Forensic Artifacts' Analysis using Graph Theory

Sophia Petra Krišáková[1,*,†], Pavol Sokol[1,†] and Rastislav Krivoš-Belluš[1,†]

[1]Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia

Abstract

The number of cyber-attacks is constantly growing, and their sophistication is increasing due to new techniques and strategies of attackers. Organisations must continuously improve their methods of detecting and responding to these attacks to protect their networks and information systems. The time between the occurrence of a security incident and its identification takes an average of 100 - 200 days, with organisations having a response time of between 50 - 70 days. Our work aims to reduce this time so that organisations can respond to security incidents more quickly. In this work, we use graph theory for forensic analysis in the Windows operating system. The main objective of the work is to identify digital evidence and the relationships between them. For this purpose, we work with datasets from various Capture the Flag (CTF) competitions. We describe the processing stages of the digital evidence and their transformation into graphs and then identify anomalies and cycles in the graphs in order to provide readers with a deeper insight.

Keywords

graph theory, graph algorithms, forensic analysis, artifact, cybersecurity

1. Introduction

In the digital world, data and network security is a key concern for organisations of all sizes and industries. With the rise of cyber-attacks and their ever-changing nature, organisations must constantly adapt to protect their assets and ensure the security of their information. Cyber attackers are constantly moving forward and developing new ways to penetrate systems and gain unauthorised access to sensitive data. As these attacks become more sophisticated, the challenge for organisations is to identify attacks as quickly as possible and respond appropriately and ideally.

One of the main issues in the response to security attacks is the time between the occurrence of a security incident, its identification, and the subsequent response. This time can be non-trivial, often measured in hundreds of days, giving attackers ample time to cause damage without being detected. In addition, even after a security incident is identified, an organisation needs time to resolve it and restore normal operations.

Our research focuses on the security incident response process, including digital forensics. The main aim is to reduce the time interval required to resolve a security incident and provide organisations with a way to respond more quickly and effectively to security incidents. In that article, we focus on how graph theory applied to individual forensic artefacts available in the Windows operating system and the NTFS file system can help us. Graph analysis allows us to identify the relationships between different digital evidence and their attributes, thereby better understanding the nature of the attack.

To achieve this objective, we specify the following partial research objectives:

- What attributes of forensic artefacts are best suited for graph representation?
- How can specific properties of graphs help identify key forensic artefacts and relationships in digital forensics?

This paper is divided into six sections. Section 2 discusses papers relevant to this research. Section 3 specifies the methods employed in this paper, including the collection and processing of the digital evidence. Section 4 outlines the graph theory applied to digital evidence and graph generation options. Section 5 discusses the lessons learned from applied graph properties to digital evidence. Section 6 provides a summary, including our suggestions for future research.

2. Related works

We often think of data analysis and machine learning as elements of artificial intelligence. It may be about analysing data differently. We must also visualise the data, preprocess it, get basic statistics, etc. It is in the visualisation that graphs can help us. Several works have already been done in forensic artefact analysis using graphs. Many of these have focused on network communication, which is different from the form of data we use in this work. Nevertheless, we can take inspiration from them

ITAT 2024 Information Technologies – Applications and Theory 2024, September 20–24, 2024, Drienica, Slovakia

*Corresponding author.

†These authors contributed equally.

sophia.petra.krisakova@upjs.sk (S. P. Krišáková); pavol.sokol@upjs.sk (P. Sokol); rastislav.krivos-bellus@upjs.sk (R. Krivoš-Belluš)

0000-0002-1967-8802 (P. Sokol)

as they offer insight into the representation of forensic artefacts using graphs and their connections. We have selected a few of these papers to get a basic overview of the current state of the art in the given field.

2.1. Security Data Analysis

Cybercrime risks have escalated with the digitization of data (books, videos, images, medical and genetic information) via laptops, tablets, smartphones, and wearables. Digital forensics recovers lost or deleted files but requires more efficient investigation resources. Current processes rely heavily on human input, slowing responses to rapid cybercrimes. Machine learning can automate digital investigations, aiding digital investigators [1].

Constantini, Gasperis, and Olivieri explored artificial intelligence and computational logic, particularly answer set programming, to automate evidence analysis in digital forensics. They demonstrated how complex investigations could be optimized and automated to assist in generating hypotheses for court cases using graph theory-based algorithms [2].

Ch. Easttom highlighted the use of graph theory in criminal investigations, describing how mathematical modelling helps understand relationships among suspects, victims, and systems [3]. Palmer, Campbell, and Gelfand further discussed graph theory's role in forensic analysis, noting the visualization benefits for investigators and its potential to support the investigation process [4].

A study on distributed graph analysis of large-scale email datasets showcased improved efficiency and accuracy in digital evidence analysis using centrality algorithms [5]. Binwal, Devi, and Singh developed algorithms for fingerprint graph representation and isomorphism testing, applicable to broader forensic analysis despite differing input data [6].

Additionally, attack graphs, used to identify potential attack paths and vulnerabilities, are proposed for practical forensic analysis, including antiforensic scenarios. These graphs help understand complex attack paths and missing evidence, demonstrated through a database attack case study [7].

Our contribution is to integrate these methods to enhance digital forensics. We focus on automating digital evidence analysis with graph theory to elucidate relationships among digital evidence and entities.

2.2. Forensic Analysis in Network Communication Using Graph Theory

While our primary focus is on NTFS file system data from Windows, we also review digital forensics in network communication, linking file system data with network communication data.

Due to the proliferation of smart devices, detecting and mitigating faults in computer networks is crucial. Anomalies, whether from security breaches, component failures, or environmental factors, must be promptly addressed. Recent studies on anomaly detection in computer networks categorize solutions and highlight trends and shortcomings, especially regarding malware in smartphone networks.

Paper [8] introduces a graph-based approach to network forensics, using a graph model of digital evidence for evidence presentation and automated reasoning. The proposed hierarchical reasoning framework infers network entity states and identifies critical entities. An interactive hypothesis testing framework aids in detecting attack activities. Experimental results show the prototype's effectiveness in extracting attack scenarios with minimal expert knowledge.

As Internet traffic grows, so do cyber crimes, necessitating advanced network forensics. One method combines network vulnerability and graph network evidence to reconstruct attack scenarios and identify multi-stage attacks, confirmed by experimental results [9].

Spectral graph theory helps understand network malware propagation, essential as device connectivity increases. Using various Laplacian matrices to track network pattern changes, one study [10] offers insights into malware spread, aiding in faster infection detection.

3. Methodology

In this chapter, we have covered how to acquire, preprocess, and explain data so that we can combine it into graphs, filter it, and analyse it later. We have also described the creation of the super timeline and the subsequent transformation of the other two datasets.

3.1. Data acquisition and description

We selected seven fictitious cases from CTF competitions focused on forensics, incident response, and threat detection, and we used disk images from these cases.

The first example is the case of the stolen Szechuan sauce from the DFIR Madness portal called **Case001 - The case of the Stolen Szechuan sauce** [11], where in this case the main goal was to find out how CITADEL's recipe got on the dark web. The company requested forensic analysis, identification of unwanted applications installed on the system, and detection of the location and time of installation. The case also offers information as to whether any content was changed, modified, deleted, or data was leaked. We worked with artefacts from the company's DC domain control server (hereafter called the "DC server") and from the Desktop - therefore we count this case as two datasets.

The other three cases **Magnet CTF 2019** [12], **Magnet CTF 2020** [13] and **Magnet CTF 2022** [14] were from the CTF (Capture the Flag) Magnet Forensics competition. It was not a classical forensic analysis, but rather answering questions like "when did we get the disk image" or "when was the software installed".

The last two cases, **NIST Data Leakage Case** [15] and **NIST Hacking Case** [16], are used to learn about different forms of data leakage and to improve techniques for investigating them. We focused on investigating a data leak case where the key is to uncover evidence of illegal activities and obtain any information generated by the suspect.

3.2. Data preprocessing

For data preprocessing, we followed the same steps in all seven cases, specifically specifying the preprocessing process for only one case. We worked with the data according to the procedure described in the paper [17].

We created the timeline using the Log2timeline [18] tool and its plugins. We modified the resulting timeline with the psort.py tool and the Python language with the pandas library. Our dataset contained records from 11 different data sources, with FILE, EVT, and REG records being the most prominently represented, accounting for 87% of all records. We divided the extracted attributes into seven categories.

We narrowed the dataset to the time of the security incident and manually identified relevant digital evidence, including the inode files: 84630, 84880, 84987, 86966, 86967, 86968, 86970, 86971, 86975, 87059, 87060, 87064, 87111, 87112, 87137, and files with the names: 'coreupdater. exe', 'FILESH 1', 'Secret', 'BETH_S 1.TXT', 'Beth-_Secret.lnk', 'SECRET 1.TXT', 'SECRET_beth.lnk', 'Szechuan', 'SZECHU1.TXT', 'Secret.lnk', 'NoJerry. lnk', 'No-Jerry.txt', 'f01b4d95cf55d32a.automatic-Destinationsms', 'SECRET_beth.txt', 'Beth_Secret.txt', 'Secret.zip', 'coreupdater.exe.2424 urv. partial'.

Next, we analysed the inodes and filenames, excluding some inodes and filenames. Finally, we used aggregation functions and created attribute combinations to analyse the data.

We only manually identified inodes and file names in the stolen Szechuan sauce recipe; the identified inodes and file names still need to be identified for the other datasets.

Each of the seven datasets is a super timeline containing 17 attributes, and the following rows are records (events). There are 17 attributes and their description by [19]:

- `Date` : the date when the event occurred
- `Time` : time the event occurred
- `Timezone` : time zone
- `MACB` : timestamps (Modification, Access, Changed, Birth)
- `Source` : source name abbreviation (e.g. REG - registry records)
- `Sourcetype` : description of the source
- `Type` : timestamp type (e.g. last entry)
- `User` : the user name (if any) that is associated with the event
- `Host` : host name (if any) that is associated with the event
- `Short` : contains a short description field in which the text is stored
- `Desc` : an array that contains most of the parsed information
- `Version` : version number of the timestamp
- `Filename` : the name of the file that is associated with the event
- `Inode` : inode number of the file being analysed
- `Notes` : a place to store additional information
- `Format` : the input module that was used to parse
- `Extra` : field with parsed information that is linked and stored here

In addition to the basic seven datasets, we created 6 additional CSV files containing records from FILE and 7 CSV files containing records from EVT. The datasets from FILE have 44 attributes, and EVT have 40 attributes. These files were created by extracting data from the original supertimeline. For example, if there was a MACB column in the original dataset, four new columns were created in the new (EVT or FILE) dataset, and the original one was deleted. The new columns are 'M', 'A', 'C', 'B'. If there was a '.ACB' record in the original data, we wrote 1 in the relevant 'A', 'C', 'B' columns and 0 in the 'M' column. In this way, we partially created binary data or columns. Not all attributes could be converted this way, so columns like 'date' or 'time' were left unchanged. The exact procedure for creating datasets is explained in [19].

For later analysis and graphing needs, we had to create additional columns in the FILE datasets - MACB, file, dir, and NTFS, which were created by concatenating some of the columns. MACB - we merged the 'broken' columns 'M', 'A', 'C', 'B' into one again. In the case of the file column, these were 'file_executable', 'file_graphic', 'file_documents', 'file_ps' and 'file_other'. For dir - 'dir_appdata', 'dir_win', 'dir_user' and 'dir_other'. NTFS - 'file_stat', 'NTFS_file_stat', 'file_entry_shell_item' and 'NTFS_USN_change'.

The analysis of the selection of the attributes mentioned above, as well as the analysis of various combinations of attributes for anomaly detection, is presented in the paper [17].

4. Graph Theory

We use standard notation for graph theory [20]. In computer/network security, graph theory models have often been used in the last decades. Some graph problems have also risen from security, e.g. k-Path Vertex Cover[21]. We will focus on modelling graphs from Artifacts' data.

4.1. Background

A Graph $G = (V, E)$ is a pair of a finite set of n vertices $V = \{v_0, v_1, \ldots, v_{n-1}\}$ and m edges $E = \{\{v_i, v_j\} | 0 \leq i, j, < n, i \neq j\}$. A directed graph has oriented edges(directed arcs), i.e. the order of vertices is important (v_i, v_j). (Edge-)Weighted graph assigns the weight to each edge, so the edges are in the form (v_i, v_j, w_{ij}).

A bipartite graph is a special type of graph where a set of vertices can be divided into two disjoint sets (partitions), where each edge has exactly one vertex from every partition.

There are several definitions and parameters: path (sequence of incident vertices and edges, starting and ending vertex (leaf), all vertices are mutually different), excentricity, etc.

The degree of the vertex v in a graph G, denoted by $\deg_G(v)$ or $d_G(v)$, is the number of edges incident to v in G. The maximum degree of a graph G, denoted by $\Delta(G)$, is the maximum value among the degrees of all vertices of the graph G. By analogy, we also denote the minimum degree of a graph G.

Walk in the graph G denotes the alternating sequence of incident vertices and edges (starting and ending with vertex). A sequence with mutually different edges is called a **trail**. A walk where all vertices differ is called a **path**. In other words, a path in a graph is a sequence of vertices for which there indeed exists an edge in the graph between every two following vertices. No two vertices (and hence no edges) are repeated. A trail in which all vertices except the first and last are distinct is called a **cycle**.

A graph is **connected** if every pair of vertices in the graph is connected. It means that there is a path between every pair of vertices. A component of a graph is a connected subgraph that is not part of any larger connected subgraph.

An edge of a graph is called a **bridge** if the number of components increases when it is removed. A vertex of a graph is called a **cut vertex/articulation point** if the number of components of the graph increases when it is removed.

Eccentricity the $ecc(v)$ of a point v is the distance of the point v from the farthest point in the G graph. The radius of the graph $rad(G)$ is the minimum eccentricity of a point in the graph G, and $diam(G)$ is the maximum eccentricity of a point in the graph G. A vertex c in G is called central if $ecc(v) = rad(G)$. The **center** of a graph $C(G)$ is the set of all vertices in the graph G with minimum eccentricity:

$$C(G) = \{v \in V \mid e(v) = \min_{u \in V} e(u)\}$$

The **center** of $C(G)$ is the set of all central vertices in the graph G. The **eccentric vertex** of a vertex v is the vertex that is the furthest from it. A vertex v is called **peripheral** if $ecc(v) = diam(G)$. **Periphery** $Per(G)$ of a graph G is the set of all peripheral vertices in the graph G.

4.2. Graph Generation from Forensic Artifacts

Typical graph generation in the security area is creating just nodes of one type and connecting them depending on communication [22] in graph or finding an attack vector in a directed graph [23]. Standard computer network-based cybersecurity applications cover traffic, security policies and vulnerabilities/threats [24].

For the artefacts, one can create nodes from any attribute (column) or any combination of attributes. Moreover, we found the most interesting results for bipartite graphs, e.g., using two node types. Depending on the dataset, we focused on different pairs of node types.

4.3. Graph from supertimeline

For the super timeline, we have chosen attributes such as `user`, `source`, `MACB`, `sourcetype` and `inode` and always two of them as vertex types for the generated bipartite graph. We can consider other attributes that will be represented by vertices in the graph, e.g. `host`, `type`, `filename`. Some of these attributes are categorical so that we can think of them in this sense. The edge represents the row's existence in data containing these two vertices (artefact). An example of the NIST Data Leakage case is shown in Fig. 2.

4.4. Graph from EVT artifacts

In the EVT dataset, we created binary combinations of three attributes: `event_id`, `user_sid`, and `execution_process` because we could not tell any vital information about the relationships between the attributes by selecting other attributes. We could also consider attributes like `inode`, `computer_name`, and `source_name`, since we assume they are finite in number and not binary values, but that probably wouldn't add any value for us. An example of a generated graph from the EVT dataset is shown in Fig. 1.

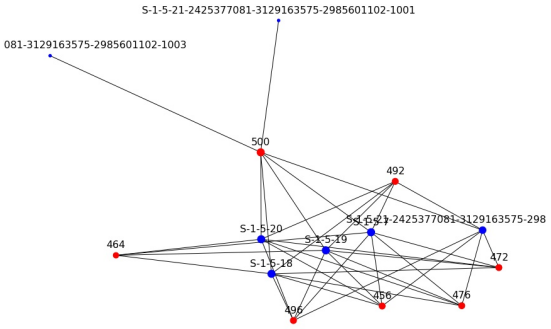

Figure 1: NIST Data Leakage Case - EVT - user_sid, execution_process_id

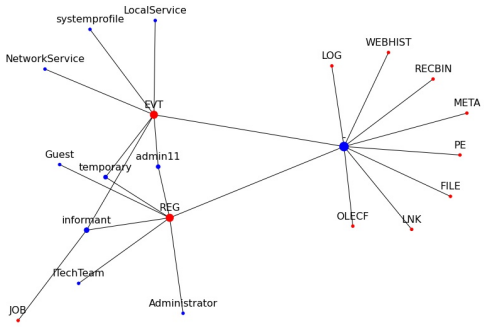

Figure 2: NIST Data Leakage Case - super timeline - user, source

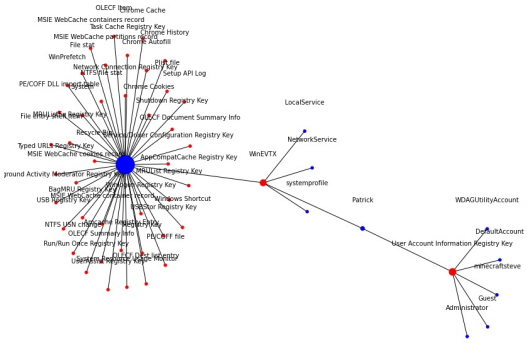

Figure 3: Magnet CTF 2022 - super timeline - user, sourcetype

4.5. Graph from FILE artefacts

In a similar way to the super timeline, we also created graphs in the FILE dataset, but with different attributes, such as MACB, dir, file, and NTFS, the creation of which we explained in Chapter 3.2. In the datasets file, it was challenging to think of other attributes that would be suitable for graphing because they were in binary form. This is why we merged some attributes, but it was impossible for all of them. We would still include the `inode` attribute in the graph creation process, but it had many unique values. Examples of these graphs are in Fig. 4 and Fig. 5.

5. Lessons learned from graph properties

From the generated graphs, we have focused on some graph-based metrics [22] and found some anomalies. In the current research, we have used only unweighted graphs.

5.1. Eccentricity

In particular, we can exploit eccentricity in graphs in the super timeline dataset created from user and source or user and source type attributes. For example, in the NIST Data Leakage Case in Fig. 2, we created a graph from the user and source attributes and then found the eccentricity of all vertices. The vertices belonging to the user attribute had the lowest eccentricity of 3: '-', 'informant', 'admin11' and 'temporary', and the vertices belonging to the source attribute had 'REG' and 'EVT'. These vertices can also be called the centre of the graph.

We also created graphs from the user and source type attributes and searched for the graph centre. We can illustrate this with all three Magnet CTF cases 2019, 2020 and 2022. Fig. 3 shows the graph from the Magnet CTF 2022 case. The lowest eccentricities in this graph were in the vertices 'WinEVTX' (source type attribute) and 'Patrick' (user attribute). We observed the same behaviour in the other Magnet CTF cases - that is, the graph centre was always identified as the 'WinEVTX' vertex and one of the users.

5.2. Degree of vertices

We exploited the degree of vertex in the datasets created from the supertimeline from the FILE source. We created three types of graphs - MACB and file, MACB and dir, MACB and NTFS. Fig. 4 shows an example of such a graph. The most interesting graphs arose when the MACB and NTFS timestamps were combined, as shown in Fig. 4. At first glance, it might seem that we should focus on the NTFS_UNS_change attribute because it is associated with timestamp C (Change), but neither the file_stat and NTFS_file_stat vertices are, despite having the highest degree of vertex - 14. The most significant vertex in this graph is file_entry_shell_item with a ver-

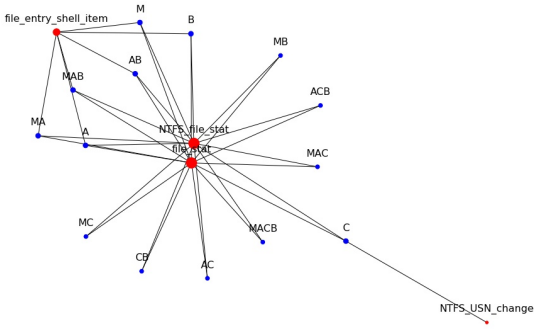

Figure 4: Szechuan Sauce - FILE - MACB, NTFS

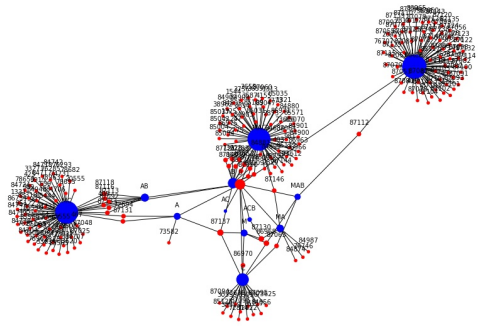

Figure 5: Szechuan Sauce - FILE - MACB, inode

tex degree of 6 because if we look further at the records containing this NTFS attribute value, we find "only" 19 unique inode values in 163 records, and just 9 of these inodes were identified as relevant to the case.

5.3. Cycles in graphs

After creating the graph from The Stolen Szechuan Sauce - FILE dataset in Fig. 5, it was not visible what specifically to focus on, so we had to apply the properties of the graph. We looked for the base cycles in the graph. We found 43 of these, and seven cycles contained inodes that were manually identified as relevant to the case by the analysis. In the same way, we analysed the graph created from the MACB and filename attributes, where we also found file names in the base cycles marked as relevant to the case by manual analysis. In this analysis, however, we must consider that the attributes are not equivalent because the MACB timestamps or combinations will always be at most 16, and the inodes are a different number, often much higher.

6. Results and future works

This paper focuses on the automation of response to security incidents, including digital forensic analysis within the Windows operating system and NTFS file system. For this purpose, we used the graphs' structure and properties to better understand the relationships between the analysed digital evidence. The paper demonstrated the possibilities of generating graphs, particularly from super timeline formats, event records, and the Master File Table. We showed that it is essential to carefully select the attributes of artefacts that can be used as vertices and to determine the corresponding edges of the graphs. At the same time, we identified several properties of graphs, whose analysis can help better understand the relationships and identify interesting or relevant digital evidence. In the future, we will enhance our models with weighted graphs.

Acknowledgment

This paper was supported by the Slovak Research and Development Agency under contract No. APVV-23-0137 and contract No. APVV-21-0336.

References

[1] Iqbal, Salman, S. A. Alharbi, Advancing automation in digital forensic investigations using machine learning forensics, Digital Forensic Science. IntechOpen (2019).

[2] S. Costantini, G. D. Gasperis, R. Olivieri, Digital forensics and investigations meet artificial intelligence, Annals of Mathematics and Artificial Intelligence (2019).

[3] C. Easttom, Utilizing graph theory to model forensic examination, International Journal of Innovative Research in Information Security (IJIRIS) 4 (2017).

[4] I. Palmer, R. Campbell, B. Gelfand, Exploring digital evidence with graph theory, in: ADFSL Conference on Digital Forensics, Security and Law, 2017.

[5] S. Ozcan, M. Astekin, N. K. Shashidhar, B. Zhou, Centrality and scalability analysis on distributed graph of large-scale e-mail dataset for digital forensics, in: IEEE International Conference on Big Data (Big Data), 2020.

[6] J. Binwal, R. Devi, B. Singh, Mathematical modelling and simulation of fingerprint analysis using graph isomorphism, domination, and graph pebbling, Advances and Applications in Discrete Mathematics (2023).

[7] C. Liu, A. Singhal, D. Wijesekera, Using attack graphs in forensic examinations, in: Seventh Inter-

national Conference on Availability, Reliability and Security, 2012.
[8] W. Wang, T. E. Daniels, Building evidence graphs for network forensics analysis, in: 21st Annual Computer Security Applications Conference (ACSAC'05), 2005.
[9] J. He, C. Chang, P. He, M. S. Pathan, Network forensics method based on evidence graph and vulnerability reasoning, MDPI, future internet (2016).
[10] C. McGee, J. Guo, Z. Wang, The application of the graph laplacian in network forensics, in: IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2021.
[11] Case 001 – the stolen szechuan sauce, 2024. URL: https://dfirmadness.com/the-stolen-szechuan-sauce/, online, [cit. 2024-02-18].
[12] Magnet ctf 2019 windows desktop, 2019. URL: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2019%20CTF%20-%20Windows-Desktop.zip, online, [cit. 2024-02-18].
[13] Magnet ctf 2020 windows, 2020. URL: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2020%20CTF%20-%20Windows.zip, online, [cit. 2024-02-18].
[14] Magnet ctf 2022 windows, 2022. URL: https://digitalcorpora.s3.amazonaws.com/corpora/scenarios/magnet/2022%20CTF%20-%20Windows.zip, online, [cit. 2024-02-18].
[15] Data leakage case, 2024. URL: https://cfreds-archive.nist.gov/data_leakage_case/data-leakage-case.html, online, [cit. 2024-02-18].
[16] Hacking case, 2024. URL: https://cfreds-archive.nist.gov/Hacking_Case.html, online, [cit. 2024-02-18].
[17] E. Marková, P. Sokol, K. Kováčová, Detection of relevant digital evidence in the forensic timelines, in: International Conference on Electronics, Computers and Artificial Intelligence (ECAI), IEEE, 2022, pp. 1–7.
[18] Plaso, 2024. URL: https://plaso.readthedocs.io/en/latest, online, [cit. 2024-04-13].
[19] E. Marková, P. Sokol, S. P. Krišáková, K. Kováčová, Dataset of windows operating system forensics artefacts, Data in Brief (2024) 110693.
[20] D. B. West, Introduction to graph theory, Prentice hall, 2001.
[21] B. Brešar, F. Kardoš, J. Katrenič, G. Semanišin, Minimum k-path vertex cover, DAM 159 (2011) 1189–1195.
[22] G. Zonneveld, L. Principi, M. Baldi, Using graph theory for improving machine learning-based detection of cyber attacks, 2024. URL: https://arxiv.org/pdf/2402.07878.
[23] T. Mézešová, P. Sokol, T. Bajtoš, Evaluation of attackers' skill levels in multi-stage attacks, Information 11 (2020) 537.
[24] V. P. Janeja, Data Analytics for Cybersecurity, Chapter 9: Cybersecurity through Network and Graph Data, 2022. URL: https://doi.org/10.1017/9781108231954.