Using Relational Graphs for Exploratory Analysis of Network Traffic Data

Milan Cermak[a], Tatiana Fritzová[b], Vít Rusňák[a], Denisa Sramkova[a]

[a]*Institute of Computer Science, Masaryk University, Sumavska 416/15, 60200, Brno, Czech Republic*
[b]*Faculty of Informatics, Masaryk University, Botanicka 68a, 60200, Brno, Czech Republic*

Abstract

The human brain is designed to perceive the surrounding world as associations. These associations between the individual pieces of information allow us to analyze and categorize new inputs and thus understand them. However, the support for association-based analysis in traditional network analysis tools is only limited or not present at all. These tools are mostly based on manual browsing, filtering, and aggregation, with only basic support for statistical analyses and visualizations for communicating the general characteristics. Yet, it is the relationship diagram that could allow the analysts to get a broader context and reveal the associations hidden in the data. In this paper, we explore the possibilities of relational analysis as a novel paradigm for network forensics. We provide a set of user requirements based on the discussion with domain experts and introduce a novel visual analysis tool utilizing multimodal graphs for modeling relationships between entities from captured packet traces. Finally, we demonstrate the relational analysis process on two use cases and discuss feedback from domain experts.

Keywords: Relational analytics, Network forensics, Visual analytics, Granef, Cybersecurity

1. Introduction

Every organization and device connected to the network constantly faces cyber threats against which they are trying to defend. On the other side, the attackers constantly react to such efforts by inventing new ways to exploit weak points. When they are successful, it is crucial to investigate the attack's type, origin, impact, and spread to prevent similar threats in the future (ENISA, 2010). For example, the investigator needs to understand how the malware got on the device or if the device has communicated with others in the network. Such investigation can be effectively achieved through network traffic analysis (also referred to as network forensics) (ENISA, 2018), which we focus on in this work. Even though many automated network traffic analysis tools are available to support the investigation (Sikos, 2020), the analyst often has to perform the analysis manually to discover all crucial information. He or she can be overwhelmed in the early stages of analysis as it is unclear where to start since the volume of data is so extensive (D'Alconzo et al., 2019). The automated tools often provide a statistical overview, intrusion detection functions, various visualizations, and the ability to filter and browse the data. However, many hidden relations in the data may remain undiscovered.

The human brain is used to perceive characteristics of the surrounding world as associations (Zhang et al., 2020). By linking different information, we can analyze a given problem and understand it. In the case of data analysis, this approach can be reflected by modeling the associations through multimodal graphs (also known as relational graphs), where nodes represent various modalities or types of entities. For example, this technique is already used in criminal investigation (Atkin, 2011; Zákopčanová et al., 2021) or social network analysis (Tabassum et al., 2018) to build situational awareness and explore broader context. Nevertheless, in the case of network traffic analysis and forensics, this approach is either not provided at all or only in a limited form. In contrast to current methods focused only on network host relations, the relational analysis of all significant attributes of the network traffic (e.g., hosts, applications, connections, and data) provides better insights and helps to reveal otherwise hidden information. To make this possible, it is necessary to face challenges related to not only the processing and storing of large volumes of network traffic data but also their visualization and interactive exploration reflecting the analysts' needs.

To explore the feasibility of relational analysis in network traffic forensics, we have designed and implemented an open-source toolkit Granef, introduced in our previous work (Cermak and Sramkova, 2021), that models attributes of the network traffic as entities, and store them in a graph database. This article extends this work and focuses on the exploratory analysis of such data in a novel user interface. We grounded the design requirements of the interface based on analyzing commonly used tools for network forensics and interviews with domain experts (cybersecurity incident analysts). As a result, the presented Granef User Interface (Granef UI) allows analysts to in-

Email addresses: cermak@ics.muni.cz (Milan Cermak), 456264@mail.muni.cz (Tatiana Fritzová), rusnak@ics.muni.cz (Vít Rusňák), denisa.sramkova@mail.muni.cz (Denisa Sramkova)

teractively analyze the network data by using predefined queries or fetching the neighbors of selected nodes (depth search). In addition, the displayed relational graph data can be searched, clustered, tagged, or visually modified to make it as easy as possible to navigate through the data. We have prepared two use cases to demonstrate the capabilities of Granef UI and evaluate it through a qualitative user study with five domain experts. Although relational analysis was a new paradigm for them, they were positive about the tool's features and quickly learned the basic principles of relational analysis.

The contributions of our work are threefold. First, we define a set of design requirements for using relational analysis in network traffic analysis. Second, we design and implement a Granef UI and provide it as open-source. Third, we demonstrate the proposed analysis approach on two use cases and evaluate it using a qualitative user study. This work can be leveraged by both cybersecurity specialists to explore the new approach to network traffic analysis, as well as computer graphics and visualization specialists who want to research the possibilities of using relational graphs for data analysis.

2. State of the Art

This paper focuses on a combination of analysis approach and visual aspects of network data analysis. Therefore, we divide this section into two parts. The first part focuses on analyzing network traffic data and briefly introduces current tools used to analyze such data and perform network forensics. The second part focuses on relational analysis and introduces relevant analytical tools applying this approach to analyzing various data types and sources.

2.1. Network Traffic Analysis

Network traffic analysis is a key element in understanding network events and hosts' behavior while performing network forensics. The crucial part is the analysis of collected data (e.g., packets or IP flows), aiming to extract the required information and gain a situational overview. Such analysis can be partly automated using anomaly or intrusion detection tools (Fernandes et al., 2018). However, these tools may not reveal details important to evidence collection, and therefore manual exploratory network data analysis plays an important role, allowing analysts to verify detected anomalies, examine contexts, or extract additional information. Since network data are considered to be big data, their processing and analysis face challenges of volume, velocity, variety, and veracity (D'Alconzo et al., 2019). In the case of network forensics (Messier, 2017), it is further emphasized by the requirement to manually analyze all the data and preserve their origin, which also limits the use of automated tools aggregating the data.

Typically, the analyst's steps during network forensics consist of listing some statistics and then focusing more on investigating suspicious or significant observations (Šrámková, 2022). Therefore, automated tools and statistical analysis are employed at the beginning of the analysis to help point out such observations. If the analyst identifies some notable network event (e.g., communication with malware C&C or unusual peak in the volume of transferred data), his or her objective is to identify the event's origin, its essence, and trace all related hosts and connections. The same applies if the analyst initially received information about the suspicious event from an external source (e.g., an intrusion detection system). In the following list, we briefly introduce common tools that are used today for network forensics and exploratory network data analysis. Further details can be found either in the tools documentation or in the survey by Sikos (2020).

- **Wireshark**[1] – It is a de facto standard for packet trace analysis widely used across many commercial and non-profit enterprises. It provides a rich feature set for in-depth analysis of network data and supports deep inspection of hundreds of network protocols in a three pane-view (packets list, packet details, and packet bytes as hex dump). In the case of TCP or UDP traffic, Wireshark organizes the captured data on a per-ethernet-frame basis and allows the user to inspect different layers by filtering and displaying the data or isolating the TCP/UDP stream.
- **Arkime**[2] – It is a large-scale, open-source, indexed packet capture and search tool with a web interface. Its main feature is the ability to track and visualize network connections and extract metadata. The main page of Arkime is the Sessions page, which displays a timeline and a list of indexed communications sessions for the filtered expression and allows the export of them as packet trace. To provide a broader picture, the SPI (Session Profile Information) View page shows aggregated statistics on different session metadata and allows their direct filtering. The tool also includes a Connection page that displays a simple relational graph of communicating hosts.
- **Brim**[3] – An open-source tool for network forensics and threat hunting, combining the Zeek network monitoring tool (The Zeek Project, 2022) and Suricata intrusion detection system (OISF, 2022). Both components produce contextual data based on packet captures, which an analyst reviews. Brim brings all of this together, presents the analyst with a graphical user interface, and provides its own query language (called ZQL) supporting aggregation functions to inspect the data. The user interface consists of a timeline showing the number of connections, a list of connections corresponding to the selection, and a pane with selected connection details. It also provides a direct link to Wireshark, where the analyst can further inspect selected network connections.

[1] `https://www.wireshark.org/`
[2] `https://arkime.com/`
[3] `https://www.brimdata.io/`

- **Elastic Stack**[4] – A universal tool for the storage and analysis of various textual data. Although it is not directly designed for network forensics, it can be used for this purpose, but the network traffic needs to be processed by another tool first (e.g., Zeek). The Elastic Stack consists of multiple tools that can work together or separately, whereas the main parts are Elasticsearch, which is used to store the data, and Kibana, providing a user interface offering many data visualizations. In the case of network forensics, Elastic Stack is mainly used to obtain a macro-view through various visualizations of aggregated data.

In addition to standalone tools, the capabilities of modern web browsers initiated the development of tools providing network traffic analysis as a service. For example, CapAnalysis (Costa, Gianluca, 2019) allows users to review large PCAP files, parse the data streams, filter out ports, protocols, or IP addresses, and associate them with geographical areas. A-packets (A-Packets, 2022) and PacketTotal (PacketTotal, 2022) provide multiple individual views on PCAP files through a set of dashboards focused on individual characteristics extracted from the data (e.g., application, TLS certificates). NetCapVis (Ulmer et al., 2019) and PCAPFunnel (Uhlár et al., 2021) provide both overview and support for explorative analysis through step-by-step data filtering.

Although there is a wide variety of tools for network data exploration that differ in features, visualization capabilities, and adoption, their focus is only on modeling relations between hosts (i.e., communications between network nodes). None of them genuinely leverages the paradigm of relational analysis.

2.2. Relational Analysis

Relational analysis has proven to be useful in various contexts and domains. Notably in criminal investigations, where this approach is traditionally used by criminalists (Atkin, 2011) and being integrated into analytical tools such as Visilant (Zákopčanová et al., 2021). In computer science, Feijs et al. (Feijs et al., 1998) showed its potential for analyzing software architectures. Ah-Pine and Marcotorchino introduced a general framework for data mining and decision making (Ah-Pine and Marcotorchino, 2010). In digital forensics, Chen and Malin (2011) used the technique in anomaly detection based on the access logs analysis. There are also general tools focused on relational analysis, such as Neo4j[5], but they are not prepared for exploratory analysis of network traffic data.

Granef is a pioneering example of applying relational analysis in network forensics. To the best of our knowledge, our approach brings a fresh perspective to traditional network analysis and introduces the principles of association-based network traffic representation into the visual analytics workflow. It is supposed to help the analysts think about the data in a way closer to the general perception of how computer networks work.

3. Data Model

Proper data representation and storage are key prerequisites for exploratory network traffic analysis using relational graphs. In our previous work, we addressed these areas, introduced data processing parts of the Granef toolkit, and simplified the network traffic data model proposed by Neise (2016) and Leichtnam et al. (2020). In general, the model follows the format of Zeek logs, preserves their relation, and eases extension by other data sources. The base model consists of the following four vertex types:

- Host – a device with an IP address observed in the network traffic capture;
- [Host-data] – information related to the host extracted from network traffic (e.g., hostname, TLS or e-mail certificates, downloaded files);
- Connection – general information about individual network connections (e.g., protocol, ports, connection time, number of packets);
- [Application] – application data extracted from the connection (e.g., DNS, HTTP, TLS data).

Edges between vertices represent their relations and preserve the information about their origin. A simplified diagram of vertices and edges is shown in Figure 1. All edges are bidirectional to allow reverse processing for querying from an arbitrary node regardless of its type. Formally, the model represents the entities in a *multimodal graph* described by Ghani et al. (2013) as "the traditional ordered pair $G = (V, E)$ comprises of a set of vertices V and edges E, but where vertices can be partitioned using a modality equivalence relation $\sim_{mod}$. This modality relation $\sim_{mod}$ is defined using the notion of *vertex type*, and the equivalence classes (partitions) defined by relation are called *modes*. We can further define a modality relation for edges based on a tuple of the modes of the two vertices an edge connects." The advantage of such an approach to data representation is a simple extension by adding new data sources (e.g., IDS alerts, OSINT data) as vertices and analyzing them with network traffic data within a unified visual environment.

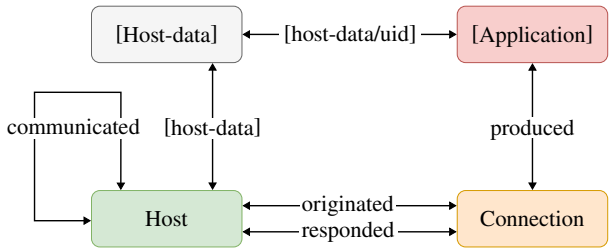

Figure 1: Simplified diagram of the relational network traffic data model (based on (Cermak and Sramkova, 2021)).

[4] https://www.elastic.co/elastic-stack/
[5] https://neo4j.com/

To extract and store the network traffic data in a defined data model, we use the transformation and data storage modules of the Granef toolkit (Cermak and Sramkova, 2021). It provides the pipeline for extracting the data from network traces and stores them in a graph database. The main components of the pipeline are the Zeek network security monitor and Dgraph[6] database. Zeek is used to extract information from a network packet trace file, such as connection information, related metadata, statistical properties, and cleartext partitions of application protocols (even encrypted ones). This extracted data is transformed into the defined data model and stored in the database. The database can be queried using Dgraph Query Language (DQL) based on GraphQL, a modern data query language providing high versatility. The toolkit provides an abstract layer API with common analysis functions such as neighbor discovery or data filtering to ease the analysis. It is worth mentioning that the initial version of the Granef toolkit provided simple graph visualization of the stored data. However, it was intended for demonstration purposes and lacked functionality for an effective exploratory analysis.

4. User Tasks and Requirements

The key requirement for exploratory analysis and data visualization is to support analysts and provide them with answers to the analytical questions as easily and fast as possible. In general, it must not only allow them to browse collected data (micro-view) but also provide an overview and broader context (macro-view). To properly design and develop such an analytical system, we have closely cooperated with domain experts from an incident response team and specialists focused on network forensics. Together with them, we have identified typical questions for network traffic analysis on which we defined requirements for exploratory analysis using relational graphs.

4.1. User Tasks

In general, any security and forensics analysis tries to answer the common questions: what, when, where, how, and who. The purpose of the analysis then determines how detailed answers are needed. In the case of analysis during incident response, it is crucial to find information as quickly as possible to recover from an incident and go back to "business as usual". In contrast, forensic analysis is usually associated with law enforcement, and the goal is to "solve the case" The analysis must therefore be more thorough and focused on more details. In the following sections, for simplicity, we will focus on the analysis of network data performed as part of incident response, but note that the principles and results we will present are applicable in both domains.

[6] `https://dgraph.io/`

As a basis for defining common user tasks, we have used recommendations by ENISA, 2010 and CISA, 2021 organizations that describe a typical analysis workflow in incident response. We have further consulted our findings through semi-structured interviews with domain experts in network and cybersecurity analysis from our university CSIRT, representing the target users of the tool. This led to the definition of the following five typical analytical questions, which are designed to verify incident severity, understand its origin, and evaluate its impact.

- *How was the host infected?* – Determine the type of attack and how it was performed.
- *Did the attacker scan for open services or vulnerabilities?* – Determine if any vulnerability of an available service has been exploited.
- *Did the host communicate to a malware C&C or another suspicious IP address?* – Identify indicators of compromise (IoC) and verify that there are no other compromised machines.
- *Did the host send a large amount of data outside the local network?* – Verify that no sensitive data has been exfiltrated.
- *Did the host communicate with other devices in the local network?* – Check whether the attacker infected other network hosts.

4.2. Requirements

We have used defined typical analytical questions to evaluate commonly used tools described in the State of the Art section and identified the functionality they provide to answer these questions. In addition, we have also analyzed the initial prototype of demonstration visualization provided by the Granef toolkit. The result of this evaluation is a set of functional and non-functional requirements, which we generalized and formulated as the following five key user requirements for exploratory analysis of network traffic data using relational graphs.

- **R1: Visualizing entities and their relationships** – The main attributes of the network traffic data will be displayed using an oriented multimodal graph. The interactive relationship visualization should allow the analyst to inspect details about any selected node and gain new observations through in-depth graph exploration. Basic statistical information (e.g., number of results, minimum, maximum, and average values of entity attributes) will be available for the displayed data.
- **R2: Facilitating graph interaction** – The user will be able to customize the graph's layout and other interface elements, including aggregation. The tool should facilitate getting initial insights about network connections. It should also allow distinguishing regular and suspicious network traffic at first glance based solely on the resulting pattern.

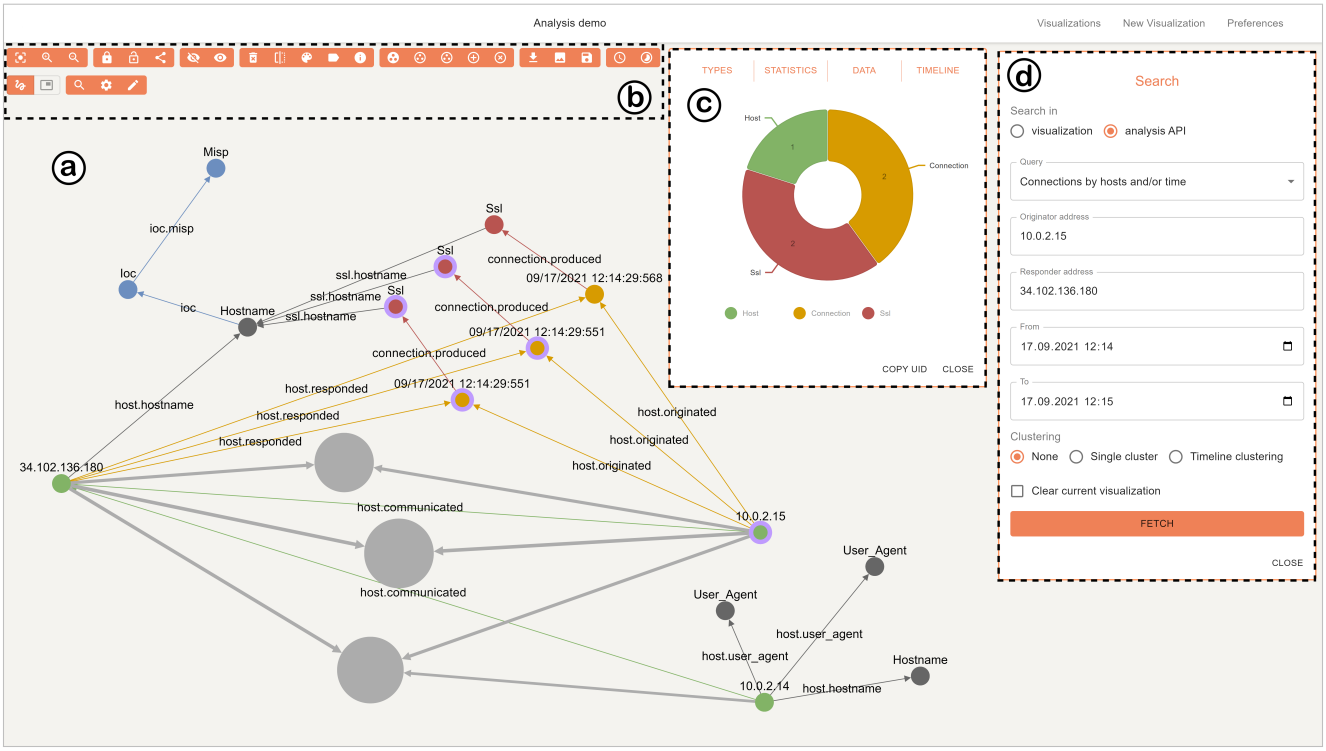

Figure 2: Granef UI's visual analytics interface: (a) *graph view* with rendering oriented relational graph; (b) *tools* menu providing different options to interact with the graph; (c) *Detail* child window providing various details related to the selected node(s) or edge(s) of the graph; (d) *Search* dialog for structured filtering and data querying.

- **R3: On-demand data enrichment** – The analyst can enrich network traffic data with additional information from external sources linked to displayed nodes (e.g., asset management, shared threat intelligence, device logs, notes). The tool should also allow the analyst to include the results of anomaly detections in a visualization.
- **R4: Visual and parametric filtering** – The data selection and filtering will be possible both by entering into a form and by direct interaction with the graph (i.e., visual query). The user will be able to filter the displayed data globally by time interval, node attributes, and graph edges, as well as locally by relationships to the selected node or group of nodes.
- **R5: Scalability** – The system must be able to display graphs with thousands of graph nodes seamlessly while keeping the high responsibility of the user interface.

5. Visual Analytics Interface

Considering defined requirements and user tasks, we have extended the Granef toolkit by Granef UI providing a visual analytics interface for exploratory analysis of stored data. The interface is designed as a web application to ensure compatibility across different systems. It is currently provided as an integrated module for the Granef toolkit. However, in the future, we plan to allow independent application operation with a connection to different graphic databases and user management. This section presents Granef UI components, describes their functionality, and discusses how they can be used during the investigation.

Since the analysts prefer working on desktop computers, the application layout is designed for a horizontal resolution of at least 1920 pixels. *Graph view*, rendering the relational graph on a canvas (Fig. 2 (a)), occupies the majority of the screen. Menus and dialog boxes are organized in a three-column layout as canvas overlays. The top-left corner occupy *Tools* (Fig. 2 (b)) for graph interaction. Child windows (*Detail*, *Search*, and *Timeline*) display in the middle and the left third of the screen. The *Detail* shows various details for the selected graph element(s) (Fig. 2 (c)), the *Search* enables the selection of visualized data and loading new data based on queries (Fig. 2 (d)), and *Timeline* enables to filter visualized data based on a time (not shown in the figure).

5.1. Graph View

The graph view is the key part of the analytical interface. It provides an interactive representation of an oriented graph that matches the visual encoding of Host, [Host-data], [Application], and Connection (i.e., graph nodes) and relations between them (i.e., oriented edges) (**R1**). Besides color encoding, nodes, and edges are accompanied by a short text label. Users can select one or more

nodes or gather them into node clusters to avoid graph cluttering. Selected nodes are encircled with a magenta ring, and their properties are shown in the *Detail* child window. Node clusters are represented as larger gray circles without any label, as shown in Fig. 2. When loaded, the graph layouting algorithm provides an initial positioning that can be adjusted by the user later on, either manually or using various layout algorithms.

Context menu. Users can interact with the chart through the context menu (see Figure 3) that is invoked by a right-click. Its content varies depending on a graph element under the cursor. If there are some nodes in the selection, the operations in the context menu apply to all selected nodes. If the user clicks on a node, actions will be applied to the node. However, the same items will be available for the selection. If no selection is defined, the context menu only offers the user to select all nodes.

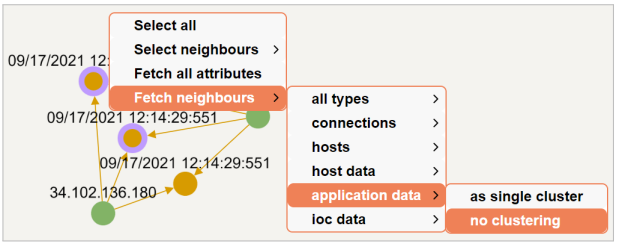

Figure 3: Context menu with actions for a group of selected nodes.

5.2. Tools

The *Tools* menu provides direct access to the most frequent interaction tasks (**R2**). Each action is represented by a button with a unique icon, and its brief description appears in a tooltip on mouseover. The related actions are grouped into nine sections as shown in Figure 4:

- *View manipulation* contains *Zoom in/out* and *Fit* buttons to adjusts the view of the graph.
- *Node locks* provide two modes for locking node positions. The global one locks the node position on the canvas regardless of further modifications in the graph. Graph-aware locking continuously recalculates and adjusts the position of the locked node due to further graph changes caused by user interaction.
- *Node hiding* allows hiding the nodes to unclutter the view and show them again.
- *Graph actions* section contains node removal feature, invert selection, tag and color assigning showing corresponding child windows where the user can assign nodes with one or more tags or colors. The last option shows the *Detail* window.
- *Clustering actions* allows the user to apply four graph clustering operations to aggregate selected nodes and unfold clustered (aggregated) nodes. The clustering actions include: clustering the outliers (nodes with the node degree 1), manual clustering of selected nodes, automated clustering of all nodes, and clustering of selected edges. The last button performs the cluster unfolding. The automated clustering is based on the Chinese Whispers algorithm (Biemann, 2006) and is further described in Section 6.2.
- *Export and Save* tools allow users to export the current graph as a serialized visualization in a JSON format or an image, or save the current analytical case, including the visualization state (i.e., definitions of nodes and edges, characteristics of the current view and data necessary for cluster manipulation).
- *Timeline controls* displays Timeline child window showing the number of the connections (y-axis) related to the time (x-axis).
- *Selection mode* allows changing the mode between rectangular and lasso (freehand).
- *Other* actions display the *Search* child window, open view configuration, and allow adding notes.

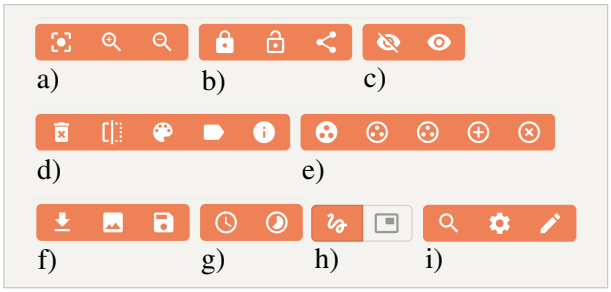

Figure 4: Tools menu – actions are grouped into ten categories: a) view manipulation, b) node locks, c) node hiding, d) graph actions, e) clustering actions, f) export and save, g) timeline controls, h) selection mode, i) other.

5.3. Child Windows

The visual analytics interface of Granef UI provides three types of child windows with a distinct focus. The *Detail* offers additional information related to the selected nodes (**R3**), *Search* allows the user to filter the data using parametric querying (**R4**), and *Timeline* allows the user to filter the data based on connection time.

5.3.1. Detail

The child window appears automatically if the user holds the mouse over the node for at least 250 milliseconds or by selecting it with a double click. In such a case, the child window remains on the screen, and the content is locked to the selected node(s). If only one node is selected, the child window displays only its attributes. If multiple nodes are selected, it contains summarized information about them. Since the nodes' metadata and

attributes are extensive and it does not make sense to display them altogether, the *Detail* is therefore divided into several tabbed views (**R3**). Such an organization also allows us to extend the functionality with new views in the future. Currently, Granef UI provides five of them: Types, Statistics, Data, Timeline, and Chord.

Types. The tab displays a donut chart (see Figure 2 (c)) summarizing the number of selected nodes by their type, with the colors corresponding to the colors of the nodes in the chart. At the bottom of the detail, there is a close button and a button that copies to the clipboard the identifiers of selected nodes for which the detail was created.

Statistics. The summary statistics tab complements the *Data* tab with charts summarizing the data obtained from the Granef analytical interface. Statistics are calculated only for some types and part of their attributes (e.g., communication protocols, a payload of requests and responses, or HTTP methods and response codes). The data is usually visualized using a bar chart or donut chart.

Data. The data tab allows the analyst to browse raw data and attributes of selected nodes. For each type of node, there is a table whose records are the individual nodes and their attributes in columns.

Timeline. The timeline shows the Connection nodes distribution in the selection regarding the time of the connection. The x-axis displays moments in time (5-minute intervals by default). The y-axis then plots the number of connections for each time instant using a bar chart. Below the chart are options where the analysis can switch between linear and logarithmic scales for the timeline bar chart and the length of the intervals.

Chord. The last tab displays a chord diagram that shows the number of connections between pairs of hosts, always considering the number of connections initiated by a particular host. This tab is displayed only if Connection and Host nodes are in the selection. The chord diagram was chosen as a convenient way to visualize the volume of data transferred between two hosts allowing the analyst to identify those who communicated heavily.

5.3.2. Search

The analyst can use two types of searches. The first serves for selecting nodes in the current visualization (**R4**). The analyst may search nodes based on their labels, tags (if assigned), and intermediate or eigenvector centrality values (similarly to Page-Rank). This search makes it easy to select nodes even if the visualization becomes large and less clear. The second search utilizes extended Granef analysis API to query data from the graph database. The analyst can either write queries in the Data Query Language (DQL) or use predefined queries with user-defined parameters (see Figure 2). When submitting a query, the user can choose whether to clear the current visualization or add a new result. In addition, the analyst can choose whether the result should be clustered. There are two clustering methods: a single cluster where all new result data will be visualized as a single cluster, or timeline clustering that clusters Connection nodes based on the connection time. These options are beneficial when the query result is expected to contain a large number of nodes that would clutter the visualization (**R5**).

5.3.3. Timeline

The timeline shows the distribution of connections in the visualization and allows to filter off the data using a visual selection of the interval (**R4**). The main element is a bar chart displaying the sum of connections (y-axis) aggregated in sampled time intervals (x-axis). The connections are aggregated in the 5-minute interval by default, but the user can change this value in the visualization settings. The user can interact with the chart in two interaction modes, either by selecting *one time interval* or *multiple intervals*. In the first case, the user can select only one moment by clicking on the graph. This comes in handy in visualizing changes in the graph over time. In the second case, it is possible to select multiple intervals with gaps between them. Inactive intervals are gray and active intervals, matching the color of the connection nodes, are orange, as shown in Figure 5 (2). The timeline only influences Connection nodes and clusters containing that type. A cluster will only be hidden if none of the contained Connection fall into active intervals.

Below the diagram are buttons for faster navigation: to display the first and last interval, the next and previous interval, and the next and previous non-empty interval (i.e., containing at least one connection). In the multiple intervals selection mode, there are buttons to show/hide nodes in all intervals and an inversion of the selected ones. Additionally, users may change visualization settings and switch between linear and logarithmic scales for the bar chart, the length of the intervals, and the time boundary for which the timeline data will be recalculated.

6. Implementation

We have implemented the proposed approach of using relational graphs for exploratory analysis of network traffic data as an open-source module for the Granef toolkit (publicly available at `https://granef.csirt.muni.cz/`). The architecture of Granef UI is described in the first part of this section. The second part describes the workflow of graph nodes clustering since it is one of the key features of the visual analysis interface.

6.1. System Architecture

We have designed Granef UI iteratively with several versions, each reflecting the feedback from the domain experts who participated in user tasks and requirements formulation. From the early phases of the design process, we

aimed to create a client-server application using modern web technologies and open-source frameworks. Granef UI builds upon the Granef toolkit and extends it with an interface for visual analysis. It provides REST API over the Dgraph database with features allowing exploratory browsing of stored data and retrieval of aggregated information. This API is used by the server side of Granef UI, which processes analytical queries and stores information necessary for efficiently running the client side of the tool provided as a web application.

Besides the visual analytics interface, which is the main view of the tool where analysts work, the application consists of three other pages. The home page contains the list of cases (i.e., graph visualizations) stored in the database. The user can create new or filter, search or delete existing ones. Next, there are two configuration pages containing the global application preferences and user-defined tag management, respectively. The latter is used for node labeling during the case investigation. In the future, we plan to add a fourth page with a brief tutorial for novice users.

Both client and server sides of the application are written in TypeScript language. The client uses React.js, with Material-UI[7] and Nivo libraries[8] for GUI components and charts. The key library for implementing graph visualizations is Cytoscape.js[9] which was chosen due to its features supporting the graph analysis, extensibility, and a live community of developers. The server side is running on Node.js. It uses the NestJS[10] and TypeORM[11] frameworks and PostgreSQL database as storage. It provides a REST API allowing to store and search the database for visualizations, preferences, annotations, and annotated nodes. We also used Docker to ease deployment and facilitate the configuration of backend and frontend parts.

6.2. Clustering of Graph Nodes

Node clustering addresses the scalability requirement (**R5**) and overcomes the issues with decreasing graph clarity, as well as performance degradation with the growing number of graph elements. The logic of creating clusters is based on selecting nodes that are grouped into one node, preserving all adjacent edges. The computation starts by identifying the nodes and edges, which will not directly appear in the visualization because these will be replaced by the cluster. Their definitions are stored with the cluster definition, so they can be used when the cluster is opened. When all the nodes and edges are identified, the new node element representing the cluster appears colored in gray. Its size is calculated based on the number of contained nodes. Next, the computation proceeds by obtaining the definitions of the new *aggregated edges* that will be adjacent to the cluster. The edges' thickness corresponds to the number of base edges it replaces. Finally, the clustering index, which maps the base node identifiers (keys) to the cluster identifiers (values), is updated.

The cluster can be opened to the original relational diagram of nodes as follows. First, all edges adjacent to the selected cluster are found in their base edges. Then the cluster can be removed from the visualization and replaced by the inner nodes. The clustering index is also updated. Next, Granef UI maps the base edges to the cluster edges, and then adds them to the visualization.

Basic node clustering algorithms find suitable nodes mainly based on the numerical values computed from node attributes. However, this is not always appropriate since the logical graph topology needs to be considered. Therefore, the Granef UI offers three clustering options. The first is the Chinese Whispers (Biemann, 2006) algorithm – a random graph-clustering applying the idea of the "Chinese Whispers" game (also known as "Telephone"). Each node in the graph is given an initial label, and the labels are then iteratively updated based on the neighbors' labels. A node adopts the most common label among its neighbors in each iteration until the labels converge or a maximum number of iterations is reached. The second algorithm creates clusters by merging so-called outliers, i.e., nodes with a single neighbor. The third algorithm was designed with knowledge of the shared nature of network traffic data. The analysts are often interested in links modeled as single nodes and their contextualization in some time range. Therefore, it was natural to cluster the connections just by time stamps, which proved practical in practice.

Besides, the application also allows loading data in separate clusters. The idea behind the algorithm is first to give the user the opportunity to inspect statistics and diagrams and thus quickly get an initial idea at a higher level of abstraction than by looking at individual nodes in turn. At the same time, this does not prematurely reduce the clarity of the visualization. If the analyst is interested in the cluster, he or she can open it and further explore it at the level of individual nodes or smaller clusters.

7. Use Cases and Evaluation

To show how the analyst may utilize Granef UI, we have proposed two use cases representing different aspects of relational analysis. The analytical process for investigating these use cases is described in the first part of this section. We have also conducted a user study with domain experts. The findings from this evaluation are summarized in the second part of this section.

7.1. Use Cases

To illustrate Granef UI capabilities, we present two scenarios focused on incident analysis based on real-world scenarios. The first one deals with the intrusion detection system (IDS) alert analysis and investigation of a malicious host and shows the data enrichment capabilities, including

[7] `https://mui.com/`
[8] `https://nivo.rocks/`
[9] `https://js.cytoscape.org/`
[10] `https://nestjs.com/`
[11] `https://typeorm.io/`

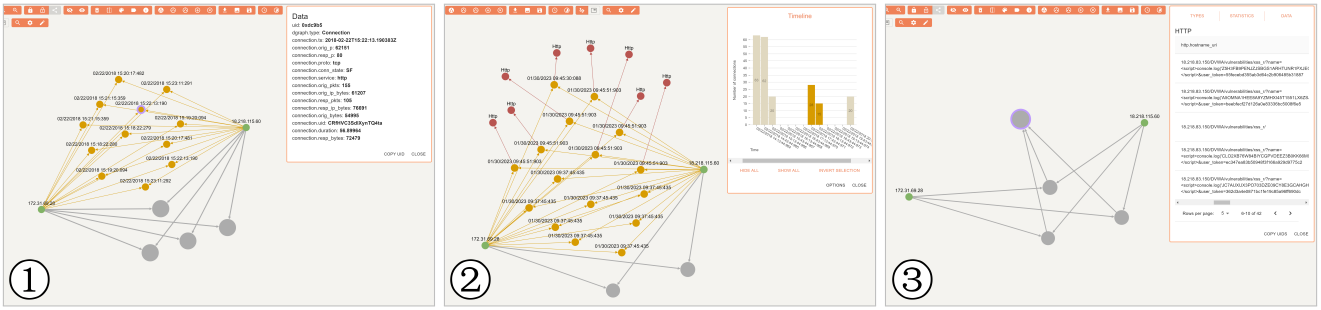

Figure 5: Visualization of analysis steps for Case Study 2: (1) initial event analysis with node data details; (2) connections and application data selection using timeline visualization; (3) clustered view of application data with details about the cluster content.

the use of external information from the threat intelligence database MISP[12]. The second scenario aims to analyze a web service attacks and documents Granef UI capabilities to process large amounts of network traffic data.

7.1.1. Use Case 1: Malicious Domain Connection

This use case is based on the SAPPAN dataset (Cermak and Obrecht, 2021) containing network traffic from a local network with multiple infected hosts communicating with the command and control center. The dataset is extended with the threat intelligence data from the MISP database describing indicators of compromise related to the used malware. The analysis starts with an IDS system alert identifying communication with a malicious domain. The goal is to verify the incident and determine its impact, as described in Section 4.1.

The first step – the false positive verification of the alert – is done by investigating related connections via the "Connections by hosts and/or time" search. The displayed relational graph shows two Host nodes and three Connection nodes, in whose detail we can see that they were normal HTTPS connections. Selection of all Connection nodes and fetching [Application] data using the context menu option adds several SSL/TLS nodes to the graph. Loading the next level of neighboring nodes (neighbors of the [Application] nodes) revealed the relation of all these connections to one hostname mentioned in the alert. Loading of hostname node neighbors confirms this as IoC and MISP nodes with details about the threat are added to the graph. These nodes, representing an external data source, are colored blue to be easily distinguishable.

To further investigate the incident, we continue to explore the relational graph by selecting neighbors of the malicious host. Fetching [Host-data] reveals an association of multiple hostnames and TLS certificates. Retrieving the neighbors shows that these nodes are also present in the MISP database, so we continue to check whether any other hosts in the local network have communicated with the malicious host. It shows to be a valid assumption, and by fetching its neighbors, the graph visualizes another Host node communicating with the malicious one. We may fetch connections between these nodes and investigate them further. The resulting graph, shown in Figure 2, displays the final result from which we can see all the communication between the malicious host and other hosts in the network.

7.1.2. Use Case 2: Web attack

In the second scenario, we explore the dataset CSE-CIC-IDS2018 (Sharafaldin et al., 2018), part Thurs-22-02-2018. The analysis starts with an alert with a brute force attack observed in a one-hour window. The investigation aims to verify the incident and check whether the attacker performed any other attacks. Figure 5 illustrates key steps in the analytical process.

We follow the same approach as in the previous use case using the "Connections by hosts and/or time" search. However, the bigger time window of the alert leads to a larger number of Connection nodes, making the visualization cluttered. Therefore, we aggregate the connections into clusters based on 15-minute intervals by "Timeline clustering". The detail of Connection nodes in the one opened cluster shows standard HTTP connections (see Figure 5 (1)). We then fetch the [Application] data and inspect them individually. The details of obtained HTTP nodes reveal that all connections refer to the login page, which confirms that the alert is a true positive.

To follow up on the analysis, we select the attacker Host node (IP: 18.218.115.60) and fetch all neighboring connection nodes (using "Timeline clustering"). As expected, more clusters will appear, so we use the Timeline child window revealing that the attacker has communicated with the server in three intervals, where the first corresponds to the alert. Repeating the procedure of alert verification for the second interval, we see the patterns ("<script>" tag in the URI) indicating the cross-site scripting attack (see Figure 5 (2)). Instead of opening a cluster and loading [Application] nodes for each connection, we can also select all clusters, load application neighbors as a single cluster, and investigate them in the Details child window (see Figure 5 (3)). Repeating the steps for the third time interval reveals an SQL injection attack. So the attacker tried three different attacks on the website.

[12] `https://www.misp-project.org/`

7.2. User Study

To verify that an analyst without more profound knowledge of the Granef UI can also effectively utilize the proposed analysis approach, we invited five experts in the cybersecurity data analysis domain to replicate the second use case. The participants had experience with cybersecurity data analysis between 5 and 12 years (8 years on average). The evaluation aimed not only to verify whether they are able to use Granef UI for alert analysis but also to check how they work with the visualization in general. We designed an online form to guide participants through the testing. It served to collect participants' feedback and present the instructions throughout the evaluation session. The sessions lasted between 45 and 60 minutes.

The evaluation sessions were realized in hybrid mode (i.e., we were present in person or online throughout the session and answered any questions). The session started with a brief introduction to the relational data model of network traffic data and a presentation of the key functions of Granef UI. Users were then given a set of simple tasks to control the visualization so that they could become more familiar with the tool and try out its features. Subsequently, the domain experts were pointed to the alert. Their task was to verify it and identify other possible attacks in the dataset. At the end of the evaluation, the participants provided feedback and filled out a System Usability Scale (SUS) questionnaire.

7.3. User Study Results and Discussion

After the initial indecision, each expert verified alert correctness, analyzed related network traffic, and identified the two additional attacks. The resulting SUS score was 78, which can be interpreted as an *Acceptable* or *Good (B+)* system in adjective interpretation or the numeric values (average score is 68). In addition, each expert stated that they could imagine working with the tool in the future and using it for certain types of network traffic analysis.

The biggest difficulty for participants was unfamiliarity with the data model, as they expected application data within the connection nodes, for example. That was a problem especially at the beginning of the alert verification task. However, after a short time, they got used to the model and could quickly analyze attacks from other time intervals and use the context menu to fetch related nodes. We attribute this initial confusion to the lack of visual demonstrations in the initial familiarization phase of the application. To avoid this in the future, we plan to extend Granef UI with an additional page containing a simple tutorial with analysis examples. A positive surprise for us was that the participants were able to use clusters intuitively, which allowed them to get an overview of the contained data quickly. Participants also intuitively started using neighbor node fetching when they did not know how to proceed. Using this naive approach, they obtained additional information that helped them to discover other attacks and finish the task.

The evaluation also revealed several bugs in implementation (e.g., incorrect alignment of elements, confusing button descriptions, and data loading indicator), which we subsequently corrected in new versions of Granef UI. We also gathered users' ideas for visualization improvements that we plan to implement in the future.

8. Conclusion

In this work, we presented a new approach to network forensics using relational graphs visualization. This data representation method offers a new way of data analysis inherent to the human brain that is used to analyze and understand a given problem by associating different information. Our work builds on the Granef toolkit (Cermak and Sramkova, 2021), which introduced processing and storing network traffic data as associations in a graph database. We designed and implemented an open-source module Granef UI, that utilizes the graph database of the Granef toolkit and allows the analyst to visualize the data in relational graphs and explore them using predefined queries or interactive fetching of related nodes. When designing the tool, we drew on discussions with domain experts and requirements derived from common tasks that an analyst performs during network forensics and incident investigation. To show how the implemented visualization tool fulfills these requirements, we presented two analytical use cases and performed a user study with five domain experts. Their results and positive feedback showed that the proposed analysis approach fulfills the defined requirements and allows the analyst to explore data at different levels of detail. At the same time, the evaluation showed that once the analysts got used to the new data model, they could quickly investigate the incident and reveal all necessary information. In the future, we plan to perform a more comprehensive evaluation with a larger sample of analysts and compare Granef with other commonly used tools to demonstrate its benefits and challenges.

We believe that Granef toolkit (available as open-source at `https://granef.csirt.muni.cz/`) extended by the visual interface will complement the portfolio of network traffic analysis tools and offer the forensics and cybersecurity community new insights into the data allowing them to uncover hidden data associations. We also expect that the relational analysis concept will be progressively used to analyze other data types, such as logs or system events. An interesting opportunity is a combination of multiple (heterogeneous) data types within a single relational analysis. As a result, the analyst could easily navigate within a unified visual interface – e.g., between data from network traffic and system processes that initiated the network connections – without combining different tools and their outputs. The broader use of the relational analysis concept thus has great potential to accelerate exploratory data analysis in both digital forensics and incident response.

Acknowledgments

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

References

A-Packets, 2022. Online PCAP file analyzer designed to visualize HTTP, Telnet, FTP. URL: https://apackets.com/. (Accessed: 2022-07-14).

Ah-Pine, J., Marcotorchino, J.F., 2010. Overview of the relational analysis approach in data-mining and multi-criteria decision making, in: Usmani, Z.U.H. (Ed.), Web Intelligence and Intelligent Agents. Intech, pp. 325–346. doi:10.5772/8387.

Atkin, H., 2011. Criminal Intelligence: Manual for Analysts. UNODC Criminal Intelligence Manual for Analysts, United Nations Office on Drugs and Crime (UNODC).

Biemann, C., 2006. Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems, in: TextGraphs-1: Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing, Association for Computational Linguistics, USA. p. 73–80.

Cermak, M., Obrecht, M., 2021. SAPPAN: Advanced Threat Data. doi:10.5281/zenodo.5547862. (Accessed: 2023-01-14).

Cermak, M., Sramkova, D., 2021. GRANEF: Utilization of a Graph Database for Network Forensics, in: Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT, INSTICC. SciTePress. pp. 785–790. doi:10.5220/0010581807850790.

Chen, Y., Malin, B., 2011. Detection of anomalous insiders in collaborative environments via relational analysis of access logs, in: Proceedings of the First ACM Conference on Data and Application Security and Privacy, Association for Computing Machinery, New York, NY, USA. p. 63–74. doi:10.1145/1943513.1943524.

Costa, Gianluca, 2019. CapAnalysis. URL: http://www.capanalysis.net/. (Accessed: 2022-07-14).

Cybersecurity and Infrastructure Security Agency, 2021. Cybersecurity Incident & Vulnerability Response Playbooks. URL: https://www.cisa.gov/sites/default/files/publications/Federal_Government_Cybersecurity_Incident_and_Vulnerability_Response_Playbooks_508C.pdf.

D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., Casas, P., 2019. A Survey on Big Data for Network Traffic Monitoring and Analysis. IEEE Transactions on Network and Service Management 16, 800–813. doi:10.1109/TNSM.2019.2933358.

European Network and information Security Agency (ENISA), 2010. Good Practice Guide for Incident Management. URL: https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management.

European Network and information Security Agency (ENISA), 2018. Introduction to Network Forensics. European Union Agency for Cybersecurity. doi:10.2824/995110.

Feijs, L., Krikhaar, R., Van Ommering, R., 1998. A relational approach to support software architecture analysis. Softw. Pract. Exper. 28, 371–400.

Fernandes, G., Rodrigues, J.J.P.C., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L., 2018. A comprehensive survey on network anomaly detection. Telecommunication Systems doi:10.1007/s11235-018-0475-8.

Ghani, S., Kwon, B.C., Lee, S., Yi, J.S., Elmqvist, N., 2013. Visual Analytics for Multimodal Social Network Analysis: A Design Study with Social Scientists. IEEE Transactions on Visualization and Computer Graphics 19, 2032–2041. doi:10.1109/TVCG.2013.223.

Leichtnam, L., Totel, E., Prigent, N., Mé, L., 2020. Sec2graph: Network Attack Detection Based on Novelty Detection on Graph Structured Data, in: Detection of Intrusions and Malware, and Vulnerability Assessment, Springer International Publishing. pp. 238–258.

Messier, R., 2017. Network Forensics. John Wiley & Sons, Ltd. doi:10.1002/9781119329190.

Neise, P., 2016. Intrusion Detection Through Relationship Analysis. Technical Report. SANS Institute. URL: https://www.sans.org/reading-room/whitepapers/detection/paper/37352.

PacketTotal, 2022. Simple, free, high-quality PCAP analysis. URL: https://packettotal.com/. (Accessed: 2022-07-14).

Sharafaldin, I., Habibi Lashkari, A., Ghorbani, A.A., 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, in: Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, INSTICC. SciTePress. pp. 108–116. doi:10.5220/0006639801080116.

Sikos, L.F., 2020. Packet analysis for network forensics: A comprehensive survey. Forensic Science International: Digital Investigation 32, 200892. doi:10.1016/j.fsidi.2019.200892.

Tabassum, S., Pereira, F.S.F., Fernandes, S., Gama, J., 2018. Social network analysis: An overview. WIREs Data Mining and Knowledge Discovery 8. doi:10.1002/widm.1256.

The Open Information Security Foundation (OISF), 2022. Suricata. URL: https://suricata.io/. (Accessed: 2022-07-14).

The Zeek Project, 2022. The Zeek Network Security Monitor. URL: https://zeek.org/. (Accessed: 2022-07-14).

Uhlár, J., Holkovič, M., Rusňák, V., 2021. PCAPFunnel: A Tool for Rapid Exploration of Packet Capture Files, in: Ebad Banissi, Anna Ursyn, e.a. (Ed.), 2021 25th International Conference Information Visualisation (IV), The Institute of Electrical and Electronics Engineers, Inc.. pp. 69–76. doi:10.1109/IV53921.2021.00021.

Ulmer, A., Sessler, D., Kohlhammer, J., 2019. NetCapVis: Web-based Progressive Visual Analytics for Network Packet Captures, in: 2019 IEEE Symposium on Visualization for Cyber Security (VizSec), pp. 1–10. doi:10.1109/VizSec48167.2019.9161633.

Zhang, H., Zeng, H., Priimagi, A., Ikkala, O., 2020. Viewpoint: Pavlovian Materials—Functional Biomimetics Inspired by Classical Conditioning. Advanced Materials 32. doi:10.1002/adma.201906619.

Zákopčanová, K., Řeháček, M., Bátrna, J., Plakinger, D., Stoppel, S., Kozlíková, B., 2021. Visilant: Visual Support for the Exploration and Analytical Process Tracking in Criminal Investigations. IEEE Transactions on Visualization and Computer Graphics 27. doi:10.1109/TVCG.2020.3030356.

Šrámková, D., 2022. Graph-based Anomaly Detection in Network Traffic. Master's thesis. Masaryk University, Faculty of Informatics, Brno. URL: https://is.muni.cz/th/mtuer/.