NetCapVis:
Web-based Progressive Visual Analytics for Network Packet Captures

Alex Ulmer* David Sessler† Jörn Kohlhammer‡

Fraunhofer IGD,
Technische Universität Darmstadt, Germany

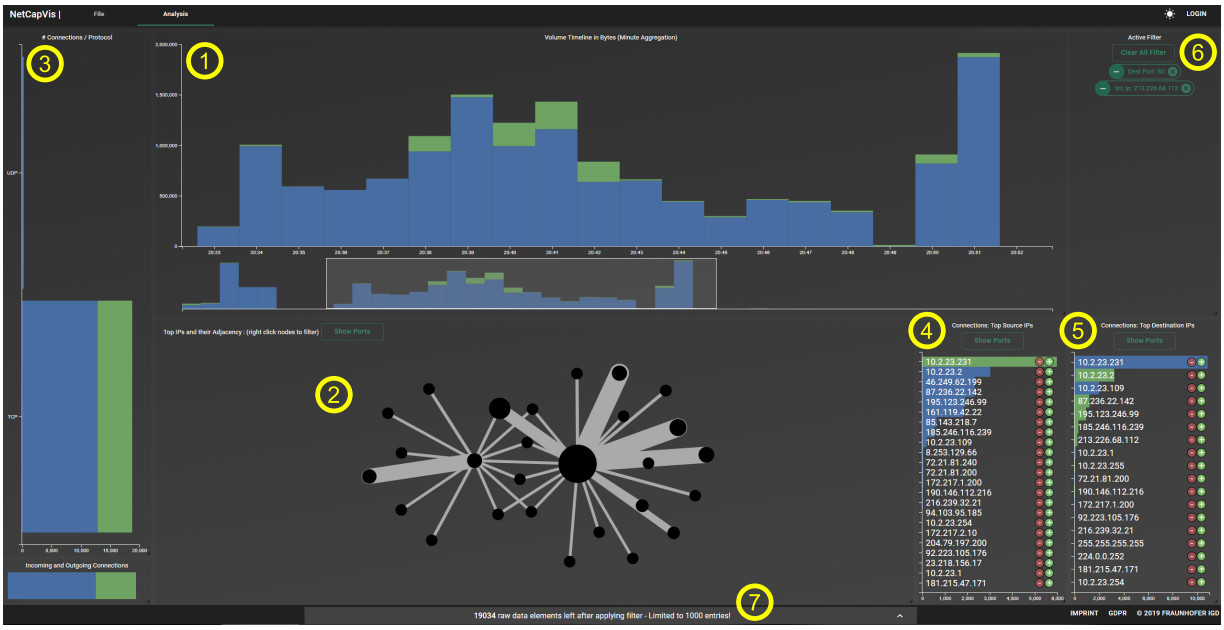

Figure 1: NetCapVis analysis interface for assisting domain experts in the analysis of PCAP data: The main visualization is the *volume timeline* (1). It is split in the context view at the bottom and the focus view at the top. The context acts as a global overview with a brush to specifiy the time frame for the focus. The *protocol view* shows the used packet protocols (2) and a stacked bar at the bottom left shows the incoming and outgoing connections. The distinction between incoming and outgoing traffic is indicated by the colors of the bars. The *graph view* (3) allows to switch between showing port connections or IP connections. Similar to that the *source view* (4) and *destination view* (5) can be toggeled between these modes and show how many connections a specific port or IP has. All views are linked and can interactively be used as filter. The *filter status* (6) displays the current filter state. At the bottom a raw data table (7) can be opened on demand.

Abstract

Network traffic log data is a key data source for forensic analysis of cybersecurity incidents. Packet Captures (PCAPs) are the raw information directly gathered from the network device. As the bandwidth and connections to other hosts rise, this data becomes very large quickly. Malware analysts and administrators are using this data frequently for their analysis. However, the currently most used tool Wireshark is displaying the data as a table, making it difficult to get an overview and focus on the significant parts. Also, the process of loading large files into Wireshark takes time and has to be repeated each time the file is closed. We believe that this problem poses an optimal setting for a client-server infrastructure with a progressive visual analytics approach. The processing can be outsourced to the server while the client is progressively updated. In this paper we present NetCapVis, an web-based progressive visual analytics system where the user can upload PCAP files, set initial filters to reduce the data before uploading and then instantly interact with the data while the rest is progressively loaded into the visualizations.

Keywords: Network Traffic, PCAP, Progressive Visual Analytics, Packet Capture, Web-Application

Index Terms: Human-centered computing—Visual analytics; Human-centered computing—Interaction design—User interface design

*alex.ulmer@igd.fraunhofer.de
†david.sessler@igd.fraunhofer.de
‡joern.kohlhammer@igd.fraunhofer.de

1 Introduction

The analysis of network traffic has been an ongoing challenge for the past years. The amount of data gathered in a few seconds can already be too much for domain experts to look through. Although automatic detection systems with machine learning algorithms were developed, they suffer from difficult setup or a high false positive rate. This is why domain experts still want to have the ability to look at the raw data. The raw data is gathered in PCAPs, which offer the most information in comparison to other formats like netflows or firewall

978-1-7281-3876-3/19/$31.00 ©2019 IEEE

logs. The main analysis tool for malware analysts and admins is Wireshark [8]. It represents the data in a table with very detailed information, so that users can reliably find what they are looking for. However, this process requires very deep domain expertise and much time because of the amount of insignificant data. The research community has developed different visual analysis systems [32] and automatic detection systems [1] to assist the analysis. We believe that these tools can be very helpful but lack accessibility for domain experts. In this paper we investigate how a client-server infrastructure can provide this accessibility even for large amounts of data. Therefore, we introduce NetCapVis, a web-based PCAP progressive visual analytics system with the future goal to become an alternative to Wireshark. Key features are the minimization of data transfer and storage of preprocessed files for a faster reload of previously uploaded files. During the upload phase, the visual interface is progressively updated so that the user can immediately start with the analysis process. For use cases with confidential data this service can be deployed for internal usage only. We evaluated the usability in an earlier stage of development with non domain experts, in addition to a smaller final evaluation with domain experts. Our prototype is freely available at https://netcapvis.igd.fraunhofer.de and works best with Google Chrome. Our main contributions are:

1. Progressive PCAP data pre-processing for efficient visual analysis with a web browser
2. Progressive web-based visual analysis system for PCAP data
3. Usability evaluation with non-experts plus interview and usability feedback from domain experts

The paper is structured as follows: In Section 2 we discuss the related work on network traffic analysis and progressive visual analytics. Section 3 covers the user, data, and task description, as well as the requirements for NetCapVis. The visual analysis system is described in Section 4. We explain our infrastucture, progressive data processing, visualizations, interactions and finally two use cases which show that our system can support the tasks. In Section 5 we summarize our evaluation methodology and the results. Section 6 discusses the challenges and limitations of our approach and shows the future work. Finally, Section 7 concludes our paper.

2 Related Work

We group the related work to our approach into two categories. First, visual analysis of network traffic is discussed to reduce the workload for malware analysts and administrators. Second, we discuss progressive visual analytics as a means to enable fluid usability with large data. In recent years this research domain gained more importance as data is getting larger. We describe which forms of progressiveness are defined in the community and which recommendations are given when creating progressive visual analytics systems [2]. We highlight related works that use similar techniques and compare them to our approach.

2.1 Network Traffic Visual Analysis

Contributions in visualization for cyber security have increased in recent years as the data is getting larger and the need for visual analysis is getting inevitable. Broad surveys by Guimarães et al. [12] and Shiravi et al. [26] summarize contributions on visualizations for network security and management, where traffic analysis plays a major role. Due to the amount of contributions in the field of network traffic analysis there are also more specific surveys like the one by Wagner et al. about visualization systems for malware analysis [30]. In the following we will discuss the most related visual analysis sytems, and refer to these surveys for further details.

ENTVis [33] by Zhou et al. is an multi-view network traffic visual analytics system with automatic anomaly detection. Similar to our approach they use multiple linked views for the most important attributes in netflow data. The major difference is that we are not using automatic detection as this is not feasible without sufficient labeled training data from one user. The authors conclusion was that their views were not scalable to large networks which led us to the decision to not use matrix views for visualizing IP or port spaces.
Another related work by Krokos et al. [15] provides a very good scalability as they can handle datasets with billions of packets. Although this approach is using 3D and requires powerful hardware, we share the goal to enable experts to visual-interactively analyze large PCAP datasets. Their performance on a dataset with over 2.4 billion packets is the current record to aim for.
Eventpad [7] is a tool to gain more high level information from PCAP files about file accesses. The goal is to assist malware analysts in finding file access patterns often used by ransomware. This approach makes use of a specific protocol in PCAP files to make these patterns visible. Our approach shares the idea of Eventpad to let the analyst find a point of interest in a visual interactive system to then switch back to Wireshark to get the detail information. Further, there are many other related works like [13] and [34] which create visualizations to uncover malware activities based on PCAP data.
The following three related approaches use a client-server architecure like our approach does.
The first one is CyberPetri [3] which was part of Ocelot [4] and got optimized for real-time data. Although, their data updates are only in 15 minute cycles, we share the update mechanism for the visualization and the aggregation for longer time frames. This prevents irritations on progressive updates and preserves the users mental map.
BubbleNet [20] is also a security dashboard with a web user interface. The authors make use of linked views to assist the user in narrowing down the perspective on the significant data. As we share the same target group, we took some of their key evaluation successes into account while developing NetCapVis.
Finally, we want to mention the online tool PacketTotal [23] which has no publication. It is currently the only web service which has the same idea as our approach. Nevertheless, this service has a file upload limitation of 50 MB and only provides static visualizations. Although the usability is disturbed by reloading pages, they have many different views and details on the data which we took into account during development.

2.2 Progressive Visual Analytics

Progressive visual analytics is a research field that gained increasing attention over the past years. As processing time increases due to larger datasets, the user needs to be updated with intermediate results. Earlier contributions to this field called this method incremental visualization [10] or progressive refinement [25] and already showed how the processing of large datasets can be handled. Later, the term progressive visual analytics was established in the community by multiple publications [28], [9]. In a recent survey Angelini et al. [2] review all previous visual analytics contributions with progressive features and categorize them. Angelini et al. define two ways of progressiveness:

- *data chunking* - the partial results show increasingly more data over time.
- *process chunking* - the partial results show all the data at all time but the quality of approximation increases.

They introduce recommendations for developing progressive visual analysis systems. We explain in detail which way we chose and which recommendations we implemented in Section 4.2. In this section we want to highlight related work in progressive visual analytics and how their findings influenced our decisions.
Glueck et al. [11] proposed a progressive loading algorithm for real-time interactions in large datasets. They use process chunking to

iteratively improve the level of detail after user interactions. Like this the user can immediately get feedback to his interaction while the final result is computing. Since our visual interface is also running in a web environment we also decided to use aggregations. The difference is though, that we can not provide a maximum level of detail as there can be thousands of packets in one second which would cause performance problems within the browser. Therefore, we compute aggregation levels, adapted to the data size, to have an optimal perceptibility and usage of screen space. Details are presented in Section 4.3.2.

Turkay et al. [29] designed a progressive and interactive visual analysis system for credit card transactions. Through an evaluation they made the observation, that visualizations should support linking and brushing as these features are already accustomed by the user. Also they observed, that it is important to preserve the users mental map while updating the views. This means that the position of previous data points and their movement to new positions should be comprehensible. Finally, the user needs to be informed if the process of updating the views is still running, to prevent surprising changes. Based on these findings all our visualizations are linked and update in parallel when a selection in one view is made. Additionally, we animated all data changes so that data that stays in the view moves to its new position. Finally, for each view a small loading icon indicates that processes are still running.

Two related works investigated the interaction in progressive visual analytics [31] [16]. One of the findings was that a progressive system definitely encourages the user to interact with the data before the final processing result is delivered. Insights are faster found as the user can already reason with parts of the data. This reinforced our belief that progressive analytics is an important feature for our use case. Both related works made the observation that latency is crucial for the usability of the visual interface. Already small delays like 500ms can have negative impact on user experience. Therefore, we put in extra effort to optimize our queries to have minimal latency.

To the best of our knowledge there are currently no progressive visual analysis systems for network traffic which are web-based and work without precomputed data which limits the users interaction freedom.

3 Data-User-Task

3.1 PCAP Data

PCAP data is a record of captured network data traveling over a computer network. This data contains all packets that were sent during the time of recording. Each PCAP file starts with a global header. This header contains general information about the capture format like the endianness of the file format and the time zone of the location where the data was recorded. The global header is followed by an arbitrary number of packets. Each of the packets start with a packet header that holds the information about the timestamp and the length of the recorded packet. Following the packet header the ethernet header starts which contains information about the source and destination MAC addresses. Underneath the ethernet header the structure may deviate. Different headers which can be associated with protocols come into play. A common example would be the IPv4 header which can hold TCP and UDP headers and their data. Finally, encapsulated by this nested header structure is the payload which contains the data associated with the protocol.

The recording or sniffing of network data can lead to very different PCAP file sizes. It depends on the network traffic, the frequency of sent packets and the size of their payloads on the one hand and the recording time span on the other. Therefore, PCAP files can become very big very fast. Browsing through such a file to check for potential malicious activities is a highly tedious task.

A widely used tool to browse PCAP data is Wireshark. It provides detailed information about the packet based on the information stored in the nested header structure. It can handle medium file sizes and gets somewhat slow for larger files. Although, Wireshark provides statistics charts over the contents of a PCAP file, it offers only limited visualizations to help understand the general scenario. There, NetCapVis can support the user with an overview to find points of interest more easily and narrow down the data to look at in Wireshark.

3.2 User Groups

In our opinion there is a wide range of users, which would benefit from an interactive visualization of network traffic. Especially if the service is easily accessible more users would try to analyze their data. Nevertheless, we focus our development on the well defined [21] user group of cyber analysts. This group is interested in finding the cause of anomalies such as malware infections or other irregular traffic in forensic analysis and real-time attack situations. The scope for our approach is the forensic analysis. Therefore, our potential users are:

- Malware analysts
- Administrators
- Security operation center analysts

For all of them the reduction of workload can be achieved by reducing the data they have to look at in detail.

3.3 Tasks

We studied the related work for interests and usual tasks of the user group. Malware analysts often create sandbox environments to let malware act freely and analyze the packet captures. This gives them more insight on how the malware works or which endpoints it communicates with. Administrators want to analyze which external hosts are connecting with their internal hosts. Important features are, how often connections are established and how much traffic is transferred. Analysts in SOCs are similar to malware analysts as they want to find out where an anomaly originated, but time is a more crucial constraint. Based on that we formulated the following tasks our prototype should support

T1: Find suspicious connections in large packet captures.

T2: Determine relevant events to filter for in Wireshark.

T3: Get an overview of the recorded network and its incoming and outgoing connections

In the current status we see our prototype as a visual interactive filtering before digging into the details with Wireshark.

3.4 Requirements

We derived the following requirements based on the user and task characterization:

R1: Give an overview of connections in the users network

R2: Give an overview of transferred volume in the users network

R3: Allow the user to filter the data for significant parts

R4: Load large files without blocking the visualizations

R5: Discriminate between incoming and outgoing traffic to the users network

Based on the explanation in Section 3.2 and 3.3, the requirements are connected to the tasks and users in the following way:

- Malware Analysts: **T2** $\rightarrow$ **R3**
- Administrators: **T2, T3** $\rightarrow$ **R1, R2, R3, R4, R5**
- SOC Analysts: **T1, T2** $\rightarrow$ **R3, R4**

4 Visual Analysis System

In this section, we briefly explain the infrastructure of our prototype and the main technologies we use. Then, we describe the progressive data uploading and continuous updates of the visualizations during the upload phase. Finally, the visualizations and interactions are shown along two use cases to demonstrate that the intended tasks can be performed.

4.1 Infrastructure and Technology

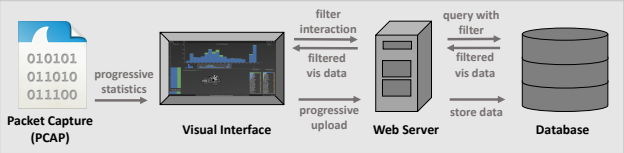

Figure 2: client-server infrastructure of NetCapVis: PCAP data is progressively uploaded, processed and stored in a database. Filter interactions are translated to database queries on the webserver and the query result is directly piped through the server to the visual interface.

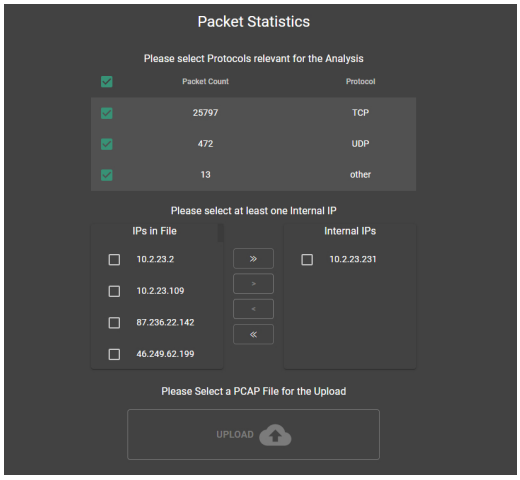

Figure 3: Packet Statistics: Top: A statistic of protocols contained in the PCAP file is shown. Users can deselect not relevant protocols for the analysis. Bottom: A list of all IP addresses is displayed in the left box, ranked by connection count. Users may mark IPs as internal by transferring them to the right box.

Our prototype is a web-service with the structure as depicted in Figure 2. At first, the raw data is processed as described in Section 3.1 and stored in a MySQL database [22]. The main computations for the visual interface are processed on a Spring Boot Java server [27] which is connected to the database. Communications between server and client are handled by a websocket using SockJS and Stomp to have lower latency. Finally, our web front-end is created with React.js [24] and Material-UI [19], while D3.js [5] is used for the visualizations. We decided to use this approach to have a highly accessible application which can be used on desktop computers and mobile devices. It also enables collaborative analysis which we want to explore in future. The service can easily be deployed internally in confidential networks so that no information has to be uploaded to external hosts. All employees inside this network can access the application without installing additional software on their computer.

4.2 Progressive Data Processing

PCAP files contain information about all packets that were captured during the recording period. Most of these packets may not be relevant for the user's analysis goal. Therefore, we have implemented filtering and data reduction techniques to upload less data from client to server.

4.2.1 Pre-Upload Filtering

We provide an interface that allows to reduce the amount of irrelevant packet data before the upload phase. In a first run, we collect statistics data over the whole PCAP file. As a result, the *packet statistics* interface presents the distribution of the protocols and IP addresses with the most connections to the user, which can be seen in Figure 3. This protocol list can be filtered by the user. This way, the user can decide to keep only protocols that are relevant for the analysis task and thus greatly reduce the data that has to be processed.
Additionally, the user can decide, which IP addresses are mapped to internal devices and mark them as such. The distinction between internal and external IPs facilitates separation and more in-depth filtering capabilities for the analysis.

4.2.2 Upload Phase

During the upload phase, we reduce the amount of data again by cutting the payload of the packets. This is primarily done to reduce the upload size but also due to privacy issues. For now this limits our analysis capabilities as we only have the header information but in future work we want to extend our tool with an option to also upload and analyze payload data. In detail, we parse the nested header structure to identify the protocol of the packet and the offset of the payload. Then we discard the payload and retain the header information. On the one hand, this has to be done due to privacy reasons and, on the other hand, the data stored in the packet headers is fully sufficient for the analysis capabilities we provide.
We support the upload of very large PCAP files by slicing them into manageable bits. These slices are then preprocessed by the client. For each of these slices we identify the packet headers, filter out undesired packet protocols and upload the header data in small batches to the server.
On the server, we parse the incoming packet header information and store it in a database. Additionally, we calculate different time aggregations for the level of detail requirements in the *timeline* view (Section 4.3.1). Each time a batch of packet headers is processed on the server, we return the data needed for all visualizations to the client. This way, the user has immediate visual feedback that some of the data was processed and is ready to be used for the analysis.
Filter operations for all visualizations are available as soon as the first partial result is returned from the server. Thus, we provide the user with all analysis capabilities after only one small portion of the data was uploaded and processed, like motivated in the requirement (R4). The upload progress is indicated by two cues. A progress bar at the top of the analysis page shows an estimation to what extent the file was already processed. Additionally, the bottom part of the *timeline* visualization indicates the progressively growing data which is ready to be analyzed.

4.2.3 Progressive Data Upload

As introduced in the related work section, there are two ways of progressiveness, *process chunking* and *data chunking* [2]. *Process chunking* is especially useful for providing a meaningful overview or early prediction of the result, when performing iterative calculations that take a lot of time. However, this approach this does not apply to data uploading.
Therefore, we use *data chunking* that allows us to upload and process large PCAP files as small bits at a time. For our case, two different *data chunking* approaches can be distinguished: *sampling-based*, and *front to back*.

Sampling-based data chunking starts with an overview of the data with a rather high uncertainty and gets gradually more detailed and certain. For the file uploading and processing this implies that we upload the PCAP data by sampling packets over the entire time span and provide an overview first. Gradually increasing the sample size, this overview would become more detailed until all packets from the file are present in the database. While this approach is great for use cases, where an overview of the data in the intermediate steps can already be used to draw reasonable conclusions, we argue that this is not the case for PCAP data. Our goal is to provide the means to detect malicious activity from the PCAP data. However, malicious activity is mostly short-lived and can only be detected and fully understood when the whole data for the time period of the attack is present. Thus, a *sampling-based* approach yields a hardly useful representation of the packet data in the intermediate steps and is therefore not optimal for our use case.
We use *front to back data chunking* to exploit the fact that the packets are inherently ordered by time. This time continuity in the data allows to make meaningful analysis even for the smallest processed parts of a PCAP file.
According to the characterization of progressive visual analytics [2], there are several points to consider. Providing early partial results is one of the main benefits of a progressive system. We provide early results that are shown milliseconds after the data uploading and processing begins, thus we satisfy the *immediacy* constraint. Furthermore, these partial results cover all available packet data for the particular time range, and therefore are *meaningful* for the analysis of this range. The early results provide data for all linked visualizations can be analyzed the same way as the data of a complete PCAP file. This ensures the *actionability* of the system on partial data.
Furthermore, we implement several recommendations listed in [2] We provide early partial results which are fully significant and interactive (Rec I). We provide an adaptable *timeline* visualization which demonstrates the succession and monitoring of partial results (Rec IV, Rec VII). We handle fluctuations by incorporating animations that help to cope with changes in the data (Rec IX). Additionally, we provide different levels of detail in the *timeline* view to adapt to the steadily growing time frame.

4.3 Visualizations and Interactions

Based on the requirements listed in Section 3.4 we derived five visualization modules for our prototype, which can be seen in Figure 1. These modules are linked with each other and are designed to support multiple requirements at once. Each visualization shows one or more attributes of the packet data and provides a different filtering functionality (R3). Interacting with one visualization impacts all other visualizations. The linked views allow to drill down the data very fast and only see the important parts.
All views except the *connection graph* view make a distinction between incoming and outgoing traffic (R5), which is indicated by different colored bar charts or stacked bar charts. *Blue* bars stand for incoming traffic while *green* bars indicate outgoing traffic.

4.3.1 Timeline

The *timeline* visualization allows to inspect the transferred packet data volume in bytes over time (R2). As one of the main visualizations it is positioned at the center in the upper half of the analysis page, shown in Figure 1 (1). The visualization contains two views the *timeline context* view at the bottom and the *timeline focus* view at the top.
The *timeline context* view provides an overview over the whole time span contained in the PCAP data. This view always shows the whole time frame of the available packet data. Thus, it also indicates the process of the progressive loading of big PCAP files by displaying the incrementally growing time frame. The *timeline context* view allows to filter the packet data for smaller time intervals. For this we provide a brush which can define arbitrary intervals by drag interactions.
The *timeline focus* view shows the zoomed version of the brushed area in the *timeline context* view. Here, the user can view the data with more details and get additional information about the raw data via tooltips. During the progressive data uploading phase the *timeline focus* view shows the same time frame as the *timeline context* view until a first time filter is set by the user. Setting a time interval prevents the *timeline focus* view from changing and copying the progressive behavior of the *timeline context* view. This way, we ensure that the user can start analyzing the partial data and does not have to wait until the upload is finished (R4).
We implement adaptive time aggregations to support level of detail for different time frames, like explained in Section 4.2.2. The aggregation of the *timeline focus* changes automatically depending on the time frame selected in the Context view. Therefore, the time aggregation of the two views may differ. We decide not to provide the raw data level of detail to the *timeline* visualization since it is not practical. Thousands of packets can be sent during a fraction of a second and browsers can not handle so many data points efficiently. We tested how many bars are optimal for comfortable reading and interaction by the user.
We found that for common desktop monitor resolutions this number should not be much higher than 50. Since minutes and hours have an inherent aggregation of 60 we decided that our maximum count of bars should also be 60. Therefore, we define our aggregation borders as follows:

0 - 60s time frame:	1s aggregation
60s - 10m time frame:	10s aggregation
10m - 1h time frame:	1m aggregation
1h - 10h time frame:	10m aggregation
>10h time frame:	1h aggregation

We decide not to define higher aggregations because PCAP records of those lengths are usually cut in multiple files due to the resulting big file sizes.

4.3.2 Protocol

This view shows the protocols present in the current selection and can be seen in Figure 1 (3). The bar length indicates the number of connections (R1). The exact numbers of incoming and outgoing connections can be displayed via tooltip. We can currently discriminate between nine most common protocols as it requires to manually implement a detection function for each protocol. The detector parses the bytes of the nested header structure based on the specifications of each sub header, to determine which protocol was sent in this packet. An extension of the detector functionality is discussed in the future work Section 6.2.
This visualization can be used to filter the data for specific protocols. This is useful to filter out dominant connections and protocols like TCP when searching for patterns and unusual behavior in the less represented protocols.

4.3.3 Incoming and Outgoing

This view is located in the bottom left corner of the analysis page and can be seen in Figure 1. A single stacked bar chart represents how many connections are incoming and outgoing (R1). We define incoming and outgoing connections as follows. The user selects a set of IPs that belong to internal devices and marks them as *internal* before the upload phase. Connections that have these IPs as the destination are indicated as incoming, and those which originate from this set of IPs as outgoing. This filter allows to greatly reduce the data for further analysis when only the outgoing or incoming traffic data is relevant for the user's analysis goal.

Figure 4: Use Case 1: On the left three peaks are found. Each peak is selected with the brush which filters the data for that time frame. Each peak gives one dominant IP address the host at 192.168.1.14 communicated with. After a quick look in Wireshark the user finds out that a malicious jpg file was downloaded from 46.4.115.108. On the right the user selects the time after the infection to find out what is happening. The user sees a periodic communication with the IP 31.31.196.236. A DNS lookup reveals that this address is located in Russia.

4.3.4 Connection Graph

A graph view of the connections makes it easy to grasp interesting patterns in the network at a glance. This view, as shown in Figure 1 (2), has two modes to show the connectivity of IP addresses and ports. The user can see which ports/IPs are communicating with each other. The size of the node shows the sum of all connections an IP address or port has to its neighbors (R1). The width of the link shows the volume of the transferred data between two nodes (R2). A detailed summary of all connection and volume values is displayed in a sidebar tooltip when hovering over a node. Right clicking on the node shows a contextmenu which allows the user either to filter for specific IPs or ports or explicitly filter them out. The graph layout is created with a force layout algorithm which allows the user to drag nodes around to rearrange the interesting subgraphs.
A graph view is excellent for finding interesting or unusual patterns in the connection data as long as the graphs stay manageable. Displaying all ports or all IP addresses for a big file is not practical, because it results in a huge graph with thousands of nodes and edges. This is bad for two reasons. First, such a big amount of edges results in more edge crossings, which make it harder to find interesting structures in the connection data. Second, we observed that low end machines start to lag because of the complex computation the graph layout algorithm has to perform for such a huge number of nodes and edges.
We solve this problem by filtering out the nodes with low connection counts and edges with low transmission volumes, when the total count of edges gets to big. Cutting the least important nodes and edges enables us to reduce the graph to a manageable size while still retaining all the necessary information to cover the use cases described in Section 4.4.

4.3.5 Source and Destination

The *source* and *destination* visualizations, presented in Figure 1 (4) and (5), show rankings of IPs or ports based on the connection count (R1). Each view can swap between the IP and port representation by clicking on the button at the top. In contrast to the *connection graph* view which also presents connection data these visualizations are decoupled. While the *connection graph* view focuses on the patterns of the connections between IPs and ports, these two views focus on analyzing the IPs and ports based on their role as the source or the destination of the traffic. This can help to identify attacks that focus on opening large numbers of connections to the same IP or identify sources that scan for ports of a specific connection. For each ranking the top 25 IPs/ports are displayed. This way, the user can identify which port or IP is most frequently used and decide if these specific connections are relevant for the analysis. Filtering can be performed by including only specific IPs/ports via clicking the ”+” button, or by excluding IPs/ports with the ”-” button. Each time the ranking gets shorter than 25 entries due to filtering, new entries are loaded from the database to fill up the view. A conventional use of these views is to filter out commonly used ports like 80 (HTTP) or 443 (HTTPS) and see what is hidden behind this dominating traffic.

4.3.6 Filter Status

The *filter status* view, Figure 1 (6), contains a list of filters that are currently active. Each filter representation provides the information whether it is a filter including an attribute constraint, indicated by a ”+”, or excluding it, indicated by a ”-”. The filtered attribute key value pair is represented in textual form. This view allows the user to remove existing filters more easily than reverting the interaction in the according visualization. It is also possible to reset all filter at once by clicking on the *clear all filter* button. The only filter that is excluded from the *filter status* view is the time range filter. There are two reasons for that. First, the *timeline context* view provides a permanent visual feedback to the selected time frame. Second, users found it more convenient to be able to clear all other filters while preserving the time range filter.

4.3.7 Raw Data Table

The *raw data table* is located at the very bottom of the analysis page, Figure 1 (7). It starts out in the collapsed mode that only displays the count of the raw packets for the selected filter. When expanding, the raw data is downloaded from the database with a limit of 1000 entries to prevent the client from lagging due to data overload. In the expanded mode a table is shown that summarizes all attributes which are used for the visualizations, namely: the timestamp of the recording, the source IP, the source port, the destination IP, the destination port, the protocol, the volume of the payload data and the information if the traffic was incoming to the defined internal IPs. Users can browse large contents of the table by navigating from page to page with the controls at the bottom of the expanded table. However, this view is most suitable to take a closer look at the packet data after only few interesting connections remain after filtering out the less relevant data.
One of our next steps is to extend this view to have additional detail on demand information. We aspire to display as much information as Wireshark offers in its detail packet view and discuss this in the future work Section 6.2. This extension is necessary to improve the usability for our user group so that they do not need to use two different tools for the same data.

Figure 5: Use Case 2: On the left the two most found destinations for traffic from our suspected IP address 192.168.1.159 are highlighted. Based on our knowledge that the first address is for a messaging service we can determine when messages were sent by our suspect. We find out that our suspect has been messaging with the IP address in second place 192.168.1.158. Then, we set the destination filter to only show data from our suspect to the outsider. On the right we can see one major peak with 13 KB directly transferred. Switching to Wireshark with the filter information we reveal that a file called "recipe.docx" was transferred.

4.4 Use Cases

In the following, we demonstrate along two use cases, how the intended tasks can be performed by using our prototype. We used two datasets from the internet [17] [18] with artificial scenarios to demonstrate that our tool works with different data and is not tailored for specific datasets. Both of these datasets are freely accesible, to make the use cases reproducible. The first dataset has nearly 29k packets with a size of 19MB. The second dataset is smaller with only 240 packets and a size of 70KB. In both use cases we focused primarily to show the task completion and not the progressiveness of the system.

4.4.1 Use Case 1

Our first use case shows an artificial scenario PCAP from the website malware-traffic-analysis.net [18]. In this scenario a host is getting infected by malware at some point during the recording and then communicates with the botnet server. The goal is to determine which IP addresses could be suspicious by using NetCapVis and then switch to Wireshark and filter for these to immediately see the significant raw data. After uploading the file the most used IP address is automatically set as an internal address, if not specified otherwise in the pre-upload statistics. This makes it easily visible that 192.168.1.14 is our host where the recording was performed. On the left of Figure 4 we see that during the recording there were three large peaks of transferred data. We set the time filter three times to only cover the peaks, and thus find out three dominant IP addresses which were communicating with our host. After a quick look in Wireshark we see that in the middle peak a HTTP request was performed to get the file "1MDiPJ.jpg" from 46.4.115.108. An internet search for "46.4.115.108:80/1MDiPJ.jpg" shows that a download of an executable file was triggered which started the communication to the botnet server. To find out with which IP address our host was communicating after the infection, we move the time filter behind the peaks like seen on the right of Figure 4. This filters out the dominant peak and makes the low volume traffic visible. We immediately see a periodic pattern with equal volume sent and received from the IP address 31.31.196.236. A DNS lookup shows that this IP is located in Russia, and is already reported to be associated with malware. This use case shows that our prototype is capable to perform the tasks of finding suspicious pattern, time frames and IP addresses (see Section 3.3).

4.4.2 Use Case 2

The second use case is an artificial scenario where a company worker is suspected to leak confidential data to an accomplice [17]. We know the IP address of the worker is 192.168.1.158 and that the company's wireless network with an instant messaging service at 64.12.24.50. During the PCAP recording an unknown device appears in the logs. We have to find out what kind of communication between the workers computer and the unknown device happened. Figure 5 shows the initial view in NetCapVis with a filter set to only show outgoing traffic from the workers IP address. We can see that the top destinations are the instant messaging service and the IP address 192.168.1.159. First, we check out the messaging service by adding it to the destination filter. We see a peak in volume close to the start of the recording hover over the bar and see the exact time. Now we switch to Wireshark for more detailed information and see that an unencrypted message witch says: "Here's the secret recipe... I just downloaded it from the file server. Just copy to a thumb drive and you're good to go >:-)". This message is forwarded by the messaging service to 192.168.1.159, the IP address in second place for outgoing traffic for the workers computer. This has to be the unknown device. Back to NetCapVis, we remove the destination filter from the messaging service and set it to the unknown device IP. Now we see one large peak of traffic where the workers computer is sending 13 KB data at a specific time to the unknown device. Inspecting the packets in Wireshark reveals that a file with the name recipe.docx was transferred. This usecase shows that the with filter mechanism of NetCapVis the analysis time can be reduced by only looking at significant parts of the data.

5 Evaluation

Figure 6: First iteration of our prototype: The layout is similar to the current version but not all views are linked and no progressive features are included.

We conducted a usability evaluation on the first iteration of our

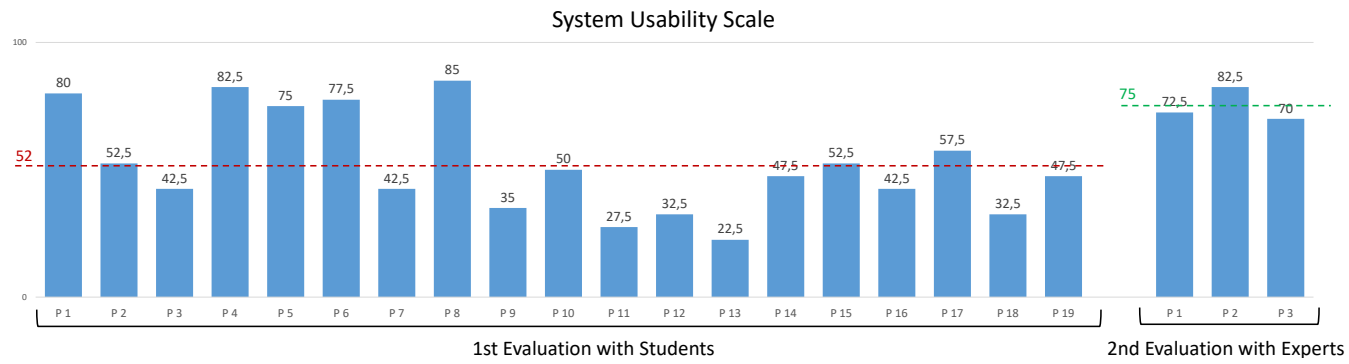

Figure 7: System usability scale scores for both evaluations: The first evaluation with students only scored 52 on average. The second evaluation with the adapted prototype and domain experts scored an average of 75.

prototype, which is shown in Figure 6. In this version, not all views were linked and the data upload feature was replaced with a fixed precomputed dataset. The goal of the evaluation was to detect usability flaws early in the development process. We adapted the prototype according to the feedback and developed additional functionalities. A second evaluation with domain experts was done with the current prototype. In this section, we describe how the evaluation has been conducted and how the results of the first evaluation were integrated into the current version.

5.1 Methodology

We asked students to analyze a dataset without a prior introduction to the tool. The selected group of students can be described as participants with medium expertise in information visualization and user-centered design, but no expertise in network traffic. The only information the participants had was that the dataset contains PCAPs. They had to identify which parts of the data are the most interesting, purely based on the visual analysis capabilities of the prototype and not on domain expertise. Our goal with this style of evaluation was to find out which designs and interactions are working well with the human cognition without influences of domain expertise. After they worked independently with the data we asked them to fill out a system usability scale (SUS) form [6]. Their feedback gave us helpful guidance in adapting our prototype. The current version was evaluated again with domain experts. At first we explained the visualizations and possible interaction, to then test the experts with the artificial scenario described in our first use case (see Section 4.4.1). In this scenario the host (where the data was collected) was infected with malware. The experts had to identify which time frame and which IP addresses are the most suspicious. Then, the experts formed a filter query in Wireshark to find out what happened exactly. After this scenario we let the experts upload one of their own files which they previously used for analysis. Based on the artificial scenario and the interaction with familiar data, we had an informal discussion with the experts about possible improvements. In the following we will describe our results.

5.2 Results

The result of the student evaluation was a bit harsher than expected, but it gave us good impulses for improvements. Remarkable is the high variance of SUS scores from the students. While assessing the student feedback we noticed that a major issue was the compatibility with different web browsers. Our prototype runs with all common browsers but there are small bugs like missing tooltips or wrong zoom behavior for some browsers. This had a big influence on the usability which led to the low ratings. Without an introduction and domain expertise the students had difficulties using the prototype. In the following we summarize the most common feedback:

- The students liked the basic layout of the interface and that different views were representing different attributes of the data.
- They said that the linked views are very helpful when filtering the data, and that this feature should be extended for all views. We integrated this feedback so that now all views are linked.
- A common feedback was that orientation is very difficult on the interface as there is so much information. It would be helpful to implement a tutorial to explain all features. Since our target group are cyber analysts, we decided to move this to a later development phase. As it turned out in the expert evaluation, it is really necessary to add a guidance at the start.
- Another feedback was to extend the raw data table to the level of detail as Wireshark provides. We are also eager to implement this feature in future work, but it was out of scope for this version of the prototype. It requires advanced algorithms to read all the information Wireshark provides for a PCAP.
- Especially interesting was the fact that the usage of the visualizations was quite different among the participants. Some stated that the graph view was useless to them and others said that they have gathered most information from the graph. Similar behavior was also visible during the expert evaluation. Our conclusion is that it is useful to have different visualizations representing the same data attributes as users have different preferences and visual interpretation.

Based on the feedback from the students we decided to give the domain experts a short tutorial of all functionalities that our prototype provides. During the test, all experts were able to find the time frame of the infection and the IP address where the malware came from. Finding the post-infection communications took considerably longer. It was not clear that there is more to see after the third peak and because the *timeline context* is dominated by the large peaks the small volume traffic is not visible (see Figure 4 right). Overall the experts felt more comfortable when analyzing their own data. After they discovered all possible interactions, they liked the overview and intuitive filter mechanisms. In the discussion, the experts said that the tool is very useful to find anomalies but needs some improvements which are summarized in the following:

- All experts told us that after they find a suspicious IP address they would like to make a DNS lookup. They said that the domain name can give a hint to the origin of the IP and that they have common indicators to trigger a deeper investigation.

- Two of the experts mentioned that the connection of ports is not as important as the connection of IPs. Only the source port is of interest because many communications are requested from a specific port but the answers are send to a random port greater than 1024. These port numbers are free to register or are dynamically assigned by the operating system.

- The experts also mentioned that when the data is progressively loaded, the *timeline focus* stayed at the start where only the first seconds of packets are seen. In fact we implemented this to be a fixed view to not irritate the user when data is still processing. But it became evident that the users are expecting that new data comes in and would like to see an overview as long as they did not make any interaction. We implemented this feature as described in Section 4.3.1.

- One expert suggested to make the timeline focus interactive. On clicking a bar the time frame should change to include only the data aggregated in this bar.

- Finally, one expert said that as long as we can not provide the same level of detail for the raw data table as Wireshark does, we could export our current filter settings as a Wireshark filter query string. This would make the switch between the two applications easier.

We also asked about the size of the PCAP files that the experts usually analyze. Surprisingly all of them stated, that they only gather PCAPs for short amounts of time after they triggered a specific action, like releasing malware in a sandbox environment. Only in rare cases, where the cause of a problem is unclear, they make long recordings. In the end, the experts were interested to use our tool further and asked if it will stay online, which is already a good sign. We think that giving a short introduction to the interface and the domain knowledge resulted in the better average SUS score of 75. This shows that our prototype is specifically designed for experts and needs an integrated tutorial to ensure a smooth first user experience.

6 Discussion and Future Work

6.1 Discussion

One of the main challenges we encountered was the combination of progressive visual analytics and the client-server infrastructure. To keep the latency of interactions low, the performance had to be optimized. Advanced communication technologies, multi-threaded algorithms and a large amount of database queries require efficient implementation and powerful hardware. Another difficulty is the client environment where the visual interface is running. Web browser have differences in executing JavaScript code, which can influence usability and user experience in a negative way. Therefore, we recommend to do evaluations in controlled environments, especially with non-experts focusing on usability like our first evaluation. It is also important to ensure that only devices with a compatible resolution and software is available. The development of data intensive algorithms on the client is not so advanced as for desktop applications. Because of that, there are currently no useful JavaScript libraries to parse PCAP files. For now, we had to implement the parsing by ourselves. Overall, the outcome of our development seems highly useful and we are eager to work towards offering a better solution than Wireshark. For now Wireshark will still be the most used tool but we believe, like [14] that with our work we improved the prefiltering of the data making it easier to look at the right spots in Wireshark.

6.2 Future Work

In addition to the evaluation suggestions and our limitations we have the following ideas for future work. First of all, we plan to extend the representation of the raw data by a detail on demand view. Currently, we provide a table-based view on the raw data with the timestamp, source and destination IPs, ports and the payload size. However, the nested headers of the packet data provide much more information, especially inside of specific protocols. Inspired by Wireshark, we plan to implement a textual summary of the header data where the nested headers can be expanded to get detailed information about all its fields.
Further, we want to provide an export function which translates our filter status into a Wireshark filter query. This way we can investigate the payloads of packets in Wireshark, after finding a potential attack. To achieve our final goal of being an alternative to Wireshark, we want to implement an option to upload the payload and provide the same detailed analysis capabilities as Wireshark does.
The evaluation suggested to make the time filtering more intuitive. Some users were confused by the different time aggregations in the *timeline* visualization. To make this more consistent we plan to display the same aggregation in the brushed area of the *timeline context ciew* as in the *timeline focus view*.
Both evaluations showed that we need some form of guidance for the first time use of our visual interface, as it provides many visual information and interactions.
We want to tackle the problem that graphs become unmanageable for big data with a simple data reduction approach. However, our approach may not be the best solution for that. An extended state of the art research on best practices for data reductions of graphs which preserve the topology could inspire us to implement better graph data reduction functionality. Alternatively, an aggregation based-approach for graphs could also be promising.
Currently, we support the recognition of the nine most commonly used protocols. With over hundred possible protocols there is room for improvement. We plan to extend the protocol recognition capability in the future by adding more detecting functions.
Automatic detection of potentially interesting patterns in the PCAP data can be used to suggest promising time frames for analysis. A guidance concept based on these statistically significant patterns may speed up the analysis workflow. We see the benefits especially for huge PCAP files, where purely visual exploration may lead to overlooking interesting patterns.
Another important feature to implement is the DNS lookup for IP addresses. Experts might know the domain names behind their local IP addresses, but for external IPs a lookup is necessary, as these names can give indications for country of origin or the company name.

7 Conclusion

In this paper we presented a new progressive visual analytics system for analyzing PCAP data. We summarized the related work in network traffic analysis and put our approach in contrast. We considered the findings of progressive approaches and created - to the best of our knowledge - the first progressive visual analysis web application to inspect packet captures. By defining specific tasks for the user group of cyber analysts, we derived five concrete requirements our approach has to support. We described our visual analysis system, interaction design and infrastructure in detail and demonstrated the usefulness through two realistic use cases. An early evaluation gave us important impulses towards the current version which we evaluated with domain experts. The positive and constructive feedback motivates us to overcome the limitations we discussed and sets the roadmap for our future work.

Acknowledgments

This research work has been funded by the German Federal Ministery of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[2] M. Angelini, G. Santucci, H. Schumann, and H.-J. Schulz. A review and characterization of progressive visual analytics. In *Informatics*, vol. 5, p. 31. Multidisciplinary Digital Publishing Institute, 2018.

[3] D. Arendt, D. Best, R. Burtner, and C. L. Paul. Cyberpetri at cdx 2016: Real-time network situation awareness. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–4. IEEE, 2016.

[4] D. L. Arendt, R. Burtner, D. M. Best, N. D. Bos, J. R. Gersh, C. D. Piatko, and C. L. Paul. Ocelot: user-centered design of a decision support visualization for network quarantine. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, 2015.

[5] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization & Computer Graphics*, (12):2301–2309, 2011.

[6] J. Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[7] 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, 2018.

[8] G. Combs et al. Wireshark-network protocol analyzer. *Version 0.99*, 5, 2008.

[9] J.-D. Fekete and R. Primet. Progressive analytics: A computation paradigm for exploratory data analysis. *arXiv preprint arXiv:1607.05162*, 2016.

[10] D. Fisher, I. Popov, S. Drucker, et al. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1673–1682. ACM, 2012.

[11] M. Glueck, A. Khan, and D. J. Wigdor. Dive in!: Enabling progressive loading for real-time navigation of data visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 561–570. ACM, 2014.

[12] V. T. Guimaraes, C. M. D. S. Freitas, R. Sadre, L. M. R. Tarouco, and L. Z. Granville. A survey on information visualization for network and service management. *IEEE Communications Surveys & Tutorials*, 18(1):285–323, 2015.

[13] N. A. Huynh, W. K. Ng, A. Ulmer, and J. Kohlhammer. Uncovering periodic network signals of cyber attacks. In *2016 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, 2016.

[14] I. V. Kotenko, M. Kolomeets, A. Chechulin, and Y. Chevalier. A visual analytics approach for the cyber forensics based on different views of the network traffic. *JoWUA*, 9(2):57–73, 2018.

[15] E. Krokos, A. Rowden, K. Whitley, and A. Varshney. Visual analytics for root dns data. In *2018 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, 2018.

[16] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131, 2014.

[17] LMG Network Forensics Contest. http://forensicscontest.com/2009/09/25/puzzle-1-anns-bad-aim, 2019.

[18] malware-traffic-analysis.net/. A source for pcap files and malware samples... http://malware-traffic-analysis.net/2017/09/19/index.html, 2019.

[19] Material-UI. React components that implement Google's Material Design. https://material-ui.com/, 2019.

[20] S. McKenna, D. Staheli, C. Fulcher, and M. Meyer. Bubblenet: A cyber security dashboard for visualizing patterns. In *Computer Graphics Forum*, vol. 35, pp. 281–290. Wiley Online Library, 2016.

[21] S. McKenna, D. Staheli, and M. Meyer. Unlocking user-centered design methods for building cyber security visualizations. In *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–8. IEEE, 2015.

[22] MySQL - Oracle. https://www.mysql.com/, 2019.

[23] PacketTotal. Simple, free, high-qulity PCAP analysis. https://packettotal.com/, 2019.

[24] React.js - Facebook Inc. https://reactjs.org/, 2019.

[25] R. Rosenbaum and H. Schumann. Progressive refinement: more than a means to overcome limited bandwidth. In *Visualization and Data Analysis 2009*, vol. 7243, p. 72430I. International Society for Optics and Photonics, 2009.

[26] H. Shiravi, A. Shiravi, and A. A. Ghorbani. A survey of visualization systems for network security. *IEEE Transactions on visualization and computer graphics*, 18(8):1313–1329, 2011.

[27] Spring Boot- Pivotal Software. https://spring.io/projects/spring-boot, 2019.

[28] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662, 2014.

[29] C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE transactions on visualization and computer graphics*, 23(1):131–140, 2016.

[30] M. Wagner, F. Fischer, R. Luh, A. Haberson, A. Rind, D. A. Keim, W. Aigner, R. Borgo, F. Ganovelli, and I. Viola. A survey of visualization systems for malware analysis. In *EG conference on visualization (EuroVis)-STARs*, pp. 105–125. The EGA, 2015.

[31] E. Zgraggen, A. Galakatos, A. Crotty, J.-D. Fekete, and T. Kraska. How progressive visualizations affect exploratory analysis. *IEEE transactions on visualization and computer graphics*, 23(8):1977–1987, 2016.

[32] H. Zhao, W. Tang, X. Zou, Y. Wang, and Y. Zu. Analysis of visualization systems for cyber security. In *Recent Developments in Intelligent Computing, Communication and Devices*, pp. 1051–1061. Springer, 2019.

[33] F. Zhou, W. Huang, Y. Zhao, Y. Shi, X. Liang, and X. Fan. Entvis: a visual analytic tool for entropy-based network traffic anomaly detection. *IEEE computer graphics and applications*, 35(6):42–50, 2015.

[34] W. Zhuo and Y. Nadjin. Malwarevis: entity-based visualization of malware network traces. In *Proceedings of the ninth international symposium on visualization for cyber security*, pp. 41–47. ACM, 2012.