Ersch. in: VizSec '13 : Visualization for Cyber Security ; Atlanta, GA, USA - October 14, 2013 / John Goodall, ... (eds.). - New York : ACM, 2013. - S. 65-72. - ISBN 978-1-4503-2173-0

http://dx.doi.org/10.1145/2517957.2517966

Finding Anomalies in Time-Series using Visual Correlation for Interactive Root Cause Analysis

Florian Stoffel
University of Konstanz
Florian.Stoffel@uni-konstanz.de

Fabian Fischer
University of Konstanz
Fabian.Fischer@uni-konstanz.de

Daniel A. Keim
University of Konstanz
Daniel.Keim@uni-konstanz.de

ABSTRACT

Monitoring computer networks often includes gathering vast amounts of time-series data from thousands of computer systems and network devices. Threshold alerting is easy to accomplish with state-of-the-art technologies. However, to find correlations and similar behaviors between the different devices is challenging. We developed a visual analytics application to tackle this challenge by integrating similarity models and analytics combined with well-known, but task-adapted, time-series visualizations. We show in a case study, how this system can be used to visually identify correlations and anomalies in large data sets and identify and investigate security-related events.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.3.8 [**Computer Graphics**]: Application; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Network Security, Visual Analytics, Correlation, Time-Series, Anomalies

1. INTRODUCTION

Nowadays, computer networks are used by almost all people in everyday life. In addition, the economic importance makes computer systems a valuable target for a large number of different targeted and wide-spread attacks. Obviously, monitoring is, therefore, indispensable in all productive environments to make sure to identify suspicious anomalies early and to be able to investigate the root causes in a timely manner. Monitoring computer networks often includes gathering vast amounts of time-series data from thousands of computer systems and network devices. While threshold alerting is easy to accomplish with state-of-the-art technologies, finding correlations and similar behaviors between the different devices is still challenging. Especially the task of analyzing the shear amount of time-series to find the related ones is often not possible interactively and there is less computational support to guide the analyst in this process. In our approach, we make use of the visual analytics [10] approach, which combines automated methods and the human capabilities in recognizing interesting patterns using background knowledge. Our system uses analytical models to highlight interesting and anomalous parts within time-series to make possible important events more visible to the analyst. Using visual exploration the analyst can benefit from the system's drill-down capabilities and similarity search across all other time-series to retrieve related data to eventually identify the root cause of the suspicious events. Subject of this work is the design of techniques, which employ network time-series correlation analysis to track down such incidents. The resulting system combines the techniques to be used for analysis and detection of incidents of various kinds.

According to Fink et al., it is very common for network analysts to utilize correlation in their daily work: "*Analysts perform standard types of correlation in the course of their normal work, such as correlating network flows to process activity.*" [5]. In the same work, the authors quote analysts, that there is only very little visual support for such tasks. In our work, we therefore concentrate on creating a framework explicitly targeted at providing support for visual correlation of network time-series data.

The three main contributions of this work are the following: (1) A visual analytics system, which provides tight coupling of analytical models and the visual representation of thousands of time-series to enhance visual correlation recognition. (2) A lens-based line chart widget designed to specifically focus on correlations of sub-segments between time-series. (3) An implementation of a time-series storage optimized for the use in a visual analytics application.

The remainder of this paper is structured as follows: In Section 2 we briefly discuss related work in the field of monitoring of system metrics and visualizing time-series, which are highly related areas of this work. To introduce the overall system of our approach, we explain the different server and client modules and in Section 3. Additionally, we briefly introduce the time-series modeling and the used techniques. We continue in Section 4 to describe the graphical user interface and the visualization components of the implemented application and show in Section 5 a case study, how the system can be used to analyze large data sets. Finally, we

conclude in Section 6 and briefly discuss limitations and future research perspectives.

2. RELATED WORK

The most extensive overview of visualization techniques for time-dependent data can be found in the book of Aigner et al. [1] providing a systematic overview and survey of many existing visualization techniques. A very compact visualization techniques is called two-tone pseudo coloring [16], which uses two discrete colors for each value of the time-series. This technique is also used and implemented in the so-called horizon charts, which properties has also been compared in [7] against line charts. However, using color to represent the value, restricts the further usage of color for highlighting critical or suspicious segments, which is an important feature in our system. Additionally, we are not so much interested in easy-to-detect peak values and the precise readability of the visual representation, which is a key advantage of horizon charts. It is more important to recognize shapes, correlations and patterns, where commonly used line charts provide a good basis.

The graphical perception for multiple time-series and line charts has been evaluated by Javed et al. [8], who showed that the presentation of time-series as small multiples is generally more efficient for comparisons across time-series with a *large* visual span. Plotting several lines in the same diagram was more efficient for comparison of *smaller* visual spans. This shows the trade-off we are confronted in our system, because we are actually interested in large visual spans to convey the overall shape *and* small visual spans to correlate interesting anomalous segments against other time-series. *ChronoLens* [18] is a highly interactive approach which enhances the exploratory analysis of times-series. The user can select parts of the line charts. The data of the selected segment is automatically transformed to show derivatives, correlations or other derived time-series for the selected focus lens area. This tightly integrates visual analysis with user interaction and provides good means to deeply analyze multiple line charts. With respect to the number of shown time-series, the *Line Graph Explorer* [12] is much more scalable, because it provides a compact overview using colored pixels positioned on a single line for each time-series. Selecting those pixel lines provide a lens mode to give more space to the selected metrics to be shown as standard line charts. This tool provides a compressed visual representation, which is very good to catch the overall global similarity of many time-series. However, if you need to explore many time-series in detail using the lens, the scalability degrades. While our system is quite similar to the *Line Graph Explorer*, we have a stronger focus on comparing specific segments across thousands of metrics using the line chart to represent the relative value and using colored highlighting to emphasize the deviation to the underlying analytical model.

Network and system performance time-series are often analyzed to identify security-related incidents in large computer networks. *ClockMap* [6] focuses on the hierarchical structure of such networks and uses a circular treemap layout with embedded temporal glyphs to represent many metrics within its context. Systems like *LiveRAC* [14] focus specifically on the analysis of monitoring data which is highly relevant from a network security perspective. They also use semantic zoom and combines it with a reorderable grid with embedded charts to represent many time-series. This system was also used by Best et al. [2] for network traffic data combined with advanced time-series analysis based on *Symbolic Aggregate approXimation* [11] to find unusual sequences to improve network security and to provide real-time situational awareness. Shafer et al. [17] also provide a visual analysis system for system monitoring to identify anomalous machines based on time-series of computer networks decomposing significant bursts and long-term trends. A good overview of related wavelet-based techniques to provide anomaly detection in network traffic for security-related applications can be found in [9]. The authors discuss and compare several applications with a focus on data traffic visualization tools to enhance security.

3. SYSTEM

This section describes the two main parts of our system, which are: A *stand-alone server* managing the time-series data and providing analysis, query and retrieval related functionality, and a *rich client*. The latter provides access to the time-series data and allows the user explore and analyze the data. Multiple clients can operate independently from each other with data form the same server instance. Both components communicate via a RMI network link.

3.1 Server

The server is a stand-alone application. It contains the components to retrieve data from various external sources, a high performance time-series data store, a modeling facility for time-series data, and extendable query and retrieval functionality. A schematic overview of the components can be seen in Figure 1.

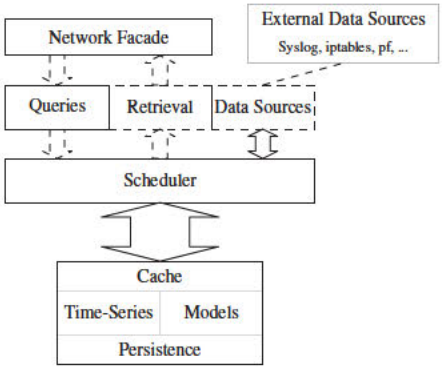

Figure 1: Schematic overview of the server components.

The core of the server is the custom time-series database, which acts as the time-series persistence layer. Although designed as a high performance retrieval system for time-series data, it can also be used to store the generated model data (see Section 3.1.1). If required, the database fills out missing values or re-samples the data with a given interval transparently on inserts of the raw data linearly. This leads to a consistent dataset without missing values, which allows the simplification of further analysis and the processing outside the server. In addition, the resulting time-series are continuous in the time domain, which is a requirement for the Fourier analysis described in Section 3.1.1.

The query facility can be adapted to the actual usage scenario, providing the best possible support for different query

types and the corresponding computation tasks. This is reflected in the API, where the client can query for supported query types and their parameters. By default, the query facility provides similarity queries. By specifying an originating time-series or its model and a time-span, the server can search in a set of given candidates or the complete, available time-series stored locally. By default, the distance of two time-series is computed by the Euclidean Distance of the normalized query region. Thanks to the high retrieval performance, the server can finish a time-series query on a dataset of around 1,1 million time-series in about a minute (10 months of data, indexed in five minutes intervals, Intel Core 2 Quad Processor, 8 GB of main memory, Intel X-18 SSD).

3.1.1 Time-Series Model

Besides the data restoration and sampling, a model of the time-series is created or updated when new data is inserted in the database. This model can be retrieved by the client and supports additional visualization and analysis methods.

In general, there are certain key observations characterizing a network time-series on two different levels. The first level is the intra-day level, where the observations refer to phenomena lasting a few hours. Some of those typical characteristics can be seen in Figure 2 on the time-series drawn with the solid line.

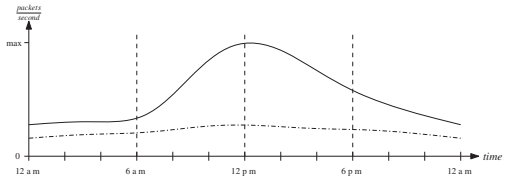

Figure 2: Two typical network traffic charts. The solid line is a exemplary labor day time-series, the dashed-dotted line resembles a typical non-labor day network time-series.

What at first stands out is the peak around noon. A typical observation is also the increase of the series values starting around 6 a.m. leading to the peak at noon. Another noticeable pattern is the slow decrease of the peak at noon, compared to the fast increase in the morning.

The second level where key observations can be made is not the intra-, but the day level. A good example of such a day level key observation can be made when comparing the overall shape of a time-series of labor- with non-labor days (solid versus dashed-dotted line in Figure 2). In the given example, one can easily distinguish the non-labor from the labor-day by not having a very high level around noon and the relatively constant level of the series.

Those observations are the motivation of creating the time-series model per-day. Each time-series is modeled by seven independent models describing one weekday. There is no distinction in holidays or vacations, which preserves the maximal generality of the model on server side. Such adjustments should be made on client side, where in the ideal case the user can interactively adjust any kind of filters or modifications on the data. This also opens possibilities for task-specific adaptations of the model, where the server is just providing general data and the client adapts them in a task specific way.

The model for one time-series contains two different models created by Fourier and wavelet transform of the time-series [3, 15]. In general both methods can be used to analyze and model time-series data. The Fourier transform decomposes the signal in components, where each of the component can be interpreted as a longer or shorter lasting phenomena in the time-series data. Besides this advantage, the frequency domain data resulting from the Fourier transform looses its time dimension. Therefore, it is almost impossible to properly model non-stationary signals which may change the frequency over time, or very short lasting phenomena in general.

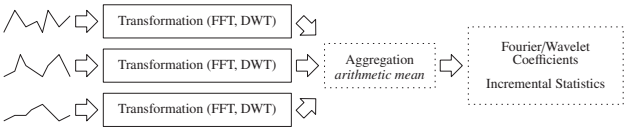

Figure 3: The model creation pipeline. From the left to the right, the raw time-series are transformed and aggregated to the model containing the Fourier and wavelet coefficients.

To overcome this limitation of the Fourier transform based models, an additional model based on the wavelet transform has been added. The major advantage of the wavelet transform is the dynamic window size, since the actual wavelet function is scaled to fit the input in data and time domain. Together, both parts of the model can accurately capture different longer lasting effects and also capture short phenomena in the time-series. To maintain the general nature of supported analysis tasks by the server and the models, there is no combination on the server side of the Fourier transform and the wavelet transform of the time-series, but band-filtering of the models is supported. By choosing such a design, the server does not restrict the available analysis tasks, but at the same time supports common, potentially computation intensive filter techniques.

To create a Fourier and wavelet model out of different days, the resulting coefficients are aggregated incrementally [13]. Besides being able to compute the incremental arithmetic mean efficiently, a comparison of different aggregation methods has been made by creating models out of 9 weeks of real network time-series data. To judge the quality of the aggregation method, the resulting models have been evaluated with the sum of squared residuals (SSR) of the models and the input time-series (see Table 1 for details).

The resulting model can be used to find anomalies by comparing the actual value of a time-series with its aggregated model. The server returns both, the Fourier and wavelet model, which keeps the design space of the application and its processing and application of the model as general as possible. The single components and computation steps for the model creation are shown in Figure 3.

3.2 Client

The client is built on top of the NetBeans Rich Client Platform (RCP)[1]. This Java framework provides a mature and flexible framework for Swing[2] based applications. Besides having a powerful window management, the platform provides mechanisms for extendable, module based applications. Building on that, the foundation of the client is built

[1] `http://platform.netbeans.org/`

[2] Java user interface toolkit

Aggregation	Mo	Tue	Wed	Thu	Fri	Sat	Sun
Median	5.344e13	8.287e13	3.005e14	4.962e14	4.022e14	5.270e13	1.225e12
Arithmetic Mean	4.453e13	6.872e13	1.752e14	4.063e14	2.926e14	4.651e13	1.051e12
Geometric Mean	9.070e14	1.420e14	4.186e14	7.360e14	5.573e14	9.620e13	2.736e12
Effective Value	5.072e13	7.247e13	1.942e14	4.043e14	3.154e14	5.151e13	1.376e12

Table 1: Comparison of the SSR of models with different aggregation methods for 9 weeks of data. For each day, the minimal SSR is highlighted with a gray cell background.

by three modules (the upper part in Figure 4): the Data Source API, the Time-Series API and the Visualization API and their respective SPIs.

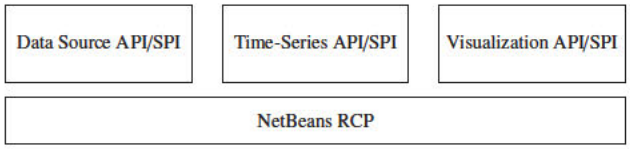

Figure 4: Schematic overview of the client components.

The Data Source API unifies the inclusion of different data sources in the client. This makes it possible to extend the application by other data sources, for example direct database access or flat files. Since the API induces no restrictions to the data handling, the different data source modules can handle the data in the most efficient and effective way.

The Time-Series API is a general contract of accessing the time-series data. This is done mostly by having direct access to the data based on the timestamps. As stated in Section 3.1, the server component can transparently fill out missing values or resample time-series data. The default time-series implementation provides the same techniques. By not requiring data sources to re-use this default implementation, it is also possible to work with un-processed time-series data.

The Visualization API is used to provide a general interface of visualizations on an abstract level. It just handles basic data management tasks and provides a high level window container. The implementation of a visualization is not restricted in terms of the painting or supported interaction techniques. The corresponding SPIs connect the different API implementations and adapt the client interface to the available modules.

All APIs are designed to be as general as possible. This makes it easy to adapt the client to different data sources, different types of visualizations with different interaction possibilities and not at least to allow efficient and effective handling of the data.

4. GRAPHICAL USER INTERFACE

For a visual analytics system, the design of the user interface and the motivation and design choices of the visualizations are crucial. The following section describes the user interface of the client and motivates and explains the design choices of our main visualization in detail.

As described in Section 3.2, the user interface is based on the NetBeans Rich Client Platform. In its general form, the underlying framework provides functionality for managing and accessing different data sources, handling general time-series data, and display data or analysis results with custom visualizations.

4.1 Overall Interface

The overall interface of the client can be seen in Figure 5. It is composed out of three main areas. On the left, covered by `A` and `B`, there is the general data control and action area. The center of the user interface, area `C`, is designated to hold the visualizations. On the right area (`D` and `E`), context sensitive displays and controls are placed. In Figure 5, there are the main controls of the visualization, `D`, and a mouse position dependent information pane, showing information of the currently hovered time-series in the visualization, `E`. The same panel is also used to display details on demand, if the current context provides such detail information. For example, if segments of one or multiple time-series are selected, the minimum and maximum values of those selections will be shown.

This clear separation makes it easy to access all features, the context sensitive parts on the right make it possible to keep the separation of the areas even with changing visualizations or actions.

Thanks to the window management provided by the NetBeans Platform, all parts of the user interface can be detached and freely placed on or even outside of the main application window. This also adds support for multi-monitor workplaces. Besides of having such support on the window level, the Visualization API contains an optional part dealing with visualization synchronization which supports multiple visualizations at the same time. Based on the actual type of the visualization, this feature allows sharing of the current data, view port or even single configuration parameters. In case of visualizations of the same type, different instances can also share their configuration.

4.2 Time-Series Overview

The main visualization of the Client is the so called ExplorerView. It can be seen in the center area of Figure 5 labeled with `C`.

The ExplorerView is built to support the following main tasks: *Shape recognition.* Similar time-series should have similar visual appearance and shape. *Correlation recognition.* Users of the client should be able to visually identify time-series with high correlation. *Pattern recognition.* The visualization should enable the user to recognize similar patterns in different time-series. In addition, those tasks should also scale for large amount of time-series.

Our visual approach takes the context of the time-series into account and allows refinements of the visual representation, which is desirable in order not to loose any information. Also, the time resolution of the ExplorerView is freely adjustable and the visual appearance can be adjusted to fit the task best. This variety is very hard to support with

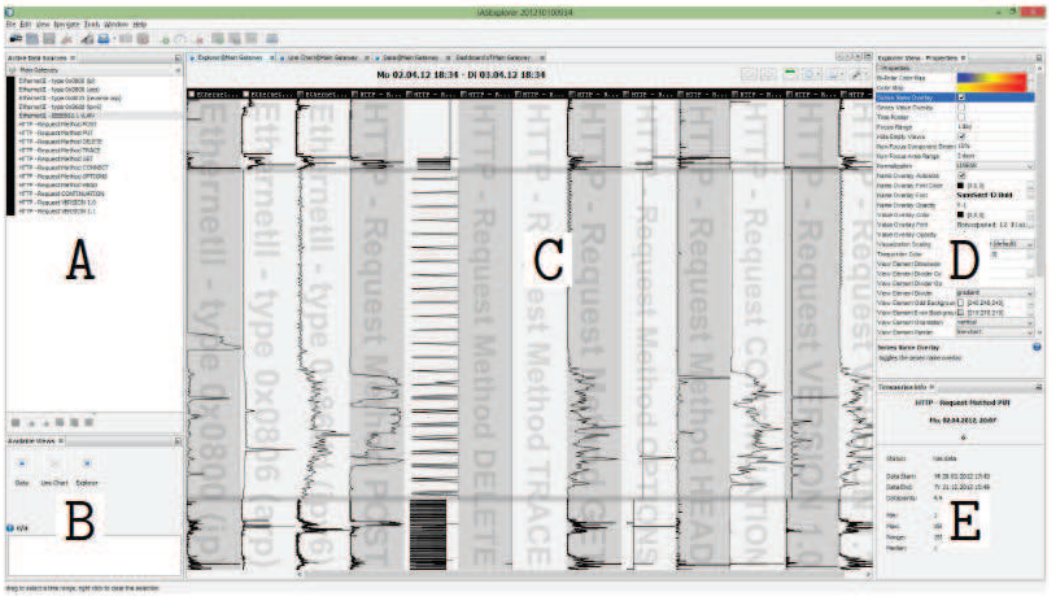

Figure 5: The main interface of the client. `A` labels the data management and time-series selection window, `B` lists different available visualizations, `C` denotes the main area where the visualizations are shown. `D` identifies the visualization configuration and the area marked with `E` is a context aware information panel.

automatic methods. The visual interface also allows exploration and browsing through the data, which should create a picture of the network condition and its usual patterns.

In addition to fulfilling the task specific requirements, line chart based time-series visualizations have two further advantages. Annotating data in line charts is straight forward by re-using the usually empty area in the background of the chart. Besides having the possibility of enriching line charts with additional data, the scaling invariance of the actual line shape facilitates level independent shape, correlation and pattern recognition.

Due to the layered network architecture, this property is desirable because a network operation can have effects on different network time-series. For example, browsing to a website generates data in (not only) the following network time-series: `ip traffic`, `tcp port 80` and `http`. Therefore, it is very likely that time-series, generated from the different layer data, are composed of parts of the same operations. Scaling the series in a fixed range, for example $[0 \dots 1]$, creates similar line charts in terms of their shape and correlation. Obviously, this also helps with the visual correlation and pattern recognition.

The line charts in the ExplorerView differ in one important aspect from common line charts. The time axis is not on the horizontal, but on the vertical. While this it not conform to the common line chart displays, it has an effect on the perception of the operator. In the Western world, people are used to read text and charts from the left to the right. This is also the case for line charts. This leads to the behavior, that viewers tend to follow a single line chart, instead of comparing them to each other even if there are multiple charts drawn next to each other. By placing the time axis not on the horizontal, but on the vertical, we force the viewer to break this habit, and try to direct the perception to comparing different line charts. The ExplorerView does not force this rotated view, the single series

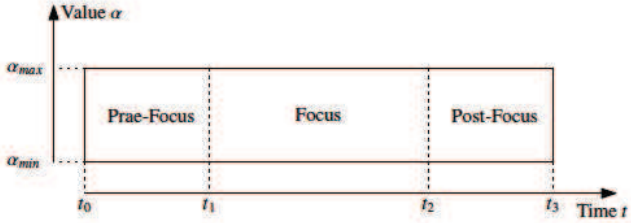

Figure 6: Focus-plus-Context line chart. Prae- and post-focus areas are building the context area. In our case it holds that $t_3 - t_2 = t_1 - t_0$

displays can also be rotated by 90 degrees, which results in a common line chart arrangement.

To account for the nature of repeating patterns in the data, it is desirable that the visualization is able to put the currently focused pattern in the larger context of the series. To support that, the line chart is divided into three parts: the prae-focus, focus and post-focus area. The prae- and post-focus area are building the context area, the focus is located in the middle of the visualization area. See Figure 6 for the construction of context and focus area.

One key issue is the blending area of the non-focus with the focus area, which is caused by the different scaling of focus and context area. There are numerous different methods of techniques that those areas of different scales have a smooth transition to each other, for example based on a Gaussian kernel or hyperbolic functions [4]. In our case, comparison and exploration requires to have the current interesting points in the focus area of the visualization. To have a steady reminder of the different scales in terms of time and to minimize artifacts introduced by distortions introduced by the time scaling techniques, the ExplorerView uses a sharp transition from the context to the focus area.

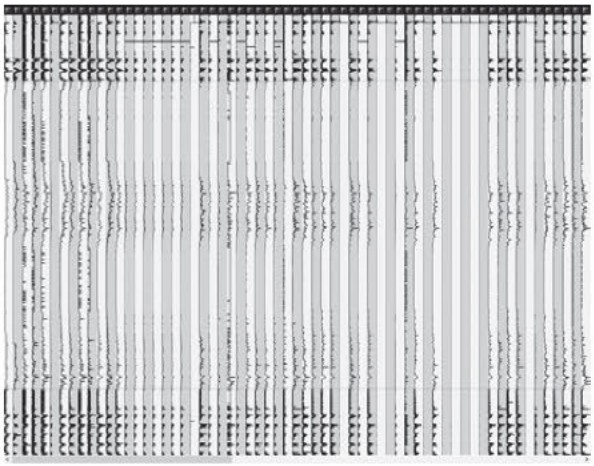

Figure 7: 63 different time-series. Each series is scaled in a way, that the data of all time-series fit on a common workstation display. The order of the time-series plots is determined by the volatility of the data in the focus area.

An additional shadowing around the area transitions can be enabled, to make the actual borders of the three areas clear to the observer. This shadowing can be seen in Figure 5.

Each time-series is displayed as a single line chart, which according to Javed [8] is the right choice for the discrimination and therefore also the compare task. In the same work, the authors show that displaying time-series with less space has little influence on the time the analyst needs to accomplish a given task. The smaller size has only an effect on the ability of estimating the value of the time-series, which is not a key issue in the tasks the ExplorerView is designed to support. Especially for tasks, where many time-series has to be considered at once, this property is important. To fit as much time-series on the available display space as possible, all plots in one ExplorerView instance can be freely re-sized to fit the needs of the task and the visual abilities of the analyst. In Figure 7, a view with 63 different time-series is displayed. Although the space used to display the time-series charts is very small, it is possible to get an impression their shapes and compare with each other.

Creating and executing queries on the displayed time-series is supported by the visualization. To issue a similarity query, the analyst can choose an area of a time-series via clicking and dragging the query time-span directly on the visualization. After selection the query range, it is possible to narrow down the search space and name the query before it is executed. The results can be displayed in any ExplorerView instance and inspected visually.

5. CASE STUDY

In this section, we describe how our system can be used with an exemplary use case performed by a fictive analyst. Although the use case is created just for the sake of showing the capabilities of our system, the data set comes from a computer network with around 20 users (for details see Section 5.1.1). Due to the general nature of network traffic, the definition of an *anomaly* can be different. In the following, we define an anomaly as a significant deviation from the usual traffic levels. The threshold of allowed deviation from the time-series to the model can be adjusted in multiples of variances of the time-series model.

Name	Service
TCP Port 25	email transport (SMTP)
TCP Port 194	IRC
TCP Port 465	email transport (SMTPS)
TCP Port 587	email transport (SMTPS)
TCP Port 6667	IRC
UDP Port 53	name resolution (DNS)
Network traffic	aggregated network throughput

Table 2: A time-series group containing some of the network time-series belonging to the most widely exploited services.

5.1 Root Cause Analysis of Anomalies in Network Time-Series

5.1.1 *Data Set*

For the following example, the Internet traffic of a small computer network with a mixed environment of around 30 workstations and servers with about 20 regular users, has been analyzed on different network layers. To do so, a so called *probe* analyzes the traffic going through a central switch by trying to match *descriptors* to the data. The analysis system contains descriptors for different protocols like TCP or UDP, SIP or HTTP, and application specifics, for example for each IRC command. For each of those, a numerical counter exists, which is incremented each time a descriptor matches. The counters are transmitted in five minute intervals to a data store, from which applications can retrieve the counter values and build a discrete time-series out of them. In the deployed system, a total of 1.6 million descriptors are contained, from which around 300,000 matched in the captured traffic of the observed network.

Since the data set contains numerical counters only, sensitive data like source ip, destination ip, or the application payload can not be stored, which protects the privacy of the users. While it is possible to use this data set for traffic and application usage analysis, it is not possible to conclude which workstations or servers are behaving anomalous. To overcome this limitation, multiple probes can be added to different subnets or in front of single servers. Unfortunately, in our environment this was not possible due to user concerns regarding their privacy.

5.1.2 *Searching for anomalies*

Our example begins with the analyst browsing through the network time-series data. Our system is capable of storing groups of time-series, so that if the active data source contains series with the given name, they can be loaded quickly. In our example, the analyst has created a time-series group containing the time-series shown in Table 2. This group contains time-series describing the most vulnerable services, which are usually target of attacks and are used to be exploited in various ways. Therefore, anomalies in those series require special attention, because they are most definitely a sign of unwanted network activity.

To support browsing through the data, the ExplorerView visualization is switched to the model difference mode, where significant deviations of a time-series from it's model are highlighted with a blue (lower value as modeled) or red

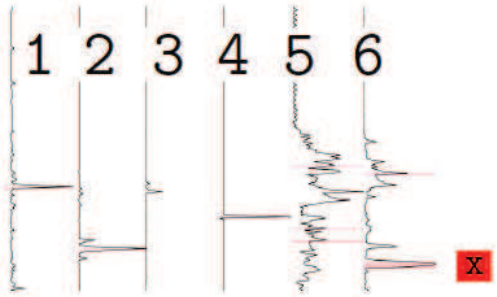

Figure 8: Detail from the EXPLORERVIEW. The time-series are: `1: TCP 25, 2: TCP 465, 3: TCP 587, 4: TCP 6667, 5: TCP UDP 53, 6: Ethernet traffic`. The time-series for port `194 TCP` is missing, because it contained no data and is excluded from the view. The red `x` is placed on the right of detected anomaly.

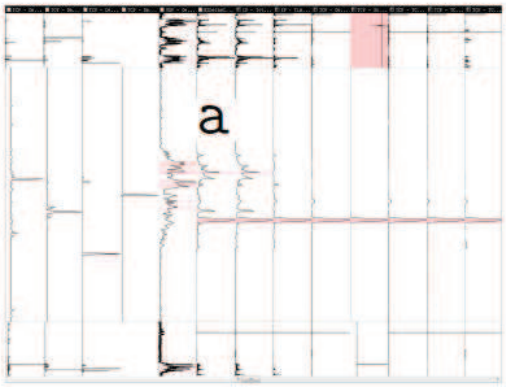

Figure 9: The EXPLORERVIEW showing the time-series selected at the beginning (the first six) and the time-series returned by the server by the similarity query (the last seven).

(higher value as modeled) background. This view mode is realized by querying the server for the time-series model, applying the reverse transformations with the configured band-filters (see Section 3.1.1), computing the differences of the model and displaying them in the background of the line-chart.

The browsing task can be performed by pressing the arrow keys or scrolling through the data with the mouse wheel. This simple interaction induces only low cognitive effort and allows the analyst to concentrate on the visual correlation of the time-series and on detecting anomalous areas via the background color of the visualization.

By browsing through the data, the analyst spots an area, where the general level of Ethernet traffic has a very significant spike, which is identified as a large deviation from the model, see Figure 8 the red `x` at series 6.

Selecting the range of the anomaly with the mouse and formulate a similarity query, which is executed on the server is the next step towards identifying the cause of the traffic spike. After the query has finished, the analyst has the possibility of getting a list of resulting time-series ordered by their similarity, or adding them in the visualization for visual correlation analysis.

Both, the visualization (Figure 9) and the list of similar time-series (Table 3) indicate a very large, unexpected

#	Time-Series
1	IP - Packet length between 0 and 255
2	IP - Packet TTL between 64 and 95
3	TCP - Destination Port 22
4	TCP - Source Port 36761
5	TCP - Destination Port 22 and packet length 0 - 255
6	TCP - Window Size 4096 - 4351
7	TCP - Window Size 3840 - 4095

Table 3: The first seven series returned by the server when the analyst queried for the anomaly region he visually identified.

transfer of data to machine outside of the monitored network on port 22 TCP. On the visualization side, the analyst can clearly see that the spike of the Ethernet time-series (marked with an `a` in Figure 9) is contained in all other visible time-series on the right of the originally queried series. By that, the analyst can conclude that there are some very good candidates to get an impression of the application and the actual use case from. This is strengthened by the fact, that there are also no anomalous spikes in the focus area of the visualization. This is additional information which can not be seen when just a list of similar time-series is returned by the server.

For all displayed time-series, the spike detected in the aggregated network traffic is an anomaly which can be easily spotted by the operator.

Having a look at those series (Table 3), it becomes clear that a large transfer of data has happened. The destination port 22 TCP is usually used for SSH based services, and there are some protocol which use SSH as transport protocol for their application data like *SFTP*[3] or rsync. Together with the detected anomaly in the aggregated network traffic, the analyst can conclude that most likely a large transfer of data from the internal network to a machine in the Internet has been executed.

It has been mentioned in the introduction of this section, that due to privacy concerns, the network data analysis has been done on a very abstract level with just counting matching descriptors. In this example, the analyst can therefore generate very plausible explanations of the observed spike in the aggregated network traffic. Our system could be able to determine also the source of the data transfer, if there would be more probes available. On the server this would not need any changes, because the modeling is done for each time-series separately. Also, the client is able to access multiple data sources simultaneously. If this is the case, the name of the time-series would be extended with the name or id of the generating probe, which makes it possible to identify the source of the displayed time-series – and in this example the source of the file transfer.

6. CONCLUSION

In this work, we presented a visual analytics system for analyzing, examining and investigating time-series data. It provides tight coupling of analytical models and visual representations capable of mining through vast amounts of time-series data. To support this task, the system features a focus plus context or lens based line chart carefully designed for

[3]SSH File Transfer Protocol

displaying correlation of sub-segments of time-series. All analytical and visual tasks are not possible without the support of a high performance time-series storage, combined with a scalable analysis framework. The usefulness of the design has been shown with a case study where the system allows an analyst to determine possible causes of a traffic anomaly.

6.1 Future Work

In the future, we plan to extend the current analytic models to provide a more sophisticated analysis. The server component could suggest certain band filters, in order to make specific classes of network anomalies visible.

The ExplorerView could also be enhanced with further visual representations, for example based on glyphs designed specifically for showing anomalies in time-series data. In addition to the automatic ordering of the series, it is also desirable to identify groups and aggregate their visual representation in order to reduce the number of visualizations shown at once. Although preliminary tests and discussions had been promising, the ExplorerView with its 90 degree rotation of the line charts should be formally evaluated to prove its usefulness.

To show the general applicability of our system and the design decisions, we currently add other data sources like DNA sequence data, where visual similarity and anomalies of the data, which can be interpreted as time-series too, play an important role for biologists.

7. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257495, "Visual Analytic Representation of Large Datasets for Enhancing Network Security" (VIS-SENSE).

8. REFERENCES

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction Series. Springer, 2011.

[2] D. M. Best, S. Bohn, D. Love, A. Wynne, and W. A. Pike. Real-time visualization of network behaviors for situational awareness. In *Proceedings of the Seventh International Symposium on Visualization for Cyber Security*, VizSec '10, pages 79–90, New York, NY, USA, 2010. ACM.

[3] P. Bloomfield. *Fourier Analysis of Time Series: An Introduction*. John Wiley and Sons, 2nd edition, 2000.

[4] S. Carpendale, J. Ligh, and E. Pattison. Achieving higher magnification in context. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pages 71–80. ACM, 2004.

[5] G. Fink, C. North, A. Endert, and S. Rose. Visualizing Cyber Security: Usable Workspaces. In *Visualization for Cyber Security, 2009. VizSec 2009. 6th International Workshop on*, pages 45–56, 2009.

[6] F. Fischer, J. Fuchs, and F. Mansmann. ClockMap: enhancing circular treemaps with temporal glyphs for time-series data. In M. Meyer and T. Weinkauf, editors, *Proceedings of the Eurographics Conference on Visualization (EuroVis 2012 Short Papers)*, pages 97–101, Vienna, Austria, 2012.

[7] J. Heer, N. Kong, and M. Agrawala. Sizing the horizon: the effects of chart size and layering on the graphical perception of time series visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1303–1312, New York, NY, USA, 2009. ACM.

[8] W. Javed, B. McDonnel, and N. Elmqvist. Graphical perception of multiple time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):927–934, Nov. 2010.

[9] G. Kaur, V. Saxena, and J. P. Gupta. Anomaly detection in network traffic and role of wavelets. In *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, volume 7, pages V7–46–V7–51, 2010.

[10] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual data mining. chapter Visual Analytics: Scope and Challenges, pages 76–90. Springer-Verlag, Berlin, Heidelberg, 2008.

[11] E. Keogh, J. Lin, and A. Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 226–233, Washington, DC, USA, 2005. IEEE Computer Society.

[12] R. Kincaid and H. Lam. Line graph explorer: scalable display of line graphs using focus+context. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '06, pages 404–411, New York, NY, USA, 2006. ACM.

[13] D. E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley Longman Publishing Co., Inc., 3 edition, 1997.

[14] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1483–1492, New York, NY, USA, 2008. ACM.

[15] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge University Press, 2000.

[16] T. Saito, H. N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. In *Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 23–, Washington, DC, USA, 2005. IEEE Computer Society.

[17] I. Shafer, K. Ren, V. N. Boddeti, Y. Abe, G. R. Ganger, and C. Faloutsos. Rainmon: an integrated approach to mining bursty timeseries monitoring data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 1158–1166, New York, NY, USA, 2012. ACM.

[18] J. Zhao, F. Chevalier, E. Pietriga, and R. Balakrishnan. Exploratory analysis of time-series with chronolenses. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2422–2431, 2011.