {'name': 'COPOD',
         'function': pyod.models.copod.COPOD,
         'contamination_values': [0.05],
         'outlier_value': 1},

        {'name': 'ECOD',
         'function': pyod.models.ecod.ECOD,
         'contamination_values': [0.05],
         'outlier_value': 1},

        {'name': 'PCA',
         'function': pyod.models.pca.PCA,
         'contamination_values': [0.05],
         'outlier_value': 1},

        {'name': 'INNE',
         'function': pyod.models.inne.INNE,
         'contamination_values': [0.05],
         'n_estimators': [50, 100, 150, 200, 250, 300, 350, 400, 450],
         'outlier_value': 1},

        {'name': 'IForest',
         'function': pyod.models.iforest.IForest,
         'contamination_values': [0.05],
         'n_estimators': [50, 100, 150, 200, 250, 300, 350, 400, 450],
         'outlier_value': 1},

        {'name': 'LOF',
         'function': pyod.models.lof.LOF,
         'contamination_values': [0.05],
         'neighbors_values': [50, 100, 150, 200],
         'metrics': [
         'canberra','jaccard',
                 'chebyshev','braycurtis',
                 'cityblock','correlation',
                 'cosine','euclidean',
                 'minkowski','sqeuclidean'
          ],
         'outlier_value': 1},

        {'name': 'LODA',
         'function': pyod.models.loda.LODA,
         'contamination_values': [0.05],
         'n_bins': [20, 40, 'auto'],
         'outlier_value': 1}
]
```

Príloha C

```
def outlier_detection(agg_data, method, agg):
  results = []

  name = method['name']
  if name == 'COPOD' or name == 'ECOD' or name == 'PCA':
    for contamination_value in method['contamination_values']:
      model1 = method['function'](contamination=contamination_value)
      model1.fit(agg_data)
      outlier_index = np.where(model1.labels_ == method['outlier_value'])
      labels = model1.labels_
      if method['outlier_value'] == 1:
        labels = [item * -1 for item in labels]
      data = {
        'name': f'{name}-{agg}-cont-{contamination_value}',
        'method': name,
        'labels': labels,
        'score': model1.decision_scores_,
        'outlier_index': outlier_index[0],
        'outlier_count': len(outlier_index[0]),
        'contamination_value': contamination_value
      }
      results.append(data)

  elif name == 'INNE' or name == 'IForest':
    for contamination_value in method['contamination_values']:
      for n_estimator in method['n_estimators']:
        model1=method['function'](n_estimators=n_estimator,\
          contamination=contamination_value)
        model1.fit(agg_data)
        outlier_index = np.where(model1.labels_ == method['outlier_value'])
        labels = model1.labels_
        if method['outlier_value'] == 1:
          labels = [item * -1 for item in labels]
        min_value = min(model1.decision_scores_)

        data = {
          'name': f'{name}-{agg}-cont-{contamination_value}-nestmtr-\
                    {n_estimator}',
          'method': name,
          'labels': labels,
          'score': model1.decision_scores_ if name == 'INNE' else\
                   model1.decision_scores_ + abs(min_value),
          'outlier_index': outlier_index[0],
          'outlier_count': len(outlier_index[0]),
          'contamination_value': contamination_value,
          'n_estimator': n_estimator
        }


        results.append(data)

  elif name == 'LODA':
    for contamination_value in method['contamination_values']:
      for n_bin in method['n_bins']:
        model1=method['function']\
               (n_bins=n_bin, contamination=contamination_value)
        model1.fit(agg_data)
```

```
      outlier_index = np.where(model1.labels_ == method['outlier_value'])
      labels = model1.labels_
      if method['outlier_value'] == 1:
        labels = [item * -1 for item in labels]

      data = {
        'name': f'{name}-{agg}-cont-{contamination_value}-n_bin-{n_bin}',
        'method': name,
        'labels': labels,
        'score': model1.decision_scores_,
        'outlier_index': outlier_index[0],
        'outlier_count': len(outlier_index[0]),
        'contamination_value': contamination_value,
        'n_bin': n_bin
      }

      results.append(data)


elif name == 'LOF':
  for metric_value in method['metrics']:
    for neighbors_value in method['neighbors_values']:
      for contamination_value in method['contamination_values']:
        model1 = method['function']\
                 (n_neighbors=neighbors_value, metric=metric_value,\
                  contamination=contamination_value, n_jobs=-1)
        model1.fit(agg_data)
        outlier_index = np.where(model1.labels_ == method['outlier_value'])
        labels = model1.labels_
        if method['outlier_value'] == 1:
          labels = [item * -1 for item in labels]
        min_value = min(model1.decision_scores_)

        data = {
          'name': f'{name}-{agg}-mtrcs-{metric_value}-cont-\
                  {contamination_value}-nghbrvl-{neighbors_value}',
          'method': name,
          'labels': labels,
          'score': model1.decision_scores_ + abs(min_value),
          'outlier_index': outlier_index[0],
          'outlier_count': len(outlier_index[0]),
          'metric_value': metric_value,
          'neighbors_value': neighbors_value,
          'contamination_value': contamination_value
        }

        results.append(data)
return results
```

Príloha D

```
def calculate_acc(data, indx_outliers):

  data[f'correct'] = list(filter\
                     (lambda x: x in data[f'outlier_index'], indx_outliers))

  data[f'TP'] = len(data[f'correct'])

  data[f'FP'] = data[f'outlier_count'] - data[f'TP']

  data[f'FN'] = len(indx_outliers) - data[f'TP']

  data[f'TN'] = len(agg_data) - data[f'TP'] - data[f'FP'] - data[f'FN']

  data[f'Recall'] = 0 if data[f'TP'] + data[f'FN'] == 0 else\
                    data[f'TP']/(data[f'TP'] + data[f'FN'])

  data[f'Precision'] = 0 if data[f'TP'] + data[f'FP'] == 0 else \
                       data[f'TP']/(data[f'TP'] + data[f'FP'])

  data[f'F1'] = 0 if data[f'Precision'] + data[f'Recall'] == 0 else \
                (2 * data[f"Precision"] * data[f"Recall"]) / \
                (data[f"Precision"] + data[f"Recall"])

  data[f'TPR'] = data[f'Recall']

  data[f'FPR'] = data[f"FP"] / (data[f"FP"] + data[f"TN"])

  data[f'Accuracy'] = (data[f"TP"] + data[f"TN"]) / \
                      (data[f"TP"] + data[f"TN"] + data[f"FP"] + data[f"FN"])
  return data
```

Príloha E

```
path = './Anomaly-detection-digital-evidence/output/Magnet_CTF_2020_Windows_Desktop_file'
# define functions for coloring sheets
def _color_green_or_transparent(val):
    color = 'lightgreen' if (type(val) == int or float) and val != 0 else 'transparent'
    return 'background-color: %s' % color
def _color_red_or_transparent(val):
    if (type(val) == int or float) and val == -1:
        return f'background-color: red; color: white'
    else:
        return f'background-color: transparent; color: black'
if not os.path.exists(f'{path}/'):
    os.mkdir(f'{path}/')
writer = pd.ExcelWriter(f'{path}/case_study_with_all_methods.xlsx')
for agg in agg_list:
    agg_data = get_aggregated_data(file_bin, agg_function=agg, group_by='inode')
    agg_data_copy = agg_data.copy()
    anomaly_inodes_indeces = []
    for anomaly_date in anomaly_inodes:
        anomaly_inodes_indeces += [agg_data_copy.index.get_loc(anomaly_date)]
    agg_data_copy = agg_data_copy.reset_index(drop=True)
    #find max value of entire data frame
    max_value = np.nanmax(agg_data_copy[agg_data_copy != np.inf])
    #replace inf and -inf in all columns with max value
    agg_data_copy.replace([np.inf, -np.inf], max_value, inplace=True)
    #find max value of entire data frame
    max_value = np.nanmax(agg_data[agg_data != np.inf])
    #replace inf and -inf in all columns with max value
    agg_data.replace([np.inf, -np.inf], max_value, inplace=True)
    # all_results will hold all results from each method and its iterations
    all_results = []
    # iterate over methods
    for method in methods:
        # append new results from all iterations
        result = outlier_detection(agg_data_copy, method, agg)
        for r in result:
            r = calculate_acc(r, anomaly_inodes_indeces)
        all_results += result
    all_results = sorted(all_results, key=lambda d: d['name'])
```

```
    # make copy of original for excel creation
    agg_data_excel = agg_data.copy()
    # get keywords from original dataframe, in order to color excel
    keywords1 = agg_data.columns.to_list()
    final_score = [ 0 for i in range(len(all_results[0]['labels']))]
    method_names = []
    # create dataframe to hold accuracy results
    columns      =      ['method','contamination_value','n_estimator','n_bin','neighbors_value',
'metric_value','TP',FP','FN','TN','Recall','Precision','F1','TPR','FPR','Accuracy']
    acc_results = pd.DataFrame(columns=columns)
    method_name = ''
    # prepare data for non removed
    for result in all_results:
        # add score and labesl to OCEAN sheet
        agg_data_excel[result['name']+'-score'] = result['score']
        agg_data_excel[result['name']+'-labels'] = result['labels']
        # append name of method to methods
        if method_name != result['method']:
            method_name = result['method']
            acc_results.loc[len(acc_results.index)] = [None for i in range(len(columns))]
        method_names.append(result['name'])
        stats = [
            result['method'],
            result['contamination_value'],
            result['n_estimator'] if 'n_estimator' in result.keys() else None,
            result['n_bin'] if 'n_bin' in result.keys() else None,
            result['neighbors_value'] if 'neighbors_value' in result.keys() else None,
            result['metric_value'] if 'metric_value' in result.keys() else None,
            result['TP'], result['FP'],result['FN'], result['TN'],
            result['Recall'],result['Precision'],result['F1'], result['TPR'], result['FPR'],
            result['Accuracy'],
        ]
        acc_results.loc[len(acc_results.index)] = list({key: result[key] if key in result.keys()
else None for key in columns }.values())
        for i in range(len(result['labels'])):
            if result['labels'][i] == -1:
                final_score[i] += 1
    # prepare data for removed zero columns
    subset_to_sort = acc_results.loc[acc_results['method'] == 'IForest']
    # Sort the subset based on the 'n_estimator' column
    sorted_subset = subset_to_sort.sort_values(by='n_estimator')
```

```
    # Reassign the sorted subset back to the original DataFrame using the index
    acc_results.loc[subset_to_sort.index] = sorted_subset.values
    subset_to_sort = acc_results.loc[acc_results['method'] == 'INNE']
    # Sort the subset based on the 'n_estimator' column
    sorted_subset = subset_to_sort.sort_values(by='n_estimator')
    # Reassign the sorted subset back to the original DataFrame using the index
    acc_results.loc[subset_to_sort.index] = sorted_subset.values
    styler = acc_results.style
    counter = 0
    for method_name in acc_results['method'].unique():
        counter += 1
        if counter%2 == 0:
            cmap1='Blues'
            cmap2='Greens'
        else:
            cmap1='Purples'
            cmap2='Oranges'
        styler.background_gradient(axis=0,                  cmap=cmap1,                  vmin=0,
subset=pd.IndexSlice[acc_results['method'] == method_name, ['TP', 'FP', 'FN', 'TN']])\
            .background_gradient(axis=0,                  cmap=cmap2,                  vmin=0,
subset=pd.IndexSlice[acc_results['method'] == method_name, ['Recall', 'Precision', 'F1', 'TPR',
'FPR', 'Accuracy']])\
    keywords2 = agg_data_excel.columns.to_list()
    keywords2 = [item for item in keywords2 if item not in keywords1 and 'score' not in item]
    # drop date, we do not want to color them
    if 'inode' in keywords1: keywords1.remove('inode')
    if 'inode' in keywords2: keywords2.remove('inode')
    # add final score to excel dataframe
    agg_data_excel['final_score'] = final_score
    agg_data_excel = agg_data_excel.sort_values(by='final_score', ascending=False)
    agg_data_excel.style.applymap(_color_green_or_transparent,
subset=keywords1).applymap(_color_red_or_transparent, subset=keywords2)\
        .to_excel(writer, sheet_name=f'{agg}', startrow=0 , startcol=0)
    styler.to_excel(writer, sheet_name=f'{agg}-acc', startrow=0 , startcol=0)
    print(f'created sheet for agg function  {agg}')
    print('----------------------------------')
    with open(f'{path}/check.txt', 'a') as the_file:
        the_file.write(f'created sheet for agg function  {agg}\n')
        the_file.write(f'----------------------------------\n')
writer.close()
print(f'done getting results')
```

Príloha F

Zdrojový kód generátora

```
import tensorflow as tf
window_size = 71
class Generator(tf.keras.utils.Sequence ):
    def __init__(self, data, batch_size, window_size):
        self.data = data
        self.batch_size = batch_size
        self.window_size = window_size
        self.num_samples = len(data) - window_size + 1
    def __len__(self):
        return math.ceil(len(self.data) / self.batch_size)
    def __getitem__(self, idx):
        start_idx = idx * self.batch_size
        end_idx = min(start_idx + self.batch_size, self.num_samples)
        batch_x = []
        batch_y = []
        for i in range(start_idx, end_idx):
            window = self.data[i:i+self.window_size]
            batch_x.append(window[:-1])
            batch_y.append(window[-1])
        return np.array(batch_x), np.array(batch_y)
model_data = Generator(sub_numpy, 1024, window_size)
```

Zdrojový kód prípravy neurónovej siete

```
from tensorflow.keras.layers import Input, LSTM, RepeatVector, Dropout,
TimeDistributed, Dense, Flatten
from tensorflow.keras.models import Model, Sequential

timesteps = window_size - 1  # Number of previous rows to consider
input_dim = 27   # Dimensionality of your input vectors

model = Sequential()
model.add(LSTM(int(27),   activation='relu',   input_shape=(timesteps,   27),
return_sequences=True))
model.add(LSTM(int(27//1.5), activation='relu', return_sequences=True))
model.add(LSTM(int(27//1.5), activation='relu', return_sequences=True))
model.add(LSTM(27, activation='sigmoid', return_sequences=False))
model.compile(optimizer='adam', loss='mse')
model.summary()
```