UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

IDENTIFIKÁCIA RELEVANTNÝCH DIGITÁLNYCH STÔP PRI FORENZNOM VYŠETROVANÍ (ŠTUDENTSKÁ VEDECKÁ A ODBORNÁ ČINNOSŤ)

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

Košice 2024

Bc. František KURIMSKÝ

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.

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.

Obsah

Zoznam ilustrácií

Zoznam tabuliek

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 vykonali 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.

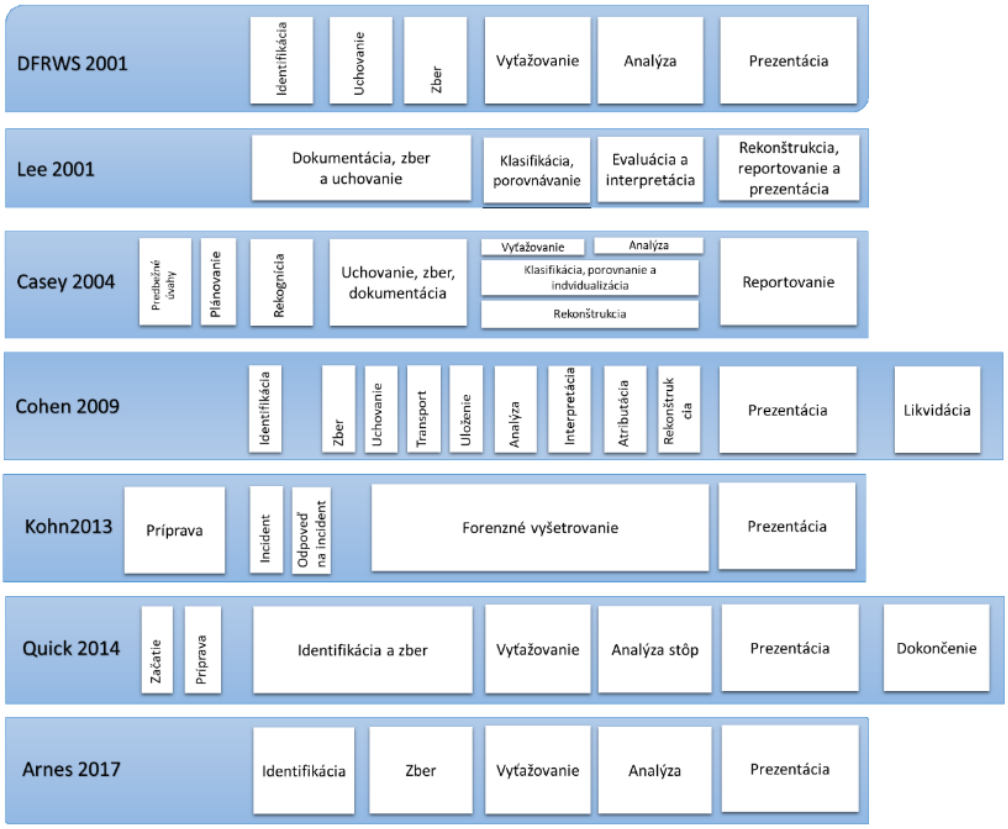
1 Digitálna forenzná analýza

Digitálna forenzná analýza je odvetvie forenzných vied, ktoré na základe vedeckých poznatkov zhromažďuje, analyzuje, dokumentuje a prezentuje digitálne dôkazy súvisiace s počítačovou trestnou činnosťou s účelom použitia týchto dôkazov 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 je 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 6 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 procesu 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 byt nepoužiteľné ako dôkazový materiál pri súdnom konaní.
- Analýza – Po fáze vyťažovania forenzný analytik analyzuje získané dáta a ich vzťah 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 digitálneho vyšetrovania objektívne, a nemal by vyjadriť svoj názor, pokiaľ to nebude žiadané. Závisí to od expertízy daného analytika [4].

V texte vyššie 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 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 a dôkazov a reťazca starostlivosti[7]. Reťazec úschovy 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 úschovy. 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, riešenie je efektívnejšie z pohľadu nákladov a jednoduchšej implementácie [4]. Správny reťazec starostlivosti 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 preskúmane? (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 poskztuje málo informácií o tom, ako tieto artefakty fungujú.

1.2.1 Metadáta

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

1.2.2 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 [10]. V Tab. 1 Príklad EXIF metadát je uvedený typický príklad EXIF metadát v čitateľnej forme [11].

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. 1 Príklad EXIF metadát [11]

1.2.3 Udalosti

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 [12].

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 [13] delíme typ event logov do piatich kategórií.

1. **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.

2. **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.
3. **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.
4. **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.
5. **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 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].

1.2.5 Ďalšie forenzné artefakty

V tejto časti sa pozrieme na ďalsie typy forenzných artefaktov. Sú nimi Shellbagy, .LNK súbory a Thumbcache.

.LNK súbory sú v podstate skratky v operačnom systéme Windows. Tieto súbory sa vytvárajú aj v prípade, keď používateľ otvorí lokálny alebo vzdialený súbor [14]. 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 [14].

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].

Shellbagy sú k dispozícii už od 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 preičinkoch, ktoré boli zo systému odstránené [15].

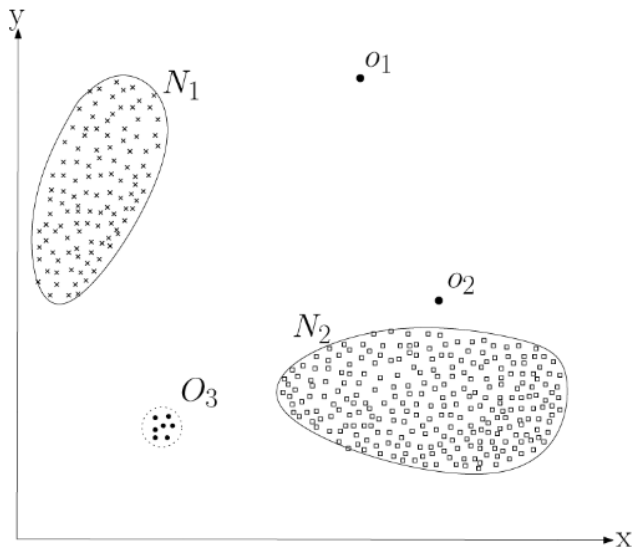
2 Detekcia anomálií

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 [16].

Prístup detekcie 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 [16].

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 [17].

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 [17]. Obr. 2 ukazuje jednoduchý príklad anomálií v dvoj dimenzionálnom datasete.



Obr. 2 Príklad anomálií v dvoj dimenzionálnom datasete [17]

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 len 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.

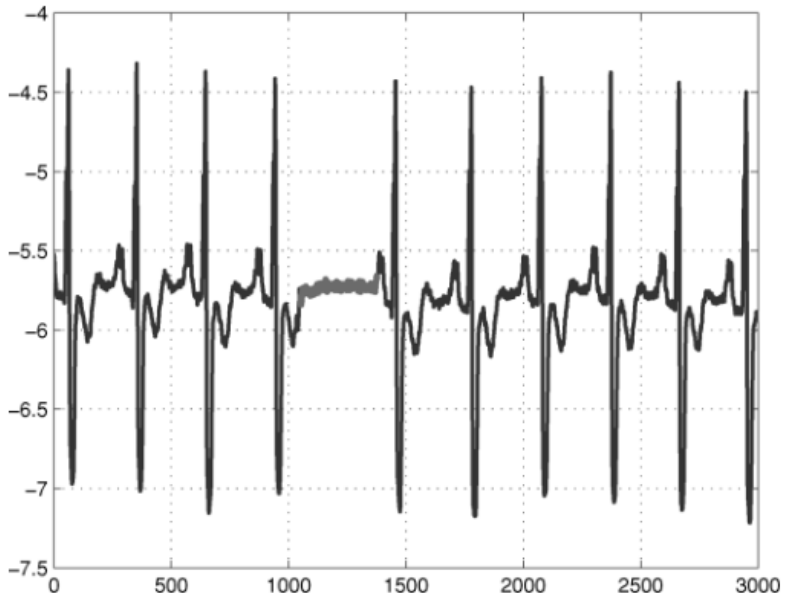
Všeobecne platí, že dátové inštancie môžu byť navzájom prepojené. Niektoré príklady sú sekvencie dát, priestorové dáta a grafy. V sekvenciách dát sú dátové inštancie lineárne usporiadané, napríklad časové rady [17].

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 anomálne 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 prípady v kolektívnej anomálii nemusia byť samy osebe 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 [17].



Obr. 3 Príklad kolektívnej anomálie [17]

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 režime polo supervízie 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 [17].

2.4 Prístupy k detekcii anomálií

V tejto podkapitole preskúmame rôzne architektúry a metódy, ktoré boli navrhnuté pre detekciu anomálií. Patria tu štatistická detekcia anomálií, detekcia anomálií na základ klasifikácie, technikou najbližšieho suseda 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 napríklad jednoducho 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 [16].

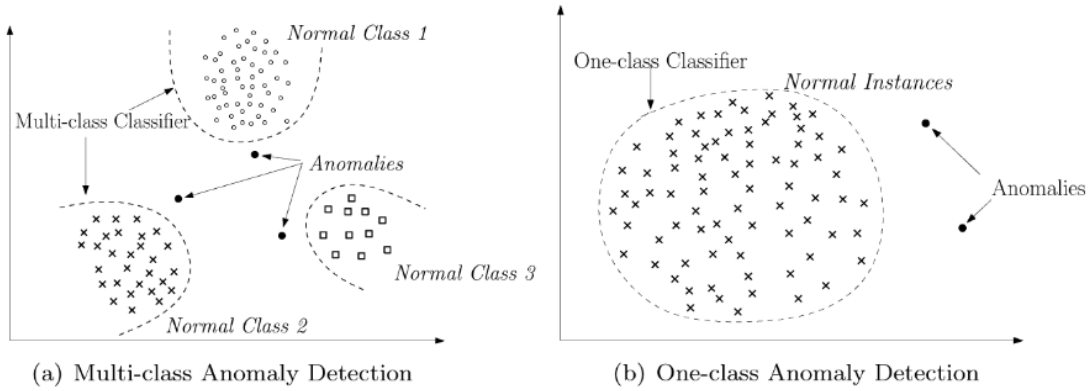
2.4.2 Detekcia anomálií na základe klasifikácie

Techniky založené na klasifikácii pre 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 pre detekciu anomálií pracujú na základe nasledujúceho všeobecného predpokladu. V danom priestore príznakov je možné naučiť klasifikátor, ktorý dokáže rozlíšiť 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 [17]. Príkladmi takejto detekcie sú neurónové siete, Bayesovské siete, SVM a prístup založený na pravidlách.



Obr. 4 Detekcia anomálií na základe klasifikácie [17]

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 [17].

2.4.4 Detekcia anomálií na základe zhlukovania

Zhlukovanie sa používa na triedenie podobných inštancií dát do takzvaných zhlukov. Jedná sa o unsupervised techniku. 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 [17].

2.5 Algoritmy a metódy detekcie anomálií

V našej práci sa zaoberáme viacerými metódami detekcie anomálii. V priebehu rokov boli navrhnuté viaceré algoritmy pre metódu hľadania anomálií bez učiteľa. V tejto časti bližšie popíšeme dôležité typy algoritmov pre 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 [18]. Tento typ detekcie anomálií delíme na algoritmy založené na vzdialenosti a hustoty.

Algoritmy založené na základe blízkosti porovnávajú body v dátach s ich susedmi. Body, ktoré sú veľmi vzdialené od svojich susedov sú označené ako anomálie [18]. Skóre anomality bodu v tomto prípade je najčastejšie určená vzdialenosťou ku k-tému najbližšiemu susedovi. Používajú sa aj iné variácie tohto prístupu, a to priemerná vzdialenosť k-najbližších susedov k bodu [19].

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 [20]

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

- Jaccardova metrika [21]

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

- 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 [20]

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

- Sqeuklidovská metrika [22]

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

- Braycurtisova metrika [23]

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

Medzi metódy založené na základe blízkosti patria:

- **Local outlier factor** (LOF) metóda je založená na základe blízkosti, konkrétne na hustote v okolí objektu a k vzdialenosti susedov. Na začiatku metóda počíta lokálne odľahlé hodnoty pre každý dátový bod, ktoré vyjadrujú stupeň odľahlosti každého bodu. Metódu je možné použiť v režime trénovania bez učiteľa [24].
- **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 [25].
- **Rotation-based Outlier Detection** (ROD) je založená na rozklade celého atribútového priestoru na rôzne kombinácie atribútov. Troj-dimenzionálne vektory reprezentujú dátové body v každom troj-dimenzionálnom 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 [26].

Myšlienka pri algoritmoch založených na **hustote** je, že anomálie vieme nájsť v regiónoch s nízkou hustotou. Podobne objekty so štandardným správaním sa v nachádzajú v oblastiach s vysokou hustotou [19].

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 [19] 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. [27].
- **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 [18].

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 a porovnaním výstupu takéhoto modelu s povodným objektom. Čím viac daný objekt nie je podobný pôvodnému, tým viac je anomálny [28].

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 viacerým detektorom môžu byť robustnejšie voči rôznorodej povahe dát [29], [30]. 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 [31].

- **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 anomálií v multimodálnych datasetoch. Metóda je rýchla, jej časová zložitosť je lineárna a priestorová zložitosť je konštantná [32].
- **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 [33].

3 Dataset a predspracovanie dát

Pre 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 [34]. 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, binarizácie dát a následnej agregácie dátových vektorov tak, aby vektory po agregácii predstavovali informáciu o správaní inodu naprieč jeho životným cyklom.

3.1 Popis datasetov

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

- **Magnet Forensics CTF 2019** [35], **Magnet Forensics CTF 2020** [36], **Magnet Forensics CTF 2022** [37]:
  Obrazy diskov sú súčasťou súťaže, kde cieľom je nájsť vlajku v určitom formáte použitím techník digitálnej forenznej analýzy. Nejde o klasický proces digitálnej forenznej analýzy, ale účastníci súťaže hľadajú odpovede na otázky typu „kedy bol získaný obraz disku“, „kedy bol nainštalovaný softvér“ a ďalšie.
- **NIST Data Leakage Case** [38] a **NIST Hacking Case** [39]:
  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** [34]:
  Cieľom 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. Všeobecné použitie nástroje je podľa definície nasledujúce [40]:

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 poddparsery: 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 [41]. 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

Na získanej časovej osi sme následne vybrali dáta zo zdroja file, teda záznamy hovoriace o zmenách v súborovom systéme. Z kategorických premenných sme vytvorili nové atribúty v binárnej podobe. Tak sme získali dáta s atribútmi ukázanými v Tab. 2 Tabuľka popisujúca atribúty dát po binarizácii

Atribút	**Popis**
Column1	Pôvodný poradový číselník 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. 2 Tabuľka popisujúca atribúty dát po binarizácii

3.4 Agregácia dát

Pri agregácii používame len binárne dáta, ktorých generovanie bolo popísané v predošlej časti. Stĺpce reprezentujú atribúty objektu, a riadky konkrétne udalosti, teda v našom prípade indexom udalosti inode. V rámci agregácií používame štandardné agregačné funkcie maximum(max), súčet(sum) a priemer(mean). Navrhli sme aj ďalšie agregácie pre lepšiu obsiahlosť dát. Uvažujme pevný atribút A. 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)

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

- kvantilový treshold (q1)

$$\begin{cases} 1, & ak\ a^{inode} \geq\ q^* \\ 0, & inak \end{cases},$$

kde q^* je kvantil, definujeme $q^* = \tilde{a}$

- kvantilový treshold 2 (q2)

$$\begin{cases} 1, & ak\ a^{inode} \geq\ q_1^*\ a\ a^{inode} \leq\ q_2^* \\ 0, & inak \end{cases},$$

kde $q_1^* = Q_p$ pre $p \geq 0{,}5$, teda $q_1^* \geq Q_{0{,}5} = \tilde{a}$ je kvantil a q_2^* je kvantil Q_{1-p}.

4 Automatizácia detekcie anomálií

V tejto časti ukážeme postup, ako implementovať agregačné funkcie v programovacom jazyku a ako pristupovať k automatizácii hľadania anomálií pomocou viacerých metód. Pri spracovávaní dát pracujeme s knižnicou pandas [42]. Vybrali sme knižnice v programovacom jazyku Python, ktoré umožňujú rýchle overenie metód. Jedná sa o scikit-learn [43] a PyOD [44]. V týchto knižniciach sme vybrali metódy, ktoré sme sa rozhodli použiť pre overenie správnosti hypotéz pri rôznych datasetoch.

4.1 Implementácia agregačných funkcií

Na implementovanie agregačných funkcií používame konštruktor DataFrame z knižnice pandas. Najpoužívanejšou funkciu pri implementácii bola funkcia groupby, ktorá umožňuje vytvoriť z veľkého množstva dát skupiny po aplikovaní určitých funkcií [45]. Vytvorili sme si funkciu get_aggregated_data, v ktorej sme implementovali všetky agregačné funkcie. Na vstupe tejto funkcie sa nachádzajú dáta v binárnej podobe, názov agregačnej funkcie od ktorej chcem získať upravené dáta, a názov atribútu, podľa ktorého vytvárame skupiny. Jedná sa o inode, názov súboru, alebo kombináciu týchto dvoch. Na Obr. 5 a Obr. 6 je kód funkcie, kde sú implementované všetky agregačné funkcie popísané v časti **Chyba! Nenašiel sa žiaden zdroj odkazov.**.

```
def get_aggregated_data(data, agg_function, col_name, val=None) -> pd.DataFrame():

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

    if agg_function == 'max':
        result = user_data.groupby([col_name]).max()

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

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

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

    if agg_function == 'z1':
        a_power_k = data.groupby(col_name).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(col_name).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)
```

Obr. 5 Implementácia agregačných funkcií 1.

```
    if agg_function == 'z3':
        a_power_k = data.groupby(col_name).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)

    if agg_function == 'q1':
        if val is None:
            a_power_k = data.groupby(col_name).sum(numeric_only=True)
            tilde_a_2 = a_power_k.median()
            result = (a_power_k >= tilde_a_2).astype(int)
        else:
            if val == 0.25 and val == 0.75:
                a_power_k = data.groupby(col_name).sum(numeric_only=True)
                quartiles__a_power_k = a_power_k.quantile([val])
                result = (a_power_k >= quartiles__a_power_k.loc[val]).astype(int)
            else:
                raise Exception("Sorry, wrong value for q1 agg")

    if agg_function == 'q2':
        a_power_k = data.groupby(col_name).sum(numeric_only=True)
        quartiles__a_power_k = a_power_k.quantile([0.1, 0.9])
        a_power_k = a_power_k.astype(float)
        result = ((a_power_k >= quartiles__a_power_k.loc[0.9]) | (a_power_k <= quartiles__a_power_k.loc[0.1])).astype(int)

    return result
```

Obr. 6 Implementácia agregačných funkcií 2.

4.2 Metódy

Pri postupe riešenia sme vybrali metódy, ktoré vieme použiť pri rôznych datasetoch. Vďaka automatizácii budeme schopný prejsť rôzne reprezentované dáta v datasetoch a rýchlejšie overiť úspešnosť týchto metód.

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
SOD	Subspace Outlier Detection	Unsupervised

Tab. 3 Výber metód pre detekciu anomálií

Pre každú metódu z Tab. 3 je možné vybrať viaceré parametre na vstupe, a tým meniť správanie sa pri detekcii. Tab. 4 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_jobs	int, optional (default=1)	Počet úloh, ktoré majú byť vykonané paralelne pre metódu fit aj predict. Ak je hodnota -1, počet úloh sa nastaví na počet dostupných jadier procesora.
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’	-
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. 4 Tabuľka popisujúca parametre pre metódy detekcie anomálií

4.3 Príprava funkcie

V nasledujúcom kroku sme si pripravili funkciu, 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 teda zahŕňa metódy z predošlej kapitoly, a umožňuje používateľovi nastaviť rôzne vstupné parameter pre každú funkciu. Obr. 3 ukazuje postupnosť.

Funkcia, akceptuje na vstupe python slovník, ktorý obsahuje podrobnosti o vstupných metódach, ich parametroch a hodnote, ktorá označuje, či je inštancia zdrojom anomálie alebo normálnych údajov. Tento slovník funguje ako konfiguračný nástroj, ktorý nám umožňuje nastaviť niekoľko metodológií a súvisiacich parametrov analýzy.

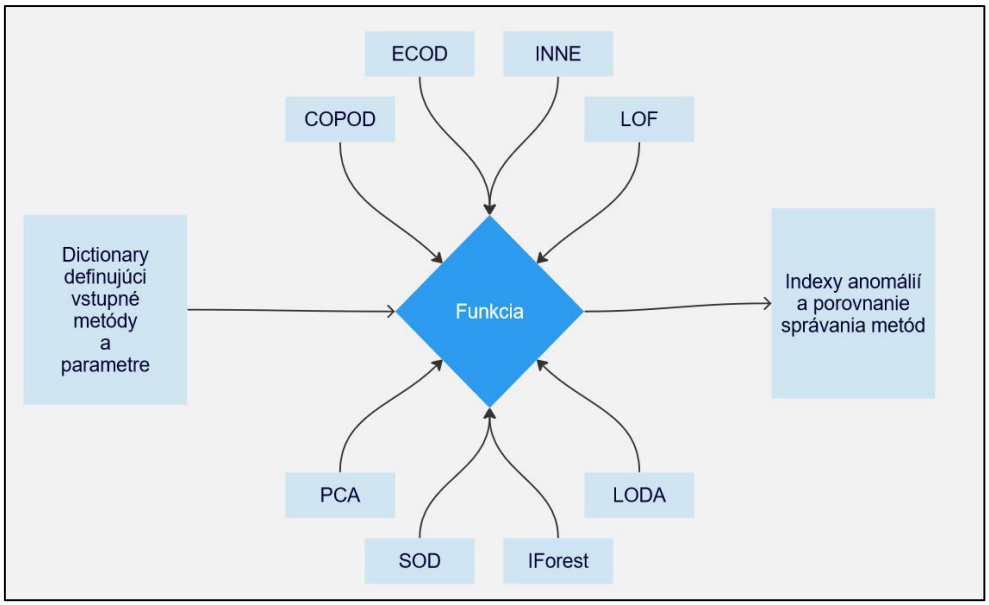
Pomocou vstupných metód a príslušných parametrov je možné vykonávať rôzne operácie a manipulácie s dátami. Dôležitou súčasťou vstupného slovníka je tiež hodnota, ktorá určuje, či je daná inštancia anomáliou alebo normálnymi dátami. Táto informácia je kľúčová pre správne vykonanie analýzy a identifikovania anomálií.

Vďaka takémuto konfigurovateľnému prístupu môžeme nastaviť rôzne kombinácie metód a parametrov, a tým prispôsobiť funkciu svojim špecifickým potrebám a požiadavkám. Umožňuje nám to analyzovať riešenia v rôznych kontextoch.

```
{
    'name': 'COPOD',
    'function': COPOD,
    'contamination_values': [
        0.001, 0.002, 0.005, 0.01, 0.02,
        0.05, 0.1]
    ,
    'outlier_value': 1
},
{
    'name': 'ECOD',
    'function': ECOD,
    'contamination_values': [
        0.001, 0.002, 0.005, 0.01, 0.02,
        0.05, 0.1
    ],
    'outlier_value': 1
},
{
    'name': 'INNE',
    'function': INNE,
    'contamination_values': [
        0.001, 0.002, 0.005, 0.01, 0.02,
        0.05
    ],
    'n_estimators': [
        10, 50, 100,
        200, 20 # remove
    ],
    'outlier_value': 1
},
{
    'name': 'IsolationForest',
    'function': IsolationForest,
    'contamination_values': [
        0.001, 0.002, 0.005, 0.01, 0.02,
        0.05
    ],
    'n_estimators': [
        10, 50, 100,
        20 # remove
    ],
    'outlier_value': -1
},
```

Obr. 1 Príklad vstupu pre funkciu automatizácie

Myšlienka tejto funkcie je založená na jednoduchom princípe spojenia viacerých metód do jednej. Na Obr. 6 môžeme vidieť princíp spracovania dát funkciou, zapojenie metód a vygenerovanie výsledkov.



Obr. 6 Princíp fungovania funkcie automatizácie

Ak je dataset, s ktorým pracujeme, ohodnotený, funkcia, o ktorej hovoríme, nám vie vypočítať hodnoty úspešnosti. Teda ak do funkcie zahrnieme aj parameter "indx_outliers", ktorý definuje indexy anomálií v danom datasete, funkcia nám vypočíta hodnoty TP (true positives - správne identifikované anomálie), FP (false positives - nesprávne identifikované anomálie), TN (true negatives - správne identifikované normálne hodnoty), FN (false negatives - nesprávne identifikované normálne hodnoty), Recall (pomer správne identifikovaných anomálií k celkovému počtu anomálií), Precision (pomer správne identifikovaných anomálií k celkovému počtu identifikovaných anomálií), F1 (harmonický priemer medzi Recall a Precision), TPR (true positive rate - pomer správne identifikovaných anomálií k celkovému počtu anomálií) a FPR (false positive rate - pomer nesprávne identifikovaných normálnych hodnôt k celkovému počtu normálnych hodnôt) a presnosť (Accuracy). Na Obr. 7 sú uvedené riadky kódu, v ktorých prebieha výpočet týchto hodnôt.

```
if indx_outliers:
    data['correct'] = list(filter(lambda x: x in outlier_index[0].tolist(), indx_outliers))
    data['TP'] = len(data['correct'])
    data['FP'] = data['outlier_count'] - data['TP']
    data['FN'] = len(indx_outliers) - data['TP']
    data['TN'] = len(agg_data_user) - data['TP']
    data['Recall'] = 0 if data['TP'] + data['FN'] == 0 else data['TP'] / (data['TP'] + data['FN'])
    data['Precision'] = 0 if data['TP'] + data['FP'] == 0 else data['TP'] / (data['TP'] + data['FP'])
    data['F1'] = 0 if data['Precision'] + data['Recall'] == 0 else (2 * data["Precision"] * data[
        "Recall"]) / (data["Precision"] + data["Recall"])
    data['TPR'] = data['Recall']
    data['FPR'] = data["FP"] / (data["FP"] + data["TN"])
    data['Accuracy'] = (data["TP"] + data["TN"]) / (data["TP"] + data["TN"] + data["FP"] + data["FN"])

results_src_user.append(data)
```

Obr. 7 Výpočet hodnôt pre každú metódu

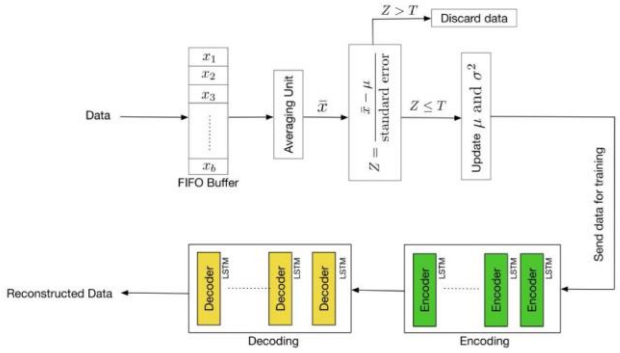
Výstupom tejto funkcie je .xlsx súbor, v ktorom sú skóre anomality pre jednotlivé modely, a označenie, či daná udalosť bola rozpoznaná ako anomália. Tento súbor obsahuje aj finálne skóre anomality, a teda súčet koľkými modelmi bol daný riadok označený ako anomálny. Takto pripravený súbor poskytuje podrobný pohľad na výkon a efektivitu jednotlivých metód pri detekcii anomálií. Ukážka výsledkov je na Obr. 8, kde žltou farbou sú zvýraznené súbory, s ktorými pracoval útočník. Final_score reprezentuje anomalitu konkrétneho záznamu pri agregácii cez kombináciu inodu a názvu súboru.

comb	final_score
20274-NTFS:\FileShare\Secret\PortalGunPlans.txt	17
86967-NTFS:\FileShare\Secret\NoJerry.txt	17
87102-NTFS:\$Recycle.Bin\S-1-5-21-2232410529-1445159330-2725690660-500\$IU2L112.txt	17
87111-NTFS:\FileShare\Secret\Beth_Secret.txt	17
1-NTFS:\$MFTMirr	16
34-NTFS:\$Extend\$RmMetadata\$TxfLog\$TxfLogContainer00000000000000000002	16
84656-NTFS:\$Extend\$UsnJrnl:$J	16
87132-NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCookies\Y8KFNFMU.txt	16
0-NTFS:\$MFT	15
19402-NTFS:\ProgramData\Start Menu	15
19515-NTFS:\ProgramData\Desktop	15
19516-NTFS:\ProgramData\Documents	15
19518-NTFS:\Documents and Settings	15
19519-NTFS:\ProgramData\Templates	15
19557-NTFS:\ProgramData\Application Data	15
29-NTFS:\$Extend\$RmMetadata\$TxfLog	15
30-NTFS:\$Extend\$RmMetadata\$Txf	15
73635-NTFS:\$Recycle.Bin\S-1-5-21-2232410529-1445159330-2725690660-500\$RU2L112.txt	15
84978-NTFS:\Users\Administrator\AppData\Local\Application Data	15
84980-NTFS:\Users\Administrator\AppData\Local\History	15
84981-NTFS:\Users\Administrator\AppData\Local\Temporary Internet Files	15
84982-NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\Temporary Internet Files	15
85025-NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Internet Explorer.lnk	15
NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar\Windows PowerS	15
86-NTFS:\Boot\memtest.exe	15
\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\9b9cdc69c1c24e2b.automaticDes	15
7085-NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Themes\CachedFiles\CachedImage_1024_768_POS4.jp	15
87089-NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\INetCache\Low\Content.IE5	15
FS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations\28c8b86deab549a1.customDestin	15
106-NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Themes\CachedFiles\CachedImage_2306_1230_POS4.j	15
87112-NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk	15
\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\1bc392b8e104a00e.automaticDes	15
10-NTFS:\$UpCase	14
11-NTFS:\$Extend	14
WinSxS\amd64_microsoft.windows.powershell.common_31bf3856ad364e35_6.3.9600.16384_none_219db062ef302354\Win	14
19535-NTFS:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools\Windows PowerShell (x86).lnk	14
\WinSxS\amd64_microsoft.windows.powershell.common_31bf3856ad364e35_6.3.9600.16384_none_219db062ef302354\W	14

Obr. 8 Výsledky funkcie

4.4 Autoencoder neurónová sieť

Jedná sa o typ neurónovej siete, ktorej hlavným cieľom je naučiť sa zachytiť dôležité vlastnosti na vstupných dátach. Funkcia encoder je zachytiť kľúčové vlastnosti. Decoder na druhej strane obnovuje data po tom, čo encoder znížil ich dimenziu. Túto neurónovú sieť sme implementovali pomocou tensorflow frameworku [46]. Na Obr. 9 je ukázaná architektúra neurónovej siete.



Obr. 9 Architektúra neurónovej siete autoencoder

Neurónová sieť na základe kontexktu o predošlých udalostiach predikuje, aká udalosť bude nasledovať. Takto sa naučí štandardné správanie operačného systému. Po natrénovaní tejto neurónovej siete sledujeme, nakoľko je predikovaná udalosť v zhode s očakávanou udalosťou. Skóre anomality vieme vypočítať ako o odchýlku, respektíve ako mean square error.

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

Výsledkom tohto prístupu je .xlsx súbor, v ktorom sú usporiadaná udalosti podľa najvyššie skóre anomality. Na Obr. 10 je ukážka výsledkov tohto prístupu. Podčiarknuté udalosti, a udalosti v oblasti vyznačenou červeným obdĺžnikom, sú v skutočnosti súvisiace s útokom.

0.29429	86971 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\NoJerry.lnk
0.29322	20479 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Internet Explorer\Quick Launch\Launch Internet Explorer Browser.lnk
0.29322	49839 - NTFS:\Windows\WinSxS\FileMaps\program_files_x86__676bbe2c7241b694.cdf-ms
0.28323	86968 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\9b9cdc69c1c24e2b.automaticDestinations-ms
0.28217	75689 - NTFS:\Windows\System32\certocm.dll
0.27207	78665 - NTFS:\Windows\System32\LogFiles\Scm\651ff2a7-84d4-4ae6-9231-bb0411d3a64f
0.27101	29435 - NTFS:\Windows\System32\certutil.exe
0.25698	85025 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Internet Explorer.lnk
0.24135	87112 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk
0.24135	73609 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\SchCache\WIN-E0PO207ERMD.C137.local.sch
0.24135	87112 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk
0.24135	86968 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\9b9cdc69c1c24e2b.automaticDestinations-ms
0.24135	84630 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\PortalGunPlans.lnk
0.24135	87060 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Szechuan Sauce.lnk
0.24135	87064 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\SECRET_beth.lnk
0.24135	87112 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk
0.24135	86971 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\NoJerry.lnk
0.24135	86968 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations\9b9cdc69c1c24e2b.automaticDestinations-ms
0.24135	86971 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\NoJerry.lnk
0.24135	86966 - NTFS:\FileShare\Secret
0.23489	33703 - NTFS:\Windows\System32\wbem\DscpspluginWkr.dll
0.23476	81133 - NTFS:\Windows\WinSxS\amd64_microsoft-windows-i..timezones.resources_31bf3856ad364e35_6.3.9600.16384_ar-sa_26a9e18016ee01fa\Window
0.23473	42271 - NTFS:\Windows\WinSxS\amd64_microsoft-windows-t..ervices-psmgmttools_31bf3856ad364e35_6.3.9600.16384_none_2526fef3eb06e744\TSPSEn
0.23032	87146 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations\590aee7bdd69b59b.customDestinations-ms
0.23032	87112 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk
0.23032	87112 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\Beth_Secret.lnk
0.23032	87146 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations\590aee7bdd69b59b.customDestinations-ms
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	2643 - NTFS:\Windows\Microsoft.NET\Framework\v4.0.30319\ASP.NETWebAdminFiles
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	84987 - NTFS:\Users\Administrator\AppData\Local\Microsoft\Windows\UsrClass.dat
0.23032	85087 - NTFS:\Users\Administrator\Searches
0.23032	49746 - NTFS:\Windows\SYSVOL\domain
0.23032	78666 - NTFS:\Windows\System32\LogFiles\Scm\72f35a0c-fd31-44c6-a5fd-74c656230c66
0.23032	73612 - NTFS:\Users\Administrator\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-21-2232410529-1445159330-2725690660-500

Obr. 10 Výsledky použitia neurónovej siete

Záver

Na základe predstavených cieľov a postupu v tejto práci sme uskutočnili komplexnú analýzu forenzných artefaktov v operačnom systéme Windows s cieľom identifikovať relevantné digitálne stopy po bezpečnostných incidentoch. Overili sme viaceré prístupy reprezentácie dát, a ich následnej agregácie. Pre detekciu anomálnych udalostí sme overili metódy založené na rôznych prístupoch. Namodelovali sme riešenie pre automatizovanú detekciu relevantných stôp zanechaných útočníkom založené na metódach hľadania anomálií a s použitím neurónovej siete. Výsledky tejto práce poskytujú cenné poznatky a návrhy na ďalší výskum v oblasti forenzného vyšetrovania a kybernetickej bezpečnosti, s dôrazom na automatizáciu procesov a zlepšenie schopností detekcie a odhaľovania kybernetických útokov.

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] 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.

[11] P. Alvarez, "Metering Mode: center weight," vol. 2, no. 3, 2004.

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

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

[14] 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/

[15] 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/

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] "Distance Metrics." Accessed: Jan. 25, 2024. [Online]. Available: https://numerics.mathdotnet.com/Distance

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

[22] "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

[23] "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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.

[33] 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.

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

[35] “Magnet CTF 2019 Windows Desktop, 2019.” [Online]. Available: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2019 CTF - Windows-Desktop.zip

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

[37] “Magnet CTF 2022 Windows Desktop, 2022.” [Online]. Available: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2022 CTF - Windows.zip

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

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

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

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

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

[43] “scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation.” Accessed: Jun. 27, 2023. [Online]. Available: https://scikit-learn.org/stable/index.html

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

[45] “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

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