Visual Decision-Support for Live Digital Forensics

Fabian Böhm* Ludwig Englbrecht† Sabrina Friedl‡ Günther Pernul§

Chair of Information Systems
University of Regensburg
Germany

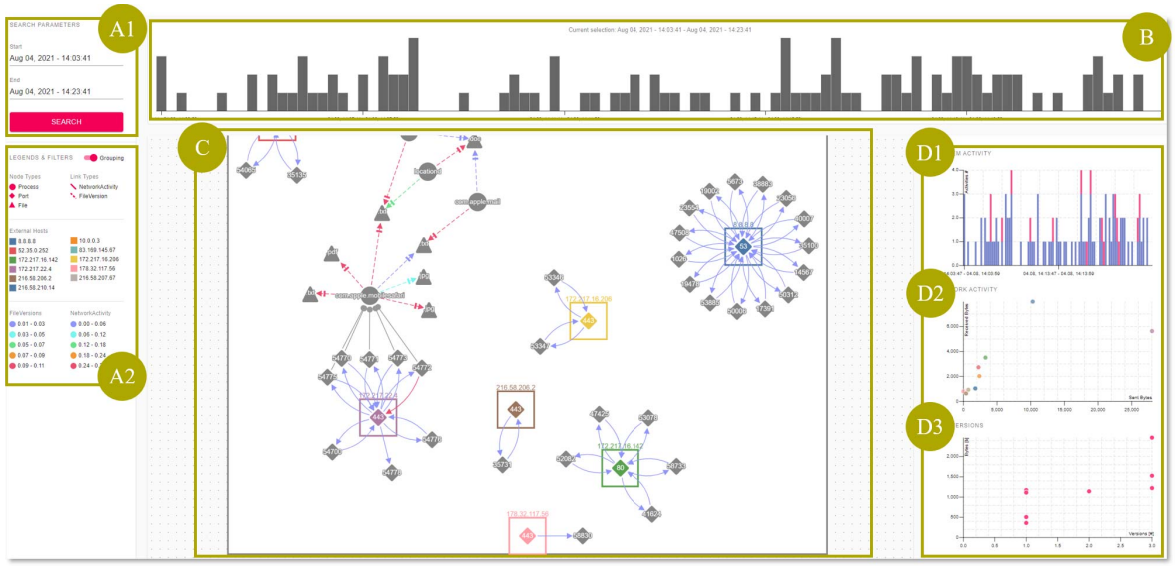

Figure 1: We introduce a visual analytics tool that supports cyber forensic experts to decide which parts of a system need further investigation. The tool provides four interactive views: *Search Parameters & Filters* (A1, A2), brushable *Overview* bar chart (B), interactive *Node-Link diagram* (C) to perceive the system's activities, and a supporting *Details-on-Demand* view (D1, D2, and D3).

ABSTRACT

Performing a live digital forensics investigation on a running system is challenging due to the time pressure under which decisions have to be made. Newly proliferating and frequently applied types of malware (e.g., fileless malware) increase the need to conduct digital forensic investigations in real-time. In the course of these investigations, forensic experts are confronted with a wide range of different forensic tools. The decision, which of those are suitable for the current situation, is often based on the cyber forensics experts' experience. Currently, there is no reliable automated solution to support this decision-making. Therefore, we derive requirements for visually supporting the decision-making process for live forensic investigations and introduce a research prototype that provides visual guidance for cyber forensic experts during a live digital forensics investigation. Our prototype collects relevant core information for live digital forensics and provides visual representations for connections between occurring events, developments over time, and detailed information on specific events. To show the applicability of our approach, we analyze an exemplary use case using the prototype and demonstrate the support through our approach.

*e-mail: fabian.boehm@ur.de
†e-mail: ludwig.englbrecht@ur.de
‡e-mail: sabrina.friedl@ur.de
§e-mail: guenther.pernul@ur.de

Index Terms: Applied computing—Computer forensics—; Human-centered computing—Visualization systems and tools—; Human-centered computing—Visual analytics—; Security and privacy—Intrusion detection systems—;

1 INTRODUCTION

Conducting live digital forensics (LDF) investigations is becoming increasingly important in the context of cyber forensics. While traditional forensic analysis methods are primarily applied to switched off devices (static analysis), LDF focuses its investigation on running devices [19]. The need for this alternative approach originates from a variety of recent developments. One of them is that live (i.e., real-time) forensics is imperative to analyze some specific malware types like fileless malware. This type of malware exists primarily in memory [24]. As soon as the infected device is turned off to perform a static forensic investigation, evidence of essential importance will be lost. Other examples where LDF is necessary are mission-critical or very large systems that cannot be shut down due to their importance to business operations. Thus, these reasons force forensic and security experts to act directly by collecting evidence while an attack is happening [15].

Experts for Digital Forensics (DF) conducting the live investigations have a wide range of forensic tools at their disposal, which they mainly apply according to their own experience. The tools are highly specified to collect and analyze a specific type of evidence and, therefore, differ widely [26]. During an LDF investigation, the targeted use of tools is essential to minimize the possibility of corrupting or even destroying evidence. For this reason, any use of a forensic tool on the device under investigation must be well-

2639-4332/21/$31.00 ©2021 IEEE
DOI 10.1109/VizSec53666.2021.00012

considered [8]. However, since the target device continues to be operational during the investigation, it is at the same time necessary to decide about the tools to be used timely. If the decision is made too slowly, this poses the risk that evidence will be corrupted or that the malware continues to spread. The live environment also leads to a high volume and velocity of data on which to base a decision for specific forensic tools.

As mentioned before, after having identified indicators for an attack or threat, forensic analysts need to decide which specialized forensic tools they should apply to work through the attack and collect evidence. This decision is based mainly on the experts' experience without any reliable automated solutions to support this decision-making. Therefore, in the context of a live forensic investigation, it is necessary to support domain experts in quickly getting an idea where on the device the attack manifests. From these locations, evidence can then be collected and analyzed. In cyber forensics, much of the respective evidence is related to the file system and changes in the file system. In a static forensic investigation, it is only possible to analyze user data on the persistent data storage, assuming that it has not yet been overwritten by further operation. However, in an LDF investigation, specific changes to persistent files, including the resulting file versions, can be incorporated into the analysis. Additionally, file system activities are often related to network communication (command & control server, leakage of information, or lateral movement of malware). It is, therefore, necessary for experts to be able to correlate the changes in the file system and the network activities of the device under investigation to identify indicators and decide on specialized forensic tools to apply. These observations lead to the main research question of our work:

RQ 1 *How can core information needed for Live Forensic investigations be acquired and systematically visualized for DF experts in order to support their decision-making?*

Our contribution to this research question is two-fold. In a first step, we derive general design requirements for an LDF investigation decision-support tool. Afterward, we present a prototypical visual decision-support tool that enables DF experts to analyze relevant volatile data from a compromised device. This approach differs from traditional DF as we provide a visualization pipeline that collects, pre-processes, and visualizes data that would no longer be available for post-mortem analysis. We collect the relevant data from a live and operative device, with physical access to the device being the only requirement. This information enables experts to make informed and justifiable decisions regarding the forensic tools to be applied.

2 Related Work

Several visual analytics (VA) tools exist that can be used for forensic purposes, such as malware or volume analysis. We show only a short selection of recent approaches that are related to our work.

The visual representations of the forensic analysis tool *Change-Link 2.0* allow experts to comprehend changes over time within shadow volume data [14]. Therefore, the authors propose metaphors to visualize directory structures, directories, file content, and directory metadata. While *Change-Link 2.0* provides insight into temporal changes of a file system, it does not show processes responsible for these changes and their external communication.

FIMETIS is a tool allowing to interactively explore file system snapshots [1]. The tool provides a security analyst with simple and straightforward analysis views for file system records, the temporal sequence of events, and data clusters. Although detailed insights into the file system are provided, the tool can't be applied for LDF as, in this context, file system snapshots are usually not available.

The research prototype *MalViz* is a tool for analyzing the behavioral patterns of malware [21]. Its intended use is to investigate the relationships and dependencies of a system's processes with an active malware. Using *MalViz*, DF experts can identify unusual patterns within the processes' activity more efficiently. *MalViz* can be applied for live forensics. However, users only gain an understanding of process activities without much-needed context about other relevant activities.

Another approach for forensic analyses of malware was introduced with *Eventpad* [4]. Eventpad's advantage is its capability to significantly reduce the complexity within network traffic samples to quickly understand the networking behavior of malware samples. Eventpad has proven to be highly effective for network data, but it does not help domain experts to identify internal indicators.

Furthermore, innovative approaches for the analysis of network packet captures (PCAPs) have been proposed [25]. They feature a web-based visualization design meant to support malware analysts and administrators in their tasks that frequently involve PCAP analyses. Although this tool is not explicitly designed with forensic analyses in mind, network traffic analysis is vital for live forensics but needs to be combined with information about the internal communications of a device.

Most of the existing visual designs in the context of forensic analysis focus on either static analysis, on a single data source, or at least on a specific type of data (e.g., network traffic or volume data). While this is undoubtedly helpful for the forensic analyst, especially concerning further investigation, there is no way to get a quick, initial overview of a system's activities during an LDF investigation. Thus, there is no support for domain experts in live forensics to help them make decisions on the forensic tool to apply.

The need for a separate visual tool is even more stressed through our design proposal within the cybersecurity field [2]. We partially build on this design idea as a reference point [20]. However, the existing design falls significantly short in several regards. First of all, it is a purely theoretical design. Thus, there is no proof that a respective tool would be feasible, especially concerning data availability. Second, our previous work is narrowed down to the proposed design and, therefore, not generalizable. It is possible and necessary to derive comprehensible requirements leading the design and development of respective visual tools.

3 Design Methodology

In this work, we design and implement a visual tool to support DF experts in deciding on the analysis tools to collect and analyze evidence. We follow the design methodology proposed by Meyer et al. [17]. Their method is strongly problem-oriented. Thus, it enables the derivation of appropriate design proposals. Additionally, an important reason for applying this methodology is to bridge the gap between domain and visualization experts [3, 23], and, to some extent, help to resolve the dichotomy of security visualization in the area of cyber forensics [16].

We identify the domain problem based on existing academic results, our own experience, and discussion with DF experts. All of these highlighted the need of experts for support in the tool selection during an LDF investigation. Again and again, the experts are confronted with not knowing which specialized forensic tool (e.g., volatility [1], autopsy [2], Cellebrite UFED [3]) they should use. They need to balance between acting as quickly as possible and at the same time not damaging or even destroying evidence by using the wrong tool. Although the DF experts are not directly involved in the design process, their precious input during informal discussions helped us determine their needs regarding data sources, possibly helpful visualization designs, and the tasks they need to fulfill and, thus, derive general requirements.

In Section 4, we firstly determine the data to be presented. In addition, in this step, we determine the concrete target group of the design proposal and define the essential tasks of this target group in

[1] https://www.volatilityfoundation.org/
[2] https://www.sleuthkit.org/autopsy/
[3] https://www.cellebrite.com/en/ufed/

the context of the domain problem. Finally, and building on these findings, general requirements for visual decision-support tools for LDF are derived.

Section 5 introduces the visual encodings we identify suitable to support the previously defined users' tasks. Our proposed design consists of multiple interactively interlocked visual representations allowing DF experts to comprise an investigated device's internal and external activities. Therefore, the prototype allows exploratory analysis leading to well-considered decisions regarding the forensic tools to apply in further and more detailed analyses.

Finally, Section 6 gives an insight into the technical implementation of the design proposal in the context of the research prototype created in this project. An agile development process is applied, in which the design is repeatedly discussed with the involved DF experts and, if necessary, adapted. An exemplary use case of the intended use of the prototype during an LDF investigation is highlighted in Section 7.

4 Requirements Analysis

In this section, we define the available data, the intended users, and their tasks as a basis for deriving requirements for actual visualization designs [18].

4.1 Data: System activities, File Versions, and Network activity

During an LDF investigation, a comprehensive picture of the current situation needs to be obtained. For this purpose, it is important to collect information about running applications, write and read operations on volatile as well as persistent storage media, and all outgoing and incoming signals (e.g., network traffic). This data must be retrieved in its raw form so that integrity is not compromised during the investigation process. In a subsequent step, this information must be correlated in a meaningful way so that the analyst can understand the relations.

We define the data at hand, which is relevant to provide a visual design supporting users in the problem domain. Analyzing a running and probably infected system is challenging. Available data during an LDF is limited to data acquired with little interaction and without installing additional software. Otherwise, the attacker could recognize that the system is being analyzed and initiate anti-forensic measures. Additionally, too much interference with the operating device might also cause evidence to be corrupted. Intending to support the decision-making process during a live investigation and to remain as undetected as possible, we identify the following data sources as relevant:

- **S1: System Logs.** Data about the activities of a running system can be used to support the decision-making process on the appropriate forensic tool. In this context, the analysis and evaluation of system logs play an important role. The system logs can be used to identify common and unusual events and to target the forensic analysis.
- **S2: Process information.** In addition to the system logs, continuous monitoring of the running processes is essential. This allows detecting irregularities in the running applications, such as a high CPU utilization or a high memory requirement. These are common indicators for an infection of the system.
- **S3: In- and outgoing network traffic.** Furthermore, information about the network activities of the device is important to detect communication with a Command & Control server or information leakage. This shows which process has sent data over which port to which destination address at what time.
- **S4: File-system activities.** Looking at the activities of the file-system provides deep insights into what files are used and modified during the execution of an application. The gathering of the file-system events does not include the actual content of the consumed or modified files but provides meta-information about the activities.
- **S5: Copy of modified files.** The acquired evidence is at the heart of a digital forensic investigation. The previously mentioned artifacts relate to volatile evidence. Those traces are not stored permanently on the disk and need to be captured and stored in a forensically sound way. We apply an additional tool for capturing the actual file content of modified files during the LDF investigation. This differs from the recording of metadata of the file-system by providing a snapshot of every file version. Although this form of backup can have an immense storage overhead, we use this data for the LDF. This makes it possible to examine the actual file contents for unusual changes in the content. Also, with this approach, a large base of usable digital evidence can be created, which can be used to prepare the DF report and consequently in court.

In order to identify suspicious activities on the system and the processes of an active attack, it is vital to understand the context of the various data sources. Only then is it possible for DF experts to apply the appropriate DF analysis tool in a targeted manner. The provided list of data sources highlights the most suitable ones but depending on the actual situation, it might be feasible to include additional data sources.

4.2 Users: Digital Forensics experts

After we are clear about the underlying data, defining the intended user group for the visual design is necessary. Subsequently, the target group has been delineated, defining the tasks they need to perform with the visual representation in the next step.

Since visual analytics is inherently user-driven, meeting the needs of users is critical. A thorough understanding of the users' needs, tasks, and work environment are required to achieve this. However, this is not easy to accomplish because problems in this area are ambiguous and, therefore users can be described in many ways.

We identify DF domain experts as the intended target group. There is a need for supporting their decision-making process, which forensic methods and tools apply during LDF. Forensic investigations are not to be carried out by security novices, as these analyses require considerable domain knowledge, which is only available among DF experts [3]. Thus, we expect the users to have the expertise to decide which forensic tool best fits the current activities on the device when supported adequately.

These experts are rarely exposed to enormous time pressure in the classic forensic investigation because they do not operate on a live system. Instead, they can perform the necessary static analyses based on memory dumps and similar static artifacts. With these copies, it is also possible to prevent evidence from being corrupted due to incorrect tool decisions by making a forensically sound copy of the hard drive apriori. However, this is not possible in the context of a live forensic investigation, and once corrupted, evidence is unrecoverable. In addition, any activity of the still-operational device can potentially corrupt evidence. This characterizes the target group of the intended visual decision-support tool as forensic domain experts who have to make a quick decision under time pressure in an ever-changing and possibly unfamiliar environment. In summary, the target group has solid domain knowledge but only limited operational knowledge regarding complex visualizations [6].

4.3 Tasks: Support forensic tool selection

The overarching task we want to support visually is to make a well-informed and thoughtful decision about the use of forensic methods, tools, and artifacts in the course of an LDF investigation. The goal is to accelerate and improve decision-making for domain experts. The

experts need to effectively identify potential indicators of malware or unusual activity to guide their further analysis process. The Live Forensic approach plays a vital role and recording volatile data or evidence that would be lost if the device were turned off. The analysis of volatile memory and the device's network activity is especially important in this as described in Section 4.1. For these tasks, experts have a wide variety of specialized forensic tools in their repertoire. In the real-time environment of an LDF investigation, a quick overview of the situation on the device plays a significant role. It is important to note that both from a technical perspective and increasingly from a legal perspective, continuous collection of the necessary data to support live forensics is not possible. Instead, the experts need a way to activate real-time data collection in case of suspicion and collect data from that moment on [16].

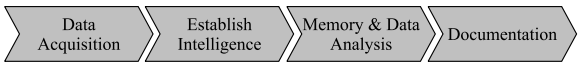

Figure 2: Forensic analysis process according to [13].

The related tasks of domain experts can be derived from the results of existing work [2, 19]. In this paper, we use a process model for LDF (see Fig. 2) to group the relevant tasks by commonly established process phases of a DF investigation. In the following, we describe the tasks in the context of the respective process phase:

1. **Data Acquisition:** DF experts need to *decide what data from the suspicious device or other systems needs to be acquired* for the investigation. This is not a simple task, especially in a live investigation since the decision needs to be made fast, and the situation can change quickly. Furthermore, the expert needs to *decide if additional software needs to be installed* to acquire appropriate data for the DF investigation. Any system interaction can be detected by an active attacker and lead to possible concealment actions. Usually, the amount of data that can be acquired without previously installed DF software is limited to the tools provided by the operating system. This leads to DF experts starting with limited data and successively acquiring additional information by selecting a suitable tool.

2. **Establish Intelligence:** Although DF investigations have a similar overall process, each attack confronts experts with different situations. During this process step, the skill is very much dependent on the expert's prior knowledge to initiate the correct next analysis step. No evidence is analyzed in-depth, but *it is decided which areas are helpful for the investigation*. There can also be a return to the previous task in which further sources are included to evaluate targeted areas. A visual representation of the current situation on the device under investigation supports analysts either *decide which areas to focus on for further analysis* or *which additional data sources need to be acquired*. A well-considered and well-informed decision at this stage is an essential success factor in resolving an attack.

3. **Memory & Data Analysis:** The previously *found and classified data is analyzed in detail* in this step. The data and information are put into context to *create a comprehensive picture of the current situation*. In particular, *unusual processes and correlations can be detected*. At this stage of a (live) forensic investigation, mainly specialized tools to are applied. Nevertheless, the domain experts still profit from a high-level overview of the device's live status to guide their analysis in the right direction.

4. **Documentation:** The *appropriate and accurate documentation of the analysis*, including every step of the expert, is a crucial aspect of the credibility of the acquired evidence. A comprehensive report supports the presentation of the facts in a court of law and the admissibility of specific evidence obtained after interacting with the compromised system.

In summary, these tasks clearly show that forensic scientists must make decisions regarding more advanced data acquisition or specialized tools at several points in the forensic analysis process. These decisions must be made repeatedly, especially in the second (*Establish Intelligence*) and third (*Memory & Data Analysis*) step of the process in an LDF investigation. It requires a clear overview of the data that can be collected without deep intervention on the investigated device. A visual representation of the data can support this overview and the associated decision-making process.

4.4 Requirements

Concluding the first part of our contribution, we derive a list of general requirements for decision-support tools within live forensic investigations. These requirements need to be fulfilled to support the users' tasks described in Section 4.3. They not only serve as a basis for a sound visualization design in the further course of this work but can be used for LDF decision-support applications in general:

- **R1: Retrieve and visualize only relevant, accessible data.** Each analysis has different prerequisites and, therefore, also different data is available in the *Data Acquisition* phase. Consequently, to keep the interference with the running system as low as possible, it is vital to retrieve only the data relevant and accessible within the DF analysis. This could be the usage of network activities, file-system activities, and information about running processes.

- **R2: Provide visual representations of the data in a timely manner.** As little time as possible should elapse between the retrieval of the data, its preparation, and the presentation. Thereby, a workflow of a real-time or (near-) real-time processing needs to be achieved to support the *Establish Intelligence* and *Memory & Data Analysis* phase.

- **R3: Document origin of and changes to the raw data.** The procedure for retrieving the files from the system must be explained in a comprehensible manner. If the analysis process results in modifications to the system and thus to the retrieved data, these must be meticulously documented. This strengthens the credibility of the data and allows the artifacts to be admitted as evidence. Additionally, it is a good approach to acquire and save the original data in its raw format in the *Data Acquisition* phase. All steps and findings must be recorded within the *Documentation* phase.

- **R4: Show significant behavioral changes over time.** All retrieved information has to be shown during the *Establish Intelligence* phase in a way that allows DF experts to detect significant changes (in- and decrease) in the device's behavior over time.

- **R5: Display available information in its situational context.** Information obtained from different data sources must be correlated with each other. The correlation can take place via timestamps and further attributes (e.g., process IDs). The resulting correlations should be visible during the *Establish Intelligence* step to the experts in the visual representation.

- **R6: Highlight conspicuous anomalies.** If, for example, processes or other entities of the monitored device behave conspicuously, this behavior should be visually identifiable. Conspicuous behavior can be when a process sends large amounts of data to a remote recipient via a wide variety of ports or

performs unusual processing of files. These findings move the focus from the *Establish Intelligence* phase to a deep *Memory & Data Analysis*.

- **R7: Allow exploratory analysis of the data.** Information is supposed to be visually abstracted for an analysis process so that an overview of a specific situation can be obtained within the *Establish Intelligence* phase. However, users should be able to explore the data and receive additional details when necessary through a *Memory & Data Analysis*.

5 Visual Design

This section describes the visualization techniques we employ for our proposed solution. The designs aim to make the data described in Section 4.1 accessible and understandable for DF experts based on the requirements from Section 4.4. With the help of the interactive design, DF experts can quickly grasp the essential indicators during an LDF investigation and decide on which other tools they need to employ for the analysis. Although data is captured and pre-processed in a real-time manner within our processing pipeline (see Sec. 6), the different views of our prototype are not updated automatically when new data from the monitored device becomes available. Instead, analysts must manually change the selected time window within A.1 to get the latest available data. In this context, the role of a DF expert differs from that of a security analyst who would have to react immediately. Therefore, we made this decision to ensure that the DF experts can focus on the analysis of the selected time range without being interrupted by live updates.

Thus, the design of the visualization techniques and the interactions follows the central guideline of the Information Seeking Mantra [22]. The overall design is depicted in Fig. 1. The period of interest is first determined by the search parameters (Fig. 1.A1). Users can then use the Overview diagram (Fig. 1.B) to get a first impression of the overall activity within the time period. More detailed information about the activities as well as the entities involved (processes, ports, files, etc.) can then be analyzed in more detail in the node-link diagram (Fig. 1.C) using the appropriate filters (Fig. 1.A2). The Details-on-Demand view (Fig. 1.D1 - D3) supports the overview as long as no entity is selected and otherwise displays detailed information about the selected object. The following sections are structured according to Fig. 1 and go into more detail on the designs regarding these visual representations.

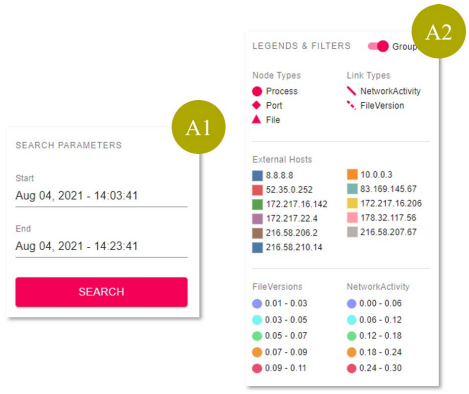

Figure 3: Search parameters (A.1) and available filters (A.2).

5.1 View A: "Search Parameters & Filters"

Fig. 3 shows search and filtering options available within our prototype. The *Search Parameters* (A.1) is the actual starting point of the design. Users select the start and end times for their analysis. Afterward, only the data that lies within this time window will be considered in the other views (**R2**, **R7**).

Fig. 3.A2 shows the *Legends & Filters* for the node-link diagram described in Section 5.3 including comprehensive filtering options (**R7**). As used in our prototype, a major disadvantage of node-link diagrams is that they tend to display only a "hairball" when the number of nodes is high or when the nodes are highly interconnected (high number of edges). Such networks can no longer be perceived by users and are thus almost useless. This shortcoming affects network diagrams regardless of the layout algorithm used. To address this drawback in some way, we implement a series of interactive filters that can be used to hide or show individual nodes, edges, or even entire classes of nodes or edges. This allows users to influence the graph's layout themselves and reduce the number of elements displayed if the automatic layout no longer produces satisfactory results. In addition to the possibility of explicitly filtering types of nodes and edges in the graph, we also offer users other interactive filters, which we describe in Section 5.3.

Figure 4: Bar chart for an overview of all activities with a specific time range brushed.

5.2 View B: "Overview"

The *Overview* chart presents the overall activity of the device in terms of file versions (**S4**) and network traffic (**S3**) for the time period selected in the *Search Parameters*. For this purpose, a simple bar chart is used in our prototype. Each bar represents the number of activities within a certain period of time. The size of this period is dynamically determined depending on the size of the total time window selected in the *Search Parameters*. The bar chart allows users to quickly perceive when the device has conspicuously high or very low activity (**R4**).

The bar chart for an overview of the device's activities is interactively interlocked with all overviews. A Brush interaction allows analysts to select a time window from the overall graph window. Fig. 4 respectively shows the *Overview* with an active Brush selection. The node-link diagram (see Section 5.3) and the detailed views (see Section 5.4) display only the activities within this brushed time frame (**R7**).

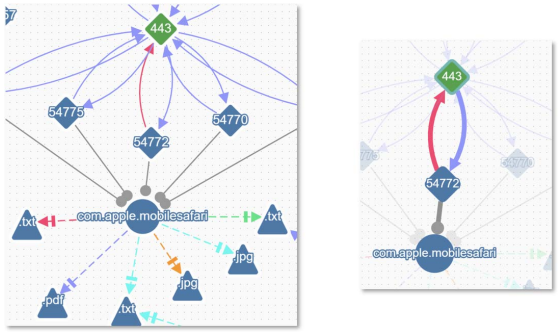

Figure 5: Detailed views of the node-link diagram. One showing the ungrouped display (left) and the other showing a highlighted node with its neighbors (right). Ports are displayed as rectangles, files as triangles, and processes as circles.

5.3 View C: "Node-link diagram"

The node-link diagram gives insight into the activities of the device during the time window selected in the *Overview*. If no brush selection is made, it displays the entire time range from the *Search Parameters*. The network-based encoding allows DF experts to perceive very active nodes (i.e., nodes with a high number of links) or suspiciously inactive nodes vice versa. Thus, users can identify possible indicators for further and more detailed analyses (**R5**, **R6**).

We apply a directed, compound node-link diagram that uses the fCoSe-layout to arrange nodes in the available display space. fCoSe is an improved version of the original CoSe layout algorithm [7]. While maintaining good results even for relatively large datasets through a force-directed layout scheme, fCoSe is more efficient and works well for directed graphs. To ensure a smooth user experience and enable exploration of parts of the graph for DF experts, the network diagram supports zooming and panning (**R7**).

Different data elements intended to be displayed are processes (**S2**), files (**S4**), and ports including the host the ports belong to (**S3**). These data elements are represented as vertices, and different glyphs indicate their type. They are colored based on the host to which they can be allocated. Most of the ports, all files, and all processes belong naturally to the device that is being monitored. However, other relevant hosts become apparent through the network connections as the device sends or receives network packages to or from their ports. When the *Grouping*-option (see Fig. 3.A2) is activated, the hosts are displayed explicitly as parent nodes for ports, processes, and files. When this option is switched off, the hosts are not displayed and indirectly influence the display through the categorical coloring of the vertices. Nodes are draggable to ensure that users can adjust the layout according to their needs or if the layout algorithm does not produce an apt result. To further give the possibility to reduce the number of nodes to be displayed, filter options are available to hide specific node types ("Node Types" filter in Fig. 3.A2) or to hide nodes of a specific host. Selecting a node displays more details (see Sec. 5.4) and highlights incoming and outgoing links from this node as well as its 1st-degree neighbors (**R5**, **R7**).

Three different types of links connecting the nodes are present as can be seen in Fig. 5 (**R5**). Each link type represents a specific activity on the investigated device:

1. "FileVersion"-links from a Process-node to a File-node indicate that the respective process edited at least one new version of this file (**S2**, **S4**).

2. "NetworkActivity"-links between two Port-Nodes represent network communication from one node to the other (**S3**).

3. "PortActivity"-links from a Process-node to a Port-node mark the attempt of a process to open a network connection via the respective port (**S1**, **S2**, **S3**).

Each type of link is displayed as a different line. Not every activity is drawn as a particular link but unique links where the respective activities are accessible through the *Details-on-Demand* window after selecting the link (**R7**). "PortActivities" are not colored, as no additional information is available. We only know whether a process used a specific port or not. However, "FileVersions" and "NetworkActivities" are colored according to their weight concerning all other displayed links of the same type (**R6**). The weight for "FileVersions" is calculated as the number of bytes edited by the process (source of the link) in the file (target of the link) divided by all file edits made within the selected time range. The weight for "NetworkActivities" is determined similarly by the sum of the package size of all packages that went through the respective link divided by the number of bytes sent between ports overall. We use a quantile color scale with only five different colors for each link type. Although this reduces the level of detail, it makes anomalous links stand out much more clearly. Types of links can, analogously to nodes, be hidden through a filter ("Link Types" filter in Fig. 3.A2), but also specific quantiles of link weights can be hidden to only view very high or very low weighted links ("Network Activity" and "File Version" filter in Fig. 3.A2).

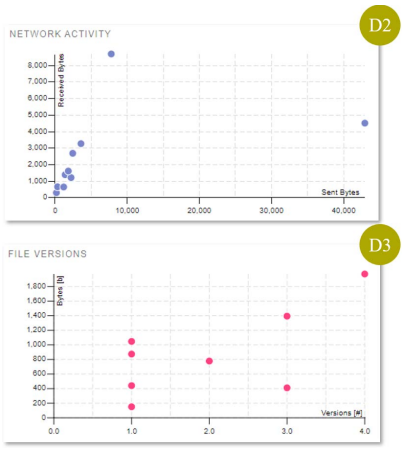

Figure 6: Scatterplot to support users in identifying abnormal network and file system activities.

5.4 View D: "Details-on-Demand"

This part of the prototype's interface serves a two-fold objective. First, it supports experts in gaining an overview of the ongoing activities as long as no specific element in the node-link diagram is selected. Second, when experts click a node or link, it displays further details about this element.

When supporting the overview task, the *Details-on-Demand* view displays three additional charts:

1. System Activity (Fig. 1.D1): This stacked bar chart gives a close-up view of the overall system activity within the brushed time period. File version activities and network activities are distinguishable through distinct colors in this chart (**R4**).

2. Network Activity (Fig. 6.D2): This scatter plot renders one dot for each host the investigated system has communicated with. This dot indicates a sent-receive-ratio in terms of bytes sent to or received from the respective host. The x-axis of this plot maps the total amount of bytes sent to a host, while the y-axis marks the sum of bytes received from it. This helps users to very efficiently spot conspicuous connections either which differ from benign behavior (**S3**, **R6**). One example is the dot on the far right of Fig. 6.D2 indicating that the device sent a lot more data to this host than to others.

3. File Versions (Fig. 6.D3): This scatter plot displays additional information about the file versions. Each dot depicts a specific file. The x-axis shows the number of times this file was edited, while the y-axis shows the sum of overall added or deleted bytes. This is calculated simply by summing up the byte differences from one version of a file to the next one. This again helps DF experts to explore the data and spot anomalous activities (**S4**, **R6**). A high number of files touched one time might indicating an ongoing data leakage action.

For details on a selected element, the charts D1, D2, and D3 are replaced with the view shown in Figure 7. This view shows all available attributes of the selected element. When the analyst clicks a link, all underlying events are accessible through this view (**R7**).

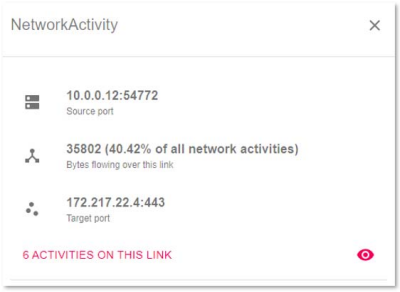

Figure 7: Displayed details after a link is selected. Clicking the "eye" symbol brings up a modal with further details on all relevant activities.

File versions can also be downloaded for further investigation with specific forensic tools (**S5**).

6 Research Prototype

We finally take a closer look at the implementation of our prototype. This includes a significant amount of work to acquire the relevant data with as little interference as possible to fulfill both **R1** and **R2**. This section describes our data acquisition approach, how the information is processed and finally sheds light on the actual implementation of the above-described visual representations. The structure of this section is based on the visualization pipeline [5]. The visualization design introduced in Section 5 represents the technical outcome of this pipeline rendered on the client-side of our prototype.

Fig. 8 shows the basic architecture of our prototype, including the applied technologies. Before we explicitly discuss the individual steps of the pipeline within our prototype, we briefly describe its overall structure. It is designed as a client[4]-server[5] application with an underlying document-oriented database[6] for persisting the raw data acquired from the *Data Collection* and the pre-processed data. The pre-processed data is the output of the *Data Analysis* step, in which a Python script continuously prepares newly available raw data. Further data analysis is performed by the specialized application SauvegardeEx, and the results are available to our prototype through an interface. A Node.js application represents the central component of the server for *Filtering* and partial *Mapping* (i.e., correlation) of the pre-processed data. The client's interface to the server is implemented through a GraphQL API. The client itself is a React-based application. Two different frameworks are used for the client-side *rendering* of the data in the form of interactive visualizations. On the one hand, the framework VisX[7] is used to display the bar chart, scatterplot, and alike. On the other hand, due to the potential complexity of the network graphs to be displayed, we resort to the specialized graph analysis framework cytoscape.js [12].

6.1 Raw data collection

A general requirement in the context of live forensics is that the system under investigation should be affected as little as possible by the investigation (**R1**). This poses a significant challenge, especially for the collection of data. For our prototype, we focus on appropriate data collection for mobile devices. In discussions with the experts involved, it became clear that these devices, in particular, are currently causing difficulties for DF experts, as there are few established procedures for collecting data for live forensics from active mobile devices. In most cases, the appropriate software must be integrated into the device before the investigation. With our approach, we trade-off between a highly detailed data collection and the need to install additional functionality or application on the device. Our prototype's complete data collection process can be performed using already available functionalities on a mobile device. The only requirement for this is physical access to the device to apply various forensic hard- and software.

Our prototype acquires data from an Apple iPhone operating on iOS 14.5. It allows extraction of information about active processes, file editing activities, and information about the ports used by the processes for communicating over the network from this smartphone. The respective information is collected using different appliances:

syslog: First of all, we acquire all system-logs by using *libimobiledevice* [8] and save them in JSON format (**S1**).

ps: Second, meta-information (like the underlying services' names, CPU usage, among others) of processes are collected using the Linux-tool *ps* which we use to extract a list of active processes every 5 seconds. *ps* is a lightweight tool and does not affect the device's performance significantly. It fulfills the data-source (**S2**).

PCAP: Besides this detailed information about the processes' activities (i.e., internal activities), we connect the iPhone to a network access point collecting network activity information (i.e., PCAP files). Acquiring and incorporating the in- and outgoing network traffic refers to data (**S3**).

syslog: Since the *netstat* service to receive information about the network communication of individual ports is not available on iOS 14.5, we include the *syslog*-appliance in our data collection process. *Syslog* can be used to collect Syslog information from the system. Besides much other information, they also contain information about processes opening and closing a network connection over a specific port. Although we do not get any information about the destination of the network connection, we at least can collect some rudimentary data about the processes network communications (**S3**).

fsmon: Highly volatile information about the file editing activities of processes and the resulting file versions is extracted using the *fsmon*-appliance (**S4**) and forwarded to *SauvegardeEx* which is a specialized application to process *fsmon* information. This mechanism creates a copy of every modified file on the system (**S5**). We will describe its functionalities within Section 6.2.

The above-described approach for data collection targets the use of existing data sources (**R1**) in a direct and timely processing of the data (**R2**) in an indirect manner. To fulfill the requirement (**R3**) of recording the origin and all changes of the data for proper traceability and documented we store the data in the following manner: All the collection processes can be triggered via a command-line instruction on the device and simply connecting the iPhone to the PCAP-collection access point. Besides the information extracted with *fsmon*, all raw data from this process step is forwarded in JSON format into a central MongoDB database.

6.2 Data Analysis

The *Data Analysis* step of our prototype mainly comprises a correlation of the previously collected raw data. This is necessary because the raw data itself does not allow visualization of connections between processes, file edits, and network communication. Therefore, two components in our architecture perform the data analysis: the *Python pre-processing* script and the *SauvegardeEx* application.

Python pre-processing : We apply an iterative, continuous preprocessing, which includes several steps for every raw data source. We will only describe exemplary ones briefly in this work. Any new document (i.e., new raw data element) that is pushed to the database by the command line tools applied in the *Data Collection*-step is further processed to ensure the data can be adequately visualized. As *ps* does only allow to extract its data encoded, we need to decode this data and reduce the amount of respective information by only persisting the relevant information for the processes (i.e., process

[4] https://github.com/bof64665/LDF_ReactFrontend
[5] https://github.com/bof64665/LDF_GraphQLServer
[6] https://www.mongodb.com/
[7] https://airbnb.io/visx
[8] https://libimobiledevice.org/

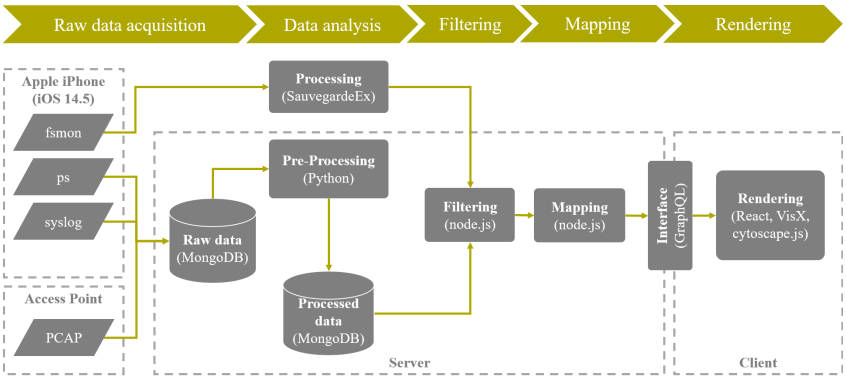

Figure 8: Architecture of the prototype.

id, name, CPU, and memory usage at a given point in time). Similarly, we parse and filter the *syslog* messages as only the messages containing information about processes opening or closing network connections via a specific port. A data reduction similar to the one applied to *ps*-data is also performed on the PCAP information. Although PCAPs contain a highly detailed description of each package leaving or reaching the iPhone, we are only interested in a comparatively small part of this description. We thus pre-process every PCAP transferred to our raw data storage and remove any currently unnecessary fields. Note that raw data is not overwritten in our process, as it might become relevant for further analysis. Thus, all pre-processed data is persisted separately from the raw data (**R3**).

SauvegardeEx: To gain useful insights into the generation and alteration of relevant files during the operation of a live system, we incorporate the tool *SauvegardeEx* [9]. *SauvegardeEx* has been used and extended in our previous works [9, 10] to support a digital forensic investigation. The client version of *SauvegardeEx* and sends every single file alteration with the actual file content to the server. Thereby, a specific file can be restored at any captured point of time, and the challenge of overwriting a freed storage area (due to deletion or updating a file on the file system) is addressed. This information can be retrieved from the *SauvegardeEx* web server through a well-defined API. Having this mechanism in place during the LDF investigation, content-specific information of every file-version can be obtained and used for the visualization.

6.3 Filtering

Filtering in the prototype is done entirely on the server-side. Based on the user's input at the start and end time of the time window to be analyzed, the relevant information is retrieved from the database and the SauvegardeEx API in this step.

6.4 Mapping

Due to the potentially tremendous amount of data that can still be relevant after the filtering, we also perform the *Mapping* almost entirely in the backend. This keeps the computational load for the frontend as low as possible and ensures that the interactive visualizations can be operated smoothly. The first step in the mapping process is thus the correlation of the data sets. Here, we correlate all data sources and obtain a data set with information about the file versions, the processes responsible for them, the ports used by the processes, and the network activities originating from these ports. The correlation is done based on the timestamps and various attributes, which allow a clear assignment of the individual events. The second sub-step is an aggregation of the correlated information into containers. Currently, we use a fixed number of 100 containers, whose size is dynamically determined depending on the selected time window.

The actual mapping and thus the last step maps the now available filtered and aggregated information to nodes and edges and all other geometric forms needed for the visualizations. However, each element additionally retains its original data to ensure access to details on-demand. This geometric information is then made available to the client via the GraphQL API.

7 Use Case

The prototypical implementation of the visual designs (see Section 6) contributes to the decision-making process according to an optimal first-hand tool by displaying the available information in its temporal and situational context (cf., R4 and R5 in Section 4.4). By applying our tool during an LDF investigation, preserving important evidence can also be supported by storing volatile data in a dedicated database. To illustrate the applicability of the prototype, we highlight its key features through an exemplary use case.

The attack pattern in this use case is based on the "Jeff Bezos Hack", where attackers applied various fraudulent techniques to obtain data from the personal iPhone X of the Amazon founder and CEO Jeff Bezos. After a meeting between Bezos and the crown prince of Saudi Arabia, Mohammad bin Salman in 2017, they exchanged phone numbers and wrote ordinary messages via WhatsApp. Shortly after Bezos received a video sent by bin Salman in 2018, his smartphone started sending large amounts of data via the Safari Mobile browser and the Apple mail program. The subsequent investigation and forensic analysis of the smartphone [11] brought to light that the compromised video (probably) contained malicious code. To this date, this was a zero-day vulnerability. The Pegasus and Galileo spyware were the most likely tools used in this attack.

We use this incident as a potential scenario for our prototype and describe the procedure during a forensic investigation from an expert's point of view with the help of the decision-support tool. We rely on the publicly available report from FTI Consulting [11] published in 2019 and extend it with additional details concerning activities of the file-system. Please note that there is no official data available from the "Jeff Bezos Hack" that would allow the comprehensive reproduction of the incident. We instead derive a set of artificial data which is available within the code repositories of our prototype. Thus, the following sections do not describe an in-depth evaluation of our prototype but rather a possible scenario where the prototype could have been applied. In the following subsections, we describe relevant steps that must be performed during an LDF analysis of similar cases. Where appropriate, relevant indicators (and their recognition) are presented and linked to the visual representations of our prototype.

[9] `github.com/LudwigEnglbrecht/sauvegardeEX`

7.1 Initialization

First, the device's user might have noticed suspicious events about half a year after receiving the WhatsApp message with a video attached. This manifested itself in strangely ambiguous WhatsApp messages (GIFs, pictures, videos) from Mohammad bin Salman, which reflected current situations from Jeff Bezos' private life that were not known to the public at that time (e.g., divorce from his wife). Ultimately, such messages prompted the victim to initiate a forensic analysis of his device. This initial suspicion is used as a starting point in the exemplary application of our prototype. After the initialization of the investigation, the core process steps (see Fig. 2) are explained and related to our prototype.

7.2 Data acquisition

In a first analysis step, the smartphone is acquired and physically available for an investigation. An initial analysis of the suspected video attachment in WhatsApp does not highlight any indications of active malware. However, there still is a possibility that Advanced Persistent Threats (APT) are used. Therefore, DF experts decide to perform a further forensic investigation. Based on these findings, a live forensic investigation is applied to extract both a decrypted and encrypted forensic image of the iPhone X. Since the forensically-sound copy of the iPhone X requires higher rights (root), the software *Cellebrite UFED 4PC* is used to gain root access without the necessity to reboot the device. After a forensically-sound copy of the current state has been stored, an in-depth live analysis of the running (and eventually compromised) system can be conducted. At this point, the raw data collection of our prototype is initiated.

7.3 Establish Intelligence

Narrow the analysis time-frame. In our prototype, in the *Search Parameters* (see Fig. 3.A1) the last 20 minutes are selected. With no additional filters applied, the views display all respective data. Meanwhile, data will be pushed continuously into the database and is available for further visual analysis. In this step, the system is intentionally left running without performing any actions. This enables to capture as much background activity (including unusual, suspicious activity) as possible. Since the attacker can be active during this time, valuable indicators and traces can be obtained. At this stage, the expert can spot and select a time-frame with a high amount of system activities in the *Overview* (Fig. 4.B).

Identify conspicuous network activity. In this step, the network scatter plot (Fig. 6.D2) reveals that the device is sending an unusually large amount of data to a specific IP address. This is illustrated by the far right dot on the referred figure. A further analysis of the traffic using *Wireshark*[10] provides more details about the amount of the data that is transmitted. Consequently, an expert can choose in this situation *Wireshark* as the following suitable, DF tool to apply.

Correlate network activity with processes and file system activity. The high-level insight by using the scatter plot (Fig. 6.D2) brings up the need for a deeper analysis of the actual data content of the connections as it seems that the device is sending a lot of data to one specific host. In the node-link diagram (Fig. 5), an expert can see that a process establishes connections to an endpoint using port 443 and sends a considerable amount of data to this site. The related application at the iPhone is the web browser *Safari*. Since the device is not used at that time of investigation to browse websites that correspond to the displayed connection, it can be concluded that this is an unusual occurrence. To investigate this, an expert could decide to use, for example, the *Telerik Fiddler*[11] tool for a detailed, in-depth inspection of the traffic. By doing this, it is possible to confirm the previously gained knowledge with *Wireshark* and to increase the level of certainty of the evidence.

[10] `https://www.wireshark.org/`

[11] `https://www.telerik.com/fiddler`

Determine files for in-depth analysis. By observing and confirming large output data on specific connections that emerged from the network analysis in the steps before, the expert now goes further by taking a close look at the file system and created file versions. In Fig. 6.D3 the scatter plot shows that more files have been created but only few files have been modified during the period of investigation. This procedure makes it possible to see whether data has been copied to a temporary directory, are compressed (zip file) or split up, to be exported via exfiltration vectors (e-mail client, safari mobile). Such file operations and the copies of all created file versions with their content can further be investigated using the tool *The Sleuth Kit* which is the expert's decision for the next tool.

7.4 Memory & Data Analysis

Based on the selected forensic tools, the actual in-depth forensic analysis takes place here. In the considered use case, the memory and the extracted data of the iPhone X are examined with the tools. This enables the expert to find out more precisely what happened (e.g., fileless malware sent via WhatsApp). The interactions with the DF tools are not in the scope of our prototype.

7.5 Documentation

Our prototype brings together relevant data for a specific point in time to help experts deciding what DF tool is the most suitable in this specific situation. In our exemplary application of the prototype to the use case, only a small selection of tool decisions was presented. In the documentation phase, the data and the investigation proceedings are to be summarized in a report and conclude the analysis. This task is not directly addressed within our approach, but the supporting aspect of our prototype was illustrated.

8 Conclusion

The proposed and implemented research prototype provides cyber forensic experts with decision-support during an LDF investigation. Thus, the developed tool provides a solution that supports the initial selection of more specific forensic tools.

The design of the tool followed a problem-oriented approach. Further, the requirements of the application are sharpened through a requirements analysis which is divided into the areas (1) Data, (2) Users, and (3) Tasks. This methodical approach allows us to derive general design requirements applying to visual decision-support systems for live forensics. We implement the requirements in a prototype showcasing how they can support forensic analysts in investigating a mobile device. We extract relevant data from an iPhone within the prototype, pre-process this data, and display it in an interactive web application. The user interface features different possibilities for the experts to explore the data and identify targets for further, in-detail forensic analysis. Several bar charts and scatter plots are arranged and interactively interlocked with a node-link diagram to ensure this support of users' tasks. An exemplary use case underlines that the research prototype fulfills the requirements.

Only little empiric evidence of the design's effectiveness is currently available. It is yet to be evaluated in a real-world setting. To do this in an appropriate way, we are integrating our prototype into an existing professional learning, hands-on workshop for DF experts. The existing workshop will be extended to include our decision-support tool. This requires a considerable amount of additional work. However, in this way, workshop participants come into contact with possible visualizations supporting their work. Thus, empirical data can be collected about the prototype. Based on this data, further development of the prototype can then be carried out.

Acknowledgments

This work is partly performed under the BMBF DEVISE project which is supported under contract by the German Federal Ministry of Education and Research (`https://devise.ur.de/`).

REFERENCES

[1] M. Beran, F. Hrdina, D. Kouril, R. Oslejsek, and K. Zakopcanova. Exploratory Analysis of File System Metadata for Rapid Investigation of Security Incidents. In *2020 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 11–20. IEEE, Salt Lake City, UT, USA, 2020. doi: 10.1109/VizSec51108.2020.00008

[2] F. Böhm, L. Englbrecht, and G. Pernul. Designing a Decision-Support Visualization for Live Digital Forensic Investigations. In A. Singhal and J. Vaidya, eds., *Data and Applications Security and Privacy XXXIV*, vol. 12122, pp. 223–240. Springer International Publishing, Cham, 2020. Series Title: Lecture Notes in Computer Science. doi: 10.1007/978-3-030-49669-2_13

[3] F. Böhm, M. Vielberth, and G. Pernul. Bridging Knowledge Gaps in Security Analytics:. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy*, pp. 98–108. SCITEPRESS - Science and Technology Publications, Online Streaming, 2021. doi: 10.5220/0010225400980108

[4] B. C. Cappers, P. N. Meessen, S. Etalle, and J. J. van Wijk. Eventpad: Rapid Malware Analysis and Reverse Engineering using Visual Analytics. In *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, Berlin, Germany, 2018. doi: 10.1109/VIZSEC.2018.8709230

[5] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. The Morgan Kaufmann series in interactive technologies. Morgan Kaufmann Publishers, San Francisco, Calif, 1999.

[6] M. Chen, D. Ebert, H. Hagen, R. S. Laramee, R. van Liere, K. Ma, W. Ribarsky, G. Scheuermann, and D. Silver. Data, information, and knowledge in visualization. *IEEE Computer Graphics And Applications*, 29(1):12–19, 2009.

[7] U. Dogrusoz, E. Giral, A. Cetintas, A. Civril, and E. Demir. A layout algorithm for undirected compound graphs. *Information Sciences*, 179(7):980–994, 2009. doi: 10.1016/j.ins.2008.11.017

[8] W. G. Eckert. *Introduction to forensic sciences*. CRC Press, Boca Raton, Fla., 1997. OCLC: 824195425.

[9] L. Englbrecht and G. Pernul. A Combined Approach for a Privacy-Aware Digital Forensic Investigation in Enterprises. *Journal of Cyber Security and Mobility*, 2021. doi: 10.13052/jcsm2245-1439.1012

[10] L. Englbrecht, S. Schönig, and G. Pernul. Supporting Process Mining with Recovered Residual Data. In *The Practice of Enterprise Modeling*, vol. 400, pp. 389–404. Springer International Publishing, Cham, 2020. doi: 10.1007/978-3-030-63479-7_27

[11] A. J. Ferrante. Project CATO. Technical report, 2019. https://assets.documentcloud.org/documents/6668313/FTI-Report-into-Jeff-Bezos-Phone-Hack.pdf.

[12] M. Franz, C. T. Lopes, G. Huck, Y. Dong, O. Sumer, and G. D. Bader. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, p. btv557, Sept. 2015. doi: 10.1093/bioinformatics/btv557

[13] K. Kent, S. Chevalier, T. Grance, and H. Dang. Guide to Integrating Forensic Techniques into Incident Response, 2006.

[14] T. R. Leschke and C. Nicholas. Change-link 2.0: a digital forensic tool for visualizing changes to shadow volume data. In *Proceedings of the Tenth Workshop on Visualization for Cyber Security - VizSec '13*, pp. 17–24. ACM Press, Atlanta, Georgia, 2013. doi: 10.1145/2517957.2517960

[15] S. Mansfield-Devine. Fileless attacks: compromising targets without malware. *Network Security*, 2017(4):7–11, 2017. doi: 10.1016/S1353-4858(17)30037-5

[16] R. Marty. *Applied security visualization*. Addison-Wesley, Upper Saddle River, NJ, 2009. OCLC: ocn227921903.

[17] M. Meyer, M. Sedlmair, P. S. Quinan, and T. Munzner. The nested blocks and guidelines model. *Information Visualization*, 14(3):234–249, 2015. doi: 10.1177/1473871613510429

[18] S. Miksch and W. Aigner. A matter of time: Applying a data–users–tasks design triangle to visual analytics of time-oriented data. *Computers & Graphics*, 38:286–290, 2014. doi: 10.1016/j.cag.2013.11.002

[19] N. R. Mistry and M. S. Dahiya. Signature based volatile memory forensics: a detection based approach for analyzing sophisticated cyber attacks. *International Journal of Information Technology*, 11(3):583–589, 2019. doi: 10.1007/s41870-018-0263-4

[20] T. Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009. doi: 10.1109/TVCG.2009.111

[21] V. T. Nguyen, A. S. Namin, and T. Dang. MalViz: an interactive visualization tool for tracing malware. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 376–379. ACM, Amsterdam Netherlands, 2018. doi: 10.1145/3213846.3229501

[22] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *The Craft of Information Visualization*, pp. 364–371. Elsevier, 2003. doi: 10.1016/B978-155860915-0/50046-9

[23] S. Simon, S. Mittelstädt, D. A. Keim, and M. Sedlmair. Bridging the Gap of Domain and Visualization Experts with a Liaison. In *Eurographics Conference on Visualization (EuroVis) - Short Papers*, pp. 1 –5, 2015. Artwork Size: 5 pages ISBN: 9783038680291 Publisher: The Eurographics Association. doi: 10.2312/EUROVISSHORT.20151137

[24] Sudhakar and S. Kumar. An emerging threat Fileless malware: a survey and research challenges. *Cybersecurity*, 3(1):1, 2020. doi: 10.1186/s42400-019-0043-x

[25] A. Ulmer, D. Sessler, and J. Kohlhammer. NetCapVis: Web-based Progressive Visual Analytics for Network Packet Captures. In *2019 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–10. IEEE, Vancouver, BC, Canada, 2019. doi: 10.1109/VizSec48167.2019.9161633

[26] T. Wu, F. Breitinger, and S. O'Shaughnessy. Digital forensic tools: Recent advances and enhancing the status quo. *Forensic Science International: Digital Investigation*, 34:300999, 2020. doi: 10.1016/j.fsidi.2020.300999