Interactive Analysis of NetFlows for Misuse Detection in Large IP Networks

Florian Mansmann, Fabian Fischer, Daniel A. Keim, Stephan Pietzko, Marcel Waldvogel

first.lastname@uni-konstanz.de

Abstract:
While more and more applications require higher network bandwidth, there is also a tendency that large portions of this bandwidth are misused for dubious purposes, such as unauthorized VoIP, file sharing, or criminal botnet activity. Automatic intrusion detection methods can detect a large portion of such misuse, but novel patterns can only be detected by humans. Moreover, interpretation of large amounts of alerts imposes new challenges on the analysts. The goal of this paper is to present the visual analysis system NFlowVis to interactively detect unwanted usage of the network infrastructure either by pivoting NetFlows using IDS alerts or by specifying usage patterns, such as sets of suspicious port numbers. Thereby, our work focuses on providing a scalable approach to store and retrieve large quantities of NetFlows by means of a database management system.

1 Introduction

Network administrators have a tendency to automate as many tasks as possible in order to keep pace with the ever increasing bandwidth requirements of modern network applications. Larger networks, in particular, have become unmanageable without smart intrusion detection systems. However, when it comes to analyzing attacks or detecting novel attacks, these systems only support the analyst in a very limited way.
While administrating our university network with several thousand hosts, we have realized that most of these systems generate a tremendous amount of alerts when being used in an open network setting with only few firewall restrictions as demanded by our users. In addition to that, it is hard to reason about the generated alerts since many of these systems are designed as blackboxes to guard the technological advance of the security provider.
In this paper we propose a novel system called *NFlowVis*, which is designed to visually present service usage and threats in the local IP network. Thereby, alerts from intrusion detection systems or defined application ports are used to identify potential attackers and visualize all their traffic to hosts within the administrated network in the next step. Note that the IP addresses in the figures are anonymized to protect the privacy of our users.
The rest of this paper is structured as follows: Section 2 presents related work, Section 3 discusses processing and querying challenges and solutions for large NetFlow data sets and Section 4 details our proposed visualization system. Afterwards, Section 5 shows how the tool is being used in practice. The last section concludes our work.

2 Related Work

Visualization for computer security is a relatively young research field. While substantial research has been conducted in the field in the last few years, for brevity this section will focus on visual network traffic monitoring and discuss the roots of the used visualization concepts. Please refer to the following books to gain deeper insight into the statistical, visualization, and application aspects of intrusion detection [Mar01, Con07, Mar08].

In the Open Source community, there are two popular tools: *NfSen* [NfS07] and *Stager* [Osl06]. Both tools comprise web frontends to display aggregated information about previously captured NetFlows. In the backend, database management systems enable efficient access to detailed information and efficient generation of aggregated reports. For visual analysis, both systems use line charts for displaying temporal overviews of network system load. While Stager only stores highly aggregated data, NfSen reverts back to the original flow data for detailed analysis.

Since network monitoring is particularly important for the health of the commercial network infrastructure, there exist a multitude of commercial systems. In contrast to the previously discussed tool, commercial systems such as *IBM Aurora*[1], *NetQoS Reporter Analyzer*[2], *Caligare Flow Inspector*[3], and *Arbor Peakflow*[4] often include methods for intrusion detection in which generated alerts can be examined through interactive reports. However, the used statistical charts and diagrams only scale to a limited number of alerts or highly aggregated information.

Visualization approaches in network monitoring aim at supporting the system administrator in the exploration of network traffic by means of interactive visual displays. *NVisionIP* [LBS$^+$05], for example, enables visual pattern recognition and drill-down functionalities to inspect suspicious machines. *TNV* [GLRK05] is a network traffic visualization tool focusing on temporal aspects by means of a time versus internal host matrix, which details traffic flows for each host and links the external communication partners on the side. The home-centric network view of *VISUAL* [BFN04] is probably closest to our proposed visualization since a matrix showing all internal hosts in the center is linked to external communication partners using straight connecting lines.

In contrast to this work, we made two major conceptual changes: a) Instead of using a matrix view for the internal hosts, we employ a *TreeMap* [Shn92] visualization, which hierarchically maps the monitored network infrastructure to prefixes of various granularity. Unlike in our previous work [MKN$^+$07], high-load entities are thereby enlarged. b) Rather than using straight lines to link the communication partners, we employ *Hierarchical Edge Bundles* [Hol06] to visually group related flows, and thereby avoid visual clutter. While we visualized flows using Hierarchical Edge Bundles with both start and end point within a TreeMap visualization in an earlier work [MFKN07], the work presented in this paper explicitly focuses on a home-centric network view, which represents the local IP prefixes or addresses in a TreeMap and places the external hosts at its border.

Abstract graph representations normally seek a way to effectively use the available screen

[1]`http://www.zurich.ibm.com/aurora`
[2]`http://netqos.com/solutions/reporteranalyzer`
[3]`http://www.caligare.com/netflow`
[4]`http://www.arbornetworks.com`

space. Thereby, linked nodes are rendered close to each other to avoid visual clutter caused by crossing edges. Cheswick et al., for example, mapped a graph of about 88 000 networks as nodes having more than 100 000 connecting edges [CBB00], obtained by measuring the quality of network connections in the Internet from different vantage points.
The study in [TN$^+$00] goes one step further in the automated analysis by applying clustering methods on graph structures, in order to reveal similar attack structures.
There exist hybrid approaches that partly take geographic information into account while calculating the graph layout on the screen. One such approach is the visualization interface of the *Skitter* application that uses polar coordinates to visualize the Internet infrastructure [Cla01]. Each AS node's polar coordinate is determined by the geographical longitude of its headquarter and by the hierarchical connectivity information.
The implementation of the node link diagrams in our tool can rather be seen as a feature than as a novel research contribution since we only apply efficient graph layout and interaction frameworks.

3 Large-Scale Processing of NetFlows

A big challenge in the analysis of network related records is the great amount of data to be handled in real-time, especially when trying to avoid packet loss. To use the proposed analysis system we are required to store all available NetFlow information in a relational database management system. This means to cope with three main problems.
Firstly, we need to receive NetFlow streams in real-time. We need to accept these streams immediately and on link speed, which are later processed in 5 minutes intervals. The underlying protocol utilizing the NetFlow streams is the stateless UDP protocol. The system receiving the streams has to be as fast as possible to accept all records even in peak times. To accomplish these requirements we set up a NetFlow collector server using a flow capture daemon (flow-tools [RFL00]) and storing the incoming NetFlow streams directly to the RAM of the server system to prevent a possible I/O bottleneck at this early stage. Having a few GB of RAM storage we can use this memory as a buffer cache to prevent packet loss and to provide more time for the next more time-consuming preprocessing and analysis steps.
Secondly, we have to preprocess the incoming NetFlow streams and to transform them to a format which can be imported to our database in a fast and efficient way. We were required to use batch import functionalities, so a very convenient data format are plain comma separated text files. In this step we also integrated an anonymization filter to use the system for scientific purposes and to prevent scientists to access unanonymized data. To overcome privacy concerns we integrated the anonymization process (especially in the testing phase) at this early stage, to ensure all data available in the database is completely anonymized. In production use it is easy for the network security officer to disable the anonymization process, which will also lead in a higher overall performance of the system. Note that this is not an inherent function of the system for operational use.
The third step of the processing workflow is to actually import the available data to the database system and to store it in a scheme which makes the data analyzable by NFlowVis.

This step is not done on the fly like the previous steps. The server system will automatically import the data to the database server in regular intervals based on the required analysis timeframe. Because of the limitations of our hardware we had to restrict the import interval to one day for now, but we are adapting our approach to an import interval to few hours using a faster database server with more memory.
The main bottleneck of the processing system is the index creation during the batch import of the data. This can be improved by importing the records to empty tables without pre-existing indices. To have a reasonable performance and response time querying the database, indices are still required. By creating the indices after the batch import of the new records is finished, we were able to drastically increase the import time. To support this importing scheme directly through the underlying database layout, we introduced a chronical timeframe table hierarchy, in which each table presents one import interval (e.g. one day). Through the heritage structure or through joining, it is basically still possible to access full years, months or several days. Additionally the system automatically creates several pre-aggregated tables during the import process to further improve the querying performance and support specific queries used in NFlowVis to visualize the data.

4 Visual Analysis of NetFlows

Keeping the general workflow of a network analyst in mind, we developed *NFlowVis* to interpret the relevance of network security alerts. The system supports this full workflow through its five analysis views with a general network *overview*, an integrated *intrusion detection view*, the *flow visualization* of attackers' connections, a detailed *host view*, and the full *NetFlow records* of the specified communications as the most detailed view. In the graphical user interface these views are represented through several tabs to emphasize the drill-down and filtering process. Figure 1 describes the design of NFlowVis, showing project selection (A), key data of the selected project (B), the Quick Lookup interface for directly querying specific IP addresses (C), fast access to external tools (D) and the data exploration views ordered according to the levels of details (E). Within the *overview* tab, the system provides several user-defined plots (F). With the help of these graphs the analyst is able to get a rough overview of the actual network situation and utilization detailing the aggregated traffic and port usage within the whole network. To visualize these time series we use line charts and grouped line-wise pixel arrangements. The use of both visualizations combines the advantages of the well known line charts and pixel visualization, which provides identification of every single minute and enables recognition of recurring patterns. The overview also provides an interactive port activity map to identify the most active ports.
The *intrusion detection view* in Figure 2 is key to system since it links our NetFlow exploration system with an intrusion detection system by showing the imported alerts. Note that the only requirement for this table is that the first row contains the IP address of an attacker, whereas the number of additional columns are only relevant for the human analyst. For further investigation of a number of attacking hosts, it is possible to select the attackers and to visualize their traffic with hosts in our network to explore their influence.

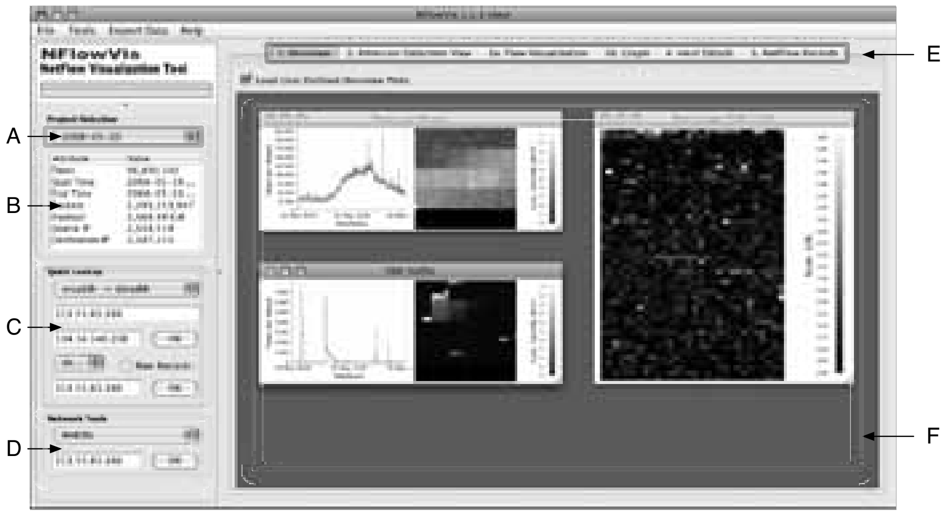

Figure 1: User interface of the NFlowVis system showing the annotated main view. In this start display the user can choose a dataset (A), see some overall statistics of this data set (B), directly access detailed data for a particular host or host combination (C), use external tools to query background information on a host (D), access a few user-defined plots (F) showing aggregated flows per minute (top left), traffic on a particular port (bottom right) or the activity on the most used ports (right) or start a detailed analysis (E).

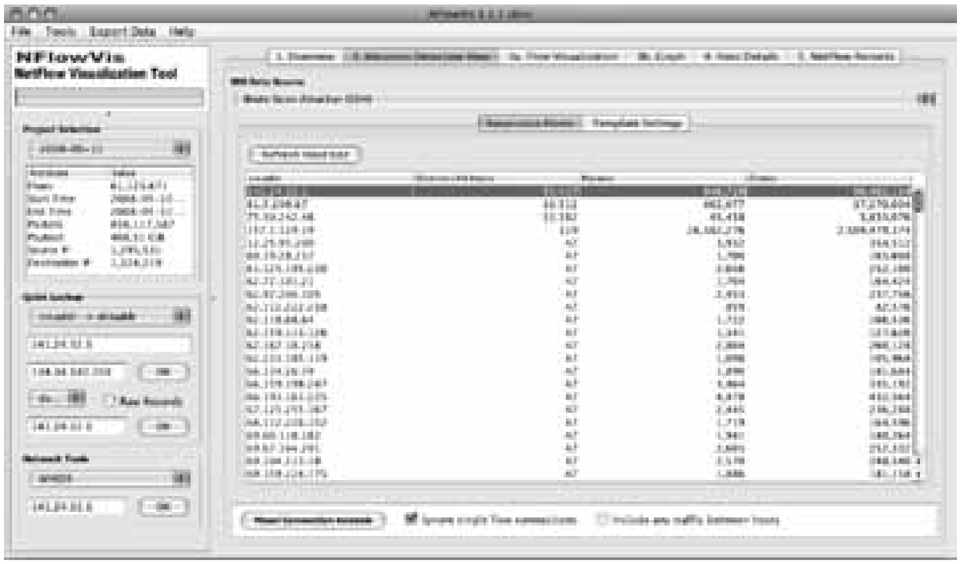

Figure 2: Alerts originating from an external IDS or warning lists in the Intrusion Detection View

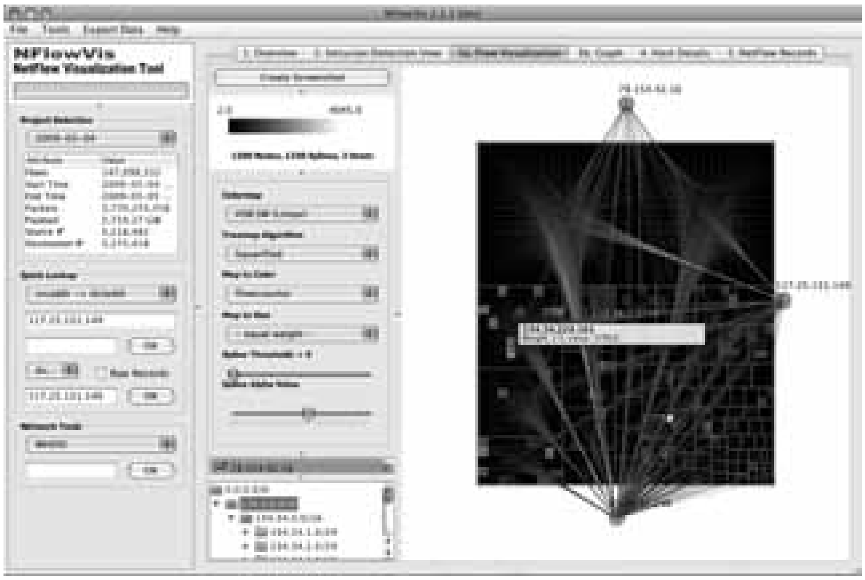

Figure 3: Flow Visualization of traffic patterns displaying the internal network structure as a TreeMap and the external hosts on the borders. The internal hosts are displayed as rectangles, which are contained within their upper level prefixes. Their size and color are configurable to the traffic payload, the number of flows or packets, or to a constant (e.g, equal-size) using the configuration on the left. The visualization shows three external hosts scanning for open SSH ports; the upper two prefixes contain a lot of scanned hosts, but the number of flows is always low (black color). In contrary to this, the prefixes in the lower part contain less scanned hosts but some hosts received a lot of flows. The yellow host, for example, received 1770 flows from the three attackers.

Besides the integration of external IDS alerts and warning lists, this view also provides a template editor to define database queries, which can directly access arbitrary tables. We included a variety of different predefined warning lists, such as grabbing all SSH traffic or other suspicious activities.
Within the *flow visualization* view shown in Figure 3, we map the monitored network to a TreeMap visualization in the center of the display and arrange the previously selected attackers at the borders. The TreeMap comprises all hosts related to the attacking hosts during the chosen timeframe, which can be defined in the project creation wizard. Flows between the attackers and the local hosts or prefixes are displayed through Splines, whose control points are the center points of the network prefixes of various levels and the attackers on the outside. The size of the TreeMap rectangles (weight) and their background color can be set to arbitrary attributes of the aggregated flow data, e.g., flow count, transferred packets, or bytes. Furthermore, splines representing traffic links smaller than a selected threshold can be discarded or made less visible by adjusting the sliders on the configuration panel on the left.
In the default configuration the Spline color correlates with the attacker's IP prefix, which better shows the behavior of attackers with similar prefixes supporting the analyst in gaining insight into the distribution of the attacking IP addresses.

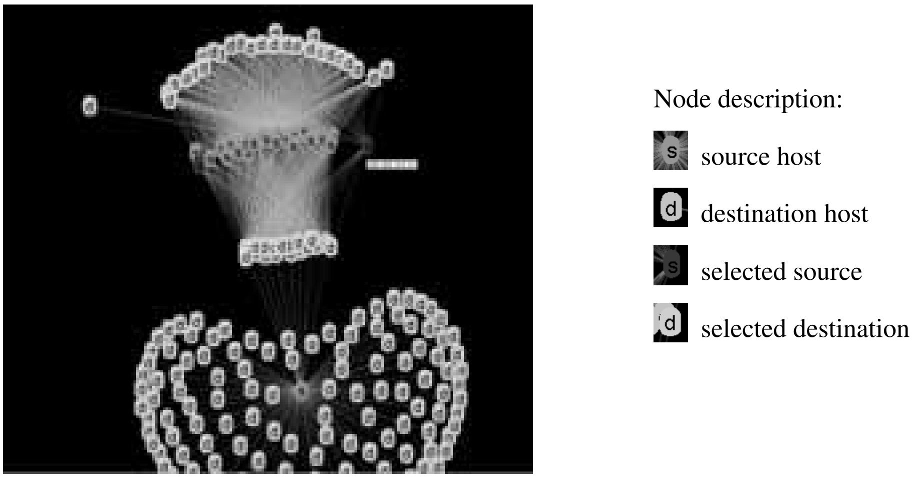

Figure 4: Graph view showing a network scan (below) and an attack from a botnet (above).

The position of the attackers is calculated based on a *k-Medoid* clustering algorithm [KR90], which identifies all attackers and clusters them based on similar destination hosts. Therefore, it is possible to arrange hosts with similar victims close to each other to minimize overlaps. Another positive effect is the meaningful grouping of collaborating attackers in the same cluster.
Figure 4 shows the graph view, which can be seen as an alternative to the previously presented flow visualization. By extracting the communicating hosts from the traffic specified in the IDS view, we generate a node link diagram using the GraphViz tool [EGK$^+$02] to efficiently calculate the layout and the Prefuse toolkit [HCL05] for displaying and interacting with the nodes. Note that the choice between using this graph layout or the previously introduced home-centric TreeMap visualization depends on the analysis task. While the graph view better presents the structure between the attackers and its victims, the home-centric visualization helps to identify properties of the attack that are influenced by the local network infrastructure. A pool of computers running an unpatched operation system, for example, could be easily identified in the home-centric network visualization due to the rectangle grouping, whereas extracting this information from the graph layout would involve interactively displaying one IP address after the other.
For further analysis of single hosts under attack, the analyst is able to select hosts in the two latter visualizations, which triggers the *host view* detailing histograms, a port activity map which visualizes the data volume on the used port numbers, and an aggregated overview of all attackers related to the chosen host (see Figure 5). Likewise, the original NetFlow records can be further analyzed by drilling-down and extracting the corresponding data in the *NetFlow records* view showing the timestamp, source/destination hosts and prots, the protocol as well as the number of data packets and octets aggregated on flow level.

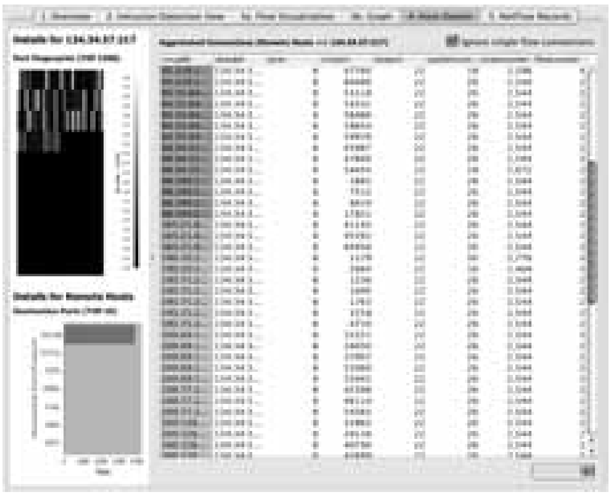

Figure 5: Host details view

5 Results and Findings

While applying the tool for monitoring our university network at the gateway, we found several interesting patterns. The first pattern that usually sticks out is caused by network scans, such as the scan for open remote desktop ports in Figure 3 or the scan for open VNC servers in Figure 5. These patterns were detected after specifying the respective ports in an SQL database query in the IDS view. While these scans can automatically be detected by relatively simple detection algorithms, the visualization can reveal further details of the structure of the attack and give additional indications whether the attack was successful or not. This is done by specifying that all traffic between the external attacker and the internal victim host is visualized. After having found a valid user and password combination, the attacker usually logs into the system and downloads a malware application to control the conquered host. While unsuccessful attacks often result in relatively little traffic, successful attacks might result in additional traffic on other ports. We show traffic properties for each internal host in its rectangle size and color to guide the analyst to these suspicious hosts, which have a higher probability of being hacked.
We were furthermore able to identify botnet attacks on open SSH ports as displayed in Figure 5. The used clustering algorithm thereby groups external hosts, which connect to similar sets of internal hosts, on the borders, thereby resulting in a more insightful visualization. Note that while this flow visualization focuses on an internal view of the network, which uses prefix information to group subnets, the graph view shown in Figure 4 might give additional hints to structures of attacks.

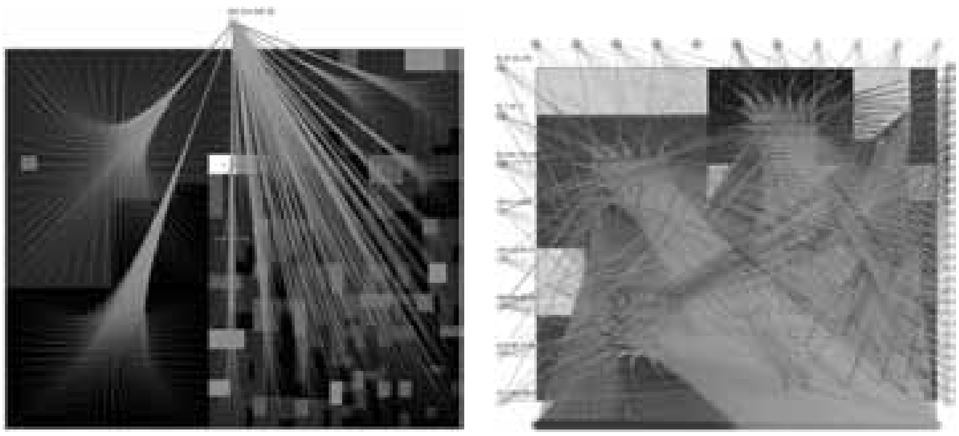

Figure 6: Application examples: Scan for Virtual Network Servers using the VNC protocol on December 8, 2008 (left) and SSH attack from a botnet on November 29, 2008 (right)

6 Conclusions

In this paper we presented the NFlowVis system for analyzing large amounts of NetFlows and intrusion detection alerts. In contrast to traditional IDS, we pursued a visual data analysis approach since this allows the experts to gain deeper insight into current threat situation and to discover novel attacks. In particular, we presented two complementary visualization approaches for the analysis of attacker and victim hosts. The first approach is comprised of a local network centric TreeMap view, which groups local network hosts according to their prefix information and allows the analyst to draw conclusions about the focus areas of attacks within the network. The second approach uses methods from graph drawing to visualize the link information between the attackers and their victims and can be especially helpful to distinguish between distributed scans and attacks.
For future work, we plan to create a database independent application, which allows administrators to analyze smaller tcpdump/NetFlow files without using a database server.
This work has been funded as part of the BW-FIT research cluster “Gigapixel displays” by the German federal state Baden-Württemberg.

Bibliography

[BFN04] R. Ball, G.A. Fink, and C. North. Home-centric visualization of network traffic for security administration. *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 55–64, 2004.

[CBB00] Bill Cheswick, H. Burch, and S. Branigan. Mapping and Visualizing the Internet. In *Proceedings of the USENIX Annual Techincal Conference*, 2000.

[Cla01] K.C. Claffy. CAIDA: Visualizing the Internet. *IEEE Internet Computing*, 05(1), 2001.

[Con07] Greg Conti. *Security Data Visualization - Graphical Techniques for Network Analysis*. No Starch Press, 2007.

[EGK+02] J. Ellson, E. Gansner, L. Koutsofios, S.C. North, and G. Woodhull. Graphviz-Open Source Graph Drawing Tools. *LECTURE NOTES IN COMPUTER SCIENCE*, pages 483–484, 2002.

[GLRK05] John R. Goodall, Wayne G. Lutters, Penny Rheingans, and Anita Komlodi. Preserving the Big Picture: Visual Network Traffic Analysis with TNV. In *VIZSEC '05: Proceedings of the IEEE Workshops on Visualization for Computer Security*, Washington, DC, USA, 2005. IEEE Computer Society.

[HCL05] J. Heer, S.K. Card, and J.A. Landay. prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430. ACM New York, NY, USA, 2005.

[Hol06] Danny Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Trans. Vis. Comput. Graph.*, 12(5):741–748, 2006.

[KR90] L. Kaufman and P.J. Rousseeuw. Finding groups in data. An introduction to cluster analysis. *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, New York: Wiley*, 1990.

[LBS+05] K. Lakkaraju, R. Bearavolu, A. Slagell, W. Yurcik, and S. North. Closing-the-Loop in NVisionIP: Integrating Discovery and Search in Security Visualizations. In *Visualization for Computer Security, IEEE Workshops on*, pages 9–9, 26 Oct. 2005.

[Mar01] David J. Marchette. *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[Mar08] Raffael Marty. *Applied Security Visualization*. Addison-Wesley Professional, 2008.

[MFKN07] F. Mansmann, F. Fischer, D. Keim, and S. North. Visualizing large-scale IP traffic flows. In *Proceedings of 12th International Workshop Vision, Modeling, and Visualization*, 2007.

[MKN+07] Florian Mansmann, Daniel A. Keim, Stephen C. North, Brian Rexroad, and Daniel Sheleheda. Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring, and Interpretation of Security Threats. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1105–1112, 2007.

[NfS07] NfSen - Netflow Sensor. A graphical web based front end for the nfdump netflow tools, 2007. `http://nfsen.sourceforge.net/`.

[Osl06] A. Oslebo. Stager A Web Based Application for Presenting Network Statistics. In *Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP*, pages 1–15, 2006.

[RFL00] Steve Romig, Mark Fullmer, and Ron Luman. The OSU Flow-tools Package and CISCO NetFlow Logs. In *LISA '00: Proceedings of the 14th USENIX conference on System administration*, pages 291–304, Berkeley, CA, USA, 2000. USENIX Association. `http://www.splintered.net/sw/flow-tools/`.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, 1992.

[TN+00] J. Toelle, O. Niggemann, et al. Supporting intrusion detection by graph clustering and graph drawing. In *Proceedings of Third International Workshop on Recent Advances in Intrusion Detection RAID*, 2000.