Excel@FIT 2021
http://excel.fit.vutbr.cz

Visualization System of Network Forensic Data

Ivan Manoilov

Abstract
The goal of this project is to develop a system for processing various forensic data inputs (such as PCAP) and presenting them in a user-friendly, graphical way. Modern systems for visualization of data are too generic for this specific use-case and can't be employed for analysis of forensic data. This problem is solved via development of the system of analytical dashboards which represent the data, processed by various back-end services, responsible for parsing input, aggregating data and transforming it to the format acceptable by dashboards. Result of this thesis would be a highly extensible system, providing deep analytical view of network traffic, which enables the end-user to see underlying anomalies or problems of the traffic.

Keywords: Data processing — Analytical dashboards — Network Forensic Data

Supplementary Material: Downloadable Code

*xmanoi00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The main goal of this project is to develop a system able to visualize network forensic data, so the user will be able to see problems and forensic activity on network. With rapid development of Internet technologies and expansion of Internet throughout the globe, cybercrime has became one of the most common of the crimes. The situation has only worsened nowadays, as one the main impacts of COVID-19 was transition of parts of our lives to the Internet. Reports of cyber-dependent crime and online fraud have increased during the COVID-19 outbreak, and rates of cybercrimes have been particularly high during months with the strictest lockdown policies [1].

Output of the developed system is composed of 4 dashboards corresponding to 4 types of network data input:

- **Network Conversations** : the basic view of the network communication data.
- **Extracted Files** : the view of the unencrypted data exchange in format of files.
- **Encrypted Traffic** : the view of the encrypted communication.
- **Resolved Domains** : the view of the DNS queries and responses.

The data for dashboards is provided from extensible back-end, which is responsible for parsing various formats of network data inputs, compressing, aggregating and presenting in acceptable for front-end format.

Tools used to achieve the same functionality as this project (**Grafana/Kibana**) provide generic way of working with data, which leads to unmet expectations of possibilities for analysis of network forensic data. Moreover, aforementioned tools enable only a small set of visualizations useful for described use-cases, therefore most of the information is presented via text/tables, which by definition are less graphical and easy for perception. The final drawback of these solutions is no support for data processing, they only consume network data in predefined formats, therefore are unable to optimise the pipeline in one of the most performance heavy parts of it.

On contrary, this project provides the users with tailored visualizations suitable for analysis of network data. Front-end works in tandem with back-end services through REST API, which enables optimization of raw data by seamlessly compressing it (60-80 % compression achieved with normal data-sets), further formats of input data are supported and with extensibility of back-end possibilities are to accept more formats.

2. Forensic Data Analysis

2.1 Forensic network data

Digital forensics are classically defined as the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations [2].

Preservation and collection of the evidence is done with the help of the **packet sniffing** tools like Wireshark. These tools usually run on server with various applications and capture incoming and outgoing traffic in format of packets. **Packet** is a group of bits that includes data and control information. It generally refers to a network layer (OSI layer 3) protocol data unit [3]. Figure 4 showcases the GUI of Wireshark as well as insides of a packet. These data are stored in format of `.pcap` file, which represents a collection of packets and metadata about them.

2.2 Analytical Dashboards

A **dashboard** is a visual display of the most important information needed to achieve one or more objectives; consolidated and arranged on a single screen so the information can be monitored at a glance [4]. An example of dashboard is displayed on Figure 5.

According to the goal of this project our main priority is to design an **analytical** dashboard. Analytical (tactical) dashboards provide aggregated data and help to analyze the data. They tend to present snapshots of data to provide a better look at the context of the data [4].

One great possibility that this type of dashboards provides us, is the aggregation of the data, particularly aggregation of data based on time. **Time series** data can be very performance-heavy for processing with little to no impact on final analysis. Therefore, time-series data can be compressed in so-called **time buckets**. This is done via partitioning the time windows into sub-windows defined by various algorithms [5], which help to select the most optimal indexes for partitioning and sizes of buckets.

3. System Architecture

The System is divided in two parts : data processing (back-end) and data representation (front-end, dashboards). High-level overview of system architecture is displayed on Figure 6.

3.1 Data processing

Data is acquired from software like Wireshark, Tshark or tcpdump. It captures raw network communication and stores it in `.pcap` format files. These files are then uploaded via POST request to server. Spring Boot in combination with Kotlin as the main back-end language contribute to system's high performance and extensibility. Parsing of `.pcap` files is done with help of `libpcap` native library, which is used by Wireshark for capturing packets and crafting `.pcap` files. Wrapper library `jnetpcap` provides hooks for communication between server and `libpcap`. This enables us to parse `.pcap` directly inside server loop and eradicates OS and HDD as mediator, which greatly improves parsing performance.

Output of parsing is plain table of the `.pcap` data. It is then stored in `TimescaleDB`. `TimescaleDB` is a framework built on top of `PostgreSQL`, which enhances database interactions with time-series data as we are working directly with time-series network data. `TimescaleDB` provides us with a great feature of compressing data based on time buckets, with help of which average data set could compressed up to **80 %**, with little to no drawbacks to analytical process. Here is a trivial SQL snippet showcasing usage of time-bucketing function:

```sql
select sourceIp,
       time_bucket(1000, packetTime) as pt,
       protocol,
       sum(octets)
 from _
 group by sourceIp, pt, protocol
```

Listing 1. Time bucket example

Here the `sourceIp, protocol` are qualifier fields, helping the algorithm to identify packets (rows), which may be merged together. If a packet with the same qualifiers is present within bucket size, which is 1000ms in the snippet, it gets merged with another packet preserving qualifier fields, but aggregating quantifier

Data set	Raw packets	Raw size (KB)	Bucket packets	Bucket size (KB)	Compression ratio
EMEA_2015-05-19-08-54-49.pcap	321949	37806	41215	5226	86 %
EMEA_2015-04-22-05-34-14.pcap	510157	59760	89766	11337	81 %
EMEA_2015-04-21-09-32-38.pcap	360849	42328	63320	8003	81 %
EMEA_2015-04-21-20-27-00.pcap	304608	35708	63101	7970	78 %

Table 1. Bucket compression statistics

fields (octets in the snippet), so the analysis will not be affected in any way.

Advantages of this feature are better described with statistics displayed in Table 1.

Then, the server operates with compressed data on demand by requests from front-end. Main features of the server logic are data aggregation and transformation. According to type of transformation, requested by front-end, the server will arrange data in a way, which front-end may seamlessly consume. Front end requests data via POST request with structure listed below:

```
requestsByClient: {
  bucketized: true,
  type: 'bar',
  mapping: {
    qualifier: 'x',
    quantifier: 'y',
    aggregator: 'z'
  },
  query: 'select client as x,
                 count(*) as y,
                 timeStamp as z
          from :pcapName
          group by x,z
          order by y'
}
```

Listing 2. Data request structure

Here:

- `bucketized` - identifies, whether data will be selected from raw(`false`) or compressed(`true`) data set
- `type` - identifies type of transformation, required by front end visualization
- `mapping` - maps query return values to aggregation return values
- `query` - query to database

3.2 Dashboards

Data is represented in format of 4 dashboards for 4 aforementioned use-cases. Dashboard example is displayed in Figure 7. Every dashboard has a so-called data map, which is a structure, representing format and source of data required for supporting visualizations. Data map consists of an array of data request map displayed in Listing 2.

Every dashboard provides user with a time line graph, which enables filtering on all visualizations by time.

Dashboards also includes visualizations suited for forensic network data analysis. Some examples are:

3.3 Bar chart

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent The bars can be plotted vertically or horizontally. A bar graph shows comparison among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value. Some bar graphs present bars clustered in groups of more than one, showing the values of more than one measured variable[6].

Bar charts are one of the most effective ways to communicate when one variable is quantitative and the other variable is categorical.[7]

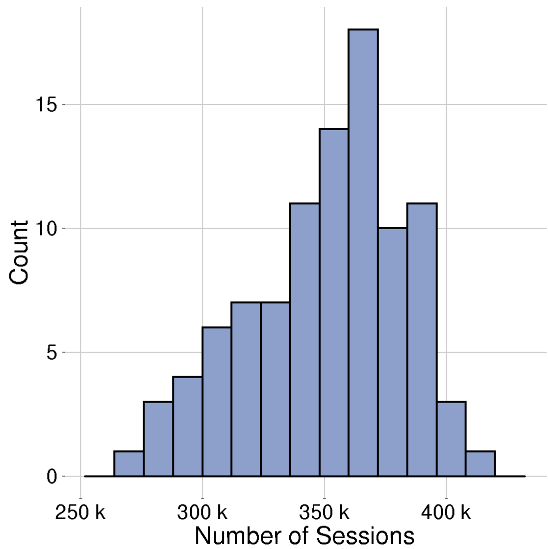

Figure 1. Bar chart example

3.4 Line graph

A line chart is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments.[8]

Line charts show time-series relationships using continuous data. They allow a quick assessment of acceleration (lines curving upward), deceleration (lines curving downward), and volatility (up/down frequency).

Figure 2. Line chart example

3.5 Sankey diagram

Sankey diagrams are traditionally used to visualize the flow of energy or materials in various networks and processes. They illustrate quantitative information about flows, their relationships, and their transformation. Sankey diagrams represent directed, weighted graphs with weight functions that satisfy flow conservation: the sum of the incoming weights for each node is equal to its outgoing weights [9].

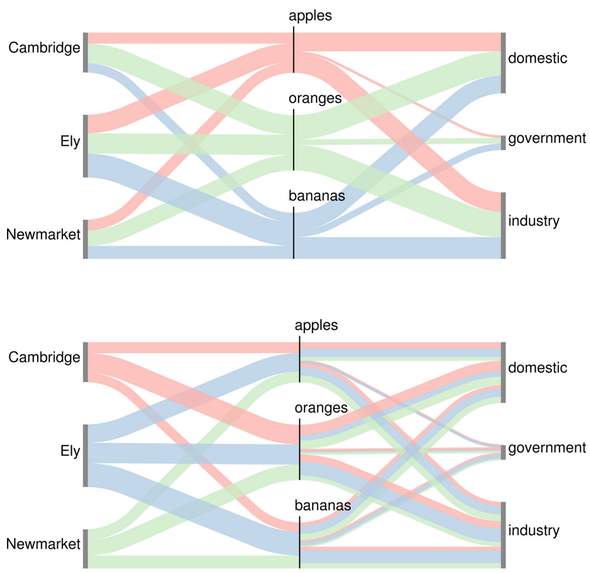

Figure 3. Sankey diagram example [10]

3.6 Libraries

All displayed visualizations are provided from `nivo React.js` wrapped framework, time-line widget is provided by `Victory Chart`. `React.js` enables us to design dashboards with several different implementations of graphs, by encapsulating logic inside `Component`.

4. Conclusions

This paper describes implementation of system designed for analysis of forensic network data. Various optimization techniques were used to enhance performance. Contemporary visualization systems were examined and several flaws were adjusted to meet user needs.

System was tested on 100 `.pcap`, where no adjustment were made to provide graphical analysis. On average 60-86 % of data was compressed with time buckets and this saved around 20-30Mb of Internet usage.

The contribution of this project is a system, which can be used on network to provide analysis of forensic activities going on. It also provides an extensible and modularized back-end, which later can be reused for different visualization project.

Project can further be improved by implementing dynamic data compression meta parameters, adding new visualizations or new input data formats parsing.

Acknowledgements

I would like to thank my supervisor Jiří Hynek for his help, collaboration and advice.

References

[1] David Buil-Gil, Fernando Miró-Llinares, Asier Moneva, Steven Kemp, and Nacho Díaz-Castaño. Cybercrime and shifts in opportunities during covid-19: a preliminary analysis in the uk. *European Societies*, 23(sup1):S47–S59, 2021.

[2] Gary Palmer. Report from the first digital forensic research workshop (dfrws). *Tech. Rep.*, 2001.

[3] *Business data communications*. Prentice Hall, Upper Saddle River, N.J., 2000.

[4] Stephen Few. *Information dashboard design: The effective visual communication of data*. O'Reilly Media, Inc., 2006.

[5] Sanjeeb Dash, Oktay Günlük, Andrea Lodi, and Andrea Tramontani. A time bucket formulation for the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(1):132–147, 2012.

[6] C.L. Bhalla. *Audio-visual Aids in Education*. Atma Ram, 1963.

[7] Jay Jacobs and Bob Rudis. *Data-driven security: analysis, visualization and dashboards*. John Wiley & Sons, 2014.

[8] B.G. Andreas. *Experimental psychology*. Wiley, 1965.

[9] P Froehlich. Interactive sankey diagrams. In *IEEE Symp. on Information Visualization*, page 233, 2005.

[10] R.C. Lupton and J.M. Allwood. Hybrid sankey diagrams: Visual analysis of multidimensional data for understanding resource use. *Resources, Conservation and Recycling*, 124:141–151, 2017.

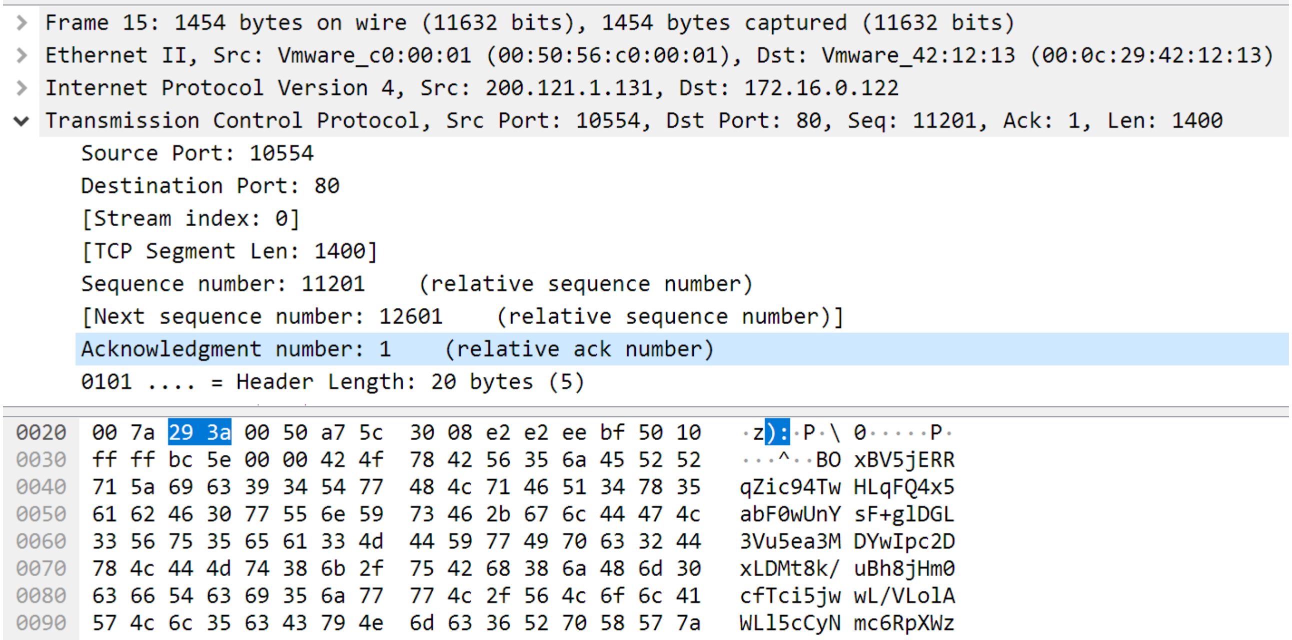

Figure 4. Screenshot of Wireshark GUI with packets data

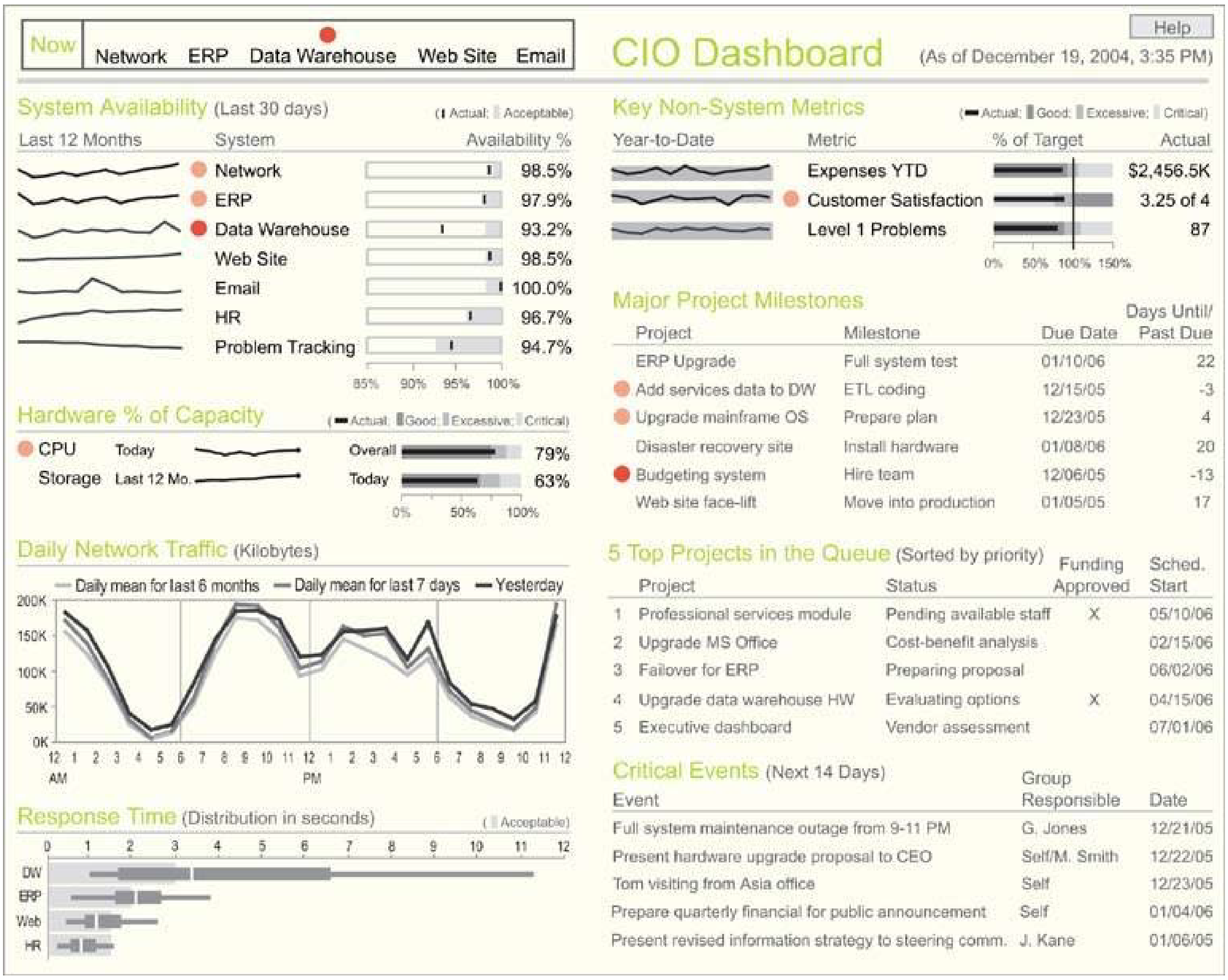

Figure 5. Dashboard example [4]

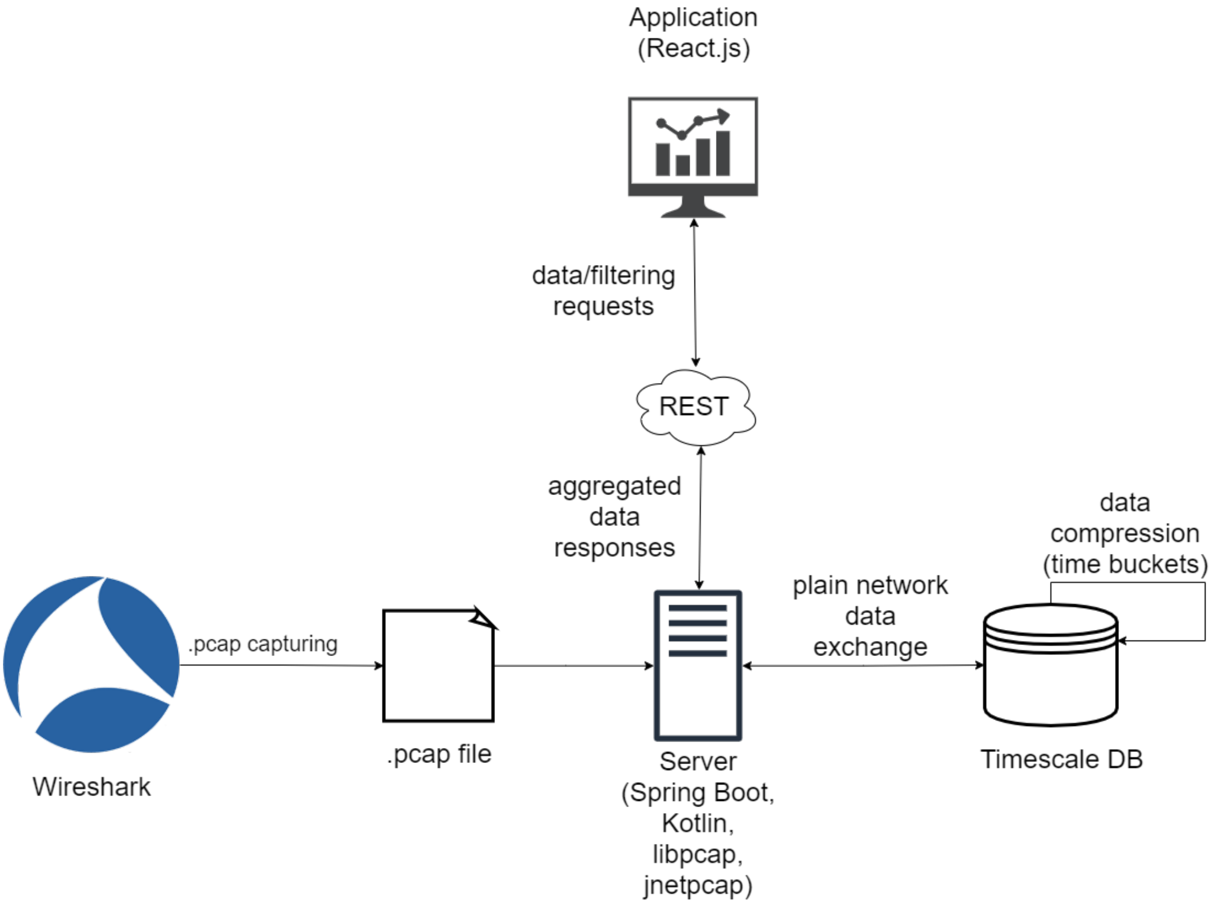

Figure 6. Architecture of the system

Figure 7. Dashboard screenshot

1. Main dashboard settings button. Enables user to select data-source, bucket size of aggregation of the data-source.
2. Selector of query type for dashboard. Option ”octets” will execute visualizations and aggregations on sum of octets for given IP address, ”packets” - on amount of packets.
3. Time line widget. Sets time filters displayed on the left as ”`startDate`” and ”`endDate`”.
4. Table of filters. Each filter can be removed independently by clicking trash icon.
5. Bar chart visualization. Just like other visualizations is based on color, which represents main aggregation value for the visualization (Source IP in this case).