UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

ČASOVÉ OSI PRI FORENZNEJ ANALÝZE OPERAČNÉHO SYSTÉMU WINDOWS

2022 **Bc. Kristína KOVÁČOVÁ**

UNIVERZITA PAVLA JOZEFA ŠAFÁRIKA V KOŠICIACH
PRÍRODOVEDECKÁ FAKULTA

ČASOVÉ OSI PRI FORENZNEJ ANALÝZE OPERAČNÉHO SYSTÉMU WINDOWS

DIPLOMOVÁ PRÁCA

Študijný program:	Informatika
Pracovisko (katedra/ústav):	ÚINF - Ústav informatiky
Vedúci diplomovej práce:	JUDr. RNDr. Pavol Sokol, PhD.

Košice 2022

Bc. Kristína KOVÁČOVÁ

Univerzita P. J. Šafárika v Košiciach
Prírodovedecká fakulta

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Kristína Kováčová
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: Časové osi pri forenznej analýze operačného systému Windows

Názov EN: Timelines in the the Windows operating system forensic analysis

Cieľ: (1) Aktuálne prístupy k spracúvaniu časových osí forenzných artefaktov operačného systému Windows.
(2) Analýza spôsobov korelácie atribútov forenzných artefaktov a vytvárania vzťahov medzi nimi.
(3) Návrh, implementácia a vyhodnotenie nástroja pre vytvorenie a vizualizáciu časovej osi forenzných artefaktov operačného systému Windows.

Literatúra:
(1) Carvey H.: Investigating Windows Systems, Academic Press, 1st edition, 2018.
(2) Debinski M., Breitinger F., Mohan P.: Timeline2GUI: A Log2Timeline CSV parser and training scenarios, Digital Investigation, Volume 28, Pages 34-43. 2019.
(3) Anson S.: Applied Incident Response, Wiley, 1st Edition, 2020.
(4) Osborne G., Thinyane H., Slay J.: Visualizing Information in Digital Forensics. In: Peterson G., Shenoi S. (eds) Advances in Digital Forensics VIII. DigitalForensics 2012. IFIP Advances in Information and Communication Technology, vol 383. Springer, Berlin, Heidelberg. 2012.

Vedúci:	RNDr. JUDr. Pavol Sokol, PhD.
Oponent:	Mgr. Ladislav Bačo
Ústav :	ÚINF - Ústav informatiky
Riaditeľ ústavu:	doc. RNDr. Ondrej Krídlo, PhD.

Dátum schválenia: 29.04.2022

doc. RNDr. Ondrej Krídlo, PhD.
riaditeľ Ústavu informatiky

Poďakovanie

Týchto pár riadkov v práci by som chcela venovať ako poďakovanie svojmu vedúcemu práce a zároveň motivátorovi, JUDr. RNDr. Pavlovi Sokolovi, PhD. za odborné vedenie, cenné rady, veľkú pomoc pri spracúvaní tejto diplomovej práce a ľudský prístup v každej situácii.

Taktiež ďakujem mojim najbližším, rodine i priateľom, ktorí mi neustálou podporou pomohli dosiahnuť vytýčené ciele.

Abstrakt v štátnom jazyku

Riešenie bezpečnostných incidentov je nevyhnutnou súčasťou informačnej a kybernetickej bezpečnosti každej organizácie. Riešenie incidentov pozostáva z viacerých fáz, medzi ktorými má nezastupiteľné miesto aj analýza dostupných digitálnych stôp. Keďže jednotlivé digitálne stopy sú zaznamenané v určitom časovom okamihu, dôležitú rolu v analýze týchto digitálnych stôp zohrávajú časové osi. Cieľom tejto záverečnej práce je bližšie preskúmať možnosti tohto spôsobu analýzy digitálnych stôp. Práca obsahuje analýzu aktuálnych prístupov k spracúvaniu časových osí forenzných artefaktov operačného systému Windows, či už z pohľadu nielen dostupných nástrojov, ale samotnej tvorby a analýzy časovej osi. Súčasťou práce je aj analýza spôsobov korelácie atribútov forenzných artefaktov a vytvárania vzťahov medzi nimi. K tomuto cieľu je v práci poskytnutý spôsob predspracovania údajov. Vytváranie vzťahov medzi jednotlivými záznamami časových osí ako aj analýza vzťahov medzi konkrétnymi atribútmi je založená na formálnej konceptovej analýze. Práca súčasne popisuje návrh a implementáciu spôsobu normalizácie, agregácie, korelácie a vizualizácie digitálnych forenzných artefaktov. Pre tento účel bol použitý forenzný obraz disku doménového kontroléra z prípadu The Stolen Szechuan Sauce z portálu DFIR Madness.

Digitálna forenzná analýza, artefakt, časová os, korelácia, formálna konceptová analýza

Abstrakt v cudzom jazyku

Handling of security incidents is an essential part of every organization's information and cyber security. Incident handling consists of several phases, among which the analysis of available digital evidence has an irreplaceable place. Because individual digital evidence is recorded at a certain point in time, timelines play an essential role in analyzing this digital evidence. This final work aims to explore the possibilities of this method of analysis of forensic timelines. The work contains an analysis of current approaches to processing timelines of forensic artifacts of the Windows operating system, either in terms of available tools or the creation and analysis of the timeline. Part of the work is also an analysis of ways to correlate the attributes of forensic artifacts and create relationships between them. For this purpose, the work provides a way to pre-process data. Making connections between individual timeline records and analyzing relationships between specific attributes is based on formal concept analysis. The work also describes the design and implementation of a method for normalization, aggregation, correlation, and visualization of digital forensic artifacts. The forensic image of the domain controller disk from The Stolen Szechuan Sauce case of DFIR Madness was used for this purpose.

Digital forensic analysis, artifact, timeline, correlation, formal concept analysis

Obsah

Slovník termínov

Forenzná veda je praktická aplikácia rôznych druhov vedných disciplín pre zodpovedanie otázok súvisiacich s právnym systémom.

Digitálna forenzná analýza je viacstupňový proces začínajúci identifikáciou digitálnych médií zo scény ako potenciálnych stôp končiaci vo fáze, v ktorej sú predložené ako dôkazy súdnym znalcom na súde v rámci civilného alebo trestného konania

Digitálna stopa je akúkoľvek informácia s vypovedacou hodnotou uloženou alebo prenášanou v digitálnej binárnej forme, ktorá môže byť predložená súdu ako vecný dôkaz s vypovedacou hodnotou.

Digitálny artefakt je akákoľvek nežiaduca alebo neúmyselná zmena údajov zavedená v digitálnom procese pomocou príslušnej techniky a/alebo technológie.

Indikátor kompromitácie (IoC) je objekt alebo aktivita, ktorá pozorovaná v počítačovej sieti alebo na zariadení naznačuje vysokú pravdepodobnosť neoprávneného prístupu do systému.

Úvod

Neustále sa zvyšujúci počet kybernetických útokov zapríčiňuje rastúci dopyt po analytikoch z oblasti kybernetickej a informačnej bezpečnosti. So zvyšujúcim sa počtom prípadov však narastá aj množstvo dát, ktoré je potrebné analyzovať. Nevyhnutné pre prácu analytika je hlavne to, aby mali rýchly prehľad o tom, čo sa deje, a získali všetky k prípadu relevantné informácie. Cieľom digitálneho vyšetrovania je pri vykonávaní digitálnej forenznej analýzy identifikovať forenzne významné artefakty, a na základe nich potom potvrdiť alebo vyvrátiť forenznú hypotézu.

Veľká časť artefaktov, ktorá sa pri zaistení jednotlivých zariadení získa, nie je pre vyšetrovanie potrebná a relevantná. Určitý nadhľad nad množstvom zaistených dát môže analytikovi priniesť časová os, na základe ktorej možno súbory záznamov reprezentovať využitím metadát v postupnom chronologickom usporiadaní. Analýza časovej osi sa považuje za kľúčovú zložku každého vyšetrovania, pretože časový sled udalostí je takmer vždy relevantný. Na to, ako možno vyhľadávať významné digitálne dôkazy v operačnom systéme Windows a vzťahy medzi nimi využitím analýzy časovej osi, sme sa zamerali v tejto diplomovej práci.

Táto záverečná práca si dáva za cieľ preskúmať možnosti analýzy časových osí pre účely forenznej analýzy. Tento cieľ je rozdelený do troch parciálnych podcieľov. Prvým podcieľom je analýza aktuálnych prístupov k spracúvaniu časových osí forenzných artefaktov operačného systému Windows. Tento cieľ je možné chápať v širších súvislostiach aj z pohľadu nielen dostupných nástrojov, ale samotnej tvorby časovej osi a jej analýzy vrátane korelácie záznamov v nej. Druhým podcieľom je analýza spôsobov korelácie atribútov forenzných artefaktov a vytvárania vzťahov medzi nimi. Pre tento účel je nevyhnutné definovať možné atribúty forenzných artefaktov a ich reprezentáciu v rámci jednotlivých záznamov uvedených v rámci časových osí. Posledným podcieľom je návrh, implementácia a vyhodnotenie nástroja pre vytvorenie a vizualizáciu časovej osi forenzných artefaktov operačného systému Windows.

V prvej časti tejto diplomovej práce sa venujeme časovým osiam a ich analýze, a taktiež času a dôležitým artefaktom z pohľadu digitálnej forenznej analýzy. Primárnym zdrojom informácií z časovej osi sú metadáta súborového systému [1]. Ďalej sú rozoberané systémové udalosti a artefakty registra operačného systému Windows.

V ďalšej časti sú rozobrané podobné práce, ktoré sa venujú nástrojom na vytváranie, analýzu a spracovanie časových osí a taktiež koreláciu získaných záznamov.

Jadro tejto záverečnej práce tvorí tretia kapitola, ktorej súčasťou je popis vzorového prípadu a samotných údajov, s ktorými pracujeme. Na tomto mieste je dôležité uviesť, že sa zameriavame len na oblasť forenznej analýzy operačného systému Windows. Vzhľadom na rozsah spracovanej problematiky, nebolo možné v rámci práce zapracovať aj forenzné artefakty obsiahnuté v operačnej pamäti a v sieťovej komunikácii. Dôležitou súčasťou tejto časti práce je samotné spracovanie datasetu. V rámci práce popisujeme postup od spracovania forenzného obrazu disku doménového kontroléra cez vytvorenie časovej osi prostredníctvom nástroja Log2timeline (plaso) až po identifikovanie atribútov pre jednotlivé záznamy forenznej osi. Po tejto fáze nasleduje agregácia jednotlivých záznamov, pri ktorej dochádza k redukcii dát a vzniku tzv. meta-záznamov (agregovaných záznamov) na základe zvolených metód.

Po vytvorení meta-záznamov sa venujeme v štvrtej kapitole tejto práce hľadaniu vzťahov medzi nimi ako aj analýzou vzťahov medzi jednotlivými atribútmi forenzných artefaktov. Pre tento účel sme využili formálnu konceptovú analýzu, najmä vytvorenie konceptov a následne identifikovanie asociačných pravidiel. Poslednou časťou tejto práce je korelácia meta-záznamov a vizualizácia vzťahov a vyhodnotenie celého systému.

1 Analýza časových osí z pohľadu forenznej analýzy

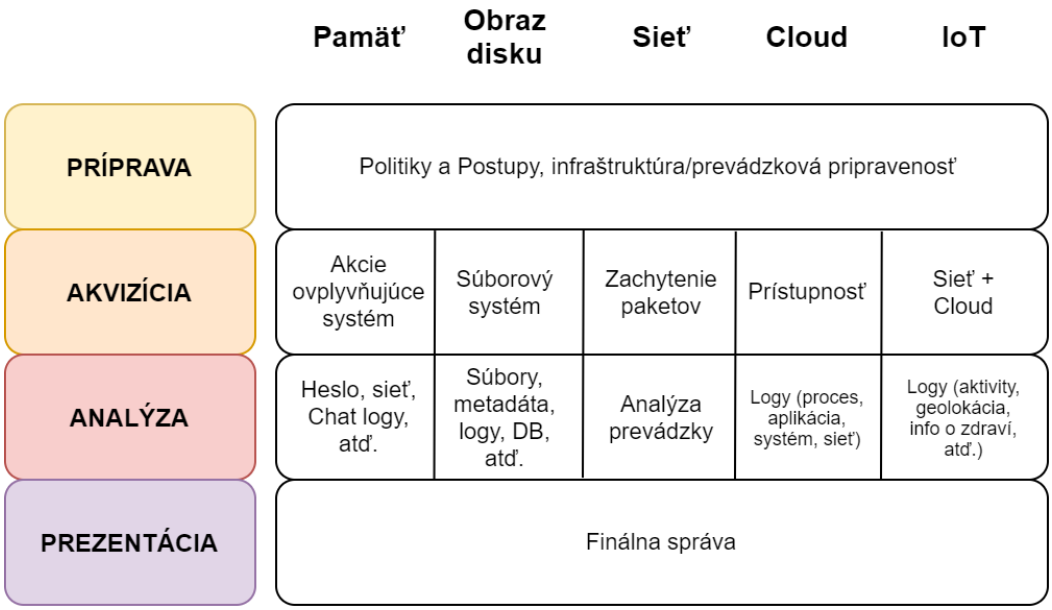
Úlohou digitálnej forenznej analýzy pri digitálnom vyšetrovaní je zodpovedanie základných vyšetrovacích otázok, a to „kedy?“, „čo?“, „prečo?“ a „ako?“. Chronologické usporiadanie udalostí dáva forenznému analytikovi možnosť urobiť si nadhľad v riešenom prípade a ľahšie tak nájsť odpovede na položené otázky. Vytvorenie a analýza časovej osi je jedným z dôležitých krokov v procese digitálnej forenznej analýzy.

1.1 Digitálna forenzná analýza

Digitálna forenzná analýza je proces pozostávajúci z niekoľkých fáz. V tejto časti popíšeme procesný model zahŕňajúci štyri kroky, a to prípravu, akvizíciu, analýzu a prezentáciu [2]. Tým že existujú rôznorodé zariadenia a zdroje digitálnych stôp, poznáme mnoho rôznych procesných modelov. Neexistuje univerzálny procesný model vhodný pre všetky typy vyšetrovania.

Príprava zahŕňa dokumentovanie a zdôvodnenie všetkých zariadení, ktoré sú relevantné pre vyšetrovanie s cieľom zistiť, ktoré z nich uchovávajú akékoľvek spoľahlivé informácie týkajúce sa vyšetrovacieho prípadu. **Zaisťovanie (akvizícia)** je fáza zameraná na zhromažďovanie stôp, teda získavanie digitálnych a iných údajov, ktoré sú relevantné alebo môžu mať určitú súvislosť s vyšetrovaným prípadom. Postupy musia byť vykonané bez zmeny údajov alebo systému, alebo aspoň s minimálnym dopadom. **Analýza** je súbor operácií vykonávaných vo vedeckom kontexte s metodickými a preukázateľnými postupmi zameranými na získanie prvkov, ktoré potvrdzujú, alebo vyvracajú obviňujúcu alebo obrannú hypotézu. **Prezentácia** je fáza zameraná na dokumentovanie aktivít a výsledkov jednotlivých fáz, čo sa uskutočňuje prostredníctvom formálnych správ [3].

Na Obrázku č. 1 môžeme vidieť, čo predstavujú jednotlivé fázy pre rôzne druhy forenzných analýz (podľa zdroja údajov).



Obrázok 1 Fázy pre rôzne druhy forenzných analýz [4]

Najzaujímavejšími fázami v rámci forenzného vyšetrovania sú akvizícia a analýza. Sú to dve fázy, ktoré je možné čiastočne automatizovať. Použitie automatických nástrojov na forenzné vyšetrovanie je obmedzené, pretože spracovanie väčšiny digitálnych stôp si stále vyžaduje odborný ľudský zásah. Technologický problém vychádza hlavne z rôznych nových typov zariadení a používaných softvérových platforiem.

1.2 Forenzné artefakty

Podľa [5] sú digitálne forenzné artefakty funkciou fyzického média, operačného systému, súborového systému a aplikácií na úrovni používateľa. Každý z nich ovplyvňuje, aké digitálne stopy sa vytvárajú a zanechávajú. Skúmaním forenzných digitálnych artefaktov sa forenzní analytici snažia pochopiť predošlé správanie na zariadení. Ako sme už spomínali v úvode, primárnym zdrojom informácií z časovej osi sú metadáta súborového systému, na ktoré sme sa z tohto dôvodu rozhodli zamerať. Ďalej sú analyzované hlavne systémové udalosti a artefakty registra operačného systému Windows.

1.2.1 Udalosti

Z pohľadu operačného systému je dôležité zaznamenávať relevantné zmeny tohto systému a činnosti v ňom vykonávané. Akúkoľvek pozorovateľnú udalosť, ku ktorej došlo v určitom časovom bode v systéme alebo sieti, najmä ak je dôležitá, označujeme ako udalosť [6]. Príkladom udalosti je odoslanie emailovej správy, či prihlásenie používateľa. Na druhej strane **bezpečnostná udalosť (security event)** je pozorovateľná udalosť v prostredí informačných a komunikačných technológií, ktorá je relevantná pre bezpečnosť [7]. Príkladom bezpečnostnej udalosti je aktualizácia operačného systému, vytvorenie procesu, resp. vytvorenie nového používateľa. Bezpečnostné udalosti zvyčajne vytvárajú určitý druh stôp, ktorý sa zaznamenáva v záznamoch (logoch) systému, a ktorý je možné analyzovať.

To, aké udalosti sa budú zaznamenávať, závisí od funkcií auditu. Zaznamenávanie udalostí je možné vypnúť s administrátorskými oprávneniami. V závislosti od povolenej úrovne zaznamenávania a nainštalovanej verzie operačného systému Windows môže zaznamenávanie udalostí z forenzného hľadiska poskytnúť vyšetrovateľom podrobnosti o aplikáciách, časové pečiatky týkajúce sa aktivity používateľov a vybrané systémové udalosti. [8,9]

ID udalosti	ID udalosti	Popis
(2000/XP/2003)	(Vista/7/8/2008/2012)	
528	4624	Úspešné prihlásenie.
529	4625	Prihlásenie zlyhalo.
680	4776	Úspešné/neúspešné overenie účtu.
624	4720	Bol vytvorený používateľský účet.
636	4732	Do lokálnej skupiny s povoleným zabezpečením bol pridaný člen.
632	4728	Do lokálnej skupiny s povoleným zabezpečením bol pridaný člen.
2934	7030	Chyby pri vytváraní služby.
2944	7040	Typ spustenia služby IPSEC Services sa zmenil z vypnutého na automatické spustenie.
2949	7045	Vytvorenie služby.

Tabuľka 1 Príklady dôležitých udalostí z pohľadu forenznej analýzy

Windows Event Logs (EVT) sú súbory, do ktorých sa ukladajú významné udalosti operačného systému Windows. Medzi zaznamenanými informáciami sa nachádzajú napr. chybové správy z aplikácií, samotného operačného systému, ale aj informácie týkajúce sa prihlásenia používateľa na zariadenie, spustenie a ukončenie služieb a mnohé iné.

V [10] sa uvádza päť typov udalostí podľa závažnosti, ktoré sú zaznamenávané v rámci Windows Event Logs. V Tabuľke č. 2 sú popísané jednotlivé typy zaznamenávaných udalostí.

Typ udalosti	Popis
Error	Udalosť, ktorá môže znamenať závažný problém, ako je strata údajov, strata funkcionality služby. Označuje, že sa vyskytol problém, ktorý môže mať vplyv na funkčnosť, a to mimo aplikácie alebo komponentu, ktorý udalosť spustil.
Warning	Udalosť, ktorá môže indikovať budúci problém. Napríklad málo diskového priestoru alebo nedostatok iných zdrojov, ktorý môže ovplyvniť službu a/alebo spôsobiť vážnejší problém, ak sa neprijmú opatrenia.
Information	Udalosť, ktorá je zaznamenaná z dôvodu úspešného priebehu chodu aplikácie. Napr. úspešné načítanie sieťového ovládača, došlo k zmene v aplikácii alebo komponente, operácia bola úspešne vykonaná, bol vytvorený prostriedok alebo bola spustená služba.
Success Audit	Udalosť, ktorá zaznamenáva úspešný auditovaný pokus o prístup k zabezpečeniu. Napríklad udelenie prístupu používateľovi na základe poskytnutia správnych poverení alebo hesla.
Failure Audit	Udalosť, ktorá zaznamenáva neúspešný auditovaný pokus o prístup k zabezpečeniu. Bezpečnostnú udalosť, ktorá sa neuskutočnila alebo nebola úspešne dokončená. Napríklad používateľovi bol zamietnutý prístup, kvôli neposkytnutiu správnych poverení alebo hesla.

Tabuľka 2 **Popis typov zaznamenaných udalostí**

Každá zo zaznamenaných udalostí môže obsahovať naviac prídavné údaje, ktoré sú zobrazené v Tabuľke č. 3.

Názov	Popis
Source	Softvér, ktorý zaznamenal udalosť. Môže to byť názov programu alebo komponentu systému, napríklad aj názov ovládača.
EventID	Číslo identifikujúce konkrétny typ udalosti. Napríklad 6005 je ID udalosti, ktorá nastane pri spustení služby Denník udalostí.
Level	Klasifikácia závažnosti zaznamenanej udalosti. Bližšie popísané v Tab. č.
User	Meno používateľa, v mene ktorého sa udalosť vyskytla. Môže to byť aj ID klienta, ak bola udalosť spôsobená procesom servera.
Operational Code	Obsahuje číselnú hodnotu, ktorá identifikuje činnosť alebo bod v rámci činnosti, ktorú aplikácia vykonávala, keď vyvolala udalosť. Napríklad inicializácia alebo ukončenie.
Log	Názov denníka, v ktorom bola udalosť zaznamenaná.
Task Category	Používa sa na reprezentáciu subkomponentu alebo činnosti vykonávateľa udalosti.
Keywords	Súbor kategórií alebo tagov, ktoré možno využiť na filtrovanie alebo vyhľadávanie, vzhľadom na udalosti.
Computer	Názov zariadenia, na ktorom došlo k udalosti. Zvyčajne je to názov lokálneho počítača, ale môže to byť aj názov počítača, ktorý udalosť presmeroval.
Date and Time	Dátum a čas, v ktorom bola udalosť zaznamenaná.
Process ID	Identifikačné číslo procesu, ktoré vygenerovalo udalosť.
Thread ID	Identifikačné číslo vlákna, ktoré vygenerovalo udalosť
Processor ID	Identifikačné číslo procesora, ktorý udalosť spracoval.
Session ID	Identifikačné číslo relácie terminálového servera, v ktorej došlo k udalosti.
Kernel Time	Čas vykonávania inštrukcií v režime jadra, zaznamenaný v jednotkách času CPU.
User Time	Čas vykonávania inštrukcií v užívateľskom režime, zaznamenaný v jednotkách času CPU.
Processor Time	Čas vykonávania inštrukcií v používateľskom režime, v taktoch CPU.
Correlation ID	Identifikuje činnosť v procese, ktorej sa udalosť týka. Tento identifikátor sa používa na špecifikáciu jednoduchých vzťahov medzi udalosťami.
Relative Correlation ID	Identifikuje súvisiacu činnosť v procese, ktorého sa udalosť týka.

Tabuľka 3 Prídavné informácie zaznamenaných udalostí s popisom

Windows Event Logs sú z pohľadu digitálnej forenznej analýzy veľmi dôležitým forenzným artefaktom, na základe ktorého je možno vyhodnotiť, čo sa v operačnom systéme Windows odohralo.

Zaznamenané udalosti môžu zodpovedať dôležité otázky forenznej analýzy, a to kto vyvolal danú udalosť (kto), čo je obsahom danej udalosti (čo), v rámci akého systému bola vykonaná udalosť (kde), prečo došlo k danej udalosti (prečo) a ako bola daná udalosť vykonaná (ako). Uložené informácie môžu odhaliť, aké zariadenie bolo použité, aký účet

bol prihlásený, trvanie používania účtu, aké aplikácie boli nainštalované alebo spustené, či je za nežiadúce správanie zodpovedný škodlivý softvér, aký typ hardvéru bol dočasne pripojený alebo odstránený alebo dôvody zlyhania systému (možné pokusy o zneužitie zraniteľnosti) a časy týchto zlyhaní.

Aj pri vytváraní časovej osi je EVT jedným zo vstupných modulov. Založený je na skripte evtparse.pl, ktorý napísal a publikoval ako jeden z nástrojov časovej osi H.Carvey [11]. Windows Event Logs sú v binárnom formáte, ktorý je rozdelený na hlavičku a sériu záznamov [12]. Hlavička záznamu udalosti má dĺžku 56 bajtov a obsahuje magickú hodnotu (magic bytes) LfLe. Vstupný modul prehľadáva súbor event log a hľadá túto magickú hodnotu. Po nájdení hodnoty LfLe vybraný parser analyzuje a extrahuje potrebné informácie na vytvorenie objektu s časovou značkou.

Od verzie operačného systému Microsoft Vista sa Event Log ukladá v novom formáte, ktorý sa označuje ako **EVTX**. Od EVT sa formát EVTX líši tým, že binárny súbor obsahuje štruktúru XML. Pomocou knižníc je možné extrahovať štruktúru XML z dokumentu namiesto čítania binárneho dokumentu [13].

Pri vytváraní časovej osi využíva upravený modul, ktorý používa knižnicu Parse::Evtx. Vyvinutá bola Andreasom Schusterom na analyzovanie tohto novovytvoreného formátu EVTX, ktorý používa systém Windows Vista a novšie.

Parser Parse::Evtx pre vyššie spomenuté typy záznamov EVTX priradzuje hodnoty zobrazené v Tabuľke č. 4 následne:

Hodnota	Typ záznamu
0	Error event
1	Warning event
2	Information event
3	Success Audit event
4	Failure Audit event

Tabuľka 4 Hodnoty EVTX záznamov priradené podľa Parse::Evtx [14]

Prídavné informácie, ktoré je parser schopný vyextrahovať sú uvedené nižšie:

- Event Identifier
- Source Name
- Message string
- Strings

- Computer Name
- Severity
- Record Number
- Event Type
- Event Category

V Tabuľke č. 5 uvádzame rozdiel medzi formátom EVT a EVTX. Ako už bolo vyššie spomenuté, EVTX formát obsahuje XML reprezentáciu daného záznamu.

Vlastnosť	EVT	EVTX
computer_name (str)	Áno	Áno
event_category (int)	Áno	Nie
event_identifier (int)	Áno	Áno
event_type (int)	Áno	Nie
event_level (int).	Nie	Áno
facility (int)	Áno	Nie
message_identifier (int)	Áno	Áno
offset (int)	Áno	Áno
record_number (int)	Áno	Áno
recovered (bool)	Áno	Áno
severity (int)	Áno	Nie
source_name (str)	Áno	Áno
strings (list[str])	Áno	Áno
user_sid (str)	Áno	Áno
xml_string (str): XML representation of the event.	Nie	Áno

Tabuľka 5 Porovnanie existencie prídavných polí pri záznamoch EVT a EVTX

1.2.2 MFT

Súborový systém NTFS (New Technology File System) je predvoleným súborovým systémom pre najpopulárnejší súborový systém Microsoft Windows NT+ a Windows Server. Poskytuje možnosť hierarchického ukladania údajov. Obsahuje súbor, ktorého úlohou je udržiavať informácie o každom súbore v súborovom systéme. Nazýva sa Master file table (MFT). Nachádza sa v koreňovom adresári súborového systému a bez použitia forenzných nástrojov je používateľom skrytý [15]. Súbor MFT je súbor metadát,

ktorý obsahuje informácie o všetkých súboroch a adresároch v aktuálnom oddiele s formátom NTFS. V rámci tohto súboru sa udržiavajú nasledujúce informácie o súboroch:

- názov súboru,
- veľkosť,
- umiestnenie súboru na disku,
- informáciu o časových pečiatkach,
- oprávnenia,
- obsah súboru (ak je súbor menší ako ~ 900 bajtov).

Každý záznam je vybavený inou sadou atribútov, ako je STANDARD_INFORMATION ($SIA) a FILENAME ($FNA) [16]. Zvyčajne operačný systém Windows, aj väčšina forenzných nástrojov, zobrazuje iba informácie o časovej pečiatke, ktoré sú uvedené v atribúte štandardných informácií (SIA) záznamu MFT. Ide o časové pečiatky, ktoré sa aktualizujú pri skopírovaní priečinku alebo súboru, jeho presunutí, vpísaní alebo pri práci s ním. Na rozdiel od časových pečiatok uchovávaných v FNA, ktoré vytvoria pri prvom vytvorení súboru na zväzku.

Pri pridaní súborov do súborového systému NTFS sa do MFT pridá viac položiek a veľkosť MFT sa zvýši. Avšak po odstránení súborov zo súborového systému NTFS sú ich položky v MFT označené ako neplatné a je možné ich znova použiť. Miesto na disku, ktoré bolo pridelené pre tieto položky, však nie je prerozdelené a veľkosť MFT sa nezmenšuje. Keď sa pridávajú nové súbory, v závislosti od veľkosti MFT sa pridávajú buď k starým (odstráneným) záznamom, alebo k novým záznamom. Dôvodom tejto funkcie je MFT je veľkoobjemový súbor, ku ktorému sa často pristupuje.

Súborový systém NTFS vyhradzuje priestor pre MFT, aby udržiaval MFT čo najviac súvislý v prípade jej rastu. Miesto vyhradené súborovým systémom NTFS pre MFT sa nazýva zóna MFT. V závislosti na priemernej veľkosti súboru a ďalších premenných môže byť najskôr vyhradená zóna MFT alebo nevyhradené miesto na disku alokované ako prvé, kým sa disk zaplní kapacitou. MFT často obsahuje množstvo vymazaných informácií vďaka čomu sa vyznačuje ako významný forenzný artefakt.

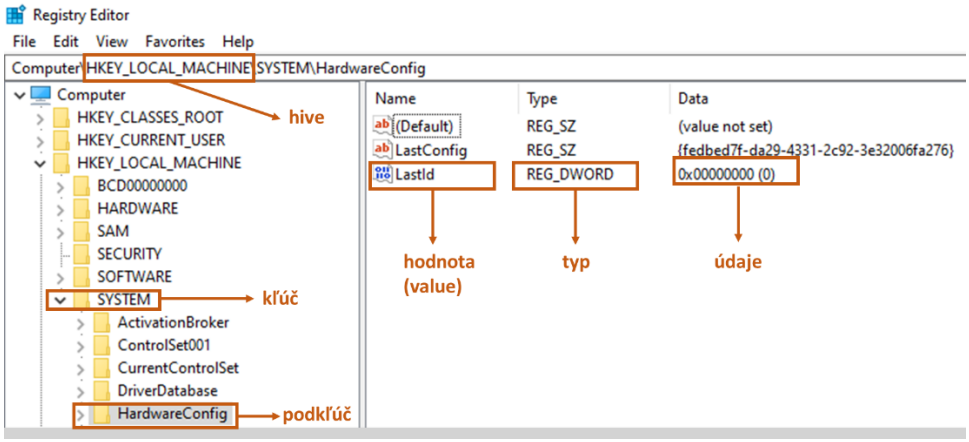
Časové pečiatky systému NTFS sa od pečiatok zaznamenaných v systémoch FAT líšia v niekoľkých veciach. Prvým zásadným rozdielom je umiestnenie metadát, ktoré sa nenachádzajú na rovnakých miestach. NTFS používa ako primárne úložisko metadát

dátumu a času súbor $MFT, zatiaľ čo systémy FAT zaznamenávajú dátumy a časy v rámci položiek priečinkov. Ďalším významným rozdielom je, že časové pečiatky systému NTFS sa zaznamenávajú v **UTC** časovej zóne bez ohľadu na časové pásmo nastavené pre operačný systém, zatiaľ čo časové pečiatky systému FAT sa zaznamenávajú v miestnom čase, čo je veľmi dôležité pri manuálnej interpretácii časových pečiatok. Okrem toho súborový systém NTFS reprezentuje časové pečiatky pomocou formátu FILETIME, ktorý predstavuje počet 100 nanosekundových intervalov od 1. januára 1601. Pri FAT sa primárne používa DOSDATETIME, ktorý má veľkosť 4 bajty a začína sa 1. januára 1980 [17].

1.2.3 Artefakty registra operačného systému Windows

Pri analýze operačného systému Windows sú veľmi hodnotnými artefaktmi tie, ktorých zdrojom sú registre. Register predstavuje hierarchickú databázu, ktorá obsahuje údaje zásadné pre činnosť operačného systému Windows. Významné sú aj pre činnosť aplikácií a služieb, ktoré sú v systéme spustené. Na Obrázku č. 2 sú zobrazené jednotlivé komponenty registra. Na obrázku je možné vidieť aj stromovú štruktúru, do ktorého sú údaje štruktúrované. Každý uzol v strome sa nazýva kľúč. Každý kľúč môže obsahovať podkľúče aj údaje, ktoré sa nazývajú hodnoty. V niektorých prípadoch je prítomnosť kľúča jediným údajom, ktoré aplikácia vyžaduje. V iných prípadoch aplikácia otvorí kľúč a použije hodnoty spojené s týmto kľúčom. Kľúč môže mať ľubovoľný počet hodnôt a hodnoty môžu byť v akejkoľvek podobe [18].

Kľúče v rámci ich binárnej štruktúry tiež obsahujú z forenzného hľadiska veľmi cenné informácie, a to čas ich posledného zápisu - **LastWrite Time**. Táto časová pečiatka je porovnateľná s časom poslednej úpravy súboru. Kedykoľvek sa vytvorí, zmení alebo vymaže kľúč databázy Registry alebo jeho hodnota, aktualizuje sa na aktuálny miestny systémový čas. Aj keď hodnota registra nie je spojená so žiadnym časom posledného zápisu, možno ju odvodiť z času posledného zápisu kľúča registra. Hodnoty zobrazené v pravej časti Obrázka č. 2 sú jednoduchšieho charakteru, obsahujú údaje konkrétneho typu, či už je to hodnota reťazca, viacnásobné hodnoty reťazca, binárne hodnoty alebo DWORD, čo je iba 32-bitová binárna hodnota.



Obrázok 2 Komponenty registra

V rámci registrov existuje päť hlavných koreňových kľúčov, ktoré sú zobrazené v Tabuľke č. 6. Každý z nich obsahuje nastavenia týkajúce sa používateľov alebo samotného operačného systému.

Názov	Funkcia
HKEY_CURRENT_USER	Obsahuje nastavenia pre aktuálneho používateľa, ako je napr. farba obrazovky, veľkosť obrazovky, nastavené pozadie, štruktúra ukladania priečinkov či nastavenia ovládacieho panela.
HKEY_USERS	Obsahuje profily každého používateľa v systéme, vrátane konfigurácií aplikácií a vizuálnych nastavení.
HKEY_LOCAL_MACHINE	Obsahuje informácie o konfigurácii a nastavení, ktoré systém používa pri spustení. Je nezávislý od prihlásenia používateľa. Obsahuje 5 podkľúčov: System, Software, SAM, Security, Hardware.
HKEY_CLASSES_ROOT	Tento kľúč obsahuje podkľúče, ktoré sú pomenované po jednom rozšírení, ktoré je možné nájsť v systéme, napríklad .png a .pdf. Tento koreňový kľúč popisuje predvolený program, ktorý sa musí použiť na otvorenie tohto rozšírenia v systéme. Ukladá tiež podrobnosti ponuky, ktorá sa zobrazí pri stlačení pravého tlačidla myši a ikonu programu.
HKEY_CURRENT_CONFIG	Obsahuje informácie o hardvéri, ktorý zariadenie používa.

Tabuľka 6 Popis piatich koreňových kľúčov

Dáta z registrov poskytujú z pohľadu forenznej analýzy množstvo ďalších zaujímavých informácií. Nachádzajú sa tam napríklad lokácie, z ktorých je možné spúšťanie aplikácií bez priamej interakcie používateľa. Veľké množstvo škodlivého softvéru to využíva na zachovanie perzistencie v systéme. Jednou z týchto lokácií je kľúč registra *HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run.*

Zdrojom dát, čo sa týka informácií ohľadom činností, ktoré vykonal používateľ, môže byť súbor NTUSER.DAT, ktorý obsahuje všetky jeho nastavenia. Obsah tohto súboru je namapovaný na hive *HKEY_USERS\SID*. Po prihlásení používateľa sa z hive *HKEY_USERS\SID* vytvorí hive *HKEY_CURRENT_USER*.

Kľúč UserAssist, ktorý sa nachádza v Software\Microsoft\Windows\ CurrentVersion\Explorer\UserAssist obsahuje dva podkľúče. Každý z nich obsahuje informácie o spúšťaných GUI aplikáciách, počet spustení, čas posledného spustenia a pod. Podkľúč GUID určuje spôsob, akým boli spustené (odkaz, dvojklik na spustiteľný súbor a pod.), a podkľúč Count obsahuje názvy súborov obfuskované funkciou ROT13, v ktorom sú znaky nahradené znakom vzdialeným od neho 13 medzier v tabuľke ASCII. Záznamy po dekódovaní nie sú spojené s konkrétnym dátumom a časom, ale dokazujú prístup používateľa k určitému súboru alebo objektu.

1.2.4 Iné forenzné artefakty

Okrem vyššie spomenutých forenzných artefaktov sú pre forenzné vyšetrovanie zaujímavé aj ďalšie artefakty, medzi ktoré zaraďujeme nasledujúce položky:

- Shortcut Files a .lnk
- Windows Task Bar Jumplisty,
- „Charm Bar“ (história vyhľadávania),
- Shellbags

Link Files, Shortcut Files, alebo Shell Link Items sú cenné forenzné artefakty. Okrem časových pečiatok súborového systému môžu v rámci svojej štruktúry obsahovať prídavné informácie o cieľovom súbore, ako sú názvy zväzkov, sériové čísla a informácie o ceste k súboru.

Jump listy môžu obsahovať automatické(*.automaticDestinations-ms) alebo custom destinations. Sú uložené vo formáte OLE (Object Linking and Embedding) Compound File (CF) alebo OLE CF. Tieto súbory majú predvolenú štruktúru, ktorá sa skladá z header, sector allocation tables a directory,

Windows ShellBags boli zavedené do operačného systému Windows 7 a sú i naďalej prítomné. Vo všeobecnosti uchovávajú informácie o preferenciách používateľa napríklad pri prehliadaní priečinkov, nastavenie prehliadacích okien alebo ikon. Všetky tieto nastavenia sa uložia do ShellBag. ShellBags obsahujú dva hlavné kľúče registra, BagMRU a Bags. V kľúči BagMRU sú uchovávané názvy priečinkov a taktiež cesta k nim. Kľúč Bags ukladá nastavenia zobrazenia, ako sú veľkosť okna či režim zobrazenia.

Ďalším typom forenzného artefaktu sú Application Compatibility Cache (Shimcache). Medzi verziami operačných systémov na zabezpečenie spätnej kompatibility starších binárnych súborov s novšími verziami operačných systémov sa používa ShimCache ako proxy vrstva [19]. Obsahuje Last Recently Used (LRU) záznamy zoradené zhora nadol. Záznam je vytvorený pri spustení súboru vo verziách operačného systému Windows XP a vyššie. V niektorých verziách operačného systému Windows môžu vznikať záznamy Shimcache pre súbory v adresári, ktorý používateľ interaktívne prechádza. Nakoniec sa do pamäte Shimcache zaznamená aj čas poslednej zmeny súboru. Vo všeobecnosti však časová pečiatka poslednej zmeny nehovorí o tom, kedy sa súbor spustil. Zo Shimcache záznamov je možné identifikovať, že súbor bol spustený, nie však kedy k tomu došlo.

Na vytvorenie časovej osi je potrebné najskôr extrahovať časové záznamy súvisiace s analyzovanými udalosťami. Najbežnejšími zdrojmi takýchto časových záznamov sú časové pečiatky v rámci súborového systému [20].

Operačný systém Windows používa niekoľko rôznych časových formátov najmä pre rôzne systémové súbory, dátové položky a hodnoty registra. Najčastejšie používané časové formáty sú:

- 64-bitový FILETIME (UTC) - počet 100 nanosekundových intervalov od 1. 1. 1601,
- 32-bitový unixový formát času (UTC) - počet sekúnd od 1. 1. 1970,
- formát založený na reťazci (miestny čas),
- SYSTEMTIME (miestny čas).

Analýza časových pečiatok súborov sú jednou z častých a používaných techník počítačových forenzných expertov na určenie času a identifikáciu podozrivého správania. Metadáta súborového systému NTFS [21], ktoré zaznamenávajú a uchovávajú časové pečiatky najnovších akcií so súbormi sa označujú ako **časové pečiatky** modifikácie, prístupu, zmeny a vytvorenia pre záznamy súborov a adresárov. Skrátene sa označujú ako časové pečiatky MACB a sú uvedené pre súborový systém FAT a NTFS v Tabuľke č. 7.

Súborový systém	M (Modification)	A (Access)	C (Creation)	B (Birth)
FAT	Vpísanie	Pristúpenie	-	Vytvorenie súboru (C)
NTFS	Modifikácia súboru	Pristúpenie	Modifikácia MFT položky	Vytvorenie súboru

Tabuľka 7 MACB časové záznamy

MACB je rozšírením časových pečiatok MAC, ktoré sa nachádzajú vo väčšine tradičných unixových systémov a ktoré sa zobrazujú príkazom *stat*. V tomto prípade ide o časové pečiatky, ktoré súvisia s časom poslednej zmeny (M), časom posledného prístupu (A) a časom vytvorenia (C). Čo sa týka aktualizácie záznamov pri MAC, časová pečiatka M sa aktualizuje pri zmene údajov súboru, časová pečiatka A sa aktualizuje pri prístupe k súboru, a časová pečiatka C sa aktualizuje pri vytvorení súboru. V súborovom systéme NTFS sú časy MACB v dvoch atribútoch Master File Table, a to v Standard Information (SI) a Filename (FN) [22]. V $STANDARD_INFO, s ktorým pracujú nástroje na zobrazovanie časových pečiatok, sú naviac uložené metadáta súboru, ako napríklad príznaky, SID a taktiež je uvedený aj vlastník súboru. Údaje môžu byť modifikované procesmi na úrovni používateľa. Vo $FILE_NAME sa okrem časov MACB nachádza napríklad aj názov a dĺžka súboru. Na rozdiel od $STANDARD_INFO, $FILE_NAME môže byť modifikovaný len jadrom operačného systému.

Pri MACB **časová pečiatka modifikácie (M)** označuje, kedy bol obsah súboru naposledy aktualizovaný. Tento časový záznam predstavuje posledný čas, kedy bol atribút MFT $DATA, ktorý obsahuje predvolené údaje súboru [23], zmenený. Napríklad, ak používateľ súbor otvorí, upraví jeho obsah a znovu uloží, bude mať súbor hodnotu Last Written (naposledy zapísané). Naopak, ak používateľ ten istý súbor otvorí, prečíta

jeho obsah a potom súbor zatvorí bez vykonania zmeny, čas posledného zápisu sa neaktualizuje.

Časová pečiatka prístupu (A) sa vzťahuje na čas posledného prístupu k obsahu, teda predstavuje posledný prístup k súboru alebo priečinku v súborovom systéme. Táto pečiatka nemusí nevyhnutne znamenať, že bol súbor otvorený. Jednoduché umiestnenie myši na názov súboru vo Windows Prieskumníkovi môže aktualizovať časovú pečiatku posledného prístupu. Jej zmena môže byť aj výsledkom automatických alebo neškodných systémových činnosti, ako je antivírusová kontrola či zálohovanie súborov. Aktualizácie časových pečiatok súborového systému NTFS môžu byť oneskorené až o jednu hodinu od prístupu súboru, avšak je to dostatočné na určenie všeobecného používania operačného systému.

Časová pečiatka zmeny (C) sa vzťahuje na zmeny metadát súboru - jeho názvu, bezpečnostných atribútov a všetkých ostatných časových značiek. Táto časová pečiatka teda udáva posledný čas, kedy došlo k úprave akéhokoľvek atribútu v zázname MFT pre súbor alebo priečinok. Dôvodom na aktualizáciu tejto časovej pečiatky môže byť zmena umiestnenia súboru na disku, pridanie ďalšieho dátového toku do súboru alebo zmena názvu súboru. **Časová pečiatka vytvorenia (B)** sa vzťahuje na čas vytvorenia položky MFT, vytvorenie súboru alebo priečinka v súborovom systéme. Jeho obsah môže byť neskôr úplne nahradený a jeho názov, umiestnenie a bezpečnostné atribúty sa môžu zmeniť. Pri kopírovaní existujúceho súboru sa napríklad časová pečiatka vytvorenia súboru novej kópie nastaví na aktuálny čas. Čas modifikácie a posledného zápisu však ostáva nezmenený.

Tabuľka č. 8 zobrazuje popis jednotlivých kombinácií časových pečiatok podľa rôznych zdrojov. Ako je možné z tejto tabuľky vidieť, rôzne zdroje sa líšia v popise, ako možno interpretovať rôzne operácie so súbormi a adresármi na základe ich časových pečiatok. Interpretácia časových pečiatok je závislá aj od verzie operačného systému Windows.

	[24]	[25]	[26]	[27]	[28]
C	File Rename, Local File Move	File Rename, Local File Move	File/Folder Rename, Local File Move, Local Folder Move	Rename, Local Move, Delete	File Rename, Local File Move
A	File Access	File Access, Volume File Move	File/Folder Access	Access	File Access
M	File Modify	-	-	-	-
BA	-	File Copy, Volume File Move	File Copy (win7/cmd, win8+), Volume File Move (cmd)	-	-
CA	Volume File Move	-	Volume File Move (explorer)	Volume Move	Volume File Move
AM	-	-	-	Modify	-
CM	-	File Modification	-	-	File Modify
ACB	File Copy	-	File Copy, Unzipped File Folder,	Copy	File Copy
MAC	-	-	Volume Folder Move, File Modification (Office), File Modification, File Contents Modification	-	-
MACB	File Creation	File Creation, File Copy, Volume File Move	File/Folder Creation, File Download, Folder Copy, Unzipped File	Create	File Creation

Tabuľka 8 Porovnanie operácií so súbormi na základe výskytu časových pečiatok M, A, C a B

1.3 Vytváranie časových osí

V digitálnej forenznej analýze prístup, ako reprezentovať súbory udalostí v postupnom chronologickom usporiadaní označujeme ako **časová os** [29]. Časový záznam pri časových osiach sa volí v závislosti od uvažovaných údajov.

Tak, ako aj pri bežnej rekonštrukcii miesta činu, je aj pri digitálnom forenznom vyšetrovaní rovnako dôležité vedieť zostaviť časovú os činností v počítačovom systéme. Časová os je prínosom pri prehľade alebo prezentácii prístupu používateľov k cieľovému systému. Sledujú sa taktiež spustenia určitých aplikácií a identifikácia systémových a dátových súborov, ku ktorým bolo pristúpené, ktoré boli modifikované alebo vymazané počas období, ktoré sú pre prípad relevantné a zaujímavé.

Digitálne forenzné vyšetrovanie sa zameriava na získanie relevantných informácií dostupných v systéme z metadát a časovej osi na identifikáciu položiek s významnou

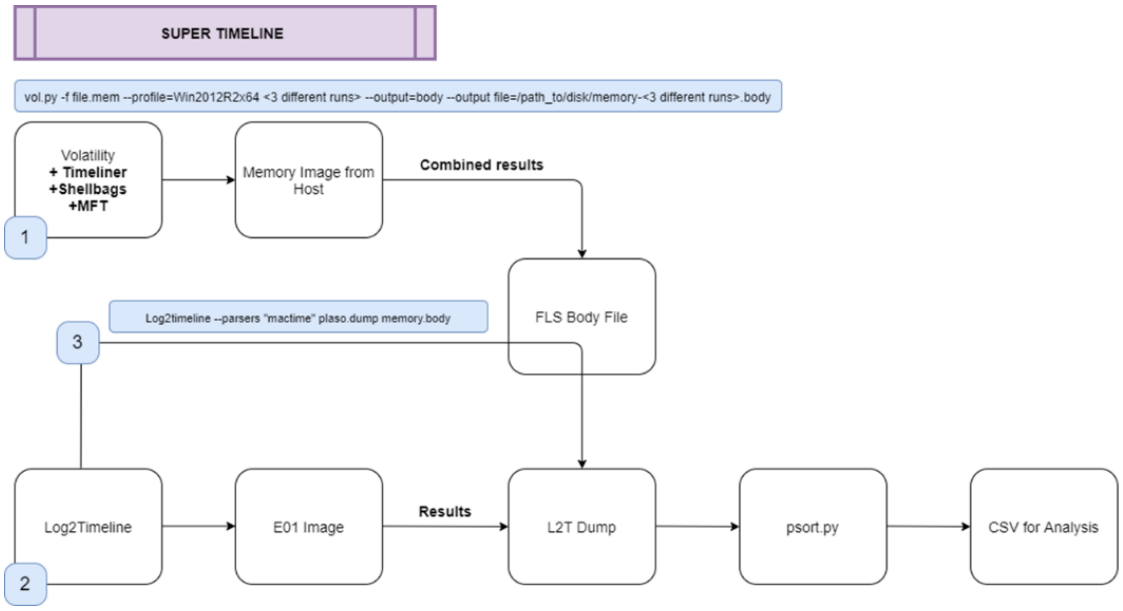
forenznou hodnotou. Na filtrovanie a indexovanie súborov sa zvyčajne používajú metadáta ako veľkosť súboru, cesta k súboru, názov súboru a podobne. Metadáta o adresároch sa používajú na zistenie napríklad časovej asociácie medzi súbormi a podobne [30].

Vytvorenie časovej osi teda možno považovať za proces iterácie nad súbormi, ich metadátami a výstupu chronologicky usporiadanej postupnosti udalostí, napríklad zobrazenie toho, kedy bol súbor vytvorený, upravený a sprístupnený alebo vymazaný. Tento druh časovej osi umožňuje získať globálny prehľad o udalostiach, ktoré sa vyskytli pred, počas a po danej udalosti.

V určitom súborovom systéme sa však môže vyskytnúť obrovské množstvo udalostí, čo znamená jeho komplikované analyzovanie. Z tohto dôvodu je veľmi dôležitá interpretácia časovej osi, ktorá by mala byť dostatočne jednoduchá a intuitívna, aby sa predišlo chybnému rozhodovaniu.

Prehľad prístupu k vytváraniu časových osí a rekonštrukcii udalostí nám prinášajú autori v [31]. Rekonštrukcia udalostí sa odkazuje na proces konvertovania stavov digitálnych objektov na udalosti, ktoré ich spôsobili. Vieme do toho zahrnúť aj schopnosť povedať o udalosti, že skutočne nastala. Zdroje časových záznamov zahŕňajú časové pečiatky MAC, ale môžu zahŕňať aj časové pečiatky zložitejších formátov, napr. z Windows registrov, SQLite databázy, logov udalostí a ďalších.

V článku [32] sa autori venujú generovaniu časových osí. Na Obrázku č. 3 je možné vidieť postup spracovania údajov využitím nástrojov Volatility, Log2timeline (plaso) a psort. Na vstupe je v tomto prípade operačná pamäť a forenzný obraz disku. Výstupom je časová os vo formáte l2tcsv. V prípade, že je k dispozícii aj obraz operačnej pamäte, je možné kombinovať výsledky získané z tohto primárneho úložiska s digitálnymi stopami získanými zo sekundárnych úložísk (diskov, externých pamäťových médií). V tomto prípade hovoríme o tvorbe **super časovej osi (super timeline).**



Obrázok 3 Schéma fázy predspracovania údajov. Prevzaté z [33]

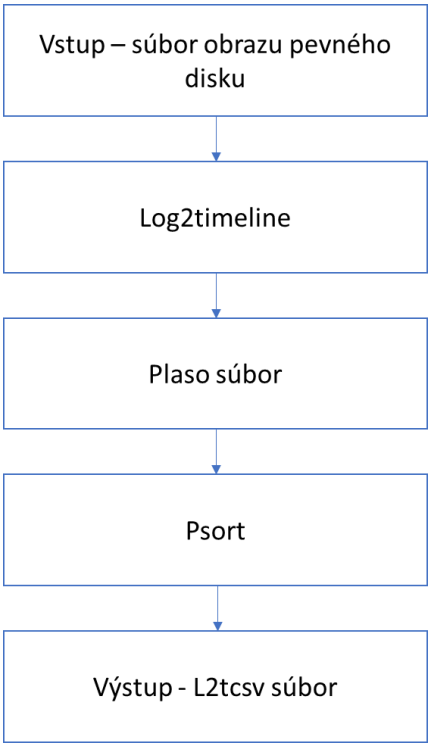
1.3.1 Nástroj Plaso

Najpoužívanejšou technológiou z hľadiska extrakcie časových značiek digitálnych udalostí z rôznych logovacích súborov, ktoré sa nachádzajú v typickom počítačovom systéme, a následne ich spojenie a utriedenie, je nástroj **Plaso (log2timeline)** [34]. Ide o nástroj, ktorý umožňuje vytvárať super časové osi, ale podporuje tiež vytváranie cielenejších časových osí. Disponuje veľkým množstvom zásuvných modulov, analyzátorov a parserov, ktoré vykonávajú predbežné zisťovanie digitálnych udalostí obsiahnutých v zariadení a umožňujú tak analytikom zamerať sa na dôležité forenzné artefakty. Niektoré z nich sú priblížené v Tabuľke č. 9.

Meno artefaktu	Dôkaz o	Popis	Parser
EVTX Logs	Vytvorenie konta Inštalácia služby(malvér) Vzdialené prihlásenie Exploity PSExec	Systémové záznamy Windowsu. V systéme je množstvo záznamov udalostí, každá zodpovedá inému silu. Napr. bezpečnostné udalosti sa sledujú v súbore Security EVTX.	**winevtx**
Prefetch	Spustenie programu Otvorenie súboru/adresára	Cesta a názov každej aplikácie, ktorá je spustená je zapísaná do súboru .pf. Každý pf súbor obsahuje posledný čas vykonania, počet spustení a popisy zariadení a súborov, ktoré program používa.	**prefetch**
Recycle Bin	Súbory, ktoré boli vymazané	C:\$Recycle.Bin Obsahuje čas vymazania súboru a názov pôvodného súboru.	**recycle_bin, recycle_bin_info2**
SRUM (System Resource Utilization Manager)	Spustenia programov Sieťová aktivita	Zaznamenávanie výkonu systému Info ako: používateľský účet spojený s vykonaním, odoslané a prijaté úlohy za aplikáciu za hodinu.	**srum**
USN Journal (Update Sequence Number Journal)	Dáta, ktoré boli k dispozícii (stiahnuté, odstránené,..)	Jednotky NTFS USN žurnál. Analýza poskytne prehľad o zmenách vykonaných v rámci volume. Údaje môžu zahŕňať zmenu, ku ktorej došlo, kedy k nej došlo, názov súboru, atribúty súboru a údaje MFT.	**usnjrnl**
Scheduled Tasks	Perzistencia Využitie účtu	Zobrazené v 2 EVTX záznamoch: 1. Security.EVTX 2. Microsoft-Windows-TaskScheduler Maintenance	**winevtx windows_task_cache**

Tabuľka 9 Opis vybraných parserov nástroja Log2timeline

Vo fáze predspracovania údajov využijeme tento nástroj na vytvorenie časovej osi. Vstupom pre nástroj Log2timeline je forenzný obraz disku. Na Obrázku č. 4 vidíme, že aplikáciou nástroja Log2timeline na obraz disku vo formáte E01 dostaneme súbor vo formáte plaso. Ten vieme prekonvertovať na čitateľný súbor vo formáte l2tcsv pomocou nástroja psort [35]. Je to nástroj príkazového riadka, ktorý dokáže spracovať plaso súbory. Umožňuje ich filtrovať, triediť a spúšťať nad nimi automatickú analýzu. Tento súbor obsahuje extrahované udalosti a rôzne metadáta o procese zberu spolu s informáciami získanými zo zdrojových údajov.



Obrázok 4 Fáza predspracovania údajov. Prevzaté z [32].

1.3.2 Formát časovej osi

V predvolenom nastavení používa log2timeline výstupný formát **l2t_csv**. Je to jednoduchý súbor CSV (oddelený čiarkou) so 17 poľami. Prvý riadok súboru CSV je hlavička, ktorá obsahuje názov každého poľa, za ktorým nasleduje každá položka s časovou pečiatkou. Každý riadok v súbore CSV odkazuje na jeden záznam s časovou

pečiatkou, čo znamená, že každý objekt časovej pečiatky môže byť rozšírený na niekoľko riadkov. Popis jednotlivých polí je uvedený v Tabuľke č. 10.

Pole	Popis
Date	dátum, kedy sa udalosť stala
Time	čas, kedy sa udalosť stala
Timezone	časová zóna, ktorá bola použitá pri volaní nástroja
MACB	časové pečiatky (**M**odification, **A**ccess, **C**reation, **B**irth)
Source	krátky názov zdroja (napr. záznamy registra - REG)
Sourcetype	popis zdroja
Type	typ časovej značky (napr. posledný prístup alebo posledný zápis)
User	ak existuje, meno používateľa spojené s udalosťou
Host	ak existuje, názov hostiteľa, ktorý je priradený k udalosti
Short	obsahuje pole s krátkym popisom, v ktorom je uložený text
Desc	pole, ktoré obsahuje väčšinu analyzovaných informácií
Version	udáva číslo verzie časovej pečiatky
Filename	názov súboru spojeného s udalosťou
Inode	udáva číslo inodu analyzovaného súboru
Notes	dodatočné miesto na ukladanie informácií pre niektoré vstupné moduly
Format	vstupný modul, ktorý bol použitý na analyzovanie
Extra	pole s parsovanými informáciami, ktoré sú tu spojené a uložené.

Tabuľka 10 Polia formátu l2t_csv s krátkym popisom podľa [32]

Dátum (Date) udalosti je uvedený vo formáte „MM/DD/RRRR“ a **Čas (Time)** udalosti vyjadrený v 24-hodinovom formáte, HH:MM:SS.

Typ zdroja (sourcetype) predstavuje komplexnejší popis zdroja. Toto pole ďalej popisuje formát, ako napríklad „Syslog“ namiesto jednoduchého „LOG“, „NTUSER.DAT Registry“ namiesto „REG“ a pod.

Zdroj (Source) obsahuje skrátený názov zdroja. Využíva sa 11 hlavných zdrojov, ktoré sú popísané v Tabuľke č. 11.

Zdroj	Popis
AMCACHE a AMCACHEPROGRAM	Forenzný artefakt, v ktorom sa ukladajú metadáta o inštalácii a spustení programu v systéme Windows. AmCache.hve obsahuje úplnú cestu k spustiteľnému súboru, NTFS $Standardlnfo Last Modification Time a haš SHA1.
EVT	Windows Event Logs (EVT) sú súbory, do ktorých sa ukladajú významné udalosti operačného systému Windows - chybové správy z aplikácií, operačného systému, ale aj informácie týkajúce sa prihlásenia používateľa na zariadenie, spustenie a ukončenie služieb.
FILE	V tomto zdroji dát sa nachádzajú záznamy o udalostiach súborového systému. Informácie o tvorení, prístupe či modifikácii súborov.
LNK	LNK súbory sú vytvárané automaticky operačným systémom Windows pri prístupe k súboru. Používateľ ich však môže vytvoriť aj manuálne. LNK súbory zvyčajne obsahujú metadáta o súbore, vrátane názvu a veľkosti súboru, pôvodnej cesty, časových pečiatok, informácií o zväzku a systéme. Užitočné sú hlavne v prípade súborov, ktoré už nie sú k dispozícii v systéme.
LOG	Všetky logovacie súbory, ktoré ukladajú informácie o udalostiach v rámci operačného systému Windows sa nachádzajú v tejto časti, spoločne označené ako LOG.
OLECF	Tento zdroj údajov nám poskytuje záznamy o Object Linking and Embedding (OLE) Compound File (CF) súboroch v rámci operačného systému. Obsahuje súbory .msp, .msi, .asd a .automaticdestination-ms, ktoré poskytujú informácie o aktualizáciách operačného systému Windows a iných programoch.
PE	V tejto časti nástroj Log2timeline ukladá informácie o súboroch PE - Portable Executable, ktoré sú extrahované parserom *pe*. Súbory vo formáte PE zahŕňajú súbory .exe, .dll, a .sys (súbory ovládačov).
RECBIN	Na tomto mieste sa nachádzajú záznamy parsované recycler parserom, ktorý extrahuje obsah, ktorý sa nachádza v Koši (Recycle Bin).
REG	Zdroj REG poskytuje záznamy o aktivite registrov operačného systému Windows.
WEBHIST	Informácie týkajúce sa prehliadania webových stránok. Nájdeme tu informácie, ktoré zahŕňajú adresy webových stránok, ku ktorým používateľ pristupoval, poštové adresy používateľa, stiahnuté súbory, a aplikácie používané na prístup k online službám.

Tabuľka 11 Popis zdrojov dát, ktoré sa nachádzajú vo formáte l2tcsv.

Typ (Type) samotnej časovej pečiatky, napríklad „Posledný prístup“, „Naposledy napísané“ alebo „Posledná úprava“ atď.

Krátky (Short) popis položky, zvyčajne obsahuje menej textu ako pole úplného popisu - Desc. Nachádzajú sa tam informácie nevyhnutné pre nástroje, ktoré tvoria časovú os a vizualizujú udalosť.

Popis (Desc) - tu je uložená väčšina informácií. Toto pole predstavuje úplný popis poľa, interpretované výsledky alebo obsah skutočného riadku zo záznamu udalosti.

Extra - dodatočné analyzované informácie sa spoja a vložia do tohto poľa, ktoré môže obsahovať rôzne extra informácie, ktoré bližšie popisujú udalosť. Niektoré vstupné moduly obsahujú dodatočné informácie o udalostiach, ako napríklad ďalšie rozdelenie udalosti na zdrojové IP adresy atď. Tieto polia sa nemusia priamo zmestiť do žiadneho iného poľa v súbore CSV, a preto sú spojené do tohto poľa. V prípade evtx záznamov sa v tomto poli nachádza celý xml reťazec.

2 Podobné práce

V rámci tejto kapitoly sa bližšie zameriame na technológie, resp. odborné a vedecké články, ktoré sú relevantné pre túto záverečnú prácu. Samotné podobné práce sme rozdelili do štyroch podkapitol, a to analýzu nástrojov, ktoré sa používajú pri tvorbe a analýze časových osí, analýzu odborných a vedeckých článkov pri tvorbe časových osí, analýze časových osí a pri korelácii jednotlivých záznamov týchto osí.

2.1 Nástroje pre zber forenzných artefaktov

V tejto časti sa nachádza porovnanie dostupných nástrojov, ktoré vedia zozbierať a spracovať zvolené dáta na požadovaný formát, ktorý vieme spracovať v ďalšej časti riešenia. V porovnaní sme zvažovali aj faktory ako operačný systém, ktorý podporujú a taktiež, či sú dostupné aj v neplatenej verzii. Porovnanie je zobrazené v Tabuľke č. 12.

Názov nástroja	Zameranie na	Funkcionalita	OS	Výstup vo formáte	Neplatený
Volatility	Obraz pamäte	Extrakcia špecifických artefaktov z dumpu pamäte	Windows, Linux, Mac Os	Plaintext	✓
Memoryze	Obraz pamäte	Zaisťovanie a analýza obrazu pamäte	Windows, Mac Os	Dump file	✓
MAGNET Process Capture	Obraz pamäte	Zaistenie pamäte zo spustených procesov	Windows	Dump file	✓
FTK Imager	Image disku	Analýza a vytvorenie kópie disku	Windows	dd, E01	✓
MAGNET Encrypted Disk Detector	Image disku	Cmdline nástroj na zaistenie, kontrolu a analýzu šifrovaných častí disku pri live analýze.	Windows		✓
EnCase	Image disku	Zaistenie diskov, hĺbková analýza.	Windows	E01	Akademická verzia
Windows Backup and Restore	Image disku	Funkcionalita OS Windows na výrobu kópie disku.	Windows	E01	✓

Tabuľka 12 Porovnanie dostupných nástrojov na spracovanie obrazu pamäte a disku

Volatility [36] je jeden z najlepších nástrojov s otvoreným kódom na analýzu operačnej pamäte v 32-bitových alebo 64-bitových systémoch. Podporuje analýzu pre systémy Linux, Windows, Mac a Android. Jeho základom je programovací jazyk Python. Dokáže analyzovať nespracované "raw" dáta, výpisy VMware (.vmem), výpisy virtuálnych boxov a veľa ďalších.

Memoryze [37] je voľne dostupný forenzný nástroj , ktorý pomáha pri identifikácii útočníka v bežiacej operačnej pamäti pri riešení kybernetického bezpečnostného incidentu. Memoryze môže získavať alebo analyzovať obrazy operačných pamätí a na živých systémoch môže do svojej analýzy zahrnúť aj stránkovací súbor.

MAGNET Process Capture [38] je bezplatný nástroj, ktorý umožňuje selektívne zachytiť pamäť z jednotlivých bežiacich procesov. Je schopný jednotlivé procesy načítať a poskytnúť menej fragmentované údaje.

FTK Imager [39] slúži na vytváranie forenzných kópií diskov bez toho, aby boli vykonané zmeny v pôvodných dátach. Vyznačuje sa taktiež rýchlosťou pri vytváraní kópií diskov a zabezpečením integrity dôkazov vytvorením ich digitálneho odtlačku (hašu).

MAGNET Encrypted Disk Detector [40] je voľne dostupný nástroj príkazového riadku, dostupný pre operačný systém Windows 7 a novší. Využíva sa na zaistenie a kontrolu zašifrovaných dát na disku pri live analýze operačného systému.

EnCase [41] je jeden z najznámejších a najpoužívanejších forenzných nástrojov, ktorý bol vytvorený spoločnosťou Guidance Software Inc. Obsahuje funkcie na zobrazovanie a zaisťovanie pevných diskov, ale aj CD a USB disku, a taktiež absolútne obnovenie údajov vo forme bitového toku. Výstupom je súbor vo formáte "E01", známy ako Encase Image File Format (Formát obrazových súborov Encase) [42]. Tento formát je známy aj pod skratkou EWF (Expert Witness Format). Súbor v tomto formáte uchováva zálohu rôznych typov získaných digitálnych stôp, ktoré zahŕňajú zachytený obraz disku, ako aj uloženie logických súborov. Pri vytváraní obrazov údajov dostupných na pevnom disku sa rozdelia kompletné údaje na menšie dátové časti. V dôsledku toho sa vytvorí viacero dátových súborov, v ktorých sú uložené rozhodujúce informácie o pevnom disku.

Windows Backup and Restore [43] je voľne dostupná funkcia operačného systému Windows, ktorá je v ňom predvolene zabudovaná. Na vytvorenie systémového obrazu vybranej jednotky je nutné, aby používala súborový systém NTFS.

2.2 Nástroje pre analýzu časových osí

Prvú skupinu podobných prác predstavujú nástroje, resp. technológie použité pre vytvorenie a analýzu časových osí.

Prvým príkladom tohto nástroja je Plaso [34], ktorý je založený na programovacom jazyku Python a ktorý používa niekoľko ďalších nástrojov na automatické vytváranie časových osí. Plaso vytvára super časové osi, ale podporuje aj vytváranie cielenejších časových osí. Časové osi podporujú digitálnych forenzných vyšetrovateľov/analytikov, aby korelovali veľké množstvo informácií nájdených v záznamoch (logoch) a iných súboroch nájdených na počítači.

Predchodcom nástroja Plaso je nástroj Log2Timeline [44]. Ide o nástroj príkazového riadka na extrahovanie udalostí z jednotlivých súborov, adresárov, napríklad bodu pripojenia alebo obrazu pamäťového média alebo zariadenia. Log2timeline vytvorí tzv. plaso súbor, ktorý možno analyzovať pomocou nástrojov pinfo a psort.

V súčasnej dobe najčastejšie používané nástroje dopĺňa nástroj Timesketch [45]. Ide o nástroj s otvoreným zdrojovým kódom pre kolaboratívnu forenznú analýzu časovej osi. Použitím „náčrtov“ (sketches) je možné jednoducho organizovať časové osi a analyzovať ich všetky súčasne. Nespracovaným údajom je možné pridávať anotácie, komentáre, značky a podobne.

Okrem vyššie uvedených nástrojov nachádzame aj odborné, resp. vedecké články, v ktorých autori popisujú iné nástroje pre tvorbu a analýzu časových osí. Prvým príkladom je článok [46]. Autori v tomto článku sa venovali vysvetleniu, že rekonštrukcia udalostí je nevyhnutným krokom pre pochopenie prípadu. Predstavili nástroj Timeline2GUI, ktorý slúži na analýzu záznamov vo formáte CSV vytvorených pomocou Log2Timeline. Taktiež predstavili tri tréningové scenáre na precvičovanie zručností pri analýze časových osí.

Iným príkladom je článok [47], v ktorom autori vyhodnotili existujúce nástroje na analýzu časových osí a identifikovali potrebu spoľahlivého nástroja určeného na analýzu časových osí. Preštudovali projekt s názvom Zeitline, predstavili jeho vlastnosti a nedostatky.

Súčasťou tejto záverečnej práce je vytvorenie nástroja pre analýzu časových osí. Nástroje ako Log2Timeline a Plaso a čiastočne Timesketch sú primárne určené na vytvorenie časových osí. Náš nástroj pracuje s výstupom z nástroja Plaso a predstavuje určitú formu nadstavby. V tomto smere je podobným nástroju Timeline2GUI, keďže vizualizuje výsledky, ktoré analyzuje. Oproti existujúcim nástrojom sa ale odlišuje jednotlivými fázami spracovania časovej osi. Tieto boli prevzaté z oblasti spracovania sieťových bezpečnostných záznamov, a to normalizácia údajov, filtrovanie údajov,

agregácia, korelácia a vytvorenie scenára alebo vizualizácia. Žiaden nám dostupných nástrojov nevyužíva takýto prístup k analýze časových osí.

2.3 Tvorba časových osí

Druhú skupinu podobných prác tvoria práce zamerané na tvorbu časových osí. Autori v rôznych odborných, resp. vedeckých článkoch využívajú rôzne prístupy k vytvoreniu časovej následnosti konkrétnych forenzných artefaktov (napr. ontologický prístup, scenáre).

Chabot a ďalší v článku [48] použili na rekonštrukciu a analýzu časových osí prístup založený na ontológii. Identifikovali sedem kritérií, ktoré musí účinný nástroj na rekonštrukcii spĺňať, aby riešil právne požiadavky, heterogenitu a problémy s objemom údajov. V článku autori predstavili prístup založený na trojvrstvovej ontológii, nazývanej ORD2I, na reprezentáciu akýchkoľvek digitálnych udalostí. Iným príkladom použitia ontológie je článok [49]. Autori v ňom prezentujú iný ontologický prístup k spracovaniu digitálnych dôkazov a zvládnutiu priebehu digitálneho vyšetrovania. Navrhovaný systém s názvom ForensicFlow poskytuje prostriedky na automatickú extrakciu artefaktov z rôznych zdrojov a rekonštrukciu grafov udalostí a artefaktov s cieľom pomôcť forenzným expertom rýchlo a efektívne načrtnúť rozsah incidentu.

V článku [50] sa autori zameriavajú na digitálne forenzné nástroje založené na príkazoch. Ich prístup bol implementovaný na zariadeniach s operačným systémom Windows, Android a iOS.

V inom článku [51] sa autori venovali predstaveniu nového prístupu založeného na štyroch rôznych úrovniach abstrakcie časovej osi udalostí a artefaktov. Sledovali jedinečnú štruktúru, ktorá poskytuje informácie týkajúce sa konkrétnej akcie vykonanej používateľom.

Autori v článku [52] extrahovali rôzne udalosti z operačnej pamäte systému a prezentujú ich v chronologickom poradí. Tiež navrhli a preskúmali niekoľko rôznych scenárov, pričom ukázali, že aktivitu používateľov, ktorá sa deje v určitom čase možno korelovať a tiež zobraziť v časovej osi.

V ďalšom článku [53] autori vytvorili dataset pomocou dynamickej behaviorálnej analýzy 400 rôznych vzoriek. Výsledný a redukovaný dataset následne vizualizovali. Výber typu grafu závisí od rozmerov údajov, typov údajov a tiež od toho, aký druh údajov

vizualizujeme. Vo svojom výskume využili graf časovej osi z D3.js, ktorý zobrazuje grafovo orientované údaje.

V inom článku [54] autori prezentujú ontologický prístup k spracovaniu digitálnych dôkazov a zvládnutiu priebehu digitálneho vyšetrovania. Navrhovaný systém s názvom ForensicFlow poskytuje prostriedky na automatickú extrakciu artefaktov z rôznych zdrojov a rekonštrukciu grafov udalostí a artefaktov s cieľom pomôcť forenzným expertom rýchlo a efektívne načrtnúť rozsah incidentu.

2.4 Analýza časových osí

Tretiu skupinu odborných a výskumných prác, ktoré súvisia s touto záverečnou prácou, sú práce zamerané na analýzu časových osí. Studiawan a kolektív autorov v článku [55] navrhli analýzu sentimentu na automatickú extrakciu zaujímavých udalostí zo záznamov (logov) vo forenznej časovej osi. Použili techniku hlbokého učenia a zakreslili výsledky analýzy sentimentu do forenzných časových osí pomocou nástroja Timesketch.

Autori následne pokračovali vo svojom výskume a v článku [56] a navrhli metódu na identifikáciu anomálií vo forenznej časovej osi. Použili hlboké autoenkódery na vytvorenie základnej línie pre bežné činnosti v súboroch so záznamami (logmi). Tieto anomálne udalosti zakresľujú do forenzných časových osí.

V ďalšom článku [57] sa autori venovali prístupu k určovaniu relevantnosti súborových artefaktov. Tento prístup umožňuje použitie predtým nájdených relevantných súborov na klasifikáciu novoobjavených súborov pri vyšetrovaní. Implementovaná technika generuje skóre relevantnosti pre podobnosť súborov pomocou metadát súborového systému každého artefaktu a súvisiacich udalostí časovej osi. Prezentovaný prístup je overený na základe troch scenárov experimentálneho použitia.

V Tabuľke č. 13 môžeme vidieť zhrnutie vyššie spomenutých podobných prác. Autori vo väčšine prípadov v rámci svojho výskumu využívali vlastný dataset. Atribút výsledok popisuje, čo bolo výstupom predmetného výskumu. Použitá metóda hovorí o prístupoch, ktoré autori zvolili.

	Výsledok	Použitá metóda	Použitý dataset
[48]	Rekonštrukcia a analýza časovej osi	Prístup založený na ontológii	Dataset skonštruovaný z obrazu disku virtuálneho stroja
[50]	Rekonštrukcia časovej osi	Prístup založený na ontológii	Windows, Android a iOS
[51]	Rekonštrukcia časovej osi	Prístup založený na abstrakcii	Obraz disku s NTFS súborovým systémom
[52]	Rekonštrukcia časovej osi	-	Obraz disku virtuálneho stroja s OS Windows 7
[53]	Rekonštrukcia časovej osi	Vizualizácia pomocou grafov	400 vzoriek malvéru/ransomvéru/spywaru
[49]	Rekonštrukcia časovej osi	Prístup založený na ontológii	Obraz disku virtuálneho stroja s OS Windows 7
[55]	Analýza časovej osi	Metóda hlbokého učenia	Casper, Jhuisi, Nssal, Honey
[56]	Analýza časovej osi	Hlboké autoenkódery	Casper, Jhuisi, Nssal, Honey
[57]	Analýza časovej osi	Prístup založený na údajoch	Generované experimentálne údaje

Tabuľka 13 Zhrnutie podobných prác z oblasti analýzy časových osí

2.5 Korelácia forenzných artefaktov v rámci časových osí

Jednou z dôležitých fáz spracovania forenzných artefaktov v rámci časových osí je hľadanie vzťahov medzi nimi, resp. ich reprezentáciami v podobe záznamov. Autori v článku [58] porovnávajú rôzne prístupy ku korelácii a spracovaniu udalostí. Tieto prístupy je možné podľa nášho názoru zovšeobecniť aj na forenzné artefakty, ktoré tieto

udalosti popisujú. Prístupy, ktorým sa venujú autori v predmetnom článku sú porovnané v Tabuľke č. 14 a bližšie sa ich popisu venujeme v nasledujúcich podkapitolách:

- korelácia na základe pravidiel (RBC),
- korelácia na základe prípadov (CBC),
- korelácia na základe konečnostavového stroja (FSM),
- korelácia na základe modelu (MBC),
- pravdepodobnostná korelácia,
- korelácia na základe kódovej knihy.

Technika	Pozitíva	Negatíva
FSM	Jednoduchá, dobrá ako základný model, ľahká na pochopenie.	Príliš jednoduché na praktické použitie, žiadna tolerancia na šum.
RBC	Transparentné správanie, blízke prirodzenému jazyku, modularita.	Časovo náročná údržba, ťažko sa učí zo skúseností.
CBC	Automatické učenie sa zo skúseností, korelácia z minulých skúseností je prirodzená.	Automatické prispôsobenie riešenia a opätovné použitie je zložité.
MBC	Spolieha sa na hlboké znalosti.	Popis správania a štruktúr môže byť zložitý v praxi.
Kódovník	Rýchla, robustná, prispôsobuje sa zmenám.	Manuálny opis správania je zdĺhavý, chýba pojem času.
Pravdepodobnostná korelácia	Dobrý teoretický základ.	Pravdepodobnostná korelácia je NP-ťažká.

Tabuľka 14 Porovnanie metód korelácie podľa [58].

Pri koreláciách na základe pravidiel (RBC) systém neustále používa súbor vopred pripravených pravidiel, ktoré využíva na vyhodnotenie prichádzajúcich upozornení. Z tohto dôvodu korelácia závisí výlučne od hĺbky a schopností súboru pravidiel. Informácie sú reprezentované v troch úrovniach:

- úroveň znalostí - doménové špecifické informácie sú k dispozícii na úrovni znalostnej bázy, ktorá je úložiskom pravidiel,

- úroveň údajov - informácie o danom probléme sú alokované v pracovnej pamäti. Na tejto úrovni sú uložené fakty,
- kontrolná úroveň -odpoveď, ako aplikovať pravidlá zo znalostnej bázy na riešenie daného problému sa nachádza v inferenčnom mechanizme.

V rámci tejto záverečnej práce budeme využívať tento prístup ku korelácií. Nami navrhnutý model (ako ukážme v nasledujúcich kapitolách) bude vychádzať z vopred vytvorených pravidiel (napr. podľa názvu súboru, konkrétneho inodu) alebo zo získaných pravidiel na základe metód formálnej konceptovej analýzy.

Druhým prístupom je korelácia na základne prípadov (CBC). V tomto prístupe sa korelačný systém snaží vyriešiť daný problém pomocou najpodobnejšieho prípadu z knižnice prípadov. Na základe toho zistí vhodné riešenie. Prípad všeobecne pozostáva z konkrétneho problému, jeho riešenia a zvyčajne aj z anotácií ohľadom toho, ako bolo riešenie odvodené. Možné je aj navrhnutie nového riešenia. Po uplatnení nového riešenia problému sa overí výsledok a ak je úspešný, použije sa nové riešenie a uloží sa do knižnice. V prípade, že nové pravidlo neuspeje, hľadá sa lepšie riešenie. Z tohto vyplýva, že tento prístup čerpá a učí sa zo skúseností a dokáže sa prispôsobiť novým a neznámym problémom.

Tretím prístupom je korelácia na základne konečnostavového stroja (FSM). Ako definíciu konečnostavového stroja (FSM) autori uvádzajú päticu, ktorá pozostáva z množiny udalostí , konečnej množiny stavov, počiatočného a koncového stavu FSM a parciálnej prechodovej funkcie. Sekvencia udalostí reprezentuje možné správanie systému. Množina správaní systému je jazyk generovaný FSM. Tento prístup umožňuje efektívne výpočty a taktiež analýzu.

Iným príkladom prístupu je korelácia na základe modelu (MBC). Pri tomto prístupe sa na koreláciu využíva vopred definovaný model reprezentujúci štruktúru a správanie pozorovaného systému. Tento prístup však nenavrhuje podrobnú techniku. Praktická implementácia by mohla využiť model založený na pravidlách. Od RBC by sa však líšil v tom, že tento model je systémový, ktorý berie forenzné artefakty ako dôsledky určitých stavov.

Ďalším príkladom prístupu je pravdepodobnostná korelácia. Tento prístup korelácie vychádza z myšlienky, že v každej praktickej oblasti existuje prvok neistoty. Pravdepodobne je to spôsobené v dôsledku nedostatočnej presnosti spôsobu modelovania reality alebo v dôsledku neúplných údajov. Z tohto dôvodu sa na udalosť vzťahuje index pravdepodobnosti, ktorý je v rozmedzí od nula až po jedna. To je namiesto toho, aby sa tvrdilo, že udalosť je absolútne pravdivá alebo nepravdivá. Korelácia je uskutočnená na základe Bayesovských sietí.

Posledným prístupom je korelácia na základe kódovej knihy. Tento prístup je podobný prístupu založenom na pravidlách. Ide o systém, ktorého cieľom je lokalizácia problémov a metódou je výber vhodnej podmnožiny "symptómových" forenzných artefaktov spojených s týmito problémami. Táto podmnožina symptómových forenzných artefaktov je kódová kniha a pre každý problém sa vytvorí binárny vektor, ktorý označuje, či každý symptóm v kódovej knihe môže byť spôsobený týmto konkrétnym problémom. Forenzné artefakty obsiahnuté v kódovej knihe sa sledujú v reálnom čase. Pri výskyte forenzného artefaktu sa jeho vektor porovná s vektorom určeným pre problém, ktorý sa nachádza v kódovej knihe. Vyberie sa vektor s najmenšou hodnotou Hammingovej vzdialenosti od sledovaného vektora, čím sa identifikuje sledovaný problém.

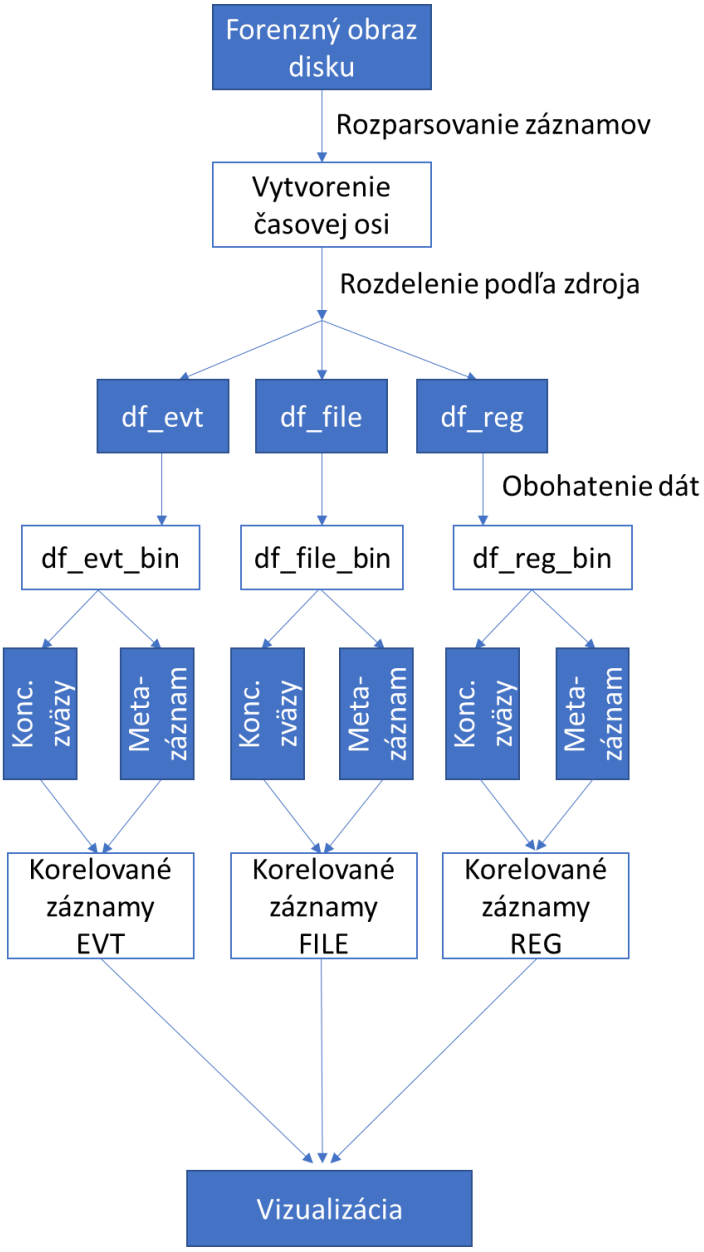
3 Návrh riešenia

V rámci návrhu modelu analýzy časových osí sme sa inšpirovali riešením otázok spojených s analýzou sieťového bezpečnostného povedomia, najmä bližším spracovaním údajov so sieťovej prevádzky. Analýza týchto údajov pozostáva z predspracovania údajov (najmä ich normalizácie a filtrácie), agregácie, korelácie, vytvorenia scenára a vizualizácie [58].

Cieľom systému je spracovanie zozbieraných forenzných artefaktov. V prvej fáze získané dáta rozparsujeme vytvorením časovej osi. Pri spracovaní ich rozdelíme do jednotlivých dátových rámcov podľa zdroja dát. Následne z nich extrahujeme prídavné atribúty a prevedieme do binárneho formátu, ktorý je vhodný na ďalšiu analýzu, v našom prípade sme využili formálnu konceptovú analýzu. cez ich spracovanie, vytvorenie časovej osi, vizualizáciu vzťahov medzi atribútmi forenzných artefaktov. Všetky fázy navrhnutého riešenia, ktoré sú zobrazené na Obrázku č. 5 sú rozobrané v ďalších častiach tejto práce. Pri každej fáze sa venujeme jej základnému popisu, vstupným a výstupným údajom, ako aj nášmu návrhu riešenia danej fázy a implementačným poznámkam.

Vstupom pre prvú fázu je forenzný obraz disku vo formáte E01(Encase Image File Format). Tento formát uchováva zálohu rôznych typov získaných digitálnych stôp, ktoré zahŕňajú zobrazenie disku, uloženie logických súborov a pod. Pri použití na vytvorenie zálohy údajov dostupných na pevnom disku sa vytvorí fyzický bitový tok údajov. V tejto práci sa budeme venovať analýze forenzného obrazu disku servera, doménového kontroléru, z prípadu The Stolen Szechuan Sauce, ktorý je popísaný v nasledujúcej podkapitole. Následne sa z forenzného obrazu vytvorí prostredníctvom nástroja Plaso časová os. Túto časovú os v ďalšej fáze rozdelíme do troch samostatných dátových rámcov:

- **df_evt** - dátový rámec pre forenzné artefakty, ktorých zdrojom je záznam udalostí,
- **df_file** - dátový rámec pre forenzné artefakty, ktorých zdrojom je súborový system NTFS,
- **df_reg** - dátový rámec pre forenzné artefakty, ktorých zdrojom je register operačného systému Windows.



Obrázok 5 Fázy navrhovaného riešenia.

V nasledujúcom kroku sa zo základných polí každého záznamu vytiahnu rôzne kategorické atribúty (napr. názov súboru, cesta súboru, inode, názov procesu) a binárne hodnoty (napr. či daný súbor bol zmazaný, umiestnenie súboru v domovskom adresári používateľa). Súčasťou tohto kroku je aj prevod kategorických premenných na binárne premenné.

Po tomto kroku dochádza k dvom paralelným vetvám. V jednej vetve sa binárne atribúty pre každý dátový rámec použijú ako vstup pre metódu formálnej konceptuálnej analýzy. Výsledkom je zoznam rôznych konceptov a ich objektov (záznamov). Bližší

popis použitia formálnej konceptovej analýzy je uvedený v štvrtej kapitole. V druhej vetve dochádza k agregácii binárnych záznamov na základe konkrétnych pravidiel, ktoré v sebe zohľadňujú určité časové okno a sú špecifické pre každý typ forenzných artefaktov (df_evt, df_file, df_reg). Bližší popis agregácie je uvedený v kapitole 3.3.

Po týchto paralelných krokoch dochádza k korelácii vytvorených agregovaných záznamov (metazáznamov). Na ich koreláciu sa používajú vopred dané pravidlá (napr. názov súboru, inode) alebo sa využívajú pravidlá vytvorené pomocou formálnej konceptovej analýzy. Posledným krokom je vytvorenie scenára a jeho následná vizualizácia.

Vstupom pre prvú fázu je forenzný obraz disku vo formáte E01(Encase Image File Format). Tento formát uchováva zálohu rôznych typov získaných digitálnych dôkazov, ktoré zahŕňajú zobrazenie disku, uloženie logických súborov a pod. Pri použití na vytvorenie zálohy údajov dostupných na pevnom disku sa vytvorí fyzický bitový tok údajov. V tejto práci sa budeme venovať analýze forenzného obrazu disku servera z prípadu The Stolen Szechuan Sauce, ktorý je popísaný v nasledujúcej podkapitole.

3.1 Popis prípadu

Aby bolo možné vytvoriť model pre analýzu časových osí operačného systému Windows sme nevyhnutne potrebovali digitálne stopy, ktoré by bolo možné použiť v našom prípade. Prvou možnosťou bolo vytvoriť tieto digitálne stopy samostatne. Výhodou tohto prístupu by bolo vytvorenie vlastného scenára. Naopak, nevýhodou nemožnosť porovnania výsledkov s inými odborníkmi, resp. výskumnými skupinami. Z tohto dôvodu sme zvolili opačný prístup a vybrali už existujúci prípad vrátane dostupných digitálnych stôp. Inšpiráciou v tomto smere nám boli aktivity DS4N6 [59]. Táto skupina sa venuje použitiu dátovej analýzy pri analýzy forenzných artefaktov.

Pre samotný účel tohto výskumu bol zvolený dataset „Case 001 - The Stolen Szechuan Sauce“. Ide o je jeden z dostupných tréningových datasetov z portálu DFIRmadness Madness [60], ktorý slúži na výučbu digitálnej forenznej analýzy, reakcie na bezpečnostné incidenty a taktiež vyhľadávania bezpečnostných hrozieb. Vzorový prípad sa venuje analýze neoprávneného vniknutia do počítačovej siete spoločnosti CITADEL, z ktorej mala uniknúť receptúra práve unikátnej „szechuan sauce“. Recept

unikol na internet s cieľom poškodiť podnik a odoprieť konkurenčnú výhodu. Jediné miesto, kde bol recept uložený, bol v osobnom počítači zakladateľa omáčky.

Úlohou je identifikovať, či boli v systéme nainštalované škodlivé aplikácie, vrátane miesta a času inštalácie softvéru. V prípade ide aj o určenie toho, či v rámci operačného systému neboli vytvorené, upravené alebo vymazané nejaké informácie, a taktiež či nedošlo k úniku dát.

V prípade sa analyzujú forenzné artefakty zo servera, doménového kontroléra, spoločnosti (DC) a taktiež z klientskeho zariadenia (DESKTOP), zachyteného sieťové spojenia a zaistených operačných pamätí. K dispozícii sú nasledujúce konkrétne digitálne stopy:

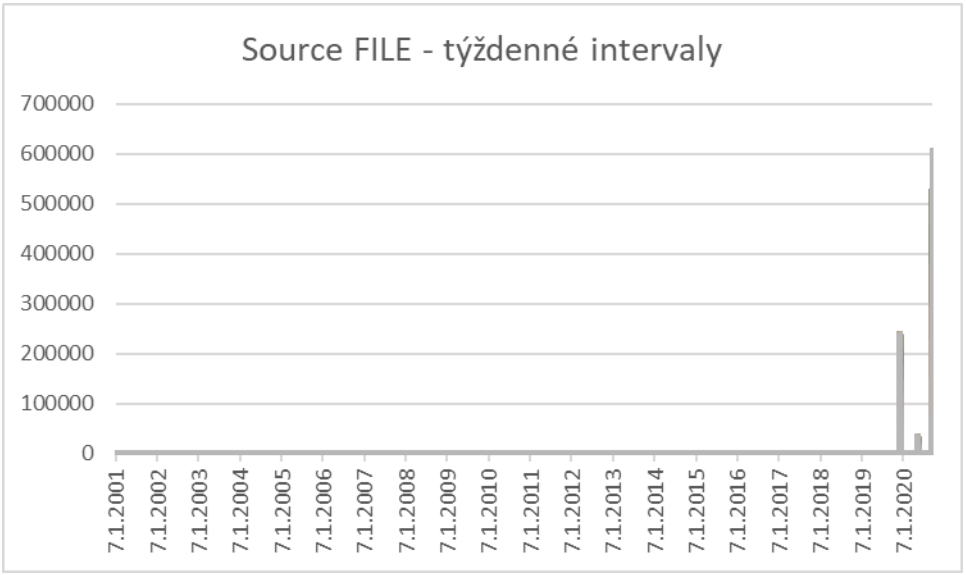
- case001-pcap.zip,
- DC01-autorunsc.zip,
- **DC01-E01.zip**,
- DC01-memory.zip,
- DC01-pagefile.zip,
- DC01-ProtectedFiles.zip,
- DESKTOP-E01.zip,
- DESKTOP-SDN1RPT-autrunsc.zip,
- DESKTOP-SDN1RPT-memory.zip,
- Desktop-SDN1RPT-pagefile.zip,
- DESKTOP-SDN1RPT-Protected Files.zip.

Pri tomto forenznom tréningu nechýbajú ani vodiace otázky, ktoré majú riešiteľov priviesť na správnu stopu, keďže na analýzu majú k dispozícii veľké množstvo dát. Nosnou informáciou na začiatku analýzy je zistiť operačné systémy analyzovaných zariadení. Identifikovaný operačný systém servera bol Windows a na klientskej stanici to bol Windows 10. Taktiež bol identifikovaný lokálny čas servera - Mountain Standard Time (UTC -6). Jednou z hlavných úloh bolo zistiť, či došlo k úniku údajov zo zariadenia a ako k nemu došlo. Na základe analýzy pcap súboru Case001.pcap v nástroji Wireshark bolo zistené, že 19.09.2020 došlo k skenovaniu IP adries spoločnosti CITADEL. Následne začal útok hrubou na službu vzdialenej plochy (RDP) z IP adresy

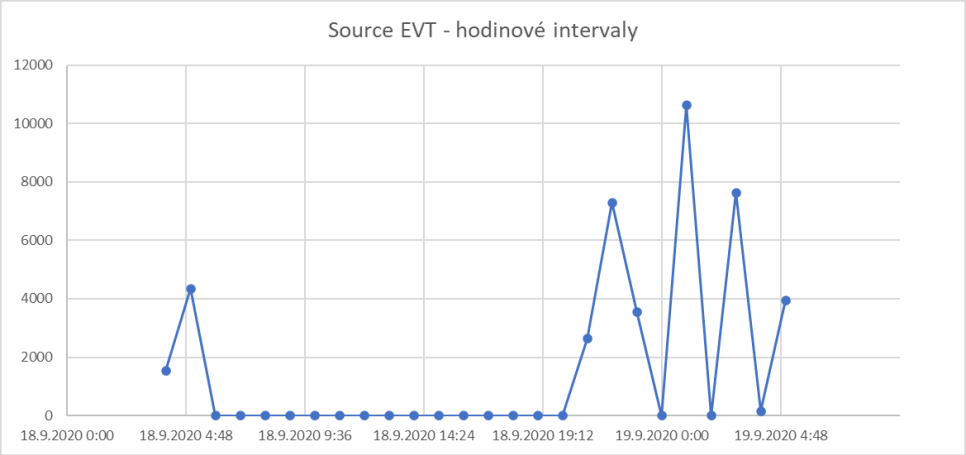
194.61.24.102. Niekoľko minút na to sa útočník úspešne prihlásil pod účtom Citadel/Administrator. Tieto prihlásenia sú zaznamenané v EVTX logoch servera - /img_20200918_037_Cdrive.E01/vol_vol3/Windows/System32/winevt/Logs/ Security.evtx. Keď sa útočník dostal na doménový kontrolér DC, otvorili druhú reláciu RDP z doménového kontroléra na klientske zariadenie s použitím rovnakých poverení. Prostredníctvom internetového prehliadača Internet Explorer bol stiahnutý na server a následne aj na klienta, škodlivý softvér s názvom coreupdater.exe. Záznam o stiahnutí a nainštalovaní softvéru (aj ako služba aj v registroch systému Windows) sa nachádza v sieťovej prevádzke – súbore Case001.pcap a taktiež obraze disku servera aj klienta, v kľúčoch registrov, ktoré je možné analyzovať pomocou nástrojov Registry Explorer alebo aj Autopsy. Škodlivý softvér bol umiestnený na disku v časti C:\Windows\System32\coreupdate.exe. Pôvodne sa však nachádzal v stiahnutých súboroch. Čo sa týka schopností škodlivého softvéru, išlo o metasploit, ktorý je schopný napr. migrácie procesov, krádeže poverení, zaznamenávania kľúčov, či obrazovky.

Na základe analýzy je možné dôjsť k záveru, že ide o meterpreter [61, 62]. Na základe MFT, žurnálu, logov a odkazov (lnk súborov) je vidieť, že došlo vytváraniu, modifikácii aj vymazaniu pôvodných súborov. Na základe analýzy odkazov vieme povedať, že k súboru SzechuanSauce.lnk bolo pristúpené 19.9.2020 o 03:32:21 (UTC -6). Analýzou žurnálu zas možno s istotou povedať, že došlo aj k úniku údajov. Útočník exfiltroval dáta v súbore Secret.zip, kde sa medzi množstvom súborov nachádzal aj súbor s tajnou receptúrou „szechuan sauce“.

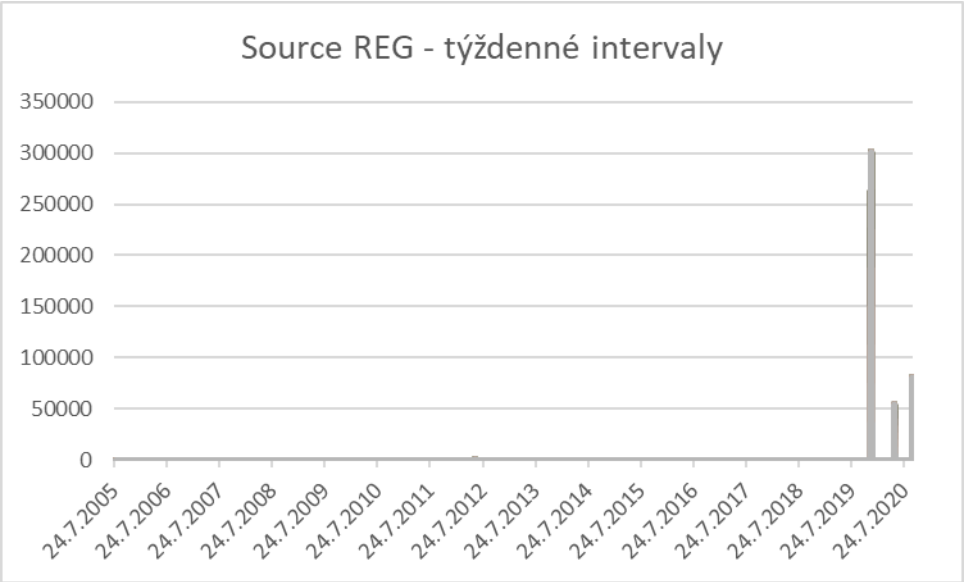
Na priblíženie aktivity na zariadení klienta pri dátach z rôznych zdrojov sme urobili prehľad počtu ich záznamov. Obrázky č. 6, 7 a 8 znázorňujú počet záznamov zo zdroja FILE, EVT, resp. REG doménového kontroléra DC agregovaných v týždenných intervaloch.



Obrázok 6 Vizualizácia počtu záznamov zo zdroja FILE v týždenných intervaloch



Obrázok 7 Vizualizácia počtu záznamov zo zdroja EVT v hodinových intervaloch



Obrázok 8 Vizualizácia počtu záznamov zo zdroja REG v týždenných intervaloch

3.2 Predspracovanie údajov

Predspracovanie údajov predstavuje dôležitú fázu pri spracovaní digitálnych stôp. V tejto fáze dochádza k výberu a úprave artefaktov a ich atribútov, ktoré budú určené na ďalšie spracovanie.

3.2.1 Fázy predspracovania údajov vo forenznej analýze

Fáza predspracovania údajov v oblasti digitálnej forenznej analýzy zabezpečuje nasledujúce aktivity [63]:

- výber atribútov,
- elimináciu atribútov,
- normalizáciu atribútov a
- reprezentáciu atribútov.

Výber atribútov predstavuje proces, ktorý umožňuje výber rôznych podmnožín exkluzívnych znakov zo súboru zozbieraných živých údajov. To umožňuje získať dôležité alebo selektívne znaky, ktoré umožňujú správnu klasifikáciu. Použitie príznakového inžinierstva tiež umožňuje identifikovať množstvo príznakov, ktoré môžu pomôcť pri klasifikácii.

Eliminácia atribútov zdôrazňuje, že daný súbor forenzných údajov možno použiť na posúdenie exkluzívnych prítomných atribútov, ktoré sú užitočné alebo relevantné pre ďalšie použitie. Na to, aby sa atribút eliminoval, prispieva viacero faktorov, ako napríklad existencia dynamická konfigurácia a rekonfigurácia zariadení. Zmeny v technických aspektoch môžu taktiež časom brániť identifikácii atribútov, ktoré je potrebné eliminovať. V tomto procese treba dbať na opatrné odstraňovanie atribútov, pretože sa môžu odstrániť aj smerodajné prvky, ktoré sú v procese užitočné. Eliminácia atribútov môže viesť aj k identifikovaniu rôznych stratégií forenzného profilovania útočníkov. Je to spôsobené tým, že zmeny správania sú spoločným aspektom.

Normalizácia atribútov predstavuje dôvod, prečo je normalizácia príznakov/vlastností údajov počas živého forenzného vyšetrovania dôležitá je, že údaje získané z rôznych zdrojov zvyčajne disponujú rôznymi vlastnosťami. Aj použitie rôznych mier vzdialeností atribútov, ako sú euklidovské vzdialenosti, manhattanské vzdialenosti atď. [64] môže týmto extrahovaným údajom priradiť rôzne váhy. Na základe toho sa normalizácia príznakov stáva dôležitou, pretože môže vyvážiť rozsah týchto atribútov na základe výpočtu podobnosti. Vo všeobecnosti tento postup zahŕňa štatistickú transformáciu zložiek atribútov tak, aby hodnoty dokázali poskytnúť správne alebo lepšie odhady atribútov.

Reprezentácia atribútov predstavuje jednu zo základných charakteristík dynamického prostredia. Súčasne predstavuje integráciu viacerých zdrojov informácií do centralizovaného procesu. Keďže sa v danom spektre analýzy agreguje niekoľko prvkov z viacerých zdrojov, je potrebné definovať jedinečný formát pre prípady v každom vzniknutom vektore. Definovať sa môže aj dátový formát, v ktorom by sa zohľadnila heterogenita údajov.

3.2.2 Vytvorenie časovej osi

Za fázu predspracovania v našom riešení považujeme vytvorenie časovej osi a výber zdrojov dát, ktorých záznamy budeme analyzovať ďalej. Taktiež výber a vytvorenie možných nových atribútov na obohatenie analýz a taktiež ich elimináciu. Na vytvorenie časovej osi využívame už spomínaný nástroj Log2timeline. V tejto časti popíšeme jeho použitie na vybranom forenznom obraze disku.

Nástroj príkazového riadku Log2timeline(plaso) má veľa možností, čo sa použitia týka. Všeobecný príklad použitia podľa dokumentácie [65]:

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

Pri aplikácia tohto nástroja na nami zvolený obraz disku sme zvolili možnosti, ktoré sú konkrétne popísané nižšie.

CITADEL-DC01 Obraz disku servera (E01)

log2timeline.py --parsers=win7_slow --status_view window --storage_file Server_timeline_all.dump 20200918_0347_CDrive.E01 --partitions "all"

Keďže ide o obraz disku zariadenia s operačným systémom Windows, zvolili sme parser `win7_slow`, ktorý v sebe zahŕňa ďalšie tri parsery, a to `win_gen, webhist a win7`. Pre získanie potrebných dát je táto voľba postačujúca, keďže zahrňujúce parsery v sebe opäť zahŕňajú ďalšie. Napr. win_gen v sebe ďalej `zahŕňa 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 o dostupných parseroch (pluginoch) je spomenuté v dokumentácii nástroja log2timeline plaso [66]. Ďalej nasleduje voľba výstupného súboru pre časovú os (Server_timeline_all.dump) a taktiež názov zdrojového súboru, v našom prípade je to `20200918_0347_CDrive.E01.`

Na získaný výstup *Server_timeline_all.dump* ďalej aplikujeme nástroj psort.py, ktorý ho prevedie do čitateľnej podoby.

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

Pri tomto nástroji je potrebné zvoliť výstupný formát (`-o`), v našom prípade je to vyššie spomínaný l2tcsv formát, pretože nám pre ďalšie fázy vyhovuje tabuľkový formát dát. Ak chcete zobraziť zoznam všetkých dostupných výstupných formátov, je možné použiť príkaz -o list. Existuje ďalších 11 formátov, ktoré nástroj poskytuje[1]. Ďalej je možné zvoliť výstupnú časovú zónu. Z tejto časti máme na výstupe súbor s čitateľnou časovou osou s názvom *`Server_timeline_all.csv`*.

Po vytvorení časovej osi sme si výstupný súbor vo formáte l2tcsv nahrali v online platforme Google Colab [67]. Je to produkt spoločnosti Google Research. Umožňuje prístup k bezplatným výpočtovým zdrojom a obzvlášť vhodný je na využitie pri strojovom učení a analýze veľkého množstva údajov. Použitý programovací jazyk v tomto prostredí je Python vo verzii 3. Načítanie súboru v tomto prostredí je zobrazené na Obrázku č. 9.

```
df = pd.read_csv (r'/content/drive/MyDrive/Server_timeline_all.csv ', error_bad_lines=False, sep= ',')
print(df.shape)
```

Obrázok 9 Načítanie súboru s časovou osou v prostredí Google Colab ako dataframe

Počet záznamov v dataframe *(print(df.shape))* pred rozdelením bol 1 263 787. Ich ukážka je na Obrázku č. 10.

```
[ ] df.sample(5)
```

	date	time	timezone	MACB	source	sourcetype	type	user	host	short	desc	version	filename	inode	notes	format	extra
	3/21/2014	18:06:10	UTC	MA..	FILE	NTFS file stat	Content Modification Time; Last Access Time	-	-	\$MFT 54829-1 $STANDARD_INFORMATION	NTFS:\$MFT File reference: 54829-1 Attribute n...	2	NTFS:\$MFT	0	-	mft	attribute_type: 16; file_system_type: NTFS; is...
	6/18/2013	15:06:35	UTC	M...	FILE	NTFS file stat	Content Modification Time	-	-	\$MFT 45107-1 $STANDARD_INFORMATION	NTFS:\$MFT File reference: 45107-1 Attribute n...	2	NTFS:\$MFT	0	-	mft	attribute_type: 16; file_system_type: NTFS; is...
	3/21/2014	18:08:23	UTC	..B	FILE	File stat	Creation Time	-	-	\Windows\WinSxS\msil_npsui.resources_31bf3856a...	NTFS:\Windows\WinSxS\msil_npsui.resources_31bf...	2	NTFS:\Windows\WinSxS\msil_npsui.resources_31bf...	72079	-	filestat	file_size: 233168; file_system_type: NTFS; is_...
	9/17/2020	17:51:23	UTC	MACB	FILE	NTFS file stat	Content Modification Time; Creation Time; Last...	-	-	\$MFT 38734-2 $FILE_NAME	NTFS:\$MFT File reference: 38734-2 Attribute n...	2	NTFS:\$MFT	0	-	mft	attribute_type: 48; file_attribute_flags: 32;...
	9/17/2020	16:47:58	UTC	MACB	FILE	NTFS file stat	Content Modification Time; Creation Time; Last...	-	-	\$MFT 13469-1 $FILE_NAME	NTFS:\$MFT File reference: 13469-1 Attribute n...	2	NTFS:\$MFT	0	-	mft	attribute_type: 48; file_attribute_flags: 2684...

Obrázok 10 Ukážka dát po načítaní v Colab notebook

Pri čistení dát sme odstránili tie, ktoré mali ako dátum uvedenú hodnotu '00/00/0000'. Táto operácia je zobrazená na Obrázku č. 11. Ide napr. o takéto záznamy:

[1] https://plaso.readthedocs.io/en/latest/sources/user/Output-and-formatting.html

- 00/00/0000 --:--:-- - .... LOG SystemInstallation Time Windows Server 2012 R2 Standard Evaluation 6.3...Windows Server 2012 R2 Standard Evaluation 6.3...

```
df.drop(df.index[df['date'] == '00/00/0000'], inplace=True)
```

Obrázok 11 Úprava dát, odstránenie nepotrebných záznamov

V ďalšej fáze sme rozdelili dáta na 11 samostatných dataframe-ov podľa zdroja, čo je zobrazené na Obrázku č. 12.

```
result = df.copy()
columnNames = ["date", "time", "timezone", "MACB", "source", "sourcetype", "type", "user", "host", "short", "desc", "version", "filename", "inode", "notes",  "format", "extra"]
df_amcache = pd.DataFrame(columns = columnNames)
df_amcacheprogram = pd.DataFrame(columns = columnNames)
df_evt = pd.DataFrame(columns = columnNames)
df_file = pd.DataFrame(columns = columnNames)
df_lnk = pd.DataFrame(columns = columnNames)
df_log = pd.DataFrame(columns = columnNames)
df_olecf = pd.DataFrame(columns = columnNames)
df_pe = pd.DataFrame(columns = columnNames)
df_recbin = pd.DataFrame(columns = columnNames)
df_reg = pd.DataFrame(columns = columnNames)
df_webhist = pd.DataFrame(columns = columnNames)

df_amcache = result.loc[result['source'] == 'AMCACHE']
df_amcacheprogram = result.loc[result['source'] == 'AMCACHEPROGRAM']
df_evt = result.loc[result['source'] == 'EVT']
df_file = result.loc[result['source'] == 'FILE']
df_lnk = result.loc[result['source'] == 'LNK']
df_log = result.loc[result['source'] == 'LOG']
df_olecf = result.loc[result['source'] == 'OLECF']
df_pe = result.loc[result['source'] == 'PE']
df_recbin = result.loc[result['source'] == 'RECBIN']
df_reg = result.loc[result['source'] == 'REG']
df_webhist = result.loc[result['source'] == 'WEBHIST']
```

Obrázok 12 Rozdelenie dataframe-u na 11 skupín podľa zdroja dát

Novovzniknuté dátové rámce podľa zdroja majú počty záznamov uvedené v Tabuľke č. 15.

Zdroj	Počet záznamov
AMCACHE	136
AMCACHEPROGRAM	3
EVT	86 180
FILE	843 863
LNK	45
LOG	194
OLECF	253
PE	181 115
RECBIN	1
REG	307 315
WEBHIST	75
Spolu	1 419 180

Tabuľka 15 Zobrazenie počtu záznamov po rozdelení dát podľa zdrojov

3.2.3 Ontologický prístup

Vo všeobecnosti bola ontológia zavedená na vysvetlenie povahy, základných atribútov a asociácií medzi všetkými bytosťami už Aristotelom. Založená je na faktoch a svojou nezávislou povahou ukazuje pochopenie, perspektívu a poznanie sveta. Pojem ontológia bol však zavedený aj do informačných a počítačových oblastí. Je vhodným prístupom na definovanie nových terminológií v akejkoľvek konkrétnej doméne [68].

Skúmanie digitálnych zariadení na získanie zmysluplných informácií je veľmi časovo náročný proces z dôvodu obrovského množstva údajov, ich rôznorodosti a rýchlych technologických inovácií. Forenzné nástroje, ktoré sú vyvinuté, generujú neštruktúrované časové osi z rôznych zdrojov údajov. Ak sú neštruktúrované, ťažko sa interpretujú najmä z dôvodu rozmanitosti sémantiky. Forenzní analytici častokrát len ťažko porozumejú novým terminológiám a získanie dôkazov je taktiež v tomto smere sťažené. Na to, aby sme uľahčili tento proces je potrebný ontologický prístup, ktorý obsahuje správne a spoľahlivé reprezentácie štruktúry dát.

Nie je možné vyvinúť ontológiu, ktorá by pokryla všetky terminológie, avšak existujúce ontológie sa môžu znova použiť na vývoj nových ontológií. Podľa [69] pozostáva vývoj konkrétnej ontológie z nasledujúcich hlavných krokov:

- definovanie tried,

- usporiadanie tried do taxonomickej hierarchie,
- opísanie slotov a hodnôt, a
- naplnenie hodnôt do slotov.

3.2.4 Ontologický prístup v našom riešení

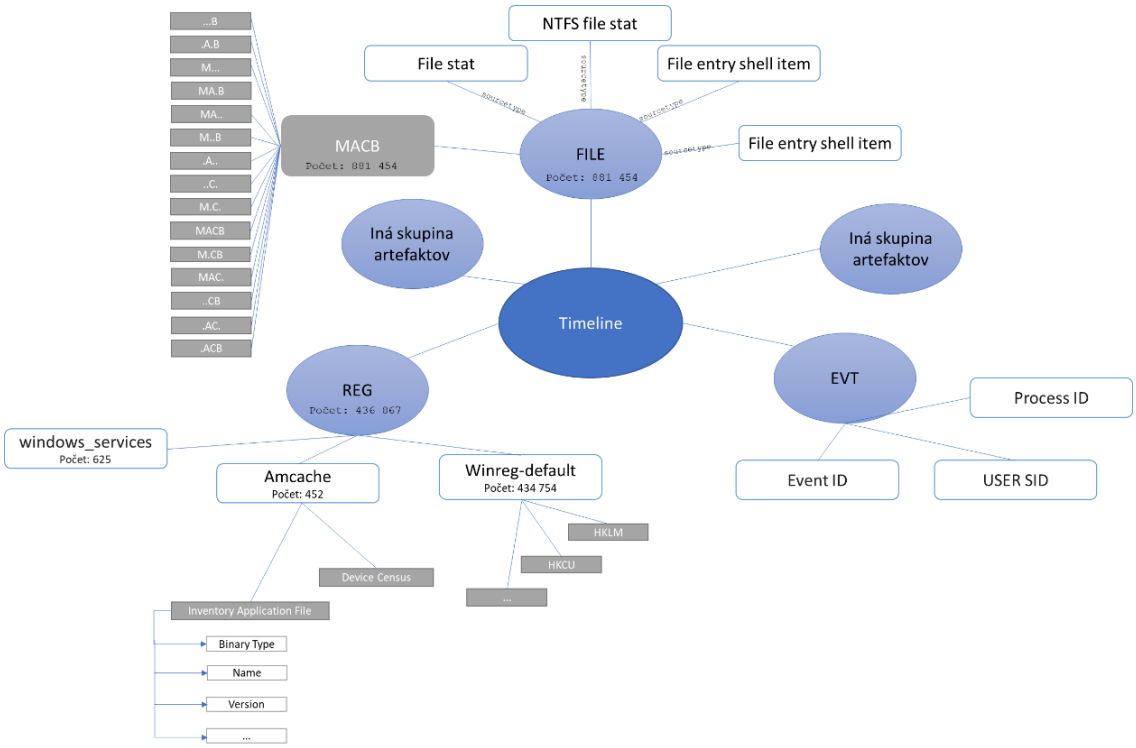
Ontologický prístup, ktorý sme sa zvolili závisí od vytvorenej časovej osi, ktorá je v práci najdôležitejšou fázou. Na základe dát, ktoré sme prostredníctvom nej získali, vieme lepšie určiť dátové vzťahy.

Hierarchicky sú ďalej jednotlivé skupiny artefaktov (zdroje dát), ktorých je v rámci časovej osi 11. V načrtnutom ontologickom riešení sú zobrazené tri hlavné skupiny artefaktov, a to FILE, EVT a REG. Vybrané boli na základe najvyššieho počtu záznamov. Dokopy tieto tri skupiny dávajú 87% všetkých záznamov.

Každý z artefaktov má špecifické atribúty, ktoré sme rozoberali v predchádzajúcej kapitole. Na základe významnosti sme vybrali skupiny atribútov, ktoré sú vhodné pri ďalšom kreovaní vzťahov medzi záznamami jednotlivých artefaktov.

V prípade **záznamu typu FILE** je významným atribútom sourcetype, pri ktorom sú zobrazené špecifické nadobudnuté hodnoty. Ďalej sú to rôzne kombinácie hodnôt atribútu MACB, ktorých je 15. Všetky atribúty záznamov typu FILE sú uvedené v Prílohe A.

Pri **záznamu typu REG** je veľmi významným atribút sourcetype, ktorého atribúty taktiež nadobúdajú jedinečné hodnoty. Na Obrázku č. 13 sú znázornené tri najpočetnejšie, a to windows_services, amcache a winreg-default.



Obrázok 13 Náčrt ontologického prístupu

V prípade **záznamu typu EVT** sú obzvlášť dôležité polia z atribútu extra, a to Event ID, user SID alebo Process ID.

3.2.5 Analýza záznamov

Na vytvorenie vlastného ontologického prístupu bolo potrebné najskôr dôkladne zanalyzovať dáta, ktoré sme v rámci riešenej prípadovej štúdie mali k dispozícii. V tejto fáze sme analyzovali dáta z obrazu disku, ktoré sú vo formáte l2tcsv so 17 odlišnými poliami, ktoré poskytujú podrobné informácie týkajúce sa rôznych činností vykonávaných používateľom na zariadení vo forme časovej osi udalostí. Na ďalšiu analýzu sme si vybrali dátové rámce zo zdrojov FILE, EVT a REG, pri ktorých sme analyzovali hodnoty jednotlivých atribútov.

3.2.6 Analýza záznamov typu FILE

Atribút s časovými pečiatkami MACB nadobúdal 15 rôznych hodnôt, a to:

'...B', '.A.B', 'M...', 'MA.B', 'MA..', 'M..B', '.A..', '..C.', 'M.C.', 'MACB', 'M.CB', 'MAC.', '..CB', '.AC.', '.ACB'.

Z analýzy tohto atribútu vyplýva, že neexistuje záznam v dátach zdroja FILE, ktorý by nemal zaznamenanú aspoň jednu z časových pečiatok M(modifikácia obsahu súboru), A(posledné pristúpenie k súboru), C(modifikácia MFT položky) alebo B(vytvorenie súboru). Keďže ide o kategorickú premennú, v ďalšej fáze bola rozdelená na štyri samostatné binárne premenné M,A,C, B, čo je zobrazené na Obrázku č. 14. Hodnoty atribútu Type, čo je slovný popis atribútu MACB nadobúdali rovnaké hodnoty, avšak boli slovne popísané. Z tohto dôvodu môžeme atribút Type vynechať pri ďalšej analýze dát zdroja FILE.

```
for item in df_file.MACB:
    list_of_letters = list(item)
    new_columns['M1'].append(list_of_letters[0])
    new_columns['A1'].append(list_of_letters[1])
    new_columns['C1'].append(list_of_letters[2])
    new_columns['B1'].append(list_of_letters[3])

df_file['M1'] = new_columns['M1']
df_file['A1'] = new_columns['A1']
df_file['C1'] = new_columns['C1']
df_file['B1'] = new_columns['B1']

df_file['M']=["1" if (x=="M") else "0" for x in df_file.M1]
df_file['A']=["1" if (x=="A") else "0" for x in df_file.A1]
df_file['C']=["1" if (x=="C") else "0" for x in df_file.C1]
df_file['B']=["1" if (x=="B") else "0" for x in df_file.B1]

df_file = df_file.drop(['M1', 'A1', 'B1', 'C1', 'MACB'], axis = 1)
```

Obrázok 14 Úprava a rozdelenie atribútu MACB

Hodnoty, ktoré sa vyskytovali v rámci **atribútu Sourcetype,** boli štyri rôzne, a to:

- 'File stat',
- 'NTFS file stat',
- 'File entry shell item',
- 'NTFS USN change'.

Tento atribút bol rozdelený na 4 samostatné binárne atribúty. Spôsob rozdelenie je zobrazený na Obrázku č. 15.

```
df_file['file_stat']=["1" if (x == 'File stat') else "0" for x in df_file.sourcetype]
df_file['NTFS_file_stat']=["1" if (x=='NTFS file stat') else "0" for x in df_file.sourcetype]
df_file['file_entry_shell_item']=["1" if (x=='File entry shell item') else "0" for x in df_file.sourcetype]
df_file['NTFS_USN_change']=["1" if (x=='NTFS USN change') else "0" for x in df_file.sourcetype]
```

Obrázok 15 Úprava atribútu Sourcetype

Hodnoty **atribútu User** pri záznamoch zdroja FILE neboli zaznamenané, z tohto dôvodu bol taktiež vynechaný z ďalšej analýzy dát zdroja FILE. Rovnaký dôvod platil pri atribúte **Host a Note**. **Atribút Version** bol vynechaný z dôvodu nemeniacej sa hodnoty. Keďže je **atribút Short** skrátenou verziou **atribútu Desc**, z ďalšej analýzy bol vynechaný. Z Desc (description - popis) atribútu boli vyextrahované atribúty názov súboru (name), typ – súbor (filef), priečinok (directory) alebo odkaz (link). Ukážka zdrojového kódu extrakcie typu súboru z atribútu Desc sa nachádza na Obrázku č. 16.

```
for item in df_file.desc:
    type_field = re.compile(r'Type:\s*(.*)')
    typef = type_field.search(item)
    if typef:
      final_type = typef.group(0).split(' ')[1:2]
      new_columns['type'].append(final_type)
    else:
      new_columns['type'].append('None')

df_file['typef'] = new_columns['type']

df_file['filef']=["1" if (x[0] == "file") else "0" for x in df_file.typef]
df_file['directory']=["1" if (x[0]=="directory") else "0" for x in df_file.typef]
df_file['link']=["1" if (x[0]=="link") else "0" for x in df_file.typef]
```

Obrázok 16 Ukážka extrakcie typu z atribútu Desc

Atribút Filename je taktiež jedinečným zdrojom údajov, z ktorých sme vyextrahovali samostatné atribúty ohľadom umiestnenia súboru (dir_appdata, dir_win, dir_user, dir_other) a atribúty ohľadom prípony súboru - **file_executable** (súbory s príponami exe, msi, cmo a bud), **file_graphic** (súbory s príponami png, jpg, psd a jpeg), **file_document** (súbory s príponami doc, docx, docm, ppt, pptx, pps, ppsx, txt, xls a xlsx), **file_ps** (súbory s príponami ps) a **file_other** (súbory s inými príponami, ako boli vymenované doteraz) . Vzorová ukážka hodnôt atribútu Filename:

- 'NTFS:\\Windows\\SoftwareDistribution\\SLS\\522D76A4-93E1-47F8-B8CE-07C937AD1A1E\\sls.cab'

- 'NTFS:\\Users\\ricksanchez\\AppData\\Local\\Packages\\Microsoft.Windows.ContentDeliveryManager_cw5n1h2txyewy\\LocalState\\TargetedContentCache\\v3\\338388\\09810508a3804cdcadef139476f1b901_1'

Atribút Inode bol ponechaný ako atribút, na základe ktorého je možné korelovať záznamy v ďalšej fáze, z dôvodu, že obsahuje jedinečné kategorické hodnoty.

Atribút Format nadobúdal šesť rôznych hodnôt, a to: 'filestat', 'mft', 'lnk/shell_items','olecf/olecf_automatic_destinations/lnk/shell_items','winreg/bagmru/shell_items' a usnjrnl'. Ide teda o kategorickú premennú, ktorú sme rozdelili na ďalších 6 binárnych hodnôt.

Podobne ako Desc, aj **atribút Extra** je zdrojom jedinečných údajov, ktoré môžu byť zdrojom pre vznik ďalších atribútov. Vyextrahovali sme z neho **atribút is_allocated,** ktorý hovorí o prítomnosti resp. vymazaní súboru z disku. Ďalej boli ako samostatné atribúty vyextrahované **haš súboru** (sha_256) a **veľkosť súboru** (file_size), ktorú sme rozdelili do štyroch kvartilov (size_Q1, size_Q2, size_Q3 a size_Q4). Súbory s uvedenou nulovou veľkosťou nadobudli **atribút size_none**. Hranice medzi kvartilmi boli vypočítané na 608.0, 3898.0 a 29779.0 bytov. Ukážka zdrojového kódu, v ktorom sa hranice počítajú sa nachádza na Obrázku č. 17. Najväčší súbor mal 1207959552 bytov. Príklad vzorových dát:

- 'file_size: 24344; file_system_type: NTFS; is_allocated: True; sha256_hash: 4156412f0ed20b33707572e12603468fd1844c89f66aaa9509f55b2dce540c7'
- 'file_attribute_flags: 32; update_reason_flags: 2147483649; update_sequence_number: 27207688; update_source_flags: 0'

```
nenulove = []
for item in df_file.file_size:
    if item != 0 or item == None:
      nenulove.append(item)

nenulove.sort()

Q1 = np.quantile(nenulove, .25)
Q2 = np.quantile(nenulove, .50)
Q3 = np.quantile(nenulove, .75)

for item in df_file.file_size:
    if int(item) == 0:
      new_columns['size_none'].append('1')
    else:
      new_columns['size_none'].append('0')

    if  int(item) > 0 and int(item) <= Q1:
      new_columns['size_Q1'].append('1')
    else:
      new_columns['size_Q1'].append('0')

    if int(item) > Q1 and int(item) <= Q2:
      new_columns['size_Q2'].append('1')
    else:
       new_columns['size_Q2'].append('0')

    if int(item) > Q2 and int(item) <= Q3:
      new_columns['size_Q3'].append('1')
    else:
      new_columns['size_Q3'].append('0')

    if int(item) > Q3:
      new_columns['size_Q4'].append('1')
    else:
      new_columns['size_Q4'].append('0')
```

Obrázok 17 Zistenie hraníc kvartilov veľkostí súborov a vytvorenie nových atribútov size_none, size_Q1, size_Q2, size_Q3 a size_Q4

Kompletný zoznam atribútov zdroja FILE je uvedený v prílohe B. V prílohe E sa nachádza taktiež zdrojový kód použitý pri úprave dát.

3.2.7 Analýza záznamov typu EVT

Záznamy typu EVT sme analyzovali taktiež na základe hodnôt atribútov, ktoré sa nachádzali v riešenej prípadovej štúdii. Atribút MACB, tak ako aj v prípade dát zo zdroja FILE, je kategorickým atribútom, ktorý je možné rozložiť na štyri binárne. Rozličných hodnôt tohto atribútu však bolo menej ako v prípade zdroja FILE. Výrazne prevyšovala hodnota M..B.

Atribút Sourcetype, teda bližší popis zdroja údajov, v tomto prípade nadobudol len hodnotu 'WinEVTX'. Táto hodnota nie je prekvapením, keďže v operačnom systéme Windows od verzie Vista sa vyskytujú len EVTX záznamy (predtým EVT). Z tohto dôvodu bol v ďalšej analýze tento atribút vynechaný.

Tak ako v prípade typu FILE, aj v tomto prípade budú pri ďalšej analýze atribúty type, user, host, version a notes vynechané. Keďže **atribút format** obsahuje len hodnotu winevtx a táto informácia by bola duplicitná, bol taktiež vynechaný. Vynechať môžeme aj **atribút inode**, keďže v tomto prípade je to inode zdrojového súboru udalosti. **Atribút short**, tak ako aj v prípade FILE, je len skrátenou verziou desc, teda ho taktiež vynecháme.

Z atribútu desc sme extrahovali nový atribút s kategorickými hodnotami, ktorý popisuje **názov zariadenia** (computer_name).

Atribút filename je pri záznamoch typu EVT kategorickou premennou, ktorá obsahuje súbor, v ktorom sú logy uložené. V tomto prípade to bolo 113 zdrojov, ktoré sme roztriedili do 6 kategórií, a to **Application** (filename_application), **System** (filename_system), **Security** (filename_security), **logy spojené so vzdialeným prístupom** (filename_rdp), **logy spojené s použitím PowerShellu** (filename_powershell) a **kategória Iné** (filename_other).

V **atribúte extra** sa pri zdroji EVT nachádza celý XML reťazec z Windows Event Logs. Z tohto dôvodu obsahuje jedinečné údaje, z ktorých sme vyextrahovali ďalšie atribúty. Bol to **atribút recovered**, ktorý nadobúda hodnotu True, ak bol súbor vymazaný. Ďalej to boli **atribúty user_sid**, ktorý identifikuje používateľa, ktorý vytvoril proces, taktiež **atribút execution_process_id**, ktorý nadobúda hodnotu ID vykonávaného procesu. Taktiež **atribúty record_number a task. Task** pracuje s identifikátorom úlohy pre časť aplikácie alebo komponentu, ktorá publikuje udalosť. Extrahované boli taktiež **atribút event id**, ktoré sme roztriedili do deviatich skupín (Tabuľka č. 16).

Kategória	Popis	Názov atribútu
Account Logon	Nastavenie určuje, či má operačný systém auditovať každé overenie platnosti poverenia účtu v počítači.	event_type_account_logon
Account Management	Nastavenie určuje, či bude auditovať každá udalosť správy účtov v počítači.	event_type_account_management
Detailed Tracking	Nastavenie určuje, či bude operačný systém auditovať udalosti súvisiace s procesmi, ako sú vytvorenie procesu, ukončenia procesu, duplikácia popisovača a nepriamy prístup k objektom.	event_type_detailed_tracking
DS Access	Nastavenie určuje, či bude operačný systém auditovať pokusy používateľa o prístup k objektom služby Active Directory.	event_type_ds_access

Logon/Logoff	Nastavenie určuje, či má operačný systém auditovať jednotlivé inštancie pokusu používateľa o odhlásenie alebo prihlásenie k počítaču.	event_type_logon_logoff
Object Access	Nastavenie určuje, či bude operačný systém auditovať pokusy používateľov o prístup k objektom mimo služby Active Directory.	event_type_object_access
Policy Change	Nastavenie určuje, či má operačný systém auditovať jednotlivé inštancie pokusov o zmenu zásad priradených používateľským právam, zásad auditu, zásad účtov, alebo zásad dôveryhodnosti.	event_type_policy_change
Privilege Use	Nastavenie určuje, či sa má auditovať každé použitie používateľských práv používateľom.	event_type_privilege_usage
System	Nastavenie určuje, či bude operačný systém auditovať udalosti typu, napr. pokus o zmenu systémového času, pokus o spustenie alebo vypnutie systému zabezpečenia, pokus o zavedenie rozšíriteľných komponent overovania,...	event_type_system
Iné		event_type_other

Tabuľka 16 Popis kategórií Event ID

Z atribútu extra sme taktiež extrahovali sedem nových atribútov, ktoré sa týkali hodnôt v poli keywords (Tabuľka č. 17).

Názov atribútu	Popis	Hodnota poľa
keywords_audit_failure	Pripojené ku všetkým neúspešným security audit udalostiam. Výskyt len pri security záznamoch.	AuditFailure 0x10000000000000L
keywords_audit_success	Pripojené ku všetkým úspešným security audit eventom. Výskyt len pri security záznamoch.	AuditSuccess 0x20000000000000L
keywords_correlation_hint	Pripojené ku všetkým neúspešným security audit udalostiam. Výskyt len pri security záznamoch.	CorrelationHint 0x10000000000000L
keywords_event_log_classic	Pripojené k udalostiam vyvolaným pomocou funkcie RaiseEvent.	EventLogClassic 0x80000000000000L
keywords_sqm	Pripojené ku všetkým udalostiam Service Quality Mechanism (SQM).	Sqm 0x08000000000000L
keywords_wdi_context	Pripojené ku všetkým kontextovým udalostiam infraštruktúry Windows Diagnostics Infrastructure (WDI).	WdiContext 0x02000000000000L
keywords_wdi_diagnostic	Pripojené ku všetkým diagnostickým udalostiam infraštruktúry Windows Diagnostics Infrastructure (WDI).	WdiDiagnostic 0x04000000000000L

Tabuľka 17 Popis atribútov vzniknutých z poľa keywords s popisom

Kompletný zoznam atribútov zdroja EVT je uvedený v prílohe C.

3.2.8 Analýza záznamov typu REG

Aj pri záznamoch typu REG sme postupovali obdobne, ako v predchádzajúcich dvoch prípadoch, a to analýzou dostupných atribútov. Atribút MACB nadobúdal v tomto prípade len hodnoty 'M... ', '.... ' a '.A...'. Aj v tomto prípade bude rozdelený na štyri osobitné binárne atribúty M,A,C, B. Na tomto mieste je dôležité upozorniť na to, že MACB sa budú vzťahovať na zdrojový súbor registra operačného systému. Pridaná hodnota týchto atribútov je výrazne nižšia ako v prípade záznamov typu FILE, kde označujú operácie so súbormi alebo adresármi.

V prípade **atribútu sourcetype** sme identifikovali 16 jedinečných hodnôt, a to:

- 'Registry Key',
- 'AppCompatCache Registry Entry',
- 'Registry Key - Service',
- 'Task Cache' ,
- 'Registry Key - Typed URLs',
- 'Registry Key - Winlogon',
- 'Registry Key - Run Key',
- 'Background Activity Moderator Registry Entry',
- 'Registry Key - User Account Information',
- 'Registry Key - UserAssist',
- 'Registry Key - BagMRU',
- 'Registry Key - MRUListEx',
- 'Registry Key - MRUList',
- 'Registry Key - Network Drive',
- 'Registry Key Shutdown Entry' a
- 'Registry Key - USB Entries'.

Hodnoty tohto atribútu majú cennú pridanú hodnotu pri riešení, keďže na základe nich vieme určiť použitie USB v systéme (USB Entries), spustenie programu (napr.

UserAssist, AppCompatCache), zobrazenia obsahu adresára (BagMRU), nastavenie perzistencie malvéru (Run Key) a štart systému (Winlogon). Z tohto dôvodu vzniklo 16 nových binárnych atribútov.

Atribút desc bol zdrojom pre 11 novovzniknutých atribútov podľa typu hodnoty registra [70]:

- REG_BINARY
- REG_DWORD
- REG_DWORD_LITTLE_ENDIAN
- REG_DWORD_BIG_ENDIAN
- REG_EXPAND_SZ
- REG_LINK
- REG_MULTI_SZ
- REG_NONE
- REG_QWORD
- REG_QWORD_LITTLE_ENDIAN
- REG_SZ

V prípade **atribútu format** bolo nájdených 26 jedinečných hodnôt, ktoré sa stali novými binárnymi atribútmi: 'winreg/winreg_default', 'winreg/appcompatcache', 'winreg/windows_services', 'winreg/windows_task_cache' 'winreg/windows_typed_urls', 'winreg/winlogon', 'winreg/msie_zone', 'winreg/windows_run', 'winreg/amcache', 'winreg/bam', 'winreg/windows_sam_users', 'winreg/userassist', 'winreg/explorer_mountpoints2' 'winreg/explorer_programscache', 'winreg/bagmru', 'winreg/mrulistex_string_and_shell_item', 'winreg/mrulistex_string', 'winreg/windows_timezone', 'winreg/mrulist_string' 'winreg/network_drives', 'winreg/mrulistex_shell_item_list', 'winreg/mrulistex_string_and_shell_item_list', 'winreg/windows_shutdown', 'winreg/windows_usb_devices', 'winreg/windows_version, 'winreg/windows_boot_execute'.

Z atribútu extra sme vyextrahovali **haš súboru**, zdroj uloženia registrov (sha_256) a **atribút filename**, ktorý bol zdrojom informácií ohľadom určenia

windows/user hiveu(dir_win, dir_user, dir_other). Atribúty **user, version, host, inode**, notes nám neprinášajú prídavnú hodnotu, a preto boli vynechané.

Každému záznamu zo všetkých spomenutých zdrojov sme taktiež vygenerovali unikátne id, ktoré predstavuje samostatný atribút, na základe ktorého sa po ďalších fázach vieme spätne vrátiť k pôvodnému záznamu. Kompletný zoznam atribútov zdroja FILE je uvedený v prílohe D.

3.3 Agregácia

Cieľom tejto fázy je znižovanie veľkého množstva dostupných záznamov. To sa vykoná prostredníctvom zoskupenia podobných artefaktov, alebo artefaktov, ktoré spolu súvisia, či už z pohľadu času alebo z pohľadu vlastností/atribútov daných artefaktov. Výsledkom agregácie je tzv. **meta-artefakt**.

3.3.1 Prístup k agregácii v rámci forenznej analýzy

Analýza operačného systému, dokonca aj s minimálnym počtom používateľov činnosti používateľa, vygeneruje milióny udalostí s časovými pečiatkami. Výzvou v tejto fáze je preto spôsob, ako minimalizovať počty udalostí len na tie relevantné, na základe ktorých je možno odpovedať na základné otázky forenznej analýzy. Jeden z prístupov [71] sa pokúšal riešiť otázku agregovania udalostí na základe manuálneho kódovania vzoru časových pečiatok. Išlo o spojenie udalostí tzv. "nízkej úrovne" spojených s udalosťami "vyššej úrovne". Napr. otvorenie súboru používateľom v operačnom systéme Windows vytvára sériu artefaktov "nízkej úrovne" vrátane záznamov v registri operačného systému Windows, modifikáciu odkazov, jumplistov a iných. Tento prístup je však časovo náročný a náchylný na potencionálne chyby. Menšie rozdiely v operačných systémoch by už mohli viesť k nesprávnym záverom.

V článku [72] autori predstavujú prístup, ako zhromažďovať a normalizovať bezpečnostné upozornenia. Naviac rozoberajú spôsob, ako ich spájať a klasifikovať. Autormi zvolený prístup zahŕňa zber výstrah z rôznych zdrojov a ich normalizáciu podľa štandardizovaných štruktúr - IDMEF (Intrusion Detection Message Exchange Format). Po normalizovaní sa hlásenia zoskupujú do tzv. meta-alertov. Je to spôsob zhlukovania súvisiacich hlásení, ktoré sa neskôr klasifikujú pomocou techník strojového učenia na útoky alebo falošne pozitívne hlásenia. Každý meta-alert disponuje množinou atribútov,

z ktorých najdôležitejším je alert_taxonomy_set typu BitArray, ktorý autori prirovnávajú k množine všetkých možných stôp, ktoré môžeme na mieste činu nájsť. Potenciálne útoky, ktoré sú znázornené meta-alertami, predstavujú podmnožinu týchto stôp. Čím viac sa súbor stôp podobá na predchádzajúci známy scenár, tým viac je pravdepodobnejšie, že ide o skutočný útok. Použitie tohto bitového poľa slúži ako podpora techník , ktoré overujú podobnosť medzi dvoma scenármi v klasifikačnej vrstve.

3.3.2 Navrhovaný spôsob agregácie

Pri agregácii využívame len binárne atribúty. Ako sme uviedli kategorické atribúty sme buď previedli na binárne (s rozšírením počtu atribútov), alebo slúžia ako prvok, podľa ktorého agregáciu vykonávame (napr. názov a inode súboru pri záznamoch typu FILE). Keďže máme k dispozícii binárne atribúty, pri agregácii vieme nad týmito hodnotami použiť agregačnú funkciu. My sme si pre naše účely zvolili funkciu max. Táto funkcia vezme na vstupe všetky binárne hodnoty agregovaných záznamov pre každý atribút a v prípade ak sa medzi nimi vyskytuje aspoň jedna 1 (True), výstupom je 1 (True). V opačnom prípade (všetky hodnoty sú 0), funkcie dá na výstupe 0 (False). To nám zabezpečí pre metazáznam (agregovaný záznam) zachovanie informácie o danom atribúte. Meta-záznam bude mať hodnoty 1 (True) len pri tých binárnych atribútoch, pri ktorých sa hodnota 1 (True) nachádzala aspoň pri jednom zo záznamov.

Dátové rámce typu FILE, EVT a REG sme agregovali samostatne. Proces agregácia sa však mení len vzhľadom na použité atribúty, ktoré sme využili. Fáza agregácie má v našom navrhovanom spôsobe riešenia dve časti.

Prvá časť spočíva vo využití funkcie pandas.DataFrame.groupby [73]. Tá dokáže zoskupiť záznamy dátového rámca na základe určitej vybranej kombinácie atribútov. Rozšírením tejto funkcie je pandas.Grouper [74], ktorý dokáže bližšie špecifikovať groupby inštrukcie pre objekt.

Pri záznamoch zo zdroja FILE sme ako kombináciu pre funkciu groupby, na základe ktorej budeme spájať záznamy, zvolili v prvom prípade **atribút inode** (s kategorickými hodnotami) a využitím groupera sme mohli spájať záznamy aj v 15-sekundových intervaloch. Ukážka zdrojového kódu je na Obrázku č. 18. Jednoducho povedané, dáta sú najskôr rozdelené na 15-sekundové intervaly na základe časového rozdielu, ktorý sa počíta z atribútu datetime, ktorým disponuje každý záznam. Na každý interval je potom aplikovaná agregácia podľa zvoleného atribútu (v tomto prípade inode).

Pri záznamoch zo zdroja FILE sme tú istú agregáciu spravili aj využitím ďalšieho atribútu s kategorickými hodnotami, ktorým je **atribút name**. Vznikli nám tak dva súbory csv formátu, ktorých spojenie prebieha v druhej časti agregácie.

```
count_series = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'), 'inode']).size()
new_df = count_series.to_frame(name = 'size').reset_index()
agg_inode = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'), 'inode']).agg(lambda x: set(x))
agg_inode['counts'] = new_df['size'].values
agg_inode.to_csv("agregovane_df_file_inode.csv")
print(agg_inode.shape)

count_series = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'),'name']).size()
new_df = count_series.to_frame(name = 'size').reset_index()
agg_name = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'), 'name']).agg(lambda x: set(x))
agg_name['counts'] = new_df['size'].values
agg_name.to_csv("agregovane_df_file_name.csv")
print(agg_name.shape)
```

Obrázok 18 Ukážka agregácie dataframe-u dc_file pomocou funkcie groupby

Myšlienkou druhej časti agregácie je spojiť súbory, ktoré vznikli v predchádzajúcej fáze a získať tak jeden súbor obsahujúci meta-záznamy. Pri dátach zo zdroja FILE nám vznikli dva súbory, kde jeden je agregovaný v 15-sekundových intervaloch podľa atribútu inode (agregovane_df_file_inode) a druhý v 15-sekundových intervaloch podľa atribútu name (agregovane_df_file_name). Pri spojení prechádzame postupne prvým súborom, kde sa pozeráme na hodnoty atribútu inode. Ak aktuálne prechádzaný inode nájdeme v súbore agregovane_df_file_name a časový rozdiel je menší alebo väčší maximálne o 15 sekúnd, tak tieto záznamy zlúčime (množinové zjednotenie všetkých atribútov).

```python
def agregacia(row: List[str]) -> Union[str, List[str]]:

    inode_querried = row[-2]
    inode_querried_datetime = dt.fromisoformat(row[-1])

    # filtrujeme podla cisla inodu
    df_name_querried = df_name.query("(inode == @inode_querried)")
    if df_name_querried.shape[0] == 0:
        return "empty"

    # pomocna funkcia pre filtrovanie podla 15 sekundoveho okna
    def datetime_rozdiel_lambda(querried_row):
        compared_datetime = dt.fromisoformat(querried_row['datetime'])
        diff = compared_datetime - inode_querried_datetime
        return abs(diff.total_seconds())

    # filtrovanie podla okna pomocou lambda vyrazu
    df_name_querried["casovy_rozdiel"] = df_name_querried.apply(datetime_rozdiel_lambda, axis=1)
    df_name_querried = df_name_querried.query("(casovy_rozdiel <= 15.0)")
    if df_name_querried.shape[0] == 0:
        return "empty"
    df_name_querried = df_name_querried.drop(["casovy_rozdiel"], axis=1)

    # vhodne usporiadamie stlpce pre dalsie spracovanie
    df_name_querried = df_name_querried[string_column_assortment]
    return dataframe_union_to_series(df_name_querried, row)

# upravujeme vstupny dataframe po riadkoch a vysledok ukladame do pola zaznamov result
result = [agregacia(row) for row in df_inode[string_column_assortment].to_numpy()]
```

Obrázok 19 Ukážka zdrojového kódu agregácie v 2.fáze

Na Obrázku č. 19 je zobrazená časť zdrojového kódu, v ktorej je pomocná metóda na úpravu vstupných dátových rámcov podľa daných filtračných podmienok - 15 sekundový časový rozdiel a zhodnosť vybraného atribútu). Vstupný parameter row: List[str] je vstupný záznam dátového rámca na zlúčenie a úpravu. Výstupný parameter Union[str, List[str]] vracia "empty", ak filtrácii nevyhovujú žiadne záznamy vstupného dátového rámca, alebo vracia záznam v podobe poľa reťazcov pozostávajúci zo zlúčených podobných záznamov vstupného dátového rámca. Počet záznamov pred agregáciou a po agregácii pri záznamoch zo zdroja FILE je zobrazený v Tabuľke č. 18.

	Počet záznamov	
Pred agregáciou	843 863	
	Agregované podľa inode	Agregované podľa name
Po agregácii - 1. fáza	214 839	293 753
Po agregácii – 2. fáza	125 262	

Tabuľka 18 Zobrazenie počtu záznamov zdroja FILE pred a po agregácii

Agregácia pri záznamoch typu REG a EVT prebieha obdobne. Tieto agregácie sa líšia len tým, podľa akého atribútu dochádza k agregácii. V prípade záznamov typu EVT sa pre agregáciu môže použiť napr. kategorický atribút event id. Po agregácii je každému meta-záznamu vygenerované jedinečné id. Zdrojový kód použitý pri agregácii zdroja FILE je uvedený v Prílohe F.

4 Hľadanie vzťahov medzi atribútmi

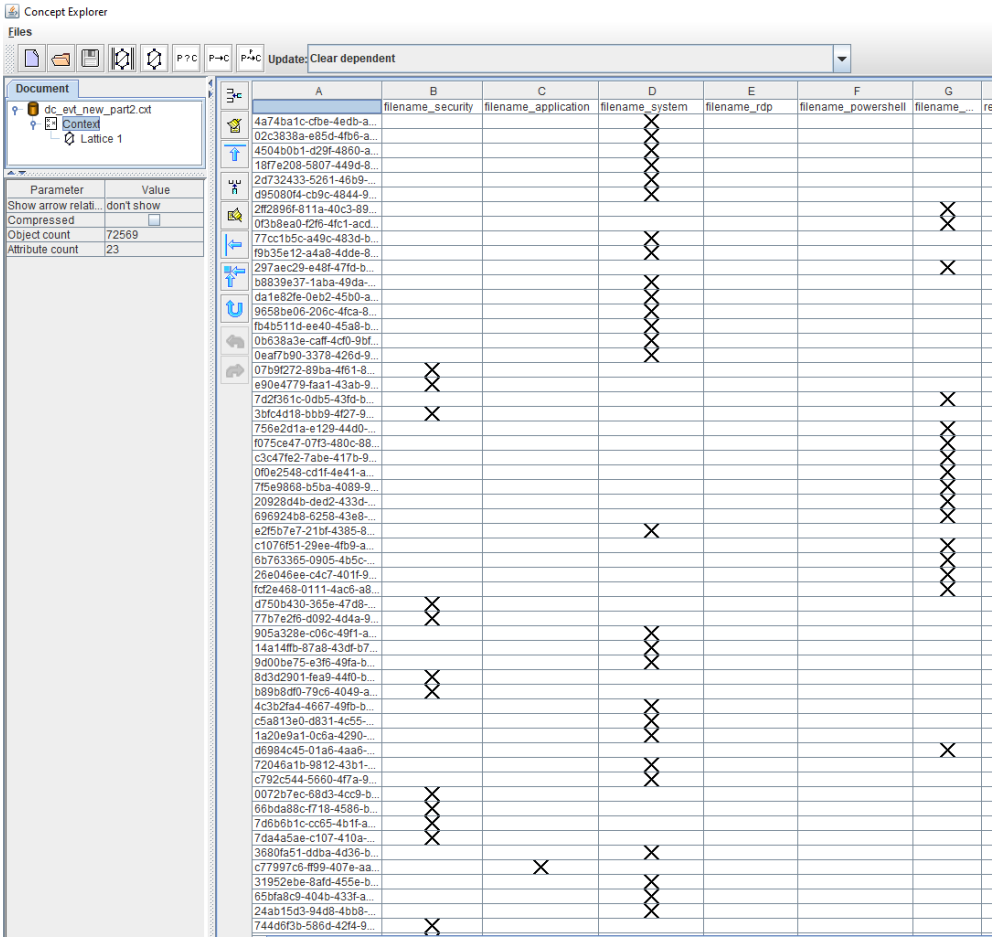
Na nájdenie možných vzťahov medzi atribútmi analyzovaných častí forenzného obrazu disku sme sa rozhodli využiť formálnu konceptovú analýzu. Je to analýza slúžiaca na analýzu dát, ktorá dokáže vizualizovať vzťahy medzi nimi, na základe súborov spoločných a odlišných atribútov údajov. Formálnu konceptovú analýzu využívajú autori napr. v článku [75] pri procese vyšetrovania založenom na kybernetickej bezpečnosti prostredníctvom vizualizácie a analýzy údajov mobilných komunikačných zariadení.

4.1 Formálna konceptová analýza

Formálna konceptová analýza (FCA) je spôsob, ako odvodiť hierarchiu pojmov alebo formálnu ontológiu zo súboru objektov a ich vlastností. Všetky pojmy v hierarchii predstavujú objekty, ktoré zdieľajú určitú množinu vlastností. Každý pojem v hierarchii predstavuje podľa umiestnenia podmnožinu alebo nadmnožinu objektov v pojmoch pod alebo nad ním. V tejto časti práce uvedieme definície pojmov formálny kontext a formálny koncept podľa [76].

Definícia Formálny koncept $K := (G, M, I)$ pozostáva z dvoch množín G a M a relácie I medzi G a M. Zložky G sú nazývané objekty a zložky M sú nazývané atribúty kontextu. Ak chceme vyjadriť, že objekt g je vo vzťahu I s atribútom M, píšeme gIm alebo $(g, m) \in I$ a čítame to ako "objekt g má atribút m".

Koncept je možné znázorniť aj ako objektovo-atribútovú tabuľku. Príklad tejto tabuľky pre záznamy typu EVT je možné vidieť na Obrázku č. 20 (zobrazenie v programe Concept Explorer).



Obrázok 20 Ukážka konceptu ako objektovo-atribútovej tabuľky pre záznamy typu EVT.

Definícia Formálny koncept kontextu $K := (G, M, I)$ je definovaný ako dvojica (A, B), kde $A \subseteq G$, $B \subseteq M$, $A = B^I$ a $B = A^I$. A nazývame extent a B intent formálneho konceptu (A, B). $\beta(G, M, I)$ označuje množinu všetkých konceptov kontextu (G, M, I).

Čiastočne usporiadaná množina formálnych konceptov sa nazýva konceptový zväz. Konštrukcia konceptový zväzu v oblasti formálnej konceptuálnej analýzy súvisí s pojmom Galoisovho spojenia [76]. To je indukované formálnym kontextom, t. j. jeho ostrým binárnym vzťahom. Tento vzťah platí aj naopak, každé Galoisovo spojenie indukuje formálny kontext ako ostrý binárny vzťah. Z formálneho kontextu môžeme zostaviť formálne koncepty, ktoré sú dvojicami extentov (t. j. podmnožín objektov) a intentov (t. j. podmnožín atribútov) získaných príslušnými operátormi tvoriacimi koncepty. Okrem toho môžeme každému objektu a každému atribútu priradiť formálny pojem.

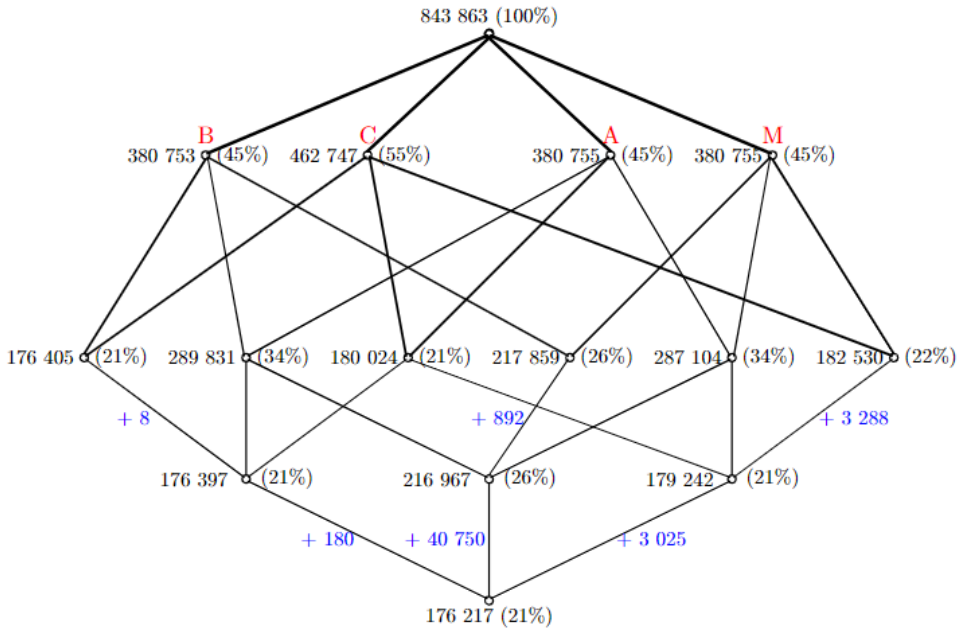
V každom konceptovom zväze existujú sup-dense a inf-dense množiny. Ich výsledky poskytujú metódu na konštrukciu diagramu ľubovoľného pojmového zväzu. V tomto diagrame potom možno získať množinu všetkých objektov patriacich do extentu konkrétneho konceptu, a to zhromaždením všetkých označení objektov vedúcich od konkrétneho prvku smerom nadol. Množinu všetkých atribútov patriacich do intentu konkrétneho konceptu možno získať zhromaždením všetkých označení atribútov vedúcich od prvku smerom nahor.

V článku [77] autori porovnávajú nástroje, ktoré možno využiť na formálnu konceptovú analýzu. V našej práci sme zvolili nástroj ConExp (Concept Explorer) [78]. Je to projekt s otvoreným kódom založený na programovacom jazyku Java. Hlavnými funkcionalitami tohto nástroja je vytváranie konceptov a ich vizualizácia. ConExp nepodporuje prepojenie s databázou, ale kontexty možno importovať a exportovať vo formáte ".CXT". Na rozloženie zväzu možno vybrať z niekoľkých algoritmov. Vytvorené diagramy možno exportovať vo formáte JPEG alebo GIF. ConExp v súčasnosti implementuje najväčší súbor operácií z knihy Formal Concept Analysis [79], vrátane výpočtu asociačných pravidiel a Duquenne-Guiguesovej bázy implikácií.

4.2 Koncepty v časti FILE

Pri analýze záznamov zo zdroja FILE sme sa najskôr zamerali na atribúty časových pečiatok MACB. Z hľadiska digitálnej forenznej analýzy pomáha analýza časových pečiatok pochopiť, aká operácia bola vykonaná so súborom (napr. vytvorenie súboru, premenovanie súboru).

Na Obrázku č. 21 je uvedený konceptový zväz MACB s 15 vrcholmi, ktoré predstavujú formálne koncepty, a 28 hranami. Výška konceptového zväzu je 4. Pri každom vrchole je uvedený aj počet objektov, ktoré sú súčasťou konceptu, a taktiež je uvedené aj ich percentuálne vyjadrenie.



Obrázok 21 Konceptový zväz atribútov MACB

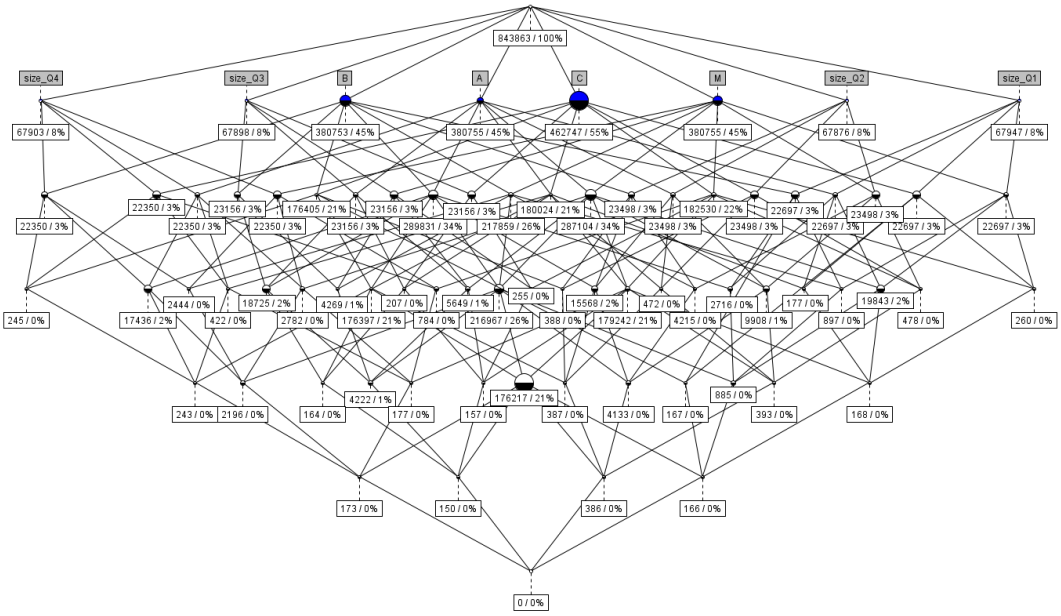
Vidíme, že 21% objektov v rámci našej prípadovej štúdie sa vzťahuje na koncept všetkých štyroch atribútov MACB, čo predstavuje operáciu vytvorenia súboru. Ak sa pozeráme na trojicu atribútov, najväčší percentuálny obsahuje koncept troch atribútov BAM, a to 26%. Koncepty atribútov CAM, ktorých kombinácia môže predstavovať napríklad modifikáciu obsahu súboru alebo premiestnenie priečinka v rámci volume, a BCA, čo predstavuje operáciu kopírovania súboru, zastupujú zhodne 21% objektov. Koncept s atribútmi BCM sa nenachádza v tomto konceptovom zväze. Táto kombinácia chýba aj v Tabuľke č.8, ktorá porovnáva operácie so súbormi na základe časových pečiatok MACB podľa viacerých zdrojov. V prípade konceptov dvojíc atribútov, najväčšie koncepty zahŕňajú 34% objektov s atribútmi aspoň BA, čo môže určovať skopírovanie súboru alebo premiestnenie súboru v rámci volume, a 34% objektov s atribútmi aspoň AM, čo môže znamenať modifikáciu súboru. Ako priamo vidíme, 45% objektov má aspoň atribút B, 55% objektov aspoň atribút C, 45% objektov aspoň atribút A a 45% objektov aspoň atribút M.

Okrem samotných vzťahov medzi typmi časových pečiatok je nevyhnutné sa pozrieť aj na koncepty v kombinácii s inými atribútmi. Konceptový zväz atribútov digitálnych dôkazov poskytuje forenznému analytikovi metódu na štúdium vzťahu medzi digitálnym dôkazom (konkrétny záznam) v jeho kontexte v závislosti od iných záznamov.

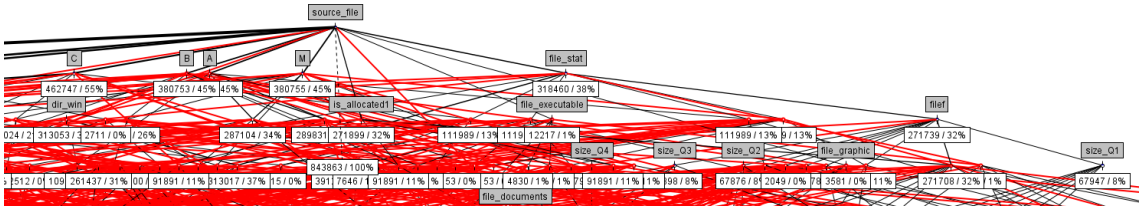
V kapitolách 3.2.5 až 3.2.8 rozoberáme všetky atribúty, na ktoré sme sa v rámci jednotlivých zdrojov zamerali.

Koncepty záznamov, ktoré vzniknú ako výsledok kombinácie atribútov MACB a atribútov ciest k súborom/adresárom, poskytujú informácie o operáciách v konkrétnych cestách. Koncepty záznamov, ktoré vznikajú kombináciou atribútov MACB a typov súborov, umožňujú lepšie pochopiť operácie s konkrétnymi typmi súborov. Koncepty záznamov vytvorených atribútmi MACB a veľkosti súborov indikujú štandardné a anomálne súbory v súborovom systéme. Ukážka takého konceptu je zobrazená na Obrázku č. 23. Možné je sa pozrieť aj na koncepty tvorené rôznymi kombináciami viacerých kategórií atribútov.

Napríklad počet záznamov v extente, ktorý je tvorený atribútmi size Q3, filef, file executable, dir win, A, M, B, C, file_stat, je jeden. Znamená to, že v adresári windows bol vytvorený iba jeden spustiteľný súbor. Na druhej strane pre nás môže byť zaujímavý extent, ktorý obsahuje atribút size_none (súbor s nulovou veľkosťou), ktorý súvisí s počtom odstránených súborov v systéme.



Obrázok 22 Konceptový zväz pre kategórie atribútov MACB a veľkostí súborov



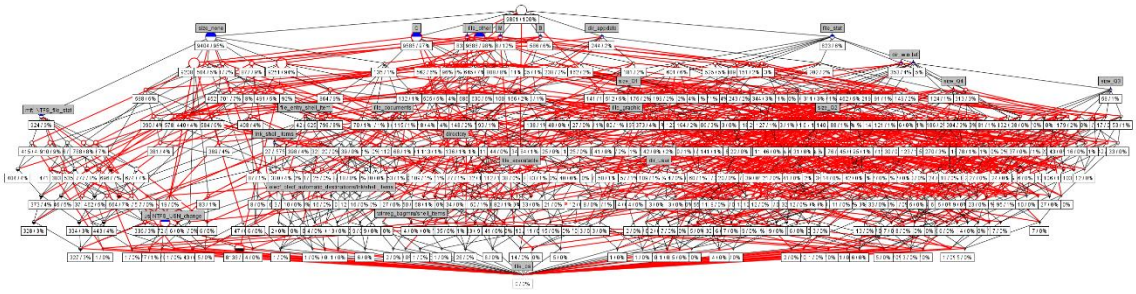
Obrázok 23 Výsek diagramu pre všetky atribúty zdroja FILE

Úplný diagram tridsiatich troch atribútov zdroja FILE, ktorého výsek je zobrazený na Obrázku č. 24 obsahuje jedenásť úrovní. Je potrebné spomenúť, že okrem diagramu je výsledkom formálnej konceptovej analýzy aj súbor asociačných pravidiel, ktoré vyjadrujú vzťahy medzi atribútmi. Ich formát vyzerá nasledovne:

< 289054 > A source_file size_none =[86%]=> < 248922 > M

Uvedené pravidlo hovorí o tom, že ak sa v analyzovaných údajoch vyskytli záznamy, ktoré obsahovali skupinu atribútov A, source_file a size_none, tak sa tam v 86% vyskytoval aj atribút M.

Vzhľadom na množstvo spracovaných údajov a cieľ práce, ktorým je demonštrovať použitie daného modelu, sme pre účely vizualizácie zúžili časové okno vstupných záznamov. Do úvahy sa vzalo časového ohraničenie bezpečnostného incidentu, čo nemá zásadný vplyv na samotný účel použitia modelu. Súčasne boli vybrané len tie pravidlá, ktoré nepokrývali veľké množstvo objektov v konceptoch. Inými slovami, vynechali sa tie koncepty, v ktorých bolo veľké množstvo záznamov. Predpokladáme, že týmto vynecháme veľké množstvo pre prípad nerelevantných záznamov. Špecifické záznamy, vyznačujúce sa zaujímavou kombináciou atribútov, budú pokryté konceptami s menším počtom objektov(záznamov). Koncept, ktorý vznikol z dát s časovým ohraničením bezpečnostného incidentu je zobrazený na Obrázku č. 25.



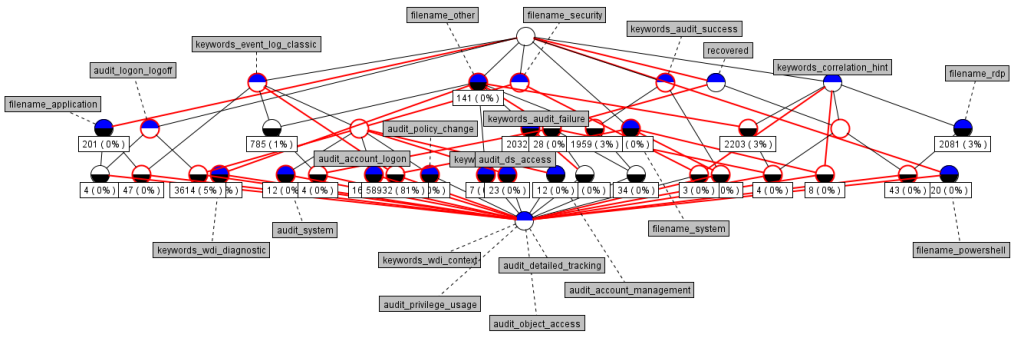
Obrázok 24 Ukážka konceptu z dát s časovým ohraničením bezpečnostného incidentu

Zo vzniknutých extentov sme teda zobrali do úvahy nasledujúcich 12, s ktorými budeme pracovať aj v ďalších fázach:

- C,M,file_documents,size_Q1,filef,file_stat,
- C,M,file_stat,
- C,M,filef,file_stat,
- C,M,size_Q1,filef,file_stat,
- C,file_documents,size_Q1,filef,file_stat,
- C,filef,file_stat,
- C,size_Q1,filef,file_stat,
- M,file_documents,size_Q1,filef,file_stat,
- M,filef,file_stat,
- M,size_Q1,filef,file_stat,
- file_documents,size_Q1,filef,file_stat a
- size_Q1,filef,file_stat.

4.3 Koncepty v časti EVT

Podobne ako pri dátach zo zdroja FILE, sme aplikovali metódy formálnej konceptovej analýzy aj na dáta zo zdroja EVT. Kvôli množstvu analyzovaných dát sme sa taktiež rozhodli vziať len tú časť dát, ktorá časovo korešponduje s bezpečnostným incidentom v prípade. Konceptový zväz reprezentuje Obrázku č. 26.



Obrázok 25 Zobrazenie konceptu dát zdroja EVT

Zápis asociačného pravidla vyzerá nasledujúco:

<počet > {atribúty} =[dôveryhodnosť] => <počet> {atribúty}

Napríklad korelačné pravidlo < 71285 > keywords_event_log_classic =[84%]=> < 59968 > filename_system znamená, že ak záznamy majú atribút keywords_event_log_classic (71 285 záznamov), potom len 59 968 záznamov má súčasne aj atribút filename_system. Dané pravidlo teda platí na 84%.

Pri analýze asociačných pravidiel používame vygenerované asociačné pravidlá zo všetkých záznamov a asociačné pravidlá zo záznamov z obdobia, ktoré ohraničuje bezpečnostný incident.

Vzhľadom na počet týchto pravidiel, uvedieme len niekoľko, ktoré podľa nášho názoru môžu hrať významnú úlohu pri forenznej analýze.

Zaujímavým aspektom je, že niektoré pravidlá sú fakty, napríklad:

- Pravidlo zo všetkých údajov: < 60189 > filename_system =[100%]=> < 59968 > keywords_event_log_classic;
- Pravidlo v čase incidentu: < 59016 > filename_system =[100%]=> < 58932 > keywords_event_log_classic;
- Pravidlo zo všetkých údajov: 7308 > audit_logon_logoff =[100%]=> < 7296 > filename_security keywords_event_log_classic;
- Pravidlo v čase incidentu: < 3618 > audit_logon_logoff =[100%]=> < 3614 > filename_security keywords_event_log_classic;
- Pravidlo zo všetkých údajov: < 2235 > keywords_audit_failure =[100%]=> < 2235 > filename_other;
- Pravidlo v čase incidentu: < 2137 > keywords_audit_failure =[100%]=> < 2137 > filename_other;

Nasledujúce korelačné pravidlo znamená, že všetkých 2124 záznamov s atribútmi filename_rdp a keywords_correlation_hint sa vyskytlo v čase bezpečnostného incidentu, a takáto kombinácia atribútov nie je príznačná pre ostatné obdobie:

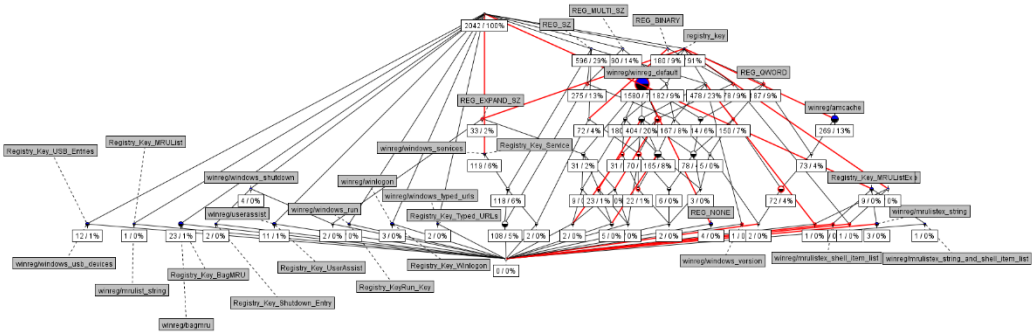
- Pravidlo zo všetkých údajov: < 2124 > filename_rdp =[100%]=> < 2124 > keywords_correlation_hint;
- Pravidlo v čase incidentu: < 2124 > filename_rdp =[100%]=> < 2124 > keywords_correlation_hint;
- Zaujímavým sa z komplexného pohľadu javia záznamy, ktoré majú atribúty recovered a filename_other. Ako ukazujú nižšie záznamy, z tohto korelačného pravidla je ťažko vyvodiť nejaký výsledok:
- Pravidlo zo všetkých údajov: < 838 > recovered =[88%]=> < 734 > filename_other;
- Pravidlo v čase incidentu: < 218 > recovered =[80%]=> < 175 > filename_other;

V rámci analýzy sa našli aj pravidlá, ktoré sa nevyskytli v čase bezpečnostného incidentu:

- Pravidlo zo všetkých údajov: 22 < 4 > audit_object_access =[100%]=> < 4 > filename_security keywords_event_log_classic;
- Pravidlo zo všetkých údajov: < 1 > recovered audit_logon_logoff =[100%]=> < 1 > filename_application;
- Naopak, zaujímavým bolo asociačné pravidlo, ktoré sa vyskytovalo len v čase bezpečnostného incidentu:
- Pravidlo v čase incidentu: < 47 > recovered keywords_correlation_hint =[91%]=> < 43 > filename_rdp;

4.4 Koncepty v časti REG

Tak ako aj v predchádzajúcich dvoch prípadoch, aj pri dátach zo zdroja REG sme využili na analýzu formálnu konceptovú analýzu. Vzniknutý konceptový zväz reprezentuje Obrázku č. 27.



Obrázok 26 Zobrazenie konceptu dát zdroja REG

Vzhľadom na počet asociačných pravidiel, uvedieme len niekoľko z nich, ktoré podľa nášho názoru môžu hrať významnú úlohu pri forenznej analýze.

Zaujímavým aspektom je, že niektoré pravidlá sú fakty, napríklad:

- Pravidlo zo všetkých údajov: < 2507 > REG_MULTI_SZ REG_SZ =[100%]=> < 2507 > dir_win;
- Pravidlo v čase incidentu: < 275 > REG_MULTI_SZ REG_SZ =[100%]=> < 275 > dir_win;

- Pravidlo zo všetkých údajov: < 160 > Task_Cache =[100%]=> < 160 > dir_win winreg/windows_task_cache;
- Pravidlo v čase incidentu: < 160 > winreg/windows_task_cache =[100%]=> < 160 > Task_Cache dir_win;

- Pravidlo zo všetkých údajov: < 9 > Registry_Key_MRUListEx =[100%]=> < 9 > dir_user;
- Pravidlo v čase incidentu: < 9 > Registry_Key_MRUListEx =[100%]=> < 9 > dir_user;

- Pravidlo zo všetkých údajov: < 819 > Registry_Key_Service =[100%]=> < 819 > dir_win winreg/windows_services;

- Pravidlo v čase incidentu: < 119 > Registry_Key_Service =[100%]=> < 119 > dir_win winreg/windows_services;

Nasledujúce korelačné pravidlo znamená, že 2 záznamy s atribútmi „winreg/windows_run“, „Registry_KeyRun_Key“, „dir_win“ sa vyskytli v čase bezpečnostného incidentu a takáto kombinácia atribútov nie je príznačná pre ostatné obdobie:

- winreg/windows_run =[100%]=> < 2 > Registry_KeyRun_Key dir_win;

V rámci analýzy sa našli aj pravidlá, ktoré sa nevyskytli v čase bezpečnostného incidentu:

- < 27 > Registry_Key_BagMRU winreg/bagmru =[96%]=> < 26 > dir_win;

Naopak, zaujímavým bolo asociačné pravidlo, ktoré sa vyskytovalo len v čase bezpečnostného incidentu:

- < 1 > winreg/windows_version =[100%]=> < 1 > registry_key REG_SZ;
- < 2 > winreg/msie_zone =[100%]=> < 2 > registry_key;

5 Korelácia meta-záznamov

Ďalšou fázou nášho riešenia po agregácii je korelácia. Cieľom tejto fázy je hľadať vzťahy medzi vzniknutými meta-artefaktmi. Hľadanie týchto vzťahov predstavuje nutnú činnosť analytickej činnosti forenzného analytika. Hľadajú sa relevantné artefakty a vzťahy medzi nimi. Na základe tejto analýzy je možné súčasne spájať pre prípad relevantné artefakty a tie, ktoré s daným prípadom nesúvisia.

Autori v [80] porovnávajú rôzne prístupy ku korelácii a spracovaniu udalostí. Tieto prístupy je možné podľa nášho názoru zovšeobecniť aj na digitálne artefakty, ktoré tieto udalosti popisujú. Bližšie sme sa týmto prístupom venovali v druhej kapitole.

5.1.1 Navrhovaný spôsob korelácie

V našom návrhu sme sa rozhodli použiť Rule-based prístup, teda nájsť vnútorné a vonkajšie vzťahy medzi artefaktmi na základe vopred zadaných pravidiel. Možným spôsobom je nájdenie do nejakej miery zhodných atribútov pri jednotlivých artefaktoch, ktoré nám poskytuje výstup z formálnej konceptovej analýzy.

Z predchádzajúcej fázy sme na výstupe dostali zo zdrojov dát FILE, REG aj EVT agregovaný súbor s meta-záznamami a taktiež vytvorené koncepty, ktoré popisujú vzťahy medzi atribútmi. Ako sme už spomenuli vyššie, vybrali sme tie, ktoré nepokrývali veľké množstvo objektov. Keďže sa analýza spúšťala nad neagregovanými záznamami, prvou časťou procesu bolo priradiť ku všetkým záznamom identifikátory patričných meta-záznamov. Keďže sme si zachovali id záznamov pôvodných dát aj meta-záznamov, priradenie bolo jednoduché. Ak sa v koncepte nachádzalo viac záznamov, ktoré patrili do jedného meta-záznamu, jeho id bolo pridané len jeden krát. V prípade dát zo zdroja FILE, sme na základe množiny id meta-záznamov, ktoré z toho vzišli, vedeli vygenerovať množinu názvov súborov (name), ktorých vzťahy nás zaujímajú. Z predchádzajúcich krokov už bolo možné zistiť taktiež aj množinu inodov, ktoré sú vo vzťahu s vybranými názvami súborov a vzniknutými konceptami. Ich extrakcia je zaznamenaná na Obrázku č. 28, ktorý obsahuje časť zdrojového kódu procesu korelácie.

```
df_agg = pd.read_csv (r'/content/drive/MyDrive/dc_final/agregovane_metazaznamy_file.csv', error_bad_lines=False, sep= ',')

my_file = open("name_pre_inode.txt", "r")
content = my_file.read()
content_list = content.split("\n")
my_file.close()

unique_names = set(content_list)
list_inode = list()
for unique in unique_names:
  file2 = open("file_inode.txt", "a")
  for value in df_agg[df_agg['name'].str.contains(unique)].inode:
    list_inode.append(value)
setik = set(list_inode)
file2.write(str(setik) + '\n')
file2.close()
```

Obrázok 27 Časť zdrojového kódu výberu inodov na vizualizáciu

V prípade dát zo zdroja FILE nám vznikli tri súbory, ktoré vyjadrovali vzťahy na základe vybraných konceptov, inodov a názvov. Ukážka vzniknutého súboru so zaznamenanými vzťahmi zobrazuje Obrázok č. 29. V tomto prípade je prvý člen názov vzťahu (vo vizualizácii názov hrany) a za ním nasledujú id meta-záznamov, ktoré budú vo vizualizácii vystupovať ako vrcholy. Ako už bolo naznačené, tieto súbory sú vstupom pre fázu vizualizácie, ktorá je popísaná v nasledujúcej podkapitole.

```
[Secret.lnk, bb5f1a2e-08a6-4018-90f9-91358359ec9e, 670270be-8118-4edb-8291-5b5e222c1e1a,
[SECRET_beth.lnk, 54bd792d-fcd4-4466-94e1-0ec597ced2fa, 488a2393-950a-4973-8537-5c2923f57
[AitEventLog.etl.002, 45d40377-c45b-4656-af62-926bdb59f189, 290c6b22-5f67-4dfa-be16-36345
[Dfsr00007.log.gz, 7714cbde-eb8e-4fbb-be6b-7e038e4aece1, 3b6c1387-121d-404d-a3c7-ed15bbe1
[Szechuan, 6d6939a6-5d92-412e-8dc3-dd6b074ec9d4, 558bf825-ab72-4200-93ce-e34500824129, 3a
[V0100001.log, c7594cbf-0854-4ede-b5d5-5986759cf1ff, 4288dae8-9aba-4872-9209-1724022d87c9
[Documents, 93aa156f-d37d-446d-8e30-ff4c85989d2c, a3a85508-137d-4f9a-acc0-9db136c0eb93, 2
[Beth_Secret.txt, 9705dcee-6983-4bbb-bb53-71312e2b3eec, 4156a40b-bb66-4d1c-ba9f-958c38640
[coreupdater.exe, 619b9388-2a19-4f6a-85fc-1fc552539946, 719b1921-7f49-4d47-848a-bb752d570
```

Obrázok 28 Ukážka súboru so zaznamenanými vzťahmi na základe atribútu name

5.2 Vyhodnotenie a vizualizácia

Identifikácia možných atribútov a hľadanie vzťahov medzi nimi je dôležitou výskumnou otázkou v tejto oblasti. Rovnako dôležitým aspektom je identifikácia relevantných digitálnych dôkazov pre daný prípad. Na tento účel sme analyzovali tieto digitálne stopy pomocou formálnej analýzy.

Ukázalo sa, že vytváranie konceptov pomáha pri analýze forenzných časových osí. Na jednej strane umožňuje všeobecné pochopenie vzťahu medzi jednotlivými atribútmi digitálnych stôp (záznamov). Existuje možnosť porovnať tieto vzťahy prostredníctvom viacerých prípadov. Na druhej strane sa možno použiť na identifikáciu

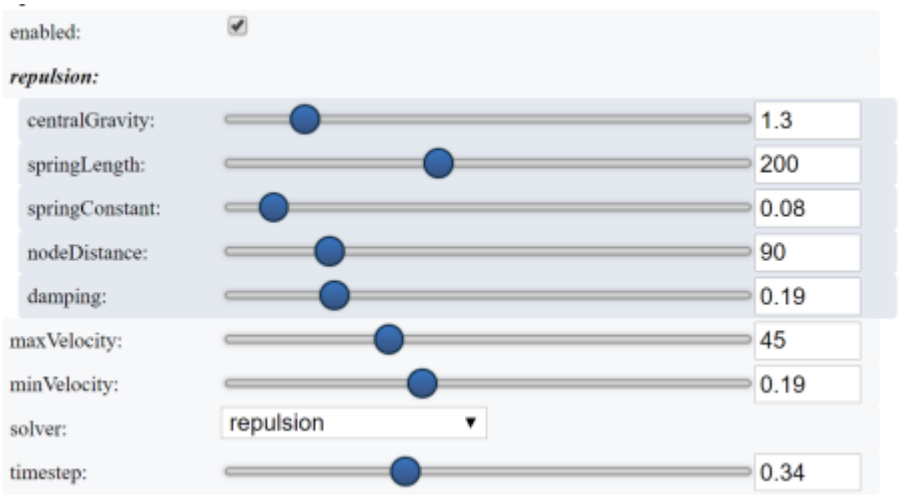
výnimočných prípadov špecifických pre súborový systém NTFS, resp. operačný systém Windows alebo typ anomálie. Tieto anomálie sú atraktívne pre forenzného analytika, pretože ho upozorňujú na digitálne dôkazy, ktoré sú v niektorých ohľadoch špecifické. Analytik tak môže rýchlo nájsť relevantné záznamy pre daný prípad a vykonať ďalšiu analýzu.

5.2.1 Vytvorenie scenára a jeho vizualizácia

Cieľom tejto fázy je vytvorenie scenárov (konkrétnych postupností meta-záznamov), ktoré je možné identifikovať v rámci systému na základe vytvorených korelačných vzťahov z predchádzajúceho kroku.

5.2.2 Navrhovaný spôsob vytvárania scenára

Na vizualizáciu vytvoreného scenára a reprezentáciu vzťahov medzi udalosťami v našom riešení sme sa rozhodli využiť python modul pyvis [81]. Určený je na rýchle generovanie vizuálnych sieťových grafov. Vďaka tomu, že je postavený na JavaScript knižnici VisJS, ponúka aj interaktívnu manipuláciu s vytvorenými sieťovými grafmi. Obsahuje taktiež možnosť dodať vizualizácii používateľské rozhranie, ktoré sa používa na dynamickú zmenu niektorých nastavení týkajúcich sa vytvoreného grafu. To môže byť užitočné pri hľadaní optimálnych parametrov nastavenia a funkcií rozloženia grafu tak, aby bol pre analytika čo najľahšie čitateľný. Zobrazený je na Obrázku č. 30. Viac o tomto module možno nájsť v článku [82]



Obrázok 29 Zobrazenie používateľského rozhrania pyvis

Vzniknuté súbory, ktoré sme dostali z predchádzajúcej fázy sme sa rozhodli vizualizovať nasledujúcim spôsobom. Ako vrcholy vo vizualizácii vystupujú id meta-záznamov. Hrany medzi nimi predstavujú vzťah k súboru, inodu alebo vyjadrujú príslušnosť ku konceptu. Metódu, ktorá medzi každý vrchol zo zoznamu vrcholov vloží do grafovej štruktúry g neorientovanú hranu s označením val zobrazuje Obrázok č. 31.

```
# importujeme kniznicu na vizualizaciu grafov
from pyvis.network import Network
# importujeme kniznicu pre efektivne grafove datove struktury
import networkx as nx
# typova struktura pre dokumentaciu
from typing import List

def add_edge_to_list(val:str, nodes_list:List[str], g: nx.DiGraph) -> None:
    # cykly cez polia vrcholov
    for u_ind in range(len(nodes_list)-1):
        for v in nodes_list[u_ind+1:]:
            g.add_edge(nodes_list[u_ind],v,title=val)
            g.add_edge(v,nodes_list[u_ind],title=val)
```

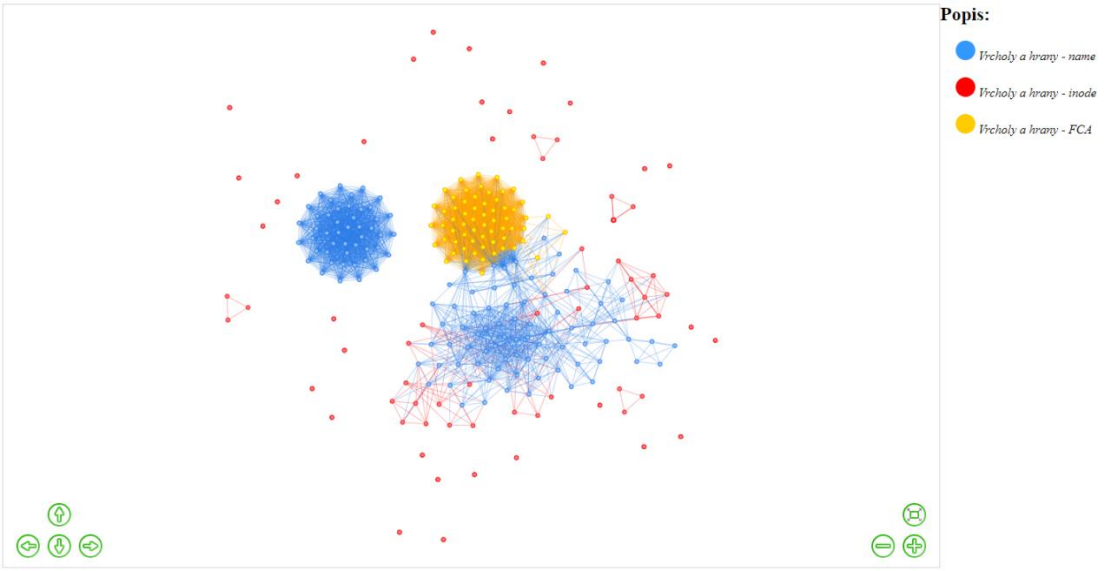
Obrázok 30 Zdrojový kód metódy na vloženie hrany neorientovanej hrany medzi vrcholy

Vstupné parametre pre túto metódu sú val – predstavuje reťazec na označenie hrany, nodes_list – predstavuje pole vrcholov a g – predstavuje grafovú štruktúru, do ktorej sa vkladajú hrany a vrcholy. Na Obrázku č. 32. je zobrazená pomocná metóda na čítanie vstupu zo súboru z predchádzajúcej fázy so zaznačenými vzťahmi.

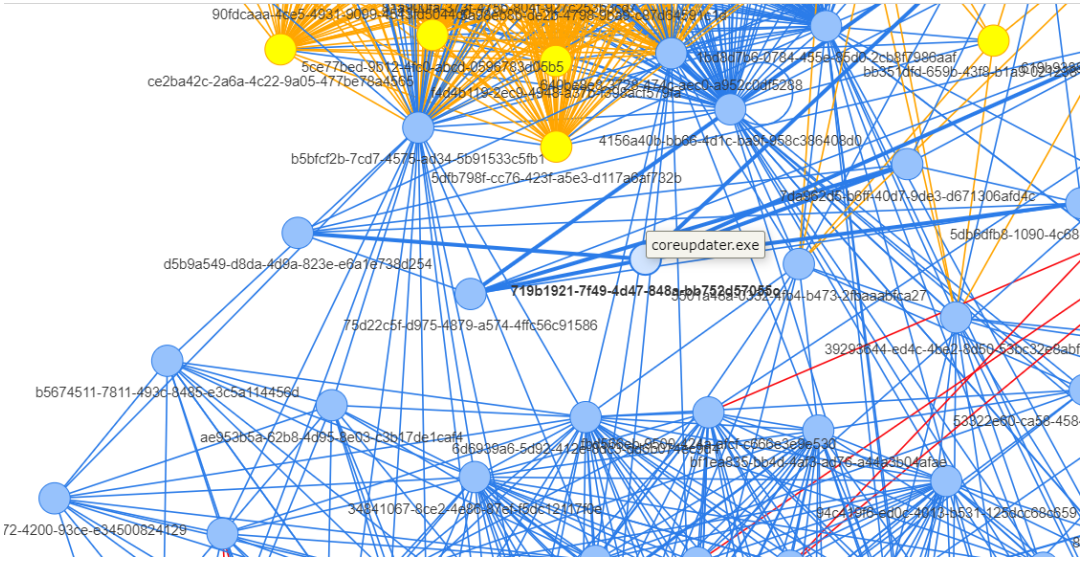
```
def standard_reader(Lines: List[str], g: nx.DiGraph, group_id:int) -> None:
    for line in Lines:
        # oddelime udaje hran od vrcholov
        line_list = line.split(',')
        line_list[0] = line_list[0].strip()[1:]
        line_list[len(line_list)-1] = line_list[len(line_list)-1].strip()[:-1]
        line_list = [s.strip() for s in line_list]

        # pridanie vrcholov do grafovej struktury
        add_nodes(line_list[1:], g, group_id)

        # pridanie vhodnych ohodnoteni hran do grafovej struktury
        add_edge_to_list(line_list[0], line_list[1:], g)
```

Obrázok 31 Ukážka zdrojového kódu pomocnej metódy na čítanie vstupu

Na Obrázku č. 33 a 34 je zobrazená samotná vizualizácia záznamov, ktoré sme dostali z fázy korelácie. Žltou farbou sú označené hrany a vrcholy pochádzajúce zo vzťahu formálnej konceptovej analýzy, červenou farbou sú zobrazené hrany a vrcholy prislúchajúce vzťahom z inodov a modrou farbou sú zobrazené vrcholy a hrany pochádzajúce zo vzťahov na základe atribútu name.



Obrázok 32 Grafová vizualizácia záznamov z fázy korelácie prostredníctvom modulu pyvis



Obrázok 33 Priblíženie vizualizácie záznamu coreupdater.exe

Na vizualizáciu časovej postupnosti záznamov, teda časový (temporal) prístup k vizualizáciu, kedy skoršia udalosť zohráva úlohu pri vzniku neskoršej udalosti, možno zobraziť vo vizualizácii orientovanými hranami v grafe. Ako sme už spomínali, ďalšie

nastavenia sú možné podľa preferencií. Zdrojový kód využitý pri vizualizácii vzťahov je uvedený v Prílohe G.

Záver

Riešenie bezpečnostných incidentov je nevyhnutnou súčasťou akýchkoľvek bezpečnostných opatrení v rámci informačnej a kybernetickej bezpečnosti organizácii. Jeho cieľom je najmä identifikovať vstupný (inicializačný) vektor útoku, postup útočníka (jeho jednotlivé kroky) a následne aj dopad pre samotnú organizáciu. Nevyhnutnou súčasťou riešenia incidentov je aj forenzná analýza a vytváranie sledu udalostí v chronologickom poradí. Inými slovami, forenzní analytici vytvárajú a analyzujú časové osi.

V rámci tejto záverečnej práce sme sa zamerali na časové osi a ich pozíciu v rámci forenznej analýzy digitálnych stôp, resp. forenzných artefaktov v rámci operačného systému Windows. Vzhľadom na rozsah skúmanej problematiky bližšie sa nevenujeme analýze artefaktov operačnej pamäte a sieťovej komunikácie a časovým osiam z nich vytvorených.

Hlavým cieľom tejto záverečnej práce je preskúmať možnosti analýzy časových osí pre účely forenznej analýzy. Tento cieľ je rozdelený do troch parciálnych podcieľov. Prvým čiastkových cieľom práce je analýza aktuálnych prístupov k spracúvaniu časových osí forenzných artefaktov operačného systému Windows. Spracovanie tohto čiastkového cieľa sa nachádza v druhej kapitole tejto práce. Vzhľadom na rozsah skúmanej problematiky, sme podobné práce rozdelili do štyroch kategórií, ktoré nie sú nevyhnutné bez prieniku. V rámci tejto kapitoly sme analyzovali nástroje pre vytvorenie a analýzu časových osí. Osobitnú pozornosť sme venovali odborných a vedeckým článkom, ktoré popisujú vytváranie časových osí, ich následnú analýzu a napokon aj koreláciu jednotlivých záznamov v rámci časových osí. Kapitola súčasne obsahuje vymedzenie vytvoreného modelu spracovania časových osí k existujúcim metódam ako aj porovnanie nami vytvoreného nástroje k nám dostupným nástrojom.

Druhým čiastkovým cieľom bola analýza spôsobov korelácie atribútov forenzných artefaktov a vytvárania vzťahov medzi nimi. Pre tento účel bolo nevyhnutné definovať možné atribúty forenzných artefaktov a ich reprezentáciu v rámci jednotlivých záznamov uvedených v rámci časových osí. Tomuto cieľu sa bližšie venujeme v tretej až piatej kapitole. V rámci tretej kapitoly bližšie popisujeme fázu predspracovania údajov.

Posledným čiastkovým cieľom práce je návrh, implementácia a vyhodnotenie nástroja pre vytvorenie a vizualizáciu časovej osi forenzných artefaktov operačného systému Windows.

Ako sme už uviedli na viacerých miestach tejto práce, rozsah spracovanej problematiky je rozsiahly a nebolo možné ho celý spracovať. Existuje viacero možných vetiev, akými je možné danú prácu rozšíriť. Prvou vetvou je analýza ostatných zdrojov digitálnych stôp, ako je operačná pamäť a sieťová komunikácia. Tieto údaje môžu výrazne obohatiť časovú os vytvorené len z forenzných artefaktov získaných zo súborového systému. Druhou vetvou je rozšírenie analýzy aj na iné operačné systémy (Linux, macOS, prípadne Android), resp. vytvorenie modelu, ktorý by v sebe zahŕňal časovú os forenzných artefaktov z rôznych operačných systémov. Posledným spôsobom rozšírenia tejto práce je úprava samotného modelu analýzy časovej osi, a to najmä v nahradení metód formálnej konceptovej analýzy inými metódami, ktoré by identifikovali vzťahy medzi forenznými artefaktami.

Zoznam použitej literatúry

[1] Inglot, B.; Liu, L. Enhanced Timeline Analysis for Digital Forensic Investigations. Inf. Secur. J. Glob. Perspect. 2014, 23, 32–44.

[2] Du, X., Le-Khac, N. A., & Scanlon, M. (2017). Evaluation of digital forensic process models with respect to digital forensics as a service. arXiv preprint arXiv:1708.01730.

[3] Costantini, S., De Gasperis, G., & Olivieri, R. (2019). Digital forensics and investigations meet artificial intelligence. Annals of Mathematics and Artificial Intelligence, 86(1), 193-229

[4] Du, X. (2020). Alleviating the Digital Forensic Backlog: A Methodology for Automated Digital Evidence Processing. School of Computer Science, University College Dublin.

[5] B. Carrier, File System Forensic Analysis, Addison-Wesley, Boston, Massachusetts, 2005.

[6] EUROPOL. Common Taxonomy for Law Enforcement and The National Network of CSIRTs, [online] [21.3.2022]. Dostupné z: https://www.europol.europa.eu/cms/sites/default/files/documents/common_taxonomy_for_law_enforcement_and_csirts_v1.3.pdf

[7] FIRST. CSIRT Services Framework, verzia 2.1, [online] [21.3.2022]. Dostupné z: https://www.first.org/standards/frameworks/csirts/csirt_services_framework_v2.1

[8] Medium. 2022. Introduction to Event Log Analysis Part 1—Windows Forensics Manual 2018. [online] Dostupné z: <https://medium.com/@lucideus/introduction-to-event-log-analysis-part-1-windows-forensics-manual-2018-b936a1a35d8a> [Accessed 30 April 2022].

[9] Fortuna, A., 2022. Windows event logs in forensic analysis. [online] Andrea Fortuna. Dostupné z: <https://www.andreafortuna.org/2017/10/20/windows-event-logs-in-forensic-analysis/> [Pristúpené 30. Marca 2022].

[10] Docs.microsoft.com. 2022. Event Types - Win32 apps. [online] Dostupné z: <https://docs.microsoft.com/en-us/windows/win32/eventlog/event-types> [Pristúpené 30. Marca 2022].

[11] Carvey, H., 2014. Windows forensic analysis toolkit. Amsterdam ; Boston: Syngress.

[12] GitHub. 2022. libevt/Windows Event Log (EVT) format.asciidoc at main · libyal/libevt. [online] Dostupné z:

<https://github.com/libyal/libevt/blob/main/documentation/Windows%20Event%20Log%20(EVT)%20format.asciidoc> [Pristúpené 30. Marca 2022].

[13] Egnyte. 2022. 32949.pdf on Egnyte. [online] Dostupné z: <https://sansorg.egnyte.com/dl/4aUTgETWOe> [Pristúpené 30. Marca 2022].

[14] Schuster, A., 2007. Introducing the Microsoft Vista event log file format. Digital Investigation, 4, pp.65-72.

[15] Carvey, H. (2014). Windows Forensics Analysis Toolkit (4th ed.). Waltham, MA USA: Syngress. dkovar. (2018).

[16] Docs.microsoft.com. 2022. Master File Table (Local File Systems) - Win32 apps. [online] Dostupné z: <https://docs.microsoft.com/en-us/windows/win32/fileio/master-file-table> [Pristúpené 30. Marca 2022].

[17] Pittman, R. D., & Shaver, D. (2010). Windows Forensic Analysis. Handbook of Digital Forensics and Investigation, 209–300. doi:10.1016/b978-0-12-374267-4.00005-7

[18] Carvey, H. (2005). The Windows Registry as a forensic resource. Digital Investigation, 2(3), 201–205. doi:10.1016/j.diin.2005.07.003

[19] Liew, S. P., & Ikeda, S. (2019). Detecting Adversary using Windows Digital Artifacts. 2019 IEEE International Conference on Big Data (Big Data). doi:10.1109/bigdata47090.2019.9006552

[20] Hargreaves, C., & Patterson, J. (2012). An automated timeline reconstruction approach for digital forensic investigations. Digital Investigation, 9, S69-S79.

[21] Kebande, V., Ikuesan, R., Karie, N., Alawadi, S., Choo, K. and Al-Dhaqm, A., 2020. Quantifying the need for supervised machine learning in conducting live forensic analysis of emergent configurations (ECO) in IoT environments. Forensic Science International: Reports, 2, p.100122.

[22] Van Buskirk, E., & Liu, V. T. (2006). Digital Evidence: Challenging the Presumption of Reliability. Journal of Digital Forensic Practice, 1(1), 19–26. doi:10.1080/15567280500541421

[23] Docs.microsoft.com. 2022. [MS-FSCC]: NTFS Attribute Types. [online] Dostupné z: <https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-fscc/a82e9105-2405-4e37-b2c3-28c773902d85> [Pristúpené 30. Marca 2022].

[24] Andreafortuna.org. 2022. [online] Dostupné z: <https://andreafortuna.org/assets/2017/10/0*8m0sfwSuNRk85grp.jpg> [Pristúpené 30. Marca 2022].

[25] Jaringankita.com. 2022. [online] Dostupné z: <http://www.jaringankita.com/blog/wp-content/uploads/2020/05/SANS_macb-1024x730.png> [Pristúpené 30. Marca 2022].

[26] Kazamiya.net. 2022. NTFS Timestamps | Forensicist. [online] Dostupné z: <http://www.kazamiya.net/en/NTFS_Timestamps> [Pristúpené 30. Marca 2022].

[27] Sans.org. 2022. SANS Digital Forensics and Incident Response Blog | Digital Forensics: Detecting time stamp manipulation | SANS Institute. [online] Dostupné z: <https://www.sans.org/blog/digital-forensics-detecting-time-stamp-manipulation/> [Pristúpené 30. Marca 2022].

[28] Fwhibbit.es. 2022. [online] Dostupné z: <https://fwhibbit.es/wp-content/uploads/2018/01/PosterSANS-1024x715.png> [Pristúpené 30. Marca 2022].

[29] Cory Altheide and Harlan Carvey. 2011. Digital forensics with open source tools. Elsevier

[30] Simson L Garfinkel. 2010. Digital forensics research: The next 10 years. Digital Investigation 7 (2010), S64–S73.

[31] Du, X., Hargreaves, C., Sheppard, J., Anda, F., Sayakkara, A., Le-Khac, N. and Scanlon, M., 2020. SoK. Proceedings of the 15th International Conference on Availability, Reliability and Security.

[32] Bhandari, S. and Jusas, V., 2020. The Phases Based Approach for Regeneration of Timeline in Digital Forensics. 2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA),.

[33] DFIR Madness. 2022. Case 001 Super Timeline Analysis - DFIR Madness. [online] Dostupné z: <https://dfirmadness.com/case-001-super-timeline-analysis/> [Pristúpené 30. Marca 2022].

[34] Plaso [online]. [cit. 2021-06-20]. Dostupné z: https://plaso.readthedocs.io/en/latest

[35] Plaso.readthedocs.io. 2022. Using psort.py (Plaso Síar Og Raðar Þessu) — Plaso (log2timeline) 20220428 documentation. [online] Dostupné z: <https://plaso.readthedocs.io/en/latest/sources/user/Using-psort.html> [Pristúpené 30. Marca 2022].

[36] Volatility [online]. [cit. 2021-06-20]. Dostupné z: https://www.volatilityfoundation.org/

[37] Memoryze [online]. [cit. 2021-06-20]. Dostupné z: https://www.fireeye.com/services/ freeware/memoryze.html

[38] MAGNET Process Capture [online]. [cit. 2021-06-20]. Dostupné z: https://www.magnetforensics.com/resources/magnet-process-capture/

[39] FTK Imager [online]. [cit. 2021-06-20]. Dostupné z: https://accessdata.com/products-services/forensic-toolkit-ftk/ftkimager

[40] MAGNET Encrypted Disk Detector [online]. [cit. 2021-06-20]. Dostupné z: https://www.magnetforensics.com/resources/encrypted-disk-detector/

[41] EnCase [online]. [cit. 2021-06-20]. Dostupné z: https://security.opentext.com/

[42] E01 formát [online]. [cit. 2021-06-20]. Dostupné z: https://www.forensicsware.com/blog/e01-file-format.html

[43] Microsoft Backup and restore [online]. [cit. 2021-06-20]. Dostupné z: https://support.microsoft.com/en-us/windows/back-up-and-restore-your-pc-ac359b36-7015-4694-de9a-c5eac1ce9d9c

[44] Forensicswiki.xyz. 2022. Log2timeline - Forensics Wiki. [online] Dostupné z: <https://forensicswiki.xyz/wiki/index.php?title=Log2timeline> [Pristúpené 30. Marca 2022].

[45] Timesketch.org. 2022. timesketch. [online] Dostupné z: <https://timesketch.org/> [Pristúpené 30. Marca 2022].

[46] Debinski, M., Breitinger, F., & Mohan, P. (2019). Timeline2GUI: A Log2Timeline CSV parser and training scenarios. Digital Investigation, 28, 34-43.

[47] Inglot, B., Liu, L.: Enhanced timeline analysis for digital forensic investigations. Information Security Journal: A Global Perspective 23(1-2), 32–44 (2014)

[48] Chabot, Y., Bertaux, A., Nicolle, C., & Kechadi, T. (2015). An ontology-based approach for the reconstruction and analysis of digital incidents timelines. Digital Investigation, 15, 83-100.

[49] Chikul, P., Bahsi, H., & Maennel, O. (2021, June). An Ontology Engineering Case Study for Advanced Digital Forensic Analysis. In International Conference on Model and Data Engineering (pp. 67-74). Springer, Cham.

[50] Bhandari, S., & Jusas, V. (2020). An Ontology Based on the Timeline of Log2timeline and Psort Using Abstraction Approach in Digital Forensics. Symmetry, 12(4), 642.

[51] Bhandari, S., & Jusas, V. (2020). An abstraction based approach for reconstruction of timeline in digital forensics. Symmetry, 12(1), 104.

[52] E. Qawasmeh and M. I. Al-Saleh, "On Producing Events Timeline for Memory Forensics: An Experimental Study," 2020 Seventh International Conference on Information Technology Trends (ITT), 2020, pp. 1-5, doi: 10.1109/ITT51279.2020.9396748.

[53] Ahmad, I., Shah, M. A., Khattak, H. A., Ameer, Z., Khan, M., & Han, K. (2020). FIViz: forensics investigation through visualization for malware in internet of things. Sustainability, 12(18), 7262.

[54] Chikul, P., Bahsi, H., & Maennel, O. (2021, June). An Ontology Engineering Case Study for Advanced Digital Forensic Analysis. In International Conference on Model and Data Engineering (pp. 67-74). Springer, Cham.

[55] Studiawan, H., Sohel, F., & Payne, C. (2020). Sentiment analysis in a forensic timeline with deep learning. IEEE Access, 8, 60664-60675.

[56] Studiawan, H., & Sohel, F. (2021). Anomaly detection in a forensic timeline with deep autoencoders. Journal of Information Security and Applications, 63, 103002.

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

[58] Ramaki, A.A., Rasoolzadegan, A., Bafghi, A.G.: A systematic mapping study on intrusion alert analysis in intrusion detection systems. ACM Comput. Surv. 51, 1–41 (2018).

[59] Ds4n6.io. (2019). DS4N6. [online] Dostupné z: https://www.ds4n6.io/doc/daisy/v0.5/daisy_demo_precook_ram.html [Pristúpené 30. Marca 2022].

[60] DFIR Madness. 2022. DFIR Madness. [online] Dostupné z: <https://dfirmadness.com/> [Pristúpené 30. Marca 2022].

[61] Virustotal.com. 2022. VirusTotal. [online] Dostupné z: <https://www.virustotal.com/gui/file/10f3b92002bb98467334161cf85d0b1730851f9256f83c27db125e9a0c1cfda6/detection> [Pristúpené 30. Marca 2022].

[62] LLC, J., 2022. Automated Malware Analysis Report for coreupdater.exe - Generated by Joe Sandbox. [online] Joesandbox.com. Dostupné z: <https://www.joesandbox.com/analysis/398583/0/html> [Pristúpené 30. Marca 2022].

[63] Kebande, V., Ikuesan, R., Karie, N., Alawadi, S., Choo, K. and Al-Dhaqm, A., 2020. Quantifying the need for supervised machine learning in conducting live forensic analysis of emergent configurations (ECO) in IoT environments. Forensic Science International: Reports, 2, p.100122.

[64] Docs.scipy.org. 2022. Distance computations (scipy.spatial.distance) — SciPy v1.8.0 Manual. [online] Dostupné z: <https://docs.scipy.org/doc/scipy/reference/spatial.distance.html> [Pristúpené 30. Marca 2022].

[65] GitHub. 2022. log2timeline/LOG2TIMELINE.txt at master · kiddinn/log2timeline. [online] Dostupné z: <https://github.com/kiddinn/log2timeline/blob/master/docs/LOG2TIMELINE.txt> [Pristúpené 30. Marca 2022].

[66] Plaso.readthedocs.io. 2022. Parsers — Plaso (log2timeline) 20220428 documentation. [online] Dostupné z: <https://plaso.readthedocs.io/en/latest/sources/user/Parsers-and-plugins.html> [Pristúpené 30. Marca 2022].

[67] Colab.research.google.com. 2022. Google Colaboratory. [online] Dostupné z: <https://colab.research.google.com/?utm_source=scs-index#scrollTo=GJBs_flRovLc> [Pristúpené 30. Marca 2022].

[68] Garbacz, P. and Kutz, O., n.d. Formal ontology in information systems.

[69] Bhandari, S. and Jusas, V. (2020). An Ontology Based on the Timeline of Log2timeline and Psort Using Abstraction Approach in Digital Forensics. Symmetry, 12(4), p.642.

[70] stevewhims (n.d.). Registry Value Types - Win32 apps. [online] docs.microsoft.com. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types [Accessed 30 Apr. 2022].

[71] Christopher Hargreaves and Jonathan Patterson. 2012. An automated timeline reconstruction approach for digital forensic investigations. Digital Investigation 9 (2012), S69–S79.

[72] Stroeh, K., Mauro Madeira, E. and Goldenstein, S., 2013. An approach to the correlation of security events based on machine learning techniques. Journal of Internet Services and Applications, 4(1), p.7.

[73] pandas.pydata.org. (n.d.). pandas.DataFrame.groupby — pandas 1.2.4 documentation. [online] Dostupné z: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.groupby.html.

[74] pandas.pydata.org. (n.d.). pandas.Grouper — pandas 1.3.3 documentation. [online] Dostupné z: https://pandas.pydata.org/docs/reference/api/pandas.Grouper.html.

[75] Waziri, V.O., Umar, A., Olalere, M.: E-fraud forensics investigation techniques with formal concept analysis. International Journal of Cyber-Security and Digital Forensics 3(4), 235–245 (2014)

[76] Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer Science & Business Media (2012)

[77] Tilley, T. (2004). Tool Support for FCA. Lecture Notes in Computer Science, 104–111. doi:10.1007/978-3-540-24651-0_11

[78] conexp.sourceforge.net. (n.d.). The Concept Explorer. [online] Dostupné z: http://conexp.sourceforge.net/index.html [Accessed 30 Apr. 2022].

[79] Ganter, B., Stumme, G. and Wille, R., 2005. Formal concept analysis. Berlin: Springer.

[80] Zamarripa López S., Calle M., Villanueva, 2010. D4.1: State-of-the-art of event correlation and event processing.

[81] pyvis.readthedocs.io. (n.d.). Tutorial — pyvis 0.1.3.1 documentation. [online] Dostupné z: https://pyvis.readthedocs.io/en/latest/tutorial.html [Accessed 30 Apr. 2022].

[82] Perrone, G., Unpingco, J. and Lu, H., 2020. Network visualizations with Pyvis and VisJS. Proceedings of the Python in Science Conference,.

Prílohy

Príloha A: Výstup z nástroja pinfo aplikovaný na obraz disku servera

Príloha B: Zoznam atribútov zdroja FILE

Príloha C: Zoznam atribútov zdroja EVT

Príloha D: Zoznam atribútov zdroja REG

Príloha E: Zdrojový kód úpravy dát zdroja FILE

Príloha F: Zdrojový kód agregácie zdroja FILE

Príloha G: Zdrojový kód vizualizácie

Príloha A

Použitý parser (plugin)	Počet záznamov
amcache	630
bagmru	27
explorer_mountpoints2	3
explorer_programscache	1
filestat	358 396
lnk	510
mft	1 074 764
mrulist_string	1
mrulistex_shell_item_list	2
mrulistex_string	3
mrulistex_string_and_shell_item	3
mrulistex_string_and_shell_item_list	1
msie_webcache	84
msie_zone	36
mstsc_rdp	3
mstsc_rdp_mru	1
networks	4
olecf_automatic_destinations	21
olecf_default	182
olecf_document_summary	8
olecf_summary	70
pe	36 679
recycle_bin	1
setupapi	90
shell_items	330
userassist	24

usnjrnl	82 085
windows_boot_execute	4
windows_run	5
windows_sam_users	2
windows_services	819
windows_shutdown	4
windows_task_cache	160
windows_timezone	2
windows_typed_urls	2
windows_usb_devices	12
windows_version	4
winevtx	172 384
winlogon	6
winreg_default	306 199
Total	**2 033 562**

Príloha B

Názov	Zdroj atribútu	Popis	Typ
Inode	Inode	pôvodný stĺpec inode	integer
Id	pridaný nový atribút	stĺpec s vygenerovaným unikátnym id (pre spätné dohľadanie záznamu)	integer
Datetime	spojený date, time	spojené stĺpce date a time	date
M	M	prvá hodnota zo stĺpca MACB	binary
A	A	druhá hodnota zo stĺpca MACB	binary
C	C	tretia hodnota zo stĺpca MACB	binary
B	B	štvrtá hodnota zo stĺpca MACB	binary
source_file	source	Identifikátor - FILE	binary
file_stat	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
NTFS_file_stat	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
file_entry_shell _item	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
NTFS_USN_ch ange	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
Name	desc	stĺpec z hodnoty vyextrahovaneh z desc	String
Filef	desc	1 ak ide o súbor, 0 ak nie	binary
Directory	desc	1 ak ide o priečinok, 0 ak nie	binary
Link	desc	1 ak ide o linkový súbor, 0 ak nie	binary
dir_appdata	filename	Popis cesty - stĺpec vyextrahovaný na základe hodnôt z desc, 1 ak je v ceste AppData	binary
dir_win	filename	Popis cesty - stĺpec vyextrahovaný na základe hodnôt z desc, 1 ak je v ceste Windows	binary
dir_user	filename	Popis cesty - stĺpec vyextrahovaný na základe hodnôt z desc, 1 ak je v ceste User	binary

dir_other	filename	Popis cesty - stĺpec vyextrahovaný na základe hodnôt z desc, 1 ak to nie je nič z troch vyššie uvedených	binary
file_executable	z filename vyextrahovaná prípona, vznikol stĺpec file_type	Typ súboru podľa prípony - stĺpec vyextrahovaný na základe hodnôt typov súborov z filename	binary
file_graphic	z filename vyextrahovaná prípona, vznikol stĺpec file_type	Typ súboru - stĺpec vyextrahovaný na základe hodnôt typov súborov z filename	binary
file_document	z filename vyextrahovaná prípona, vznikol stĺpec file_type	Typ súboru - stĺpec vyextrahovaný na základe hodnôt typov súborov z filename	binary
file_ps	z filename vyextrahovaná prípona, vznikol stĺpec file_type	Typ súboru - stĺpec vyextrahovaný na základe hodnôt typov súborov z filename (powershell file)	binary
file_other	z filename vyextrahovaná prípona, vznikol stĺpec file_type	Typ súboru - zvyšok	binary
mft	format	stĺpec vytvorený na základe hodnôt stĺpca format	binary
lnk_shell_items	format	stĺpec vytvorený na základe hodnôt stĺpca format	binary
olecf_olecf_automatic_destinations/lnk/shell_items	format	stĺpec vytvorený na základe hodnôt stĺpca format	binary
winreg_bagmru/shell_items	format	stĺpec vytvorený na základe hodnôt stĺpca format	binary
usnjrnl	format	stĺpec vytvorený na základe hodnôt stĺpca format	binary
is_allocated	extra	stĺpec z vyextrahovanej hodnoty zo stĺpca extra (1 ak sa nachádza informácia)	binary
is_allocated1	extra	1 ak hodnota v is_allocated stĺpci bola jedna (súbor sa nachádza v súborovom systéme)	binary
is_allocated0	extra	1 ak hodnota v is_allocated stĺpci bola nula (súbor bol vymazaný	binary
size_none	extra	1 ak uvedená veľkosť bola 0 (ak je vymazaný súbor -> 0 veľkosť)	binary
size_Q1	extra	1 ak veľkosť súboru padla do 1. kvartilu	binary
size_Q2	extra	1 ak veľkosť súboru padla do 2. kvartilu	binary
size_Q3	extra	1 ak veľkosť súboru padla do 3. kvartilu	binary
size_Q4	extra	1 ak veľkosť súboru padla do 4. kvartilu	binary
sha_256	extra	Haš súboru v sha_256	string

Príloha C

Názov	Zdroj	Popis	Typ
inode	inode	pôvodný inode stĺpec, označenie Inodu v rámci súborového systému NTFS	integer
id	novovytvorený	jedinečne vygenerované unikátne id	integer
datetime	date + time	časová pečiatka, spojené stĺpce date a time, časové pásmo UTC	datetime
source_evt	source	pôvodný stĺpec source, kde boli len hodnoty EVT	binary
WinEVTX	sourcetype	pôvodný stĺpec sourcetype, všade hodnoty WinEVTX	binary
computer_name	desc	vytiahnuté z desc, resp. stĺpca extra	string
source_name	desc	vytiahnuté z desc, resp. stĺpca extra - zdroj logov (napr. Service Control Manager)	string
filename_security	filename	vytiahnuté zo stĺpca filename, log v rámci Security.evtx	binary
filename_application	filename	vytiahnuté zo stĺpca filename, log v rámci Application.evtx	binary
filename_system	filename	vytiahnuté zo stĺpca filename, log v rámci System.evtx	binary
filename_rdp	filename	vytiahnuté zo stĺpca filename, log v rámci Microsoft-Windows-RemoteDesktopServices-RdpCoreTS%4Operational.evtx	binary
filename_powershell	Filename	vytiahnuté zo stĺpca filename, log v rámci Microsoft-Windows-PowerShell%4Operational.evtx	binary
filename_other	filename	vytiahnuté zo stĺpca filename, log sa nevyskytoval ani v jednej z vyššie spomenutých skupín pri filename	binary
recovered	extra	vytiahnuté zo stĺpca extra, True ak bol záznam obnovený (recovered data = deleted data)	binary
user_sid	extra	vytiahnuté zo stĺpca extra, identifikácia používateľa, ktorý vytvoril proces	string
execution_process_id	extra	vytiahnuté zo stĺpca extra, ID vykonávaného procesu	integer

channel	extra	vytiahnuté zo stĺpca extra, napr. System	string
event_type_account_logon	extra	rozdelenie na základe EventID	binary
event_type_account_management	extra	rozdelenie na základe EventID	binary
event_type_detailed_tracking	extra	rozdelenie na základe EventID	binary
event_type_logon_logoff	extra	rozdelenie na základe EventID	binary
event_type_ds_access	extra	rozdelenie na základe EventID	binary
event_type_object_access	extra	rozdelenie na základe EventID	binary
event_type_policy_change	extra	rozdelenie na základe EventID	binary
event_type_privilege_usage	extra	rozdelenie na základe EventID	binary
event_type_system	extra	rozdelenie na základe EventID	binary
event_type_other	extra	rozdelenie na základe EventID	binary
task	extra	pracuje s identifikátor úlohy pre časť aplikácie alebo komponentu, ktorá publikuje udalosť.	integer
keywords_audit_failure	extra	AuditFailure = 0x10000000000000L	binary
keywords_audit_success	extra	AuditSuccess = 0x20000000000000L	binary
keywords_correlation_hint	extra	CorrelationHint = 0x10000000000000L	binary
keywords_event_log_classic	extra	EventLogClassic = 0x80000000000000L	binary
keywords_sqm	extra	Sqm = 0x08000000000000L	binary
keywords_wdi_context	extra	WdiContext = 0x02000000000000L	binary
keywords_wdi_diagnostic	extra	WdiDiagnostic = 0x04000000000000L	binary
record_number	extra	jednoznačný identifikátor	integer

Príloha D

Názov	Zdroj	Popis	Typ
Inode	inode	Hodnota pôvodného stĺpca inode	integer
M	MACB	prvá hodnota zo stĺpca MACB	binary
A	MACB	druhá hodnota zo stĺpca MACB	binary
C	MACB	tretia hodnota zo stĺpca MACB	binary
B	MACB	štvrtá hodnota zo stĺpca MACB	binary
source_reg	source	pôvodný stĺpec source, kde boli len hodnoty REG	binary
registry_key	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
AppCompatCache_Registry_Entry	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_Service	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Task_Cache	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
Registry_Key_Typed_URLs	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_Winlogon	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_KeyRun_Key	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
Background_Activity_Moderator_Registry_Entry	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_User_Account_Information	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_UserAssist	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary

Registry_Key_BagMRU	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_MRUListEx	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_MRUList	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype	binary
Registry_Key_Network Drive	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_Shutdown_Entry	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
Registry_Key_USB_Entries	sourcetype	stĺpec vytvorený na základe hodnôt zo stĺpca sourcetype,	binary
valueName	desc	zaznamenaný kľúč registra	string
REG_BINARY	desc	atribút podľa typu hodnoty registra	binary
REG_DWORD	desc	atribút podľa typu hodnoty registra	binary
REG_DWORD_LITTLE_ENDIAN	desc	atribút podľa typu hodnoty registra	binary
REG_DWORD_BIG_ENDIAN	desc	atribút podľa typu hodnoty registra	binary
REG_EXPAND_SZ	desc	atribút podľa typu hodnoty registra	binary
REG_LINK	desc	atribút podľa typu hodnoty registra	binary
REG_MULTI_SZ	desc	atribút podľa typu hodnoty registra	binary
REG_NONE	desc	atribút podľa typu hodnoty registra	binary
REG_QWORD	desc	atribút podľa typu hodnoty registra	binary
REG_QWORD_LITTLE_ENDIAN	desc	atribút podľa typu hodnoty registra	binary
REG_SZ	desc	atribút podľa typu hodnoty registra	binary
dir_win	filename	Windows hive	binary
dir_user	filename	User hive	binary
dir_other	filename	Hive, nie user ani windows	binary

winreg/winreg_default	format	jedinečná hodnota z atribútu format	binary
winreg/appcompatcache	format	jedinečná hodnota z atribútu format	binary
winreg/windows_services	format	jedinečná hodnota z atribútu format	binary
winreg/windows_task_cache	format	jedinečná hodnota z atribútu format	binary
winreg/windows_typed_urls	format	jedinečná hodnota z atribútu format	binary
winreg/winlogon	format	jedinečná hodnota z atribútu format	binary
winreg/msie_zone	format	jedinečná hodnota z atribútu format	binary
winreg/windows_run	format	jedinečná hodnota z atribútu format	binary
winreg/amcache	format	jedinečná hodnota z atribútu format	binary
winreg/bam	format	jedinečná hodnota z atribútu format	binary
winreg/windows_sam_users	format	jedinečná hodnota z atribútu format	binary
winreg/userassist	format	jedinečná hodnota z atribútu format	binary
winreg/explorer_mountpoints2	format	jedinečná hodnota z atribútu format	binary
winreg/explorer_programscache	format	jedinečná hodnota z atribútu format	binary
winreg/bagmru	format	jedinečná hodnota z atribútu format	binary
winreg/mrulistex_string_and_shell_item	format	jedinečná hodnota z atribútu format	binary
winreg/mrulistex_string	format	jedinečná hodnota z atribútu format	binary
winreg/windows_timezone	format	jedinečná hodnota z atribútu format	binary
winreg/mrulist_string	format	jedinečná hodnota z atribútu format	binary
winreg/network_drives	format	jedinečná hodnota z atribútu format	binary
winreg/mrulistex_shell_item_list	format	jedinečná hodnota z atribútu format	binary
winreg/mrulistex_string_and_shell_item_list	format	jedinečná hodnota z atribútu format	binary
winreg/windows_shutdown	format	jedinečná hodnota z atribútu format	binary

winreg/windows_usb_devices	format	jedinečná hodnota z atribútu format	binary
winreg/windows_version	format	jedinečná hodnota z atribútu format	binary
winreg/windows_boot_execute	format	jedinečná hodnota z atribútu format	binary
sha_256	extra	Sha_256 haš	string

Príloha E

```
# -*- coding: utf-8 -*-
"""FILE.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1AlUJk4DTBIrrPPqKeSadC0MoHeX1j
91V

# Úvodné záležitosti, importy, náhľad na dataframe
"""

import pandas as pd
from itertools import product
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

df = pd.read_csv (r'/content/drive/MyDrive/dc01-super-timeline.csv',
error_bad_lines=False, sep= ',')
print(df.shape)
df.drop(df.index[df['date'] == '00/00/0000'], inplace=True)

"""# Rozdelenie df na základe source stĺpca"""

result = df.copy()
columnNames = ["date", "time", "timezone", "MACB", "source",
"sourcetype", "type", "user", "host", "short", "desc", "version",
"filename", "inode", "notes", "format", "extra"]
df_amcache = pd.DataFrame(columns = columnNames)
df_amcacheprogram = pd.DataFrame(columns = columnNames)
df_evt = pd.DataFrame(columns = columnNames)
df_file = pd.DataFrame(columns = columnNames)
df_lnk = pd.DataFrame(columns = columnNames)
df_log = pd.DataFrame(columns = columnNames)
df_olecf = pd.DataFrame(columns = columnNames)
df_pe = pd.DataFrame(columns = columnNames)
df_recbin = pd.DataFrame(columns = columnNames)
df_reg = pd.DataFrame(columns = columnNames)
df_webhist = pd.DataFrame(columns = columnNames)

df_amcache = result.loc[result['source'] == 'AMCACHE']
df_amcacheprogram = result.loc[result['source'] == 'AMCACHEPROGRAM']
df_evt = result.loc[result['source'] == 'EVT']
df_file = result.loc[result['source'] == 'FILE']
```

```
df_lnk = result.loc[result['source'] == 'LNK']
df_log = result.loc[result['source'] == 'LOG']
df_olecf = result.loc[result['source'] == 'OLECF']
df_pe = result.loc[result['source'] == 'PE']
df_recbin = result.loc[result['source'] == 'RECBIN']
df_reg = result.loc[result['source'] == 'REG']
df_webhist = result.loc[result['source'] == 'WEBHIST']

"""# Nové stĺpce

## Úvodné záležitosti - nastavenie stĺpcov, importy
"""

from numpy.lib import type_check
import pandas as pd
import re
import os
import uuid

df_file['id'] = [uuid.uuid4() for _ in range(len(df_file.index))]

df_file['datetime'] = pd.to_datetime(df_file['date'] + ' ' +
df_file['time'])


new_columns = {"M":[], "A":[], "C":[], "B":[], "M1":[], "A1":[],
"C1":[], "B1":[], "File stat":[], "NTFS file stat":[], "File entry shell
item":[], "NTFS USN change":[], "filestat":[], "mft":[],
"lnk/shell_items":[],
"olecf/olecf_automatic_destinations/lnk/shell_items":[],"winreg/bagmru
/shell_items":[], "usnjrnl":[], "file_path":[],'is_allocated':[],
'is_allocated0':[], 'is_allocated1':[], 'file_size':[], 'name':[],
'shell_path':[], 'type':[], 'file':[], 'directory':[], 'link':[],
'dir_type':[],
"dir_appdata":[], "dir_win":[], "dir_user":[], "dir_other":[],
"source_file":[], "size_Q1":[], "size_Q2":[], "size_Q3":[],
"size_Q4":[], "size_none":[], 'sha_256':[], 'file_type':[],
'source_file':[]}

"""## MACB """

for item in df_file.MACB:
    list_of_letters = list(item)
    new_columns['M1'].append(list_of_letters[0])
    new_columns['A1'].append(list_of_letters[1])
    new_columns['C1'].append(list_of_letters[2])
    new_columns['B1'].append(list_of_letters[3])

df_file['M1'] = new_columns['M1']
df_file['A1'] = new_columns['A1']
df_file['C1'] = new_columns['C1']
```

```
df_file['B1'] = new_columns['B1']

df_file['M']=["1" if (x=="M") else "0" for x in df_file.M1]
df_file['A']=["1" if (x=="A") else "0" for x in df_file.A1]
df_file['C']=["1" if (x=="C") else "0" for x in df_file.C1]
df_file['B']=["1" if (x=="B") else "0" for x in df_file.B1]

df_file = df_file.drop(['M1', 'A1', 'B1', 'C1', 'MACB'], axis = 1)

"""## Source file ako atribút """

for item in df_file.source:
  if item:
    new_columns['source_file'].append('1')

df_file['source_file'] = new_columns['source_file']

"""## Sourcetype """

df_file['file_stat']=["1" if (x == 'File stat') else "0" for x in
df_file.sourcetype]
df_file['NTFS_file_stat']=["1" if (x=='NTFS file stat') else "0" for x
in df_file.sourcetype]
df_file['file_entry_shell_item']=["1" if (x=='File entry shell item')
else "0" for x in df_file.sourcetype]
df_file['NTFS_USN_change']=["1" if (x=='NTFS USN change') else "0" for
x in df_file.sourcetype]

"""## Desc - name """

for item in df_file.desc:
    division = item.split(' ')
    file_path = division[0]
    if file_path is not None:
      final_type = file_path.split('\\')
      if len(final_type) > 2:
        filef = final_type[len(final_type)-1]
        new_columns['file_path'].append(filef)
      else:
        new_columns['file_path'].append(file_path)
    else:
      new_columns['file_path'].append('')

df_file['file_path'] = new_columns['file_path']
print(df_file['file_path'])


for item in df_file.desc:
    name_field = re.compile(r'Name:\s*(.*)')
```

```
    name = name_field.search(item)              # Run a regex search
anywhere inside a string
    #print(final_name)
    if name:
      final_name = name.group(0).split(' ')[1:2]
      new_columns['name'].append(str(final_name))
    else:
      new_columns['name'].append(np.nan)


df_file['name'] = new_columns['name']
df_file['name'] = df_file['name'].str.replace('[', '')
df_file['name'] = df_file['name'].str.replace(']', '')
df_file['name'] = df_file['name'].str.replace('\'', '')


df_file['name'] = df_file['name'].fillna(df_file['file_path'])

"""## Desc - type"""

for item in df_file.desc:
    type_field = re.compile(r'Type:\s*(.*)')
    typef = type_field.search(item)
    if typef:
      final_type = typef.group(0).split(' ')[1:2]
      new_columns['type'].append(final_type)
    else:
      new_columns['type'].append('None')

df_file['typef'] = new_columns['type']

df_file['filef']=["1" if (x[0] == "file") else "0" for x in
df_file.typef]
df_file['directory']=["1" if (x[0]=="directory") else "0" for x in
df_file.typef]
df_file['link']=["1" if (x[0]=="link") else "0" for x in df_file.typef]

"""## Filename - dir_type



"""

for item in df_file.filename:
    final_type = item.split('\\')
    if 'AppData' in final_type:
      new_columns['dir_type'].append('AppData')
    elif 'AppData' not in item and 'Windows' in item:
      new_columns['dir_type'].append('Windows')
```

```
    elif 'Users' in item and 'AppData' not in item:
      new_columns['dir_type'].append('Users')
    else:
      new_columns['dir_type'].append('Other')

df_file['dir_type'] = new_columns['dir_type']

df_file['dir_appdata']=["1" if (x == "AppData") else "0" for x in
df_file.dir_type]
df_file['dir_win']=["1" if (x=="Windows") else "0" for x in
df_file.dir_type]
df_file['dir_user']=["1" if (x=="Users") else "0" for x in
df_file.dir_type]
df_file['dir_other']=["1" if (x=="Other") else "0" for x in
df_file.dir_type]

"""## file_type """

for item in df_file.name:
  ext = item.split('.')
  #print(ext)
  if len(ext) > 1:
    #print(ext[len(ext)-1])
    new_columns['file_type'].append(ext[len(ext)-1])
  else:
    new_columns['file_type'].append('')


df_file['file_type'] = new_columns['file_type']
print(df_file['file_type'])

df_file['file_executable']=["1" if (x == "exe" or x == "msi" or x ==
"cmo" or x == "bud") else "0" for x in df_file.file_type]
df_file['file_graphic']=["1" if (x == "png" or x == "jpg" or x == "psd"
or x == "jpeg") else "0" for x in df_file.file_type]
df_file['file_documents']=["1" if (x == "doc" or x == "docx" or x
=="docm" or x =="docm" or x =="ppt" or x =="pptx" or x =="pps" or x
=="ppsx" or x =="txt" or x =="xls" or x =="xlsx") else "0" for x in
df_file.file_type]
df_file['file_ps']=["1" if (x == "ps") else "0" for x in
df_file.file_type]
df_file['file_other']=["1" if (x != "ps" and x != "msi" and x != "cmo"
and x != "bud" and x != "exe" and x != "png" and x != "jpg" and x !=
"psd" and x != "jpeg" and x != "doc" and x != "docx" and x != "docm" and
x != "docm" and x != "ppt" and x != "pptx" and x != "pps" and x != "ppsx"
and x != "txt" and x != "xls" and x != "xlsx") else "0" for x in
df_file.file_type]

"""## Format """

for item in df_file.format:
    if item == 'mft':
```

```
      #print('mft')
      new_columns['mft'].append("1")
    else:
      new_columns['mft'].append("0")

    if item == 'lnk/shell_items':
      new_columns['lnk/shell_items'].append("1")
      #print('lnk')
    else:
      new_columns['lnk/shell_items'].append("0")

    if item == 'olecf/olecf_automatic_destinations/lnk/shell_items':

new_columns['olecf/olecf_automatic_destinations/lnk/shell_items'].appe
nd("1")
      #print('olecf')
    else:

new_columns['olecf/olecf_automatic_destinations/lnk/shell_items'].appe
nd("0")

    if item == 'winreg/bagmru/shell_items':
      new_columns['winreg/bagmru/shell_items'].append("1")
      #print('winreg')
    else:
      new_columns['winreg/bagmru/shell_items'].append("0")

    if item == 'usnjrnl':
      new_columns['usnjrnl'].append("1")
      #print('usn')
    else:
      new_columns['usnjrnl'].append("0")

    if item == 'filestat':
      #print('file')
      new_columns['filestat'].append("1")
    else:
      new_columns['filestat'].append("0")

#df_file['filestat'] = new_columns['filestat']
df_file['mft'] = new_columns['mft']
df_file['lnk_shell_items'] = new_columns['lnk/shell_items']
df_file['olecf_olecf_automatic_destinations/lnk/shell_items']         =
new_columns['olecf/olecf_automatic_destinations/lnk/shell_items']
df_file['winreg_bagmru/shell_items']                                  =
new_columns['winreg/bagmru/shell_items']
df_file['usnjrnl'] = new_columns['usnjrnl']

"""## Extra - is_allocated """
```

```
from pandas.core.arrays.categorical import contains
for item in df_file.extra:
    is_allocated_field = re.compile(r'is_allocated:(.*?);')
    is_allocated = is_allocated_field.search(item)
    if is_allocated:
      new_columns['is_allocated'].append(is_allocated.group(1))
    else:
      new_columns['is_allocated'].append('None')


df_file['is_allocated'] = new_columns['is_allocated']
df_file['is_allocated1']=["1" if ('True' in x) else "0" for x in
df_file.is_allocated]
df_file['is_allocated0']=["1" if ('False' in x) else "0" for x in
df_file.is_allocated]

df_file.is_allocated1

"""## Extra - file_size"""

for item in df_file.extra:
    file_size_field = re.compile(r'file_size:(.*?);')
    file_size = file_size_field.search(item)
    if file_size:
      new_columns['file_size'].append(int(file_size.group(1)))
    else:
      new_columns['file_size'].append(0)

df_file['file_size'] = new_columns['file_size']
max_filesize = int(df_file.file_size.max())

nenulove = []
for item in df_file.file_size:
    if item != 0 or item == None:
      nenulove.append(item)

nenulove.sort()

Q1 = np.quantile(nenulove, .25)
Q2 = np.quantile(nenulove, .50)
Q3 = np.quantile(nenulove, .75)

for item in df_file.file_size:
    if int(item) == 0:
      new_columns['size_none'].append('1')
    else:
      new_columns['size_none'].append('0')

    if  int(item) > 0 and int(item) <= Q1:
```

```
      new_columns['size_Q1'].append('1')
    else:
      new_columns['size_Q1'].append('0')

    if int(item) > Q1 and int(item) <= Q2:
      new_columns['size_Q2'].append('1')
    else:
       new_columns['size_Q2'].append('0')

    if int(item) > Q2 and int(item) <= Q3:
      new_columns['size_Q3'].append('1')
    else:
      new_columns['size_Q3'].append('0')

    if int(item) > Q3:
      new_columns['size_Q4'].append('1')
    else:
      new_columns['size_Q4'].append('0')

df_file['size_none'] = new_columns['size_none']
df_file['size_Q1'] = new_columns['size_Q1']
df_file['size_Q2'] = new_columns['size_Q2']
df_file['size_Q3'] = new_columns['size_Q3']
df_file['size_Q4'] = new_columns['size_Q4']

print(Q1)
print(Q2)
print(Q3)
print(max_filesize)

sucet = 0
for item in df_file['file_size']:
    if item == 0:
      sucet = sucet + 1
print("Počet nulových záznamov je:"  + str(sucet))

"""## Sha256"""

for item in df_file.extra:
    sha_256_field = re.compile(r'sha256_hash:\s*(.*)')
    sha_256 = sha_256_field.search(item)
    if sha_256:
      f_sha256 = sha_256.group(0)
      final_sha =f_sha256.split(':')
      new_columns['sha_256'].append(final_sha[1])
    else:
      new_columns['sha_256'].append('')

df_file['sha_256'] = new_columns['sha_256']
```

```
import sys
uniqueValues = df_file['sha_256'].unique()
with open('unique_hashes.txt', 'a') as f:
  print(str(uniqueValues), file=f)


uniqueValues = df_file['sha_256'].unique()
print(str(uniqueValues))

pd.set_option('display.max_colwidth', None)
df_file['filename'].sample(5)

df_file['desc'].sample(5)

df_file.file_type

"""## Záverečné záležitosti - odstránenie nepotrebných stĺpcov a
vytvorenie nového df"""

df_file.to_csv("dc_file_old.csv")

df_file = df_file.drop(['date', 'time', 'timezone', 'sourcetype',
'short', 'desc', 'user', 'host', 'notes', 'version', 'type', 'source',
'format', 'extra', 'typef', 'dir_type', 'file_type', 'file_size',
'filename', 'file_path', 'is_allocated'], axis = 1)

df_file.to_csv("dc_file_new.csv")

"""## Agregácia"""

#from google.colab import drive
#import pandas as pd
#from itertools import product
#import numpy as np
#drive.mount('/content/drive')
#df_file                          =                          pd.read_csv
(r'/content/drive/MyDrive/diplomka_final/dc_file_new.csv',
error_bad_lines=False, sep= ',')

#df_file.reset_index(drop=True, inplace=True)
#print(df_file)

count_series                                                          =
df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'),
'inode']).size()
new_df = count_series.to_frame(name = 'size').reset_index()
agg_inode                                                             =
df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'),
'inode']).agg(lambda x: set(x))
agg_inode['counts'] = new_df['size'].values
```

```
agg_inode.to_csv("agregovane_df_file_inode.csv")
print(agg_inode.shape)

count_series = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'),'name']).size()
new_df = count_series.to_frame(name = 'size').reset_index()
agg_name = df_file.set_index('datetime').groupby([pd.Grouper(freq='15s'), 'name']).agg(lambda x: set(x))
agg_name['counts'] = new_df['size'].values
agg_name.to_csv("agregovane_df_file_name.csv")
print(agg_name.shape)


"""## Po agregácii"""

import pandas as pd
from itertools import product
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

df_agg_name = pd.read_csv(r'/content/drive/MyDrive/agregovane_df_file_name.csv', error_bad_lines=False, sep= ',')

"""# Odstránenie zátvoriek a množín pri binárnych dátach"""

df_agg_inode = df_agg_inode.rename(columns = {'name': 'nazov'}, inplace = False)
df_agg_name = df_agg_name.rename(columns = {'name': 'nazov'}, inplace = False)

uniqueValues = df_agg_name['name'].unique()
print(len(uniqueValues))

from datetime import datetime
columns = ['M', 'A', 'C', 'B', 'source_file', 'file_stat', 'NTFS_file_stat', 'file_entry_shell_item', 'NTFS_USN_change', 'filef', 'directory', 'link', 'dir_appdata', 'dir_win', 'dir_user', 'dir_other', 'file_executable', 'file_graphic', 'file_documents', 'file_ps', 'file_other', 'mft', 'lnk_shell_items', 'olecf_olecf_automatic_destinations/lnk/shell_items', 'winreg_bagmru/shell_items', 'usnjrnl', 'is_allocated1', 'is_allocated0', 'size_none', 'size_Q1', 'size_Q2', 'size_Q3', 'size_Q4']

for id_inode,row_inode in df_agg_inode.iterrows():
  value = row_inode.inode
  time_inode = row_inode.datetime
```

```
  time_inode_obj = datetime.strptime(time_inode, '%Y-%m-%d %H:%M:%S')
  for id_name,row_name in df_agg_name.iterrows():
    set_inodes = row_name.inode
    if str(value) in set_inodes:
      time_name = row_name.datetime
      time_name_obj = datetime.strptime(time_name, '%Y-%m-%d %H:%M:%S')
      time_diff = time_inode_obj - time_name_obj
      if time_diff.seconds <= 15 and time_diff.seconds >= -15:
        name_inode = str(row_inode.nazov)[2:-2]
        name_name = str(row_name.nazov)
        df_agg_name.at[id_name, 'nazov'] = name_name + "," + name_inode

df_agg_name.to_csv("agg.csv")
df_agg_name.to_csv("/content/drive/MyDrive/agg.csv", index=False)

"""# Vzťahy záznamy

Priradenie UID každému meta-záznamu
"""

import uuid

unique_inodes = df_agg['inode'].unique()
print(unique_inodes)
df_agg['uid_agg'] = [uuid.uuid4() for _ in range(len(df_agg.index))]
df_agg.to_csv('agregovane_metazaznamy_file.csv')

import pandas as pd
from itertools import product
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

df_agg                          =                          pd.read_csv
(r'/content/drive/MyDrive/dc_final/agregovane_metazaznamy_file.csv',
error_bad_lines=False, sep= ',')

my_file = open("coreupdate_1.txt", "r")
content = my_file.read()
content_list = content.split(",")
my_file.close()

unique = set(content_list)
for id in unique:
  list_relations_inode = list()
  file1 = open("file_relations_id_1.txt", "a")
  for value in df_agg[df_agg['id'].str.contains(id)].uid_agg:
    list_relations_inode.append(str(value))
```

```
    setik = set(list_relations_inode)
  file1.write(str(setik) + ',')
file1.close()

my_file = open("coreupdate_2.txt", "r")
content = my_file.read()
content_list = content.split(",")
my_file.close()


unique = set(content_list)
for id in unique:
  list_relations_inode = list()
  file1 = open("file_relations_id_12.txt", "a")
  for value in df_agg[df_agg['id'].str.contains(id)].uid_agg:
    list_relations_inode.append(str(value))
    setik = set(list_relations_inode)
  file1.write(str(setik) + ',')
file1.close()


import pandas as pd
from itertools import product
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

df_agg                          =                          pd.read_csv
(r'/content/drive/MyDrive/dc_final/agregovane_metazaznamy_file.csv',
error_bad_lines=False, sep= ',')
unique_inodes = df_agg['inode'].unique()

for inode in unique_inodes:
  #print(inode)
  list_relations_inode = list()
  list_relations_inode.append(inode)
  file1 = open("file_relations_inode.txt", "a")
  #list_relations_inode.append(str(df_agg.loc[df_agg['inode']       ==
inode].shape[0]))
  for id in df_agg.loc[df_agg['inode'] == inode].uid_agg:
    list_relations_inode.append(id)
  file1.write(str(list_relations_inode) + '\n')
file1.close()


import regex as re
from itertools import product
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')

df_agg                              =                              pd.read_csv
(r'/content/drive/MyDrive/dc_final/agregovane_metazaznamy_file.csv',
error_bad_lines=False, sep= ',')

df_agg_name['name'] = df_agg_name['name'].str.replace('\\','\\\\')

unique_names = df_agg_name['name'].unique()

for name in unique_names:
  list_relations_name = list()
  list_relations_name.append(name)
  file1 = open("file_relations_name.txt", "a")
  for id in df_agg[df_agg['name'].str.contains(name)].uid_agg:
    list_relations_name.append(id)
  file1.write(str(list_relations_name) + '\n')
file1.close()

import uuid

unique_inodes = df_agg['inode'].unique()
df_agg['uid_agg'] = [uuid.uuid4() for _ in range(len(df_agg.index))]

from datetime import datetime
columns   =   ['M',   'A',   'C',   'B',   'source_file',   'file_stat',
'NTFS_file_stat', 'file_entry_shell_item', 'NTFS_USN_change', 'filef',
'directory', 'link', 'dir_appdata', 'dir_win', 'dir_user', 'dir_other',
'file_executable',    'file_graphic',    'file_documents',    'file_ps',
'file_other',                 'mft',                'lnk_shell_items',
'olecf_olecf_automatic_destinations/lnk/shell_items',
'winreg_bagmru/shell_items',        'usnjrnl',        'is_allocated1',
'is_allocated0',   'size_none',   'size_Q1',   'size_Q2',   'size_Q3',
'size_Q4']

unique_inodes = df_agg['inode'].unique()

for inode in unique_inodes:
  list_relations_inode = list()
  list_relations_inode.append(inode)
  file1 = open("file_relations_inode.txt", "a")
  for row_name in df_agg.itertuples():
    agg_inodes = row_name.inode
    uid_row = row_name.uid_agg
    if inode== agg_inodes:
      list_relations_inode.append(uid_row)
  file1.write(str(list_relations_inode) + '\n')
file1.close()
```

Príloha F

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# importujeme kniznicu pre pracu s dataframemami
import pandas as pd
# importujeme kniznicu pre pracu s datovymi formatmi
from datetime import datetime as dt
# importujeme datovy typ uuid pre spracuvavanie
from uuid import UUID
# importujeme kniznicu pre serializaciu a ulozenie struktur s datami pre
neskorie spracovanie
import pickle
# typova struktura pre dokumentaciu
from typing import List, Union

# vypnutie vypisu neopodstatnenych warningov
pd.options.mode.chained_assignment = None  # default='warn'

# nacitanie dataframov
df_name = pd.read_csv('agregovane_df_file_name_upravene.csv')
df_inode = pd.read_csv('agregovane_df_file_inode.csv')

# pociatocna filtracia a uprava dat
max_inode_search = df_inode.inode.max()
min_inode_search = df_inode.inode.min()
df_name = df_name.drop(df_name.columns[0], axis=1)
#df_inode = df_inode.drop(df_inode.columns[0], axis=1)
df_name = df_name.query("(inode <= @max_inode_search) & (inode >=
@min_inode_search)")
df_inode = df_inode.query("(inode <= @max_inode_search) & (inode >=
@min_inode_search)")
df_inode = df_inode.sort_values(by='inode', ascending=False)

# zoznam ustaleneho poradia stlpcov pre pohodlnejsie spracovanie
string_column_assortment = [ 'id', 'M', 'A', 'C', 'B', 'source_file',
       'file_stat', 'NTFS_file_stat', 'file_entry_shell_item',
       'NTFS_USN_change', 'filef', 'directory', 'link', 'dir_appdata',
       'dir_win',    'dir_user',    'dir_other',    'file_executable',
'file_graphic',
       'file_documents',      'file_ps',      'file_other',      'mft',
'lnk_shell_items',
       'olecf_olecf_automatic_destinations/lnk/shell_items',
       'winreg_bagmru/shell_items', 'usnjrnl', 'is_allocated1',
       'is_allocated0', 'size_none', 'size_Q1', 'size_Q2', 'size_Q3',
       'size_Q4', 'sha_256', 'name', 'counts', 'inode','datetime']

def     dataframe_union_to_series(df:pd.DataFrame,     row:List[str])->
List[str]:
```

```
    """
    Pomocna metoda na zlucenie riadkov dataframu df podla danych
kriterii.

    Parameters
    ----------
    df : pd.DataFrame
        Povodne data na zlucenie.
    row : List[str]
        Vysledny riadok zlucenych udajov z df podla danych kriterii.

    Returns
    -------
    List[str]
        Zluceny riadok dat z df.

    """
    # prechadzame riadkami a zlucujeme (mnozinove zjednotenie buniek)
zaznamy
    row_traverser_int = 0
    for col_string in string_column_assortment[0:-3]:
        for cell_set in df[col_string]:
            row[row_traverser_int]                                    =
str(eval(row[row_traverser_int]).union(eval(cell_set)))
        row_traverser_int = row_traverser_int + 1

    row[-3] = row[-3] + df.iloc[:,-3].sum()
    return row

def agregacia(row: List[str]) -> Union[str, List[str]]:
    """
    Pomocna metoda na upravu vstupnych dataframov podla danych
filtracnych
    podmienok (15 sekundovy casovy rozdiel + zhodnost inodov).

    Parameters
    ----------
    row : List[str]
        Vstupny zaznam dataframu na zlucenie a upravu.

    Returns
    -------
    Union[str, List[str]]
        Vracia "empty" ak filtracii nevyhovuju ziadne zaznamy
        vstupneho dataframu alebo zaznam v podobe pola retazcov
        pozostavajuci zo zlucenych podobnych zaznamov vstupneho
dataframu.

    """
    inode_querried = row[-2]
```

```
    inode_querried_datetime = dt.fromisoformat(row[-1])

    # filtrujeme podla cisla inodu
    df_name_querried = df_name.query("(inode == @inode_querried)")
    if df_name_querried.shape[0] == 0:
        return "empty"

    # pomocna funkcia pre filtrovanie podla 15 sekundoveho okna
    def datetime_rozdiel_lambda(querried_row):
        compared_datetime = dt.fromisoformat(querried_row['datetime'])
        diff = compared_datetime - inode_querried_datetime
        return abs(diff.total_seconds())

    # filtrovanie podla okna pomocou lambda vyrazu
    df_name_querried["casovy_rozdiel"]                                    =
df_name_querried.apply(datetime_rozdiel_lambda, axis=1)
    df_name_querried   =   df_name_querried.query("(casovy_rozdiel   <=
15.0)")
    if df_name_querried.shape[0] == 0:
        return "empty"
    df_name_querried    =    df_name_querried.drop(["casovy_rozdiel"],
axis=1)
    #df_name_querried["name"]            =            "{'"            +
df_name_querried["name"].astype("string") + "'}"

    # vhodne usporiadamie stlpce pre dalsie spracovanie
    df_name_querried = df_name_querried[string_column_assortment]
    return dataframe_union_to_series(df_name_querried, row)

# upravujeme vstupny dataframe po riadkoch a vysledok ukladame do pola
zaznamov result
result        =        [agregacia(row)        for        row        in
df_inode[string_column_assortment].to_numpy()]

# serializujeme a ukladame pole result
pickle.dump(result, open("agregacia_vysledok.pickle","wb"))
# pre nacitanie:
# pickle.load( open( "agregacia_vysledok.pickle", "rb" ) )
```

Príloha G

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# importujeme kniznicu na vizualizaciu grafov
from pyvis.network import Network
# importujeme kniznicu pre efektivne grafove datove struktury
import networkx as nx
# typova struktura pre dokumentaciu
from typing import List

def add_edge_to_list(val:str, nodes_list:List[str], g: nx.DiGraph) ->
None:
    """
    Pomocna metoda, ktora medzi kazdy vrchol v nodes_list vlozi do
    grafovej struktury g neorientovanu hranu s oznacenim val.

    Parameters
    ----------
    val : str
        Retazec, co sluzi na oznacenie hrany.
    nodes_list : List[str]
        Pole vrcholov.
    g : nx.DiGraph
        Grafova struktura, do ktorej sa vkladaju hrany a vrcholy.

    Returns
    -------
    None

    """
    # cykly cez polia vrcholov
    for u_ind in range(len(nodes_list)-1):
        for v in nodes_list[u_ind+1:]:
            g.add_edge(nodes_list[u_ind],v,title=val)
            g.add_edge(v,nodes_list[u_ind],title=val)

def add_nodes(nodes_list, g, group_id:int) -> None:
    for node in nodes_list:
        g.add_node(node, label = node, group=group_id)

def fca_reader(Lines: List[str], g: nx.DiGraph, group_id:int):
    """
    Pomocna metoda na citanie vstupu textoveho suboru 'typu' fca.

    Parameters
    ----------
    Lines : List[str]
```

```
        Pole riadkov vstupneho suboru (kazdy riadok reprezentovany ako
str)
    g : nx.DiGraph
        Grafova struktura na ulozenie dat z textoveho Lines.
    group_id : int
        Identifikator pre data rovnakeho typu z Lines
        (vhodne pre prehladnejsie vykreslenie).

    Returns
    -------
    None

    """
    # prechadzame riadkami
    for line in Lines:
        # oddelime udaje hran od vrcholov
        line_list_tuple = line.split(';')
        line_list_tuple[0] = line_list_tuple[0].strip()[1:]
        line_list = line_list_tuple[1].strip().split(',')
        line_list[len(line_list)-1]     =     line_list[len(line_list)-
1].strip()[:-1]
        line_list = [s.strip() for s in line_list]

        # pridanie vrcholov do grafovej struktury
        add_nodes(line_list, g, group_id)

        # pridanie vhodnych ohodnoteni hran do grafovej struktury
        add_edge_to_list(line_list_tuple[0], line_list, g)

def standard_reader(Lines: List[str], g: nx.DiGraph, group_id:int) ->
None:
    """
    Pomocna metoda na citanie vstupu textoveho suboru 'typu' inode a
name.

    Parameters
    ----------
    Lines : List[str]
        Pole riadkov vstupneho suboru (kazdy riadok reprezentovany ako
str)
    g : nx.DiGraph
        Grafova struktura na ulozenie dat z textoveho Lines.
    group_id : int
        Identifikator pre data rovnakeho typu z Lines
        (vhodne pre prehladnejsie vykreslenie).

    Returns
    -------
    None
```

```
    """
    for line in Lines:
        # oddelime udaje hran od vrcholov
        line_list = line.split(',')
        line_list[0] = line_list[0].strip()[1:]
        line_list[len(line_list)-1]     =     line_list[len(line_list)-
1].strip()[:-1]
        line_list = [s.strip() for s in line_list]

        # pridanie vrcholov do grafovej struktury
        add_nodes(line_list[1:], g, group_id)

        # pridanie vhodnych ohodnoteni hran do grafovej struktury
        add_edge_to_list(line_list[0], line_list[1:], g)

# cesta ku suborum s datami
PATH = 'C:\Users\krist\OneDrive\Počítač\diplomka\vizualizacia\'

# mena datovych suborov
FILE_NAME_1 = "fca_relations"
FILE_NAME_2 = "file_relations_inode"
FILE_NAME_3 = "file_relations_name"

# moznosti uprav vykreslenia
custom_options = """
var options = {
  "nodes": {
    "font": {
      "size": 9
    },
    "size": 18
  },
  "edges": {
    "color": {
      "inherit": true
    },
    "smooth": false
  },
  "interaction": {
    "hover": true,
    "navigationButtons": true
  },
  "physics": {
    "hierarchicalRepulsion": {
      "centralGravity": 0
    },
    "minVelocity": 0.75,
    "solver": "hierarchicalRepulsion"
  }
```

```
}
"""

# spracovanie datovych suborov
g = nx.DiGraph()

with open(PATH + FILE_NAME_1 + '.txt') as fp:
    Lines = fp.readlines()
    fca_reader(Lines,g,1)

with open(PATH + FILE_NAME_2 + '.txt') as fp:
    Lines = fp.readlines()
    standard_reader(Lines,g,2)

with open(PATH + FILE_NAME_3 + '.txt') as fp:
    Lines = fp.readlines()
    standard_reader(Lines,g,3)

# vykreslenie dat
nt = Network('720px', '1200px')

nt.from_nx(g,show_edge_weights=False)
nt.toggle_physics(False)
#nt.show_buttons(filter_=True)
nt.set_options(custom_options)
nt.show('full_result.html')
```