Lessons Learned: Visualizing Cyber Situation Awareness in a Network Security Domain

Christopher G. Healey[1(✉)], Lihua Hao[1], and Steve E. Hutchinson[2]

[1] Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206, USA
healey@ncsu.edu
[2] ICF International, U.S. Army Research Laboratory, Adelphi, USA
http://www.csc.ncsu.edu/faculty/healey

Abstract. This chapter discusses lesson learned working with cyber situation awareness and network security domain experts to integrate visualizations into their current workflows. Working closely with network security experts, we discovered a critical set of requirements that a visualization must meet to be considered for use by the these domain experts. We next present two separate examples of visualizations that address these requirements: a flexible web-based application that visualizes network traffic and security data through analyst-driven correlated charts and graphs, and a set of ensemble-based extensions to visualize network traffic and security alerts using existing and future ensemble visualization algorithms.

1 Introduction

The use of computer networks continues to grow, and with it the rise of sophisticated network attacks. Network security analytics has become an important area of computer science, and more recently data visualization. To maintain the security and stability of a network system, analysts continuously collect vast amount of data that capture important characteristics about their networks, then analyze the data to detect attacks, intrusions, and suspicious activity hidden in the traffic. Visualization has been proposed as an important component of this effort, since it allows for interactive exploration and analysis of large amounts of data, and can help analysts detect unexpected patterns more efficiently and effectively than traditional, text-based representations [3,13,19].

During this MURI project, we collaborated with network security experts from ARL to explore the use of visualization to improve cyber situation awareness in a network security environment. Numerous important findings resulted from this collaboration. In particular, we discovered that visualizations designed for network security analytics must meet a number of unique requirements, if they are to be adopted by network security analysts. Based on these requirements, visualizations that are simple and efficient in their representation of network data can provide powerful tools to support exploration and discovery in an effective and time critical manner. We discuss these requirements in detail,

P. Liu et al. (Eds.): Cyber Sitation Awareness, LNCS 10030, pp. 47–65, 2017.
DOI: 10.1007/978-3-319-61152-5_3

then summarize two approaches we developed to visualize network security data: analyst-drive web-based charts and graphs, and ensemble analysis techniques.

2 Visualization for Cyber Situation Awareness

The visualization community has focused recent attention on the areas of cyber security and cyber situation awareness. Early visual analysis of cyber security data often relied on text-based approaches that present data in text tables or lists. Unfortunately, these approaches do not scale well, and they cannot fully represent important patterns and relationships in complex network or security data. Follow-on work applied more sophisticated visualization approaches like node-link graphs, parallel coordinates, and treemaps to highlight different security properties, patterns in network traffic, and hierarchical data relationships. Because the amount of data generated can be overwhelming, many tools adopt a well-known information visualization approach: overview, zoom and filter, and details on demand. This approach starts by presenting an overview of the data. This allows an analyst to filter and zoom to focus on a subset of the data, then request additional details about the subset as needed. Current security visualization systems often consist of multiple visualizations, each designed to investigate different aspects of a system's security state from different perspectives and at different levels of detail.

2.1 Security Visualization Surveys

Visualization for cyber environments has matured to a point where survey papers on the area are available. These papers provide useful overviews, and also propose ways to organize or categorize techniques along different dimensions.

Shiravi et al.'s survey of visualization techniques for network security provides a useful overview of current visualization systems, and proposes a number of broad categories for data sources and visualization techniques [13]. One axis subdivides techniques by data source: network traces, security events, user and asset context (e.g., vulnerability scans or identity management), network activity, network events, and logs. A second axis considers use cases: host/server monitoring, internal/external monitoring, port activity, attack patterns, and routing behavior. Numerous techniques are described as examples of different data sources and use cases. The authors specifically address the issue of situation awareness in their future work, noting that many visualization systems try to prioritize important situations and project critical events as ways to summarize the massive amounts of data generated within a network. They distinguish between situation awareness, which they define as "a state of knowledge", and situation assessment, defined as "the process of attaining situation awareness." Converting raw data into visual forms is one method of situation assessment, meant to present information to an analyst to enhance their situation awareness.

Dang and Dang also surveyed security visualization techniques, focusing on web-based environments [3]. Dang chose to classify systems based on where they run: client-side, server-side, or web application. Client-side systems are normally simple, focusing on defending web users from attacks like phishing. Server-side visualizations are designed for system administrators or cyber security analysts with an assumed level of technical knowledge. These visualizations are usually larger and more complex, focusing on multivariate displays that present multiple properties of a network to the analyst. Most network security visualization tools fall into the server-side category. A final class of system is security for web applications. This is a complicated problem, since it can involve web developers, administrators, security analysts, and end users. Dang also subdivided server-side visualizations by their main goal: network management, monitoring, analysis, and intrusion detection; by visualization algorithm: pixel, chart, graph, and 3D; and by data source: network packet, NetFlows, and application-generated data. Various techniques exist at the intersection of each category.

New security and cyber situation visualization systems are constantly being proposed. These range from simple visualization approaches like charts and maps [4], node-link graphs [10], and timelines [9,11], to more complex representations like parallel coordinates [1,16], treemaps [7,8], and hierarchical visualizations [2].

Although security visualization systems aim to support more flexible user interactivity and correlation of various data sources, many of them still force an analyst to choose from a fairly limited set of static representations. For example, Phan et al. use charts, but with fixed attributes on the x and y-axes [9]. General purpose commercial visualization systems like Tableau, ArcSight, SpotFire, or SAS VA [6,12,14,15] offer a more flexible collection of visualizations, but they do not include visualization and human perception guidelines, so representing data effectively requires visualization expertise on the part of the analyst. Finally, many systems lack a scalable data management architecture, so the entire dataset must be loaded into memory prior to analysis and visualizing, increasing data transfer cost and limiting dataset size.

3 Visualization Design Philosophy

Our design philosophy is based on discussions with cyber security analysts at various research institutions and government agencies. The analysts overwhelming agreed that, intuitively, visualizations should be very useful. In practice, however, they had rarely realized significant improvements by integrating visualizations into their workflow. A common comment was: "Researchers come to us and say, Here's a visualization tool, let's fit your problem to this tool. But what we really need is a tool built to fit our problem." This is not unique to the security domain, but it suggests that security analysts may be more sensitive to deviations from their existing analysis strategies, and therefore less receptive to general-purpose visualization tools and techniques.

This is not to say, however, that visualization researchers should simply provide what the security analysts ask for. The analysts have high-level suggestions

about how they want to visualize their data, but they do not have the visualization experience or expertise to design and evaluate specific solutions to meet their needs. To address these, we initiated a collaboration with colleagues at ARL to build visualizations that: (1) meet the needs of the analysts, but also (2) harness the knowledge and best practices that exist in the visualization community.

Again, this approach is not unique, but it offers an opportunity to study its strengths and weaknesses in the context of a cyber security domain. In particular, we were curious to see which general techniques (if any) we could start with, and how extensively these techniques would need to be modified before they would be useful for an analyst. Seen this way, our approach does not focus explicitly on network security data, but rather on network security analysts. By supporting the analysts' situation awareness needs, we are implicitly addressing the goal of effectively visualizing their data.

From our discussions, we defined a set of requirements for a successful visualization tool. Interestingly, these do not inform explicit design decisions. For example, they do not define which data attributes we should visualize and how those attributes should be represented. Instead, they implicitly constrain a visualization's design through a high-level set of suggestions about what a real analyst is (and is not) likely to use. We summarized these comments into six general categories:

1. **Mental Models.** A visualization must "fit" the mental models the analysts use to investigate problems. Analysts are unlikely to change how they attack a problem in order to use a visualization tool.
2. **Working Environment.** The visualization must integrate into the analyst's current working environment. For example, many analysts use a web browser to view data stored in formats defined by their network monitoring tools.
3. **Configurability.** Static, pre-defined presentations of the data are typically not useful. Analysts need to look at the data from different perspectives that are driven by the problems they are currently investigating.
4. **Accessibility.** The visualizations should be familiar to an analyst. Complex representations with a steep learning curve are unlikely to be used, except in very specific situations where a significant cost-benefit advantage can be found.
5. **Scalability.** The visualizations must support query and retrieval from multiple data sources, each of which may contain very large numbers of records.
6. **Integration.** Analysts will not replace their current problem-solving strategies with new visualization tools. Instead, the visualizations must augment these strategies with useful support.

4 Web-Based Alert Visualization

The configurability, accessibility, scalability, and integration requirements of our design demand flexible user interaction that combines and visualizes multiple large data sources. The working environment requirement further dictates that this happen within the analyst's current workflow. To achieve this, we initially

designed and implemented a visualization system that combines MySQL, PHP, HTML5, and JavaScript to generate web-based network security visualizations through combinations of analyst-configurable charts focused on analyzing suspicious network activity.

4.1 Web Visualization

Based on our integration requirement, we built our visualizations as a web application using HTML5's canvas element. This works well, since it requires no external plug-ins and runs in any modern web browser.

We visualize network data using 2D charts. Basic charts are one of the most well known and widely used visualization techniques [17,18]. This supports our accessibility requirement, because: (1) charts are common in other security visualization systems the analysts have seen, and (2) charts are an effective visualization method for presenting the values, trends, patterns, and relationships the analysts want to explore.

Numerous JavaScript-based libraries exist to visualize data as 2D charts, for example, HighCharts, Google Charts, Flot Charts, and RGraph. Unfortunately, these libraries are designed for general information visualization, so they do not support analysis at multiple levels of abstraction or correlation between multiple charts. To address this, we extended RGraph [5] to generate security visualizations with flexible user interaction, data retrieval via MySQL, and the ability to correlate between multiple charts.

RGraph cannot automatically choose chart types based on the data to visualize. This capability might be useful, since analysts do not want to manually select the chart type if this decision is obvious. On the other hand, the analysts do not want to be restricted to specific, pre-defined visualizations. To support these conflicting needs, we classify our charts based on the different use cases in Fig. 1: (1) pie and bar charts for proportion and frequency comparison over a single attribute, (2) bar charts for value comparison over a secondary attribute, (3) scatterplots for correlation between two attributes, and (4) Gantt charts for range value comparisons. A visualization is created based on analysis goals, and not on the specific data being visualized. The analyst is free to change this initial selection, and more importantly, to interactively manipulate which data attributes are mapped to the primary and secondary dimensions.

Once a request is finalized, the system: (1) generates SQL queries to extract the target data from one or more data sources, (2) initializes chart properties like background grids and glyph size, color, and type, and (3) visualizes the data.

4.2 Charts

In a general information visualization tool, the viewer is usually asked to define exactly the visualization they want. We automatically choose an initial chart type based on: (1) existing knowledge on the strengths, limitations, and uses of different types of charts, and (2) the data the analyst chooses to visualize. For example, if the analyst asks to see the distribution of a single data attribute,

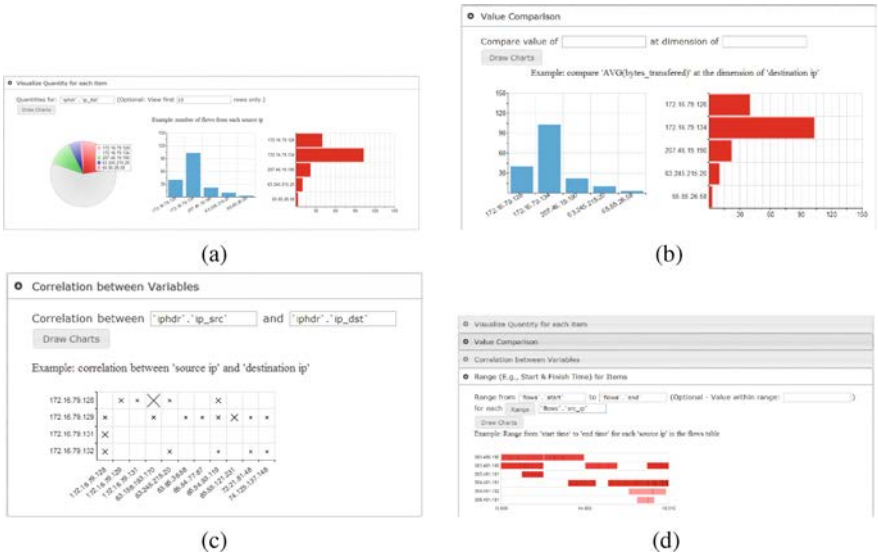


Fig. 1. Charts classified by use case: (a) pie and bar chart, analysis of proportion; (b) bar chart, value comparison along one dimension; (c) scatterplot, correlation analysis; (d) Gantt chart, range in at least one dimension

the system recommends a pie chart or bar chart. If the analyst asks to see the relationship across two data attributes, the system recommends a scatterplot or a Gantt chart.

The axes of the charts are initialized based on properties of the data attributes, for example, a categorical attribute on a bar chart's x-axis and an aggregated count on the y-axis. If the attributes were event timestamp and destination IP, time would be assigned to the x-axis and destination IP to the y-axis of a scatterplot (Fig. 2a). Visualizing start and end times for flows across destination IP produces a Gantt chart with time on the x-axis, destination IP on the y-axis, and rectangular range glyphs representing different flows (Fig. 2b). Or, if two attributes like source and destination IP are selected, the attributes are mapped to a scatterplot's x and y-axes, with data points shown for flows between pairs of values (Fig. 2c, d).

Data elements sharing the same x and y values are grouped together and displayed as a count. For example, in a scatterplot of traffic between source and destination IPs, the size of each tick mark indicates the number of connections between two addresses (Fig. 2c, d). In a Gantt chart, the opacity of each range bar indicates the number of flows that occurred over the time range for a particular destination IP (Fig. 2b).

More importantly, the analyst is free to change any of these initial choices. The system will interpret their modifications similar to the processing we perform for automatically chosen attributes. This allows the analyst to automatically start with the most appropriate chart type (pie, bar, scatterplot, or Gantt) based

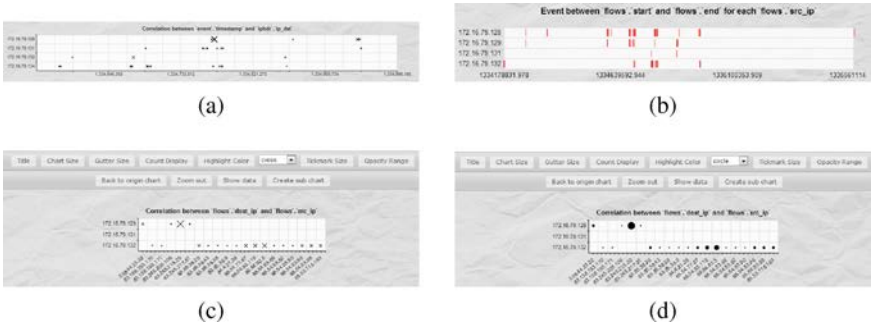


Fig. 2. Scatterplot and Gantt chart: (a) a scatterplot of connection counts over time by destination IP; (b) a Gantt chart of time ranges for flows by source IP; (c) a scatterplot with frequency count mapped to the size of an × glyph; (d) a scatterplot with count mapped to circle size

on their analysis task, the properties of the attributes they assign to a chart's axes, and on any secondary information they ask to visualize at each data point.

4.3 Correlation over Multiple Views

Analysts normally conduct a sequence of investigations, pursuing new findings as they are uncovered by correlating multiple data sources and exploring the data at multiple levels of detail. This necessitates visualizations with multiple views and flexible user interaction. We correlate multiple data sources by generating correlated SQL queries and extending the RGraph library to support dependencies between different charts (Fig. 3).

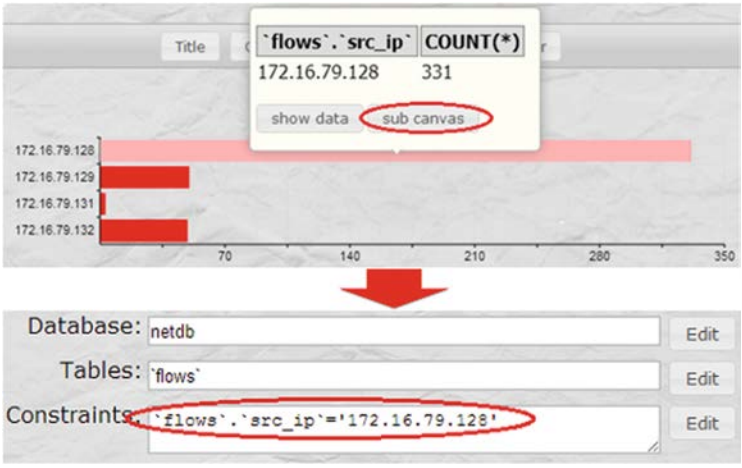


Fig. 3. New constraints to create a correlated chart

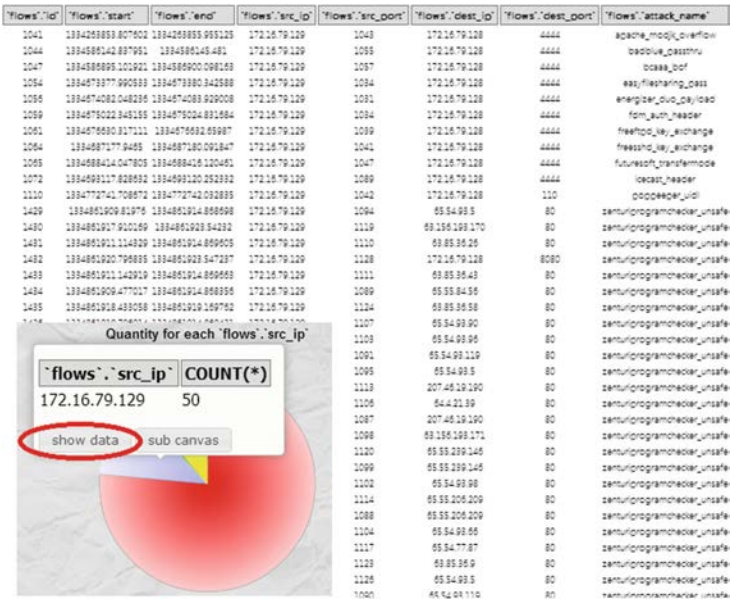


Fig. 4. Pie chart with a raw data spreadsheet

As an analyst examines a chart, they will form new hypothesis about the cause or effect of activity in the network. Correlated charts allow the analyst to immediately generate new visualizations from the current view to explore these hypotheses. In this way, the system allows an analyst to conduct a series of analysis steps, each one building on previous findings, with new visualizations being generated on demand to support the current investigations (Fig. 4).

5 Example Analysis Session

To demonstrate our web-based visualization system, we obtained trap data being used by network security colleagues at NCSU to act as input for their automated intrusion detection algorithms. This provided us with a real-world dataset, and also offers the possibility of comparing results from an automated system to a human analyst's performance, both with and without visualization support. One of our NCSU colleagues served as the analyst in this example scenario. Visualization starts at an abstract level: the distribution of alerts at different destination IP addresses; then follows the analyst's explorations of different hypotheses as he highlights and zooms into subregions of interest, creates correlated charts to drill down and analyze data at a more detailed level: visual analysis for alerts to a specific destination IP; and imports additional supporting data into the visualization: port and source IP. By including a new database flow table, the analysis of a subset of interest is extended to a larger set of data sources: analysis of flows related to interesting alerts. The visualization system supports the analyst by

generating different types of charts on demand, based on the analyst's current interest and needs. The analyst can view the data both visually and in raw text form to examine qualitative and quantitative aspects of the current region of interest. The data sources used in this example include:

- **event.** Signature id and timestamp for each alert.
- **flows.** Network flow information, including source and destination IP, port, and start and end time.
- **iphdr.** Source and destination IP and other information related to packet headers.
- **tcphdr.** TCP related information such as source and destination port.

The analyst begins by selecting the database and the tables to use for the first visualization, as well as the constraints needed to correlate the tables and filter the rows to explore. Based on these tables and constraints, the analyst can determine the types of analysis he wants to pursue and the data attributes to visualize. The analyst initially visualizes the number of alerts for each destination IP, selecting *ip_dst* from table *iphdr* as the "aggregate for" attribute. A SQL query is automatically generated to extract data for the chart.

Choosing "Draw Charts" displays the aggregated results as pie and bar charts (Fig. 5). This supports visual analysis of the data from different perspectives. Pie charts highlight the relative number of alerts for different destination IPs, while bar charts facilitate a comparison of the absolute number of alerts by destination IP. The charts are linked: highlighting at a bar in the bar chart will highlight the corresponding section in the pie chart, and vice-versa.

By default, aggregation results are sorted by the number of alerts for different destination IPs in reverse order, allowing the analysts to focus on the first few rows with the largest number of alerts. This is based on the assumption that analysts are more interested in addresses where a significant amount of traffic or number of alerts occurs. The analysts can reverse the sort order to search for low-occurrence events.

The pie and bar charts indicate that the majority of the alerts (910) occur at destination IP 172.16.79.134. Choosing "Show Data" displays all 910 rows as a spreadsheet in a new window (Fig. 6a). To further analyze alerts associated with this destination IP, the analyst chooses "Sub Canvas" to open a new window with the initial query information (the database, tables, and constraints) predefined. The constraint $iphdr.ip\_dst = 172.16.79.134$ is added to restrict the query for further analysis over this target destination IP. The analyst can continue to add new constraints or tables to the query as he requests visualizations to continue his analysis.

Next, the analyst chooses to visualize alerts from different source IPs attached to the target destination IP. He uses destination port to analyze the correlation between source and destination through the use of a scatterplot. Figure 6b shows there is only one source IP with alerts related to the target destination IP, and that most alerts are sent to port 21, shown as the large $\times$ symbol in the scatterplot.

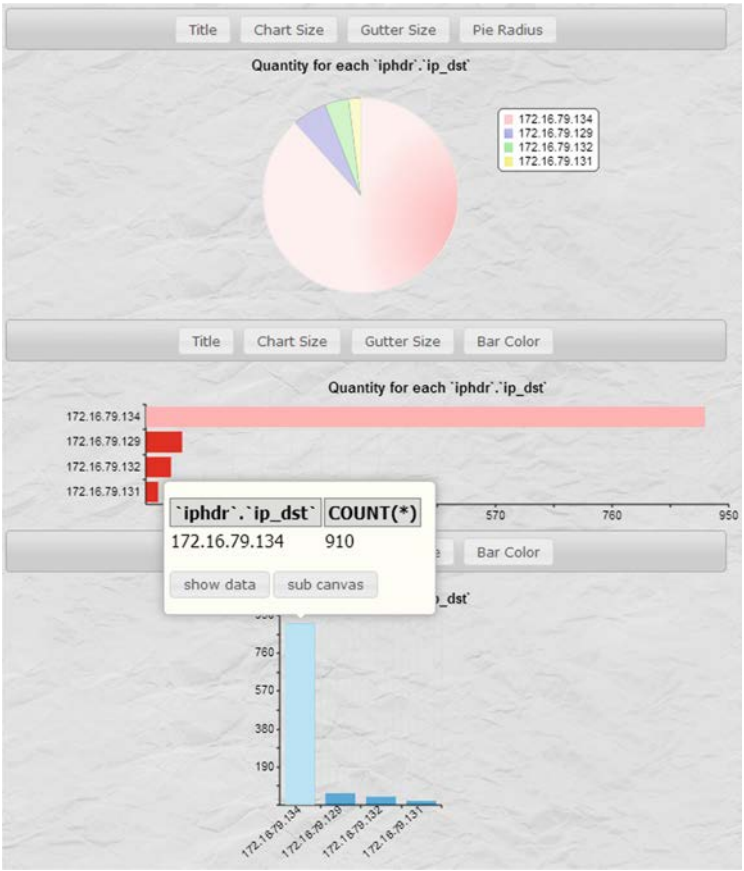
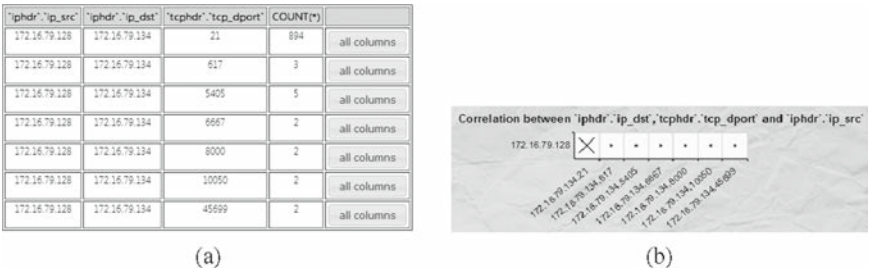


Fig. 5. Aggregated results visualized as a pie chart and horizontal and vertical bar charts



(a) (b)

Fig. 6. Detail analysis: (a) raw data spreadsheet; (b) scatterplot of relationships between source IP and destination port correlated to destination IP 172.16.79.134

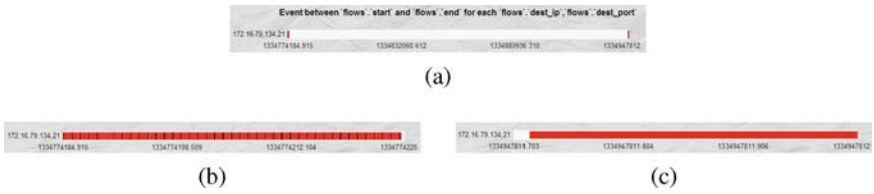


Fig. 7. Gantt chart with alerts for network flows at the destination IP and port of interest; (a) two flows; (b) zoom on the left flow, showing numerous alerts; (c) zoom on the right flow, showing one alert (Color figure online)

The analyst looks more closely at traffic related to the target destination IP on port 21 by visualizing netflows and their associated alerts in a Gantt chart. Collections of overlapping flows are drawn in red with endpoints at the flow set's start and end times. Alerts appear as black vertical bars overlaid on top of the flows at the time the alert was detected. Figure 7a shows most of the flows are distributed over two time ranges. By zooming in on each flow separately (Fig. 7b, c), the analyst realizes that the vast majority of the alerts occur in the left flow (Fig. 7b). The alerts in this flow are considered suspicious, and are flagged for more detailed investigation.

This example demonstrates how our system allows an analyst to follow a sequence of steps based on their own strategies and preferences to investigate alerts. Although it is possible to filter the raw data to "highlight" this result directly, our colleagues suggested it might be difficult to recognize that the majority of the alerts occur for a specific destination IP, source IP, port, and time range from a 910-line alert spreadsheet. The visualizations allow an analyst to follow this pattern step-by-step, uncovering more detailed information as they progress.

6 Ensemble Alert Visualization

A relatively new area of research that has grown rapidly in recent years is ensemble analysis. In numerous disciplines, scientists collect data produced by a series of runs of a simulation or an experiment, each with slightly different initial conditions or parameterizations. This collection of related datasets—an *ensemble*—has been widely used to simulate complex systems, explore unknowns in initial conditions, investigate parameter sensitivity, mitigate uncertainty, and compare structural characteristics of models. Each individual dataset forms a *member* of the ensemble. Ensemble visualization is an active area of research in visualization, specifically designed for exploring and comparing both within and between members of massive ensemble datasets.

Although network security data and ensemble data look quite different at first glance, they are similar in terms of their analytic challenges and goals. Both are large and time-dependent, necessitating analysis in the time dimension and approaches to support scalability. Ensemble visualization focuses on comparison

and aggregation of related ensemble members, while security visualization focus on exploration of correlations between network traffic. Although visualization of scientific ensemble and network data at the most detailed level will likely be different, high-level ensemble overviews and frameworks could allow an analyst to quickly identify and drill down on subsets of interesting or suspicious network traffic. If we view network security data as a type of ensemble, there is an important opportunity to apply ongoing and future ensemble visualization research to improve network security analytics.

6.1 Network Ensemble Data

To harness ensemble visualization for network security analytics, we must address the challenge of terminology mapping: defining a network ensemble from security datasets, and dividing the data into a series of related network traffic, analogous to members in an ensemble. We focus on two common types of network security data: alerts and flows. An alerts dataset contains source and destination IP, port, time, protocol, message, and classification. A flows dataset contain source and destination IP, port, flags, start time, end time, protocol, number of packets, and number of bytes. An alert belongs to a flow if it is detected within the flow's time range and has the same source and destination IP. Based on this, we define two types of ensembles: an alert ensemble and a flow ensemble.

To maintain the requirement of configurability, we propose a general framework to construct network data ensembles. This allows analysts to configure details of the ensemble, if they choose to do so. A network ensemble consists of related network traffic (analogous to ensemble members), each representing a temporal sequence of alerts or flows that fall within equal-sized time windows. Analyzing relationships (similarities) between this network traffic is one important goal of cyber situation awareness.

Specifically, we offer two ways to define members in a network ensemble. The first method combines data properties to identify members. For example, we could define source IP and source port to specify members. Now, alerts or flows sent from each source IP+port form a network ensemble member. The second method divides the ensemble time window into a number of smaller time windows, each identifying a member in the ensemble. For example, if the dataset consists of network traffic for a 24-hour period, hourly traffic can represent an ensemble member. Finally, analysts can control the data values stored within each member. For example, we can analyze inter-member relationships in an alert ensemble by comparing changes in the of numbers of alerts over the time dimension.

6.2 Alert Ensembles

To construct an alert ensemble, an analyst defines the SQL tables that contain the alert data of interest, a time dimension column, correlations between the tables if more than one table is selected, and any additional constraints needed to form the ensemble. The system will automatically identify the time window that

covers the ensemble's data. The analyst can choose one or more table columns to define ensemble members, or evenly subdivide the ensemble's time window into a number of smaller time periods, each representing a member.

To analyze relationships between members in an alert ensemble, we subdivide the time range of every member into a user-specified number of time-steps and aggregate alerts within each time-step. For example, each destination IP+port combination could form a member in the ensemble. Given a time window divided into 30 time-steps, the number of alerts is calculated at every time-step. Comparing alert members is performed by comparing changes in the number of alerts over time.

6.3 Flow Ensembles

Flow ensembles are defined in a manner similar to alert ensembles. An analyst chooses the SQL tables that contain the data, defines how to correlate alerts and flow traffic, identifies time dimension columns for the alerts and flows, and provides any additional constraints to extract the data to analyze. A flow has start and end times and contains a number of alerts, so comparing pairs of flows is more complicated than comparing numbers of alerts. Instead of aggregating data across time-steps, we view every member in a flow ensemble as a sequence of individual flows. Calculating member similarity aligns flows between pairs of members prior to comparison using dynamic time warping (DTW).

A member m_i in a flow ensemble is a sequence of l_i flows $m_i = (f_{i,1}, f_{i,2}, \ldots, f_{i,l_i})$. Each flow has start time t_s^i and end time t_e^i, and contains zero or more alerts. This makes the comparison of flow traffic more complicated than aggregated alerts. Manhattan distance is not applicable for flow sequence comparison because the sequences may have different lengths and may not be aligned in time. To calculate the DTW distance between flow members, we must first calculate the dissimilarity between pairs of flows. Let $dis(f_u, f_v)$ be the dissimilarity between flows f_u and f_v. We propose a simple flow comparison that calculates $dis(f_u, f_v)$ based on three metrics:

1. **Duration.** Given f_i's duration $dur_i = t_e^i - t_s^i$, the duration dissimilarity between f_u and f_v is

$$dis_{\text{dur}}^{u,v} = \frac{|\, dur_u - dur_v \,|}{\max(dur_u, dur_v)} \tag{1}$$

2. **Density.** Given f_i containing n_i alerts, the density of alerts in f_i is $den_i = n_i/dur_i$. The density dissimilarity between f_u and f_v is

$$dis_{\text{den}}^{u,v} = \frac{|den_u - den_v|}{\max(den_u, den_v)} \tag{2}$$

3. **Distribution.** Given start and end times t_s^i and t_e^i for a flow f_i that receives n_i alerts at times $t_1^i, t_2^i, \ldots, t_{n_i}^i$, the intervals between alerts are $I_i = \{t_s^i - t_1^i, t_2^i - t_1^i, \ldots, t_e^i - t_{n_i}^i\}$. We use $\sigma_i = \sum_{i=1}^{n_i}(I_i - I_\mu)^2/n_i$, the variance of the

intervals between alerts, to compute distribution dissimilarity between f_u and f_v as

$$dis^{u,v}_{\text{dist}} = \frac{|\sigma_u - \sigma_u|}{\max(\sigma_u, \sigma_v)} \tag{3}$$

The individual dissimilarities are averaged and normalized to generate an overall dissimilarity between f_u and f_v. We allow an analyst to tune the weights w_{dur}, w_{den}, and w_{dist} during averaging:

$$\begin{aligned} dis(f_u, f_v) &= w_{\text{dur}} dis^{u,v}_{dur} + w_{\text{dens}} dis^{u,v}_{\text{dens}} + w_{\text{dist}} dis^{u,v}_{\text{dist}} \\ 0 &\leq w_{\text{dur}}, w_{\text{dens}}, w_{\text{dist}} \leq 1 \\ w_{\text{dur}} &+ w_{\text{dens}} + w_{\text{dist}} = 1 \end{aligned} \tag{4}$$

6.4 Ensemble Member Visualization

Ensemble visualization often contains detailed representations of one or more ensemble members. These detailed member visualizations may be very different depending on the types of members contained in a ensemble. To maintain consistency with our existing web-based network security visualization system, we use 2D charts to visualize network traffic. Specifically, we generate line charts to visualize members in an alert ensemble and Gantt charts for members in a flow ensemble.

Alert Member Visualization. An alert ensemble member is a sequence of aggregated alert counts calculated at every time-step. Figure 8a visualizes a 100-member alert ensemble for a single source IP of interest constructed by an analyst. Destination IP+port is used to define individual members. Time unfolds on the x-axis, and the number of alerts at each time step is plotted on the y-axis. Color represents the destination IP of each alert's parent flow.

Figure 8a highlights a number of similar patterns: flows that start with numerous alerts but end with few alerts; flows with two spikes in alerts near their center; and an outlier flow (in orange) with numerous alerts over its center.

The analyst chooses to assign the 100 alerts members to $k = 20$ clusters. Figure 8b visualizes the 20 clusters, averaging the number of alerts within each cluster at every time-step. This visualization provides a general understanding of changes in the numbers of alerts over time. The highlighted purple line at the bottom of the graph represents a large cluster that contains 65 members. A follow-on visualization of this cluster's members (Fig. 8c) confirms that changes in the number of alerts over time among the 65 members are similar.

Flow Member Visualization. We chose to visualize flow traffic and associated alerts using Gantt charts. The x-axis represents time, and the y-axis represents member (e.g., a combined IP+port). Flows are visualized as colored bars with endpoints at the flow's start and end times. Alerts appear as black vertical tick marks at the time the alert was detected.

Analysts choose clusters of flow members to visualize, based on results from flow similarity clustering algorithms. For example, Fig. 9 visualizes two clusters,

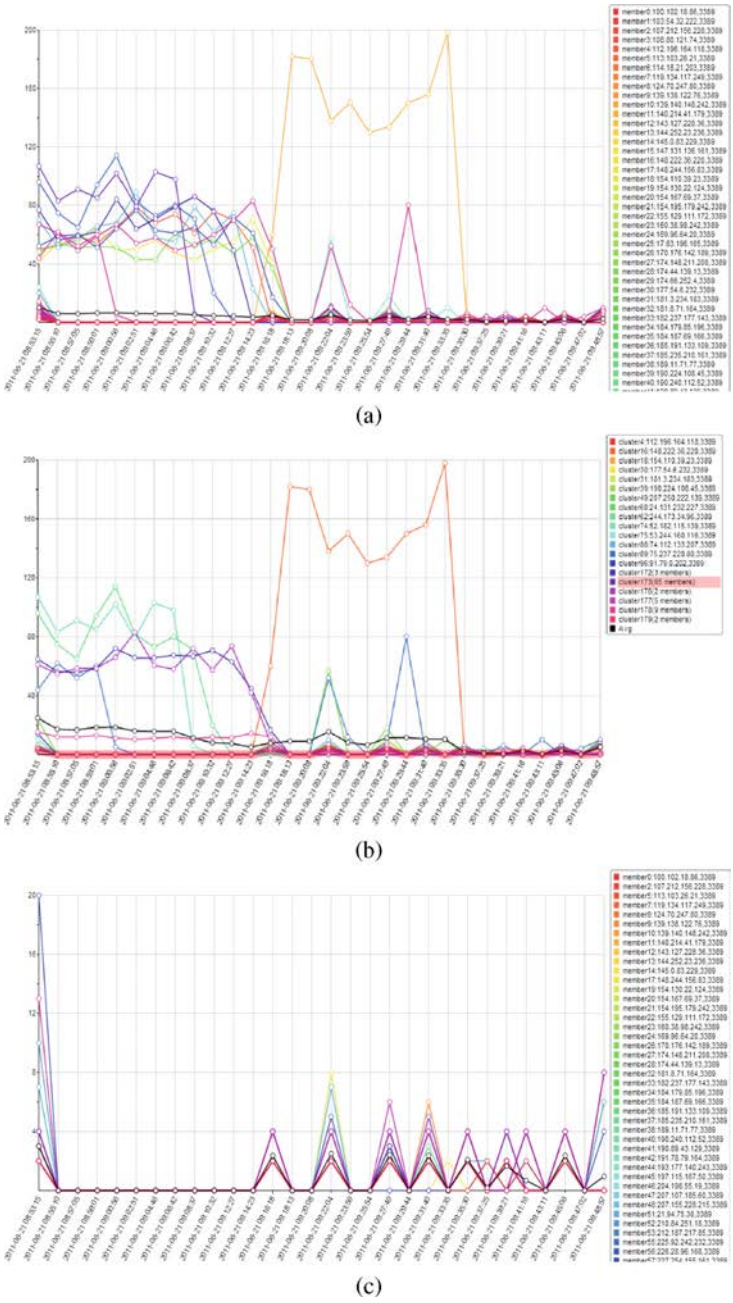


Fig. 8. Alert member visualization: (a) a 100-member ensemble; (b) visualization of members assigned to $k = 20$ clusters; (c) visualizing each alert member in a cluster containing 65 members with similar alert patterns, but from different flows (Color figure online)

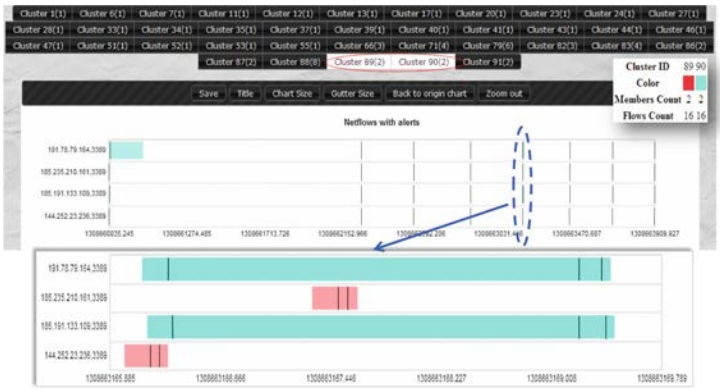


Fig. 9. Visualization of four flow ensemble members from two different clusters shown in blue and red (Color figure online)

each with two flow members. Red and blue identify the two clusters. Zooming into a group of flows that occur at a similar time, but belong to different members, produces a detailed visualization that shows the flows in each cluster are similar based on time duration, alert density, and alert distribution. Notice that without dynamic time warping, the flows in the red cluster would not be considered similar, since they start and end at different times.

7 Practical Application

We extended our web-based system to support visualizing network security ensembles. As before, data management occurs on a remote server running MySQL and PHP. The visualizations are based on interactive 2D charts using HTML5 and Javascript. We used anonymized network traffic from one floor of our Computer Science building to test our system on real-world alert and flow patterns.

As a practical example of our system, consider the alert ensemble discussed above that retrieves alerts data for source IP `64.120.250.242` and uses destination IP+port to define alerts sent to a common destination and port as a member in the ensemble. Since the analyst is not interested in traffic with a small number of alerts, he sorts ensemble members by number of alerts and analyzes only the top 100 members. The system generates SQL queries to extract the relevant data and calculate dissimilarities between pairs of members. It generates a dissimilarity matrix visualization (similar to Fig. 10b), clusters the flows with an agglomerative clustering algorithm, then presents a line chart characterizing overall dissimilarity for different numbers of clusters k. The analyst uses the line chart to study inter-member relationships, combining the 100 members into $k = 20$ clusters, then visualizes the largest cluster, containing 65 members with similar changes in the number of alerts over time (Fig. 8c). Ensemble

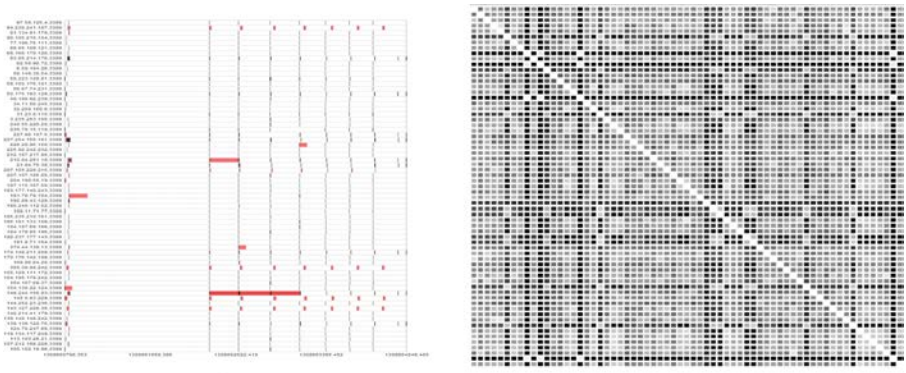


Fig. 10. Flow ensemble: (a) visualizing ensemble members in a Gantt chart; (b) dissimilarity matrix for the 65 members

visualization makes it more efficient to detect similar changes in alerts patterns, something that is not as obvious from a general visualization of alert traffic.

To gain a more in-depth understanding of the alert traffic covered by the 65-member cluster, the analyst takes flow traffic into consideration by requesting a flow ensemble visualization. The alert visualization system exports SQL queries to extract the alerts in the 65-alert member. The analyst uses these constraints to retrieve associated flows to construct a flow ensemble. Members in the ensemble are defined by destination IP+port. Equal weights are assigned to the three metrics w_{dur}, w_{dens}, and w_{dist} during flow comparison (Eq. (4)). In this way, every member in the flow ensemble is correlated with the member in the alert ensemble that is sent to the same destination.

Figure 10a visualizes the 65 members in the flow ensemble as 65 individual Gantt charts with time on the x-axis and destination IP+port on the y-axis. At this level of detail, flow traffic for most members looks similar, making it difficult to visually distinguish the flows without zooming in. The dissimilarity matrix (Fig. 10b) indicates that network traffic sent to different destinations are fairly strongly differentiated when we include the flow data (i.e., there are numerous dark cells in the dissimilarity matrix).

Based on the dissimilarity matrix, the analyst decides that members of the ensemble are similar if their dissimilarities are smaller than 0.21. This combines the 65 members into $k = 38$ clusters. As expected, flows in members from the same cluster are similar. For example, in Fig. 11, the flows in a cluster with six members have very similar density and distributions of alerts, and relatively similar durations (as shown in the top-right overview visualization).

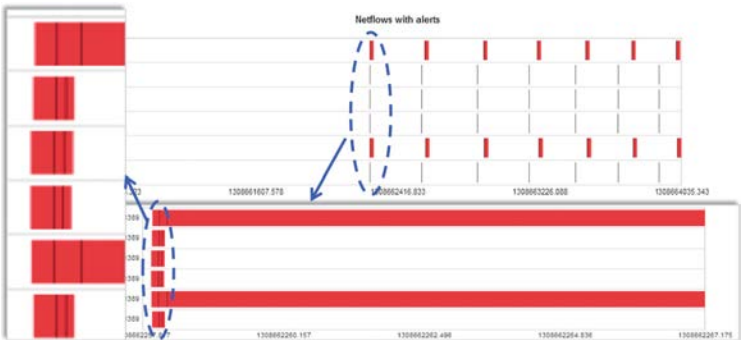


Fig. 11. A cluster of six members in the flows ensemble. Corresponding flows has very similar patterns

8 Conclusions

Data visualization converts raw data into images that allow a viewer to "see" data values and the relationships they form. The images allow viewers to use their visual perception to identify features in the data, to manage ambiguity, and to apply domain knowledge in ways that would be difficult to do algorithmically. Visualization can be formalized as a mapping: data passed through a data–feature mapping generates a visual representation—a visualization—that displays individual data values and the patterns they form.

Numerous situation awareness tools use visualization techniques like charts, maps, and flow diagrams to present information to an analyst. The challenge is to determine how best to integrate visualization techniques into a cyber situation awareness domain. Many tools adopt a well-known information visualization approach: overview, zoom and filter, and details on demand. Techniques utilized recently for the security and situation awareness domains include: charts and maps, node-link graphs, timelines, parallel coordinates, treemaps and hierarchical visualization.

We identified an initial set of requirements for a successful visualization tool. These do not define which data attributes we should visualize or how those attributes should be represented. Instead, they implicitly constrain a visualization's design through a high-level set of suggestions about what a real analyst is, and is not, likely to use. A visualization must "fit" the mental models the analysts use to investigate problems. It must integrate into the analyst's current working environment. Pre-defined presentations of the data are typically not useful. Visualizations should be familiar to an analyst. The system must support query and retrieval from multiple data sources; the visualizations must integrate into existing strategies with useful support. We demonstrate a prototype system for analyzing network alerts based on these guidelines, using both basic charts and graphs, and ensemble approaches to compare and combine alert and flow traffic based on inter-member relationships. In both cases, data retrieval and

data-feature mapping are driven by the analyst, to ensure they visualize their current data of interest in ways that highlight exactly the data correlations they want to analyze.

References

1. Bradshaw, J.M., Carvalho, M., Bunch, L., Eskridge, T., Feltovich, P.J., Johnson, M., Kidwell, D.: Sol: an agent-based framework for cyber situation awareness. Künstliche Intellienz **26**(2), 127–140 (2012)
2. Cockburn, A., Karlson, A., Bederson, B.B.: A review of overview+detail zooming and focus+context interfaces. ACM Comput. Surv. **41**(1) (2008). Article 2
3. Dang, T.K., Dang, T.T.: A survey on security visualization techniques for web information systems. Int. J. Web Inf. Syst. **9**(1), 6–31 (2013)
4. Goodall, J., Sowul, M.: VIAssist: visual analytics for cyber defense. In: IEEE Conference on Technologies for Homeland Security (HST 2009), Boston, pp. 143–150 (2009)
5. Heyes, R.: RGraph: HTML5 and JavaScript charts (2017). https://www.rgraph.net
6. HP ArcSight ESM. http://www8.hp.com/us/en/software-solutions/arcsight-esm-enterprise-security-management/
7. Kan, Z., Hu, C., Wang, Z., Wang, G., Huang, X.: NetVis: a network security management visualization tool based on Treemap. In: 2nd International Conference on Advanced Computer Control (ICACC 2010), Shenyang, pp. 18–21 (2010)
8. Mansmann, F., Fisher, F., Keim, D.A., North, S.C.: Visual support for analyzing network traffic and intrusion detection events using TreeMap and graph representations. In: Symposium on Computer-Human Interaction for Management of Information (CHIMIT 2009), Baltimore, article 3 (2009)
9. McPherson, J., Ma, K., Krystosk, P., Bartoletti, T., Christensen, M.: PortVis: a tool for port-based detection of security events. In: Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004), Washington, DC, pp. 73–81 (2004)
10. Minarik, P., Dymacek, T.: NetFlow data visualization based on graphs. In: Goodall, J.R., Conti, G., Ma, K.-L. (eds.) VizSec 2008. LNCS, vol. 5210, pp. 144–151. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85933-8_14
11. Phan, D., Gerth, J., Lee, M., Paepcke, A., Winograd, T.: Visual analysis of network flow data with timelines and event plots. In: Goodall, J.R., Conti, G., Ma, K.-L. (eds.) VizSEC 2007, pp. 85–99. Springer, Heidelberg (2008)
12. SAS Visual Analytics. http://www.sas.com/en_us/software/business-intelligence/visual-analytics.html
13. Shiravi, H., Shiravi, A., Ghorbani, A.: A survey of visualization systems for network security. IEEE Trans. Vis. Comput. Graph. **18**(8), 1313–1329 (2012)
14. Tableau Software. http://www.tableau.com
15. Tibco Spotfire. http://spotfire.tibco.com
16. Tricaud, S., Nance, K., Saadé, P.: Visualizing network activity using parallel coordinates. In: 44th Hawaii International Conference on System Sciences (HICSS 2011), Poipu, pp. 1–8 (2011)
17. Tufte, E.R.: The Visual Display of Quantitative Information. Graphics Press, Cheshire (1983)
18. Tufte, E.R.: Envisioning Information. Graphics Press, Cheshire (1990)
19. Zhang, Y., Xiao, Y., Chen, M., Zhang, J., Deng, H.: A survey of security visualization for computer network logs. Secur. Commun. Netw. **5**(4), 404–421 (2012)