VisFlowCluster-IP: Connectivity-Based Visual Clustering of Network Hosts*

Xiaoxin Yin, William Yurcik, and Adam Slagell

National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
{xiaoxin,byurcik,slagell}@ncsa.uiuc.edu

Abstract. With the increasing number of hostile network attacks, anomaly detection for network security has become an urgent task. As there have not been highly effective solutions for automatic intrusion detection, especially for detecting newly emerging attacks, network traffic visualization has become a promising technique for assisting network administrators to monitor network traffic and detect abnormal behaviors.

In this paper we present *VisFlowCluster-IP*, a powerful tool for visualizing network traffic flows using network logs. It models the network as a graph by modeling hosts as graph nodes. It utilizes the force model to arrange graph nodes on a two-dimensional space, so that groups of related nodes can be visually clustered in a manner apparent to human eyes. We also propose an automated method for finding clusters of closely connected hosts in the visualization space. We present three real cases that validate the effectiveness of *VisFlowCluster-IP* in identifying abnormal behaviors.

1 Introduction

There has been tremendous growth of network applications and services in the last two decades. At the same time, the number of hostile attacks is increasing, and these attacks are hidden by the vast majority of legitimate traffic. There have been many studies on intrusion detection systems (IDS) which fall into the following two categories. The first category is *misuse detection systems* [6, 7, 14, 18], which use predefined signatures to detect intrusions. However, they are not effective for detecting new intrusions and viruses, and the ever growing signature database becomes problematic. The second category is *anomaly detection systems* [15, 17], which attempt to model normal behaviors and give alerts for any behavior that deviates from the model. However, the "normal behaviors" of different persons and tasks may not be similar, so it is difficult to accurately profile normal network traffic. Attempts to detect anomalies based on profiles of network traffic often suffer from unacceptable false positive rates.

As fully automated intrusion detection systems have not been able to provide a highly effective solution to protect network systems, humans are still in-the-loop of

* This research was supported in part by a grant from the Office of Naval Research (ONR) under the auspices of the National Center for Advanced Secure Systems Research (NCASSR) <http://www.ncassr.org>.

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

inspecting large numbers of alerts to identify real threats. In comparison with automated systems, the human mind is capable of rapid visual processing, especially for detecting abnormal or extraordinary visual patterns. Tools that visually depict network traffic patterns leverage this capability for anomaly detection. They can provide a user with the capability to drill down into the data to extract information about potential attacks. Almost all security events leave traces in network traffic. It is a highly challenging task to find these traces in high-volume traffic. Visualization tools can represent high-volume traffic as static graphs or dynamic animations, in order to help network administrators sift through gigabytes of daily network traffic to identify anomalous patterns.

In this paper we propose *VisFlowCluster-IP*, a tool that visualizes network connectivity using visual clustering techniques to assist network administrators with monitoring abnormal behaviors. There have been many studies on visualizing network traffic and connectivity [2, 4, 23, 25]. In these approaches, hosts in a network are represented as nodes and traffic as edges or flows in a graph. However, they fix hosts in certain locations according to their IP addresses, without considering relationships and interactions between different hosts.

In [9] and [13], approaches have been proposed to arrange graph nodes with a *force model* in order to better capture the relationships and patterns among the nodes. This technique has been widely used in graph drawing and visualization [19, 24]. *VisFlowCluster-IP* uses this model for arranging hosts in network visualization. It models each host in a network as a particle in a two-dimensional space, and defines the attraction (or repulsion) force between two hosts according to their relationship. Then it lets particles interact with each other until a reasonably stable arrangement is achieved. In the force model method, related hosts can be visually clustered and certain traffic patterns will thus become apparent to human eyes. *VisFlowCluster-IP* can also detect clusters of particles that are located close to each other in the space of visualization, which correspond to groups of hosts that are closely related to each other.

Another shortcoming of most existing approaches for visualizing network traffic [2, 4, 23, 25] is that most of them only use the traffic volume between two hosts to measure the relationship between them. Although this model can identify hosts that communicate with each other, it is not good at finding hosts that exhibit similar behaviors, because two hosts with similar behaviors may not have traffic between them. For example, two hosts in a group often both have high traffic volume to some servers in that group, but usually do not have much traffic between them. Another example is that two basketball fans often have high traffic to web sites of NBA, NCAA, and ESPN, but they seldom communicate with each other. *VisFlowCluster-IP* provides functionality to address this problem. It defines the relationship between two hosts based the external hosts they access, as well as traffic between them.

With the above techniques, *VisFlowCluster-IP* can visualize the network hosts and connections in a 2-D space, so that hosts that exhibit similar behaviors will be arranged close to each other and form visual clusters. This capability is not available in existing visualization systems for anomaly detection. We apply *VisFlowCluster-IP* to the traffic logs from our network. Some abnormal visual patterns are identified from the visualizations, with which we easily find the corresponding abnormal patterns in network traffic. These experiments show that *VisFlowCluster-IP* is able to convert certain

abnormal traffic patterns into abnormal visual patterns that are apparent to human eyes. Our experiments also show that *VisFlowCluster-IP* can automatically identify these abnormal visual patterns as visual clusters and report such clusters to users. However, *VisFlowCluster-IP* also reports some clusters that correspond to normal behaviors and are false alerts. The user still needs to make judgments for each cluster based on both the visualization and the network traffic data.

The remainder of this paper is organized as follows. Section 2 summarizes related work. We present background information in Section 3. Section 4 describes the visualization approach of *VisFlowCluster-IP* and the approach for identifying clusters of closely related hosts. Section 5 presents experimental results that validate the capability of *VisFlowCluster-IP* for security monitoring. We end with a summary and conclusions in Section 6.

2 Related Work

Currently, there are two main approaches to intrusion detection: *misuse detection* and *anomaly detection*. Misuse detection [6, 7, 14, 18] finds intrusions by directly matching known attack patterns. The major drawback of this rule-based approach is that it is only effective at finding known attacks with predefined signatures. In anomaly detection [15, 17], the normal behavior of a system is stored as a profile. Any statistically significant deviations from this profile are reported as possible attacks. However, these alarms may also be legitimate but atypical system activity, which often leads to unacceptable false positive alarm rates.

Because of the limited density of information conveyed through text, visualization techniques to present computer network data sets to humans have been a growing area of research. It is well-known that seeing enables humans to glean knowledge and deeper insight from data. Early work on visualizing networks has been motivated by network management and analysis of bandwidth characteristics [3, 5, 10]. A wide spectra of knowledge about visualization of cyberspace is provided in [8], such as topology of connectivity and geography of data flows.

In [20], the authors present a visualization of network routing information that can be used to detect inter-domain routing attacks. In [21, 22], they explore further in this field and propose different ways for visualizing routing data in order to detect intrusions. An approach for comprehensively visualizing computer network security is presented in [11], where the authors visualize the overall behavioral characteristics of users for intrusion detection. They represent the substantial characteristics of user behavior with a visual attribute mapping capable of identifying intrusions that otherwise would be obscured. However, the host representation employed in [11] is not scalable in terms of the number of hosts and traffic volume. In [1], parallel axes are used to visualize network logs about traffic involving a single machine (web server). In [16], the authors present NVisionIP, which shows network traffic in a host-centric view, providing both an overview and detailed views with its on-demand zoom and filtering capabilities.

Linkages among different hosts and events in a computer network contain important information for traffic analysis and intrusion detection. Approaches for link analysis are proposed in [2, 4, 25]. [2] focuses on visualizing linkages in a network, and [4] focuses

on detecting attacks based on fingerprints. In [25], the authors present VisFlowConnect-IP, a tool for visualizing connectivity between internal and external hosts using animation. It uses animation to visualize network traffic in real-time and focuses on short-term link analysis. In contrast, *VisFlowCluster-IP* is a complementary, static off-line visualization tool, which uses visual clustering techniques to highlight clusters of hosts closely related to each other. It focuses on long-term link relationships among the hosts.

These tools provide effective ways to visualize hosts and traffic in a network. However, they all use approaches where all nodes are fixed on certain locations or lines, and the locations of nodes are independent from their relationships with other nodes. This prevents the visualization tool from arranging hosts in a nice layout that captures the relationships among them. This problem is discussed in [23], which first performs hierarchical clustering on hosts and then visualizes the clusters. However, the visualization approach in [23] simply shows the hierarchical structures of clusters, instead of arranging them for better understanding.

3 Preliminaries

3.1 NetFlow Source Data

The source data used by *VisFlowCluster-IP* is NetFlow data. We consider two formats of NetFlow data, one from Cisco routers and the other from a freely available software named *Argus* (<http://www.qosient.com/argus/>). A distinct flow is defined as either a unidirectional TCP connection (where a sequence of packets take the same path) or individual UDP packets with the same IP and port in a short period of time.

The input to *VisFlowCluster-IP* is a stream of NetFlow records either from a log file or a streaming socket. A NetFlow agent is used to retrieve the NetFlow records and feed them into *VisFlowCluster-IP*. Each record contains the following information: (1) IP addresses and ports of the source and destination, (2) number of bytes and packets, (3) start and end times, and (4) protocol type.

3.2 Problem Settings

The goal of *VisFlowCluster-IP* is to visualize hosts and network traffic in a way that groups of related hosts can be easily identified by humans. Each host in the network is modeled as a node in a graph, and an edge is added between two nodes if there is certain relationship between them. In *VisFlowCluster-IP*, two nodes are related if they communicate with each other, or they both access the same server(s) outside the network. We translate relationships between nodes into attraction/repulsion forces, which are used to arrange nodes in our visualization.

4 Visualizing Network Flows

4.1 Constructing Graph

Given a network log containing a list of NetFlow records, *VisFlowCluster-IP* constructs a graph with the following procedure.

VisFlowCluster-IP first creates a graph node for each host in the network to be visualized (e.g., each host in NCSA in our visualization), which is called a *particle node*. It also creates a graph node for each host outside the target network, which is called a *hub node*. A hub node is only for visualization and does not participate in the arrangements of graph nodes.

Suppose a graph contains particle nodes $p_1, \ldots, p_n$, and hub nodes $h_1, \ldots, h_m$. The volume of traffic between two nodes x and y (in either direction) is represented by $T(x, y)$. Two nodes are considered to be highly related if there is high traffic volume between them, or they both have high traffic volume to many hub nodes. Thus we define the edge weight between two particle nodes p_1 and p_2 as a combined function of the traffic volume between them and the traffic volume between them and their common neighbor nodes.

$$w(p_1, p_2) = \alpha \cdot \log T(p_1, p_2) + (1 - \alpha) \cdot \log \left(\sum_{j=1}^{m} T(p_1, h_j) \cdot T(p_2, h_j) \right) \quad (1)$$

The user may adjust the weights of internal traffic and external traffic by choosing an appropriate value for α.

4.2 Arranging Nodes

VisFlowCluster-IP uses the *force model* [9, 13] to arrange particle nodes in a graph, in order to reorganize the graph so that groups of nodes related to each other can be clustered on the screen and made apparent to humans. In such a model, two particles attract each other if they have an edge between them, and repulse each other if not. All particle nodes keep interacting with each other in many iterations, until a reasonably stable arrangement is achieved. The following procedure is used for arranging nodes.

Initially, *VisFlowCluster-IP* assigns a default location for each particle node. Because some parts of an IP address may indicate information about the host, the location of each host is determined by its IP. Suppose the IP of a host h is $a.b.c.d$. Suppose H_w and H_h are two hash functions whose input domains are all integers and value ranges are the width and height of the visualization panel respectively. In *VisFlowCluster-IP*, the x coordinate of h is $H_w(256 \times ((256 \times a) + b) + c)$, and the y coordinate is $H_h(d)$. Since hosts belonging to the same group or cluster often belong to the same class C, they will be located on one or a few vertical lines. Some hosts that are certain types of servers often have the same values on last byte of IP (e.g., gateway or DNS server), and thus they will be located on certain horizontal lines.

After the initial assignments, *VisFlowCluster-IP* lets the particle nodes interact with each other in order to achieve an arrangement in which groups of related hosts are visually clustered. As in previous approaches of force models [9, 13, 24], *VisFlowCluster-IP* adjusts the location of each particle according to the attraction and repulsion forces it receives in each iteration, and it performs many iterations until achieving a reasonably stable arrangement.

Suppose the location of a particle node p_i is $\boldsymbol{p_i}$. The force between two particle nodes p_1 and p_2 is defined as follows.

- **Repulsion:** There is a repulsion force between any two particle nodes, which is a decreasing function of the distance between them. The magnitude of the repulsion force is

$$Repul(p_1, p_2) = \frac{C_{repul}}{\sqrt{||\boldsymbol{p_1} - \boldsymbol{p_2}||}} \tag{2}$$

  in which C_{rep} is a constant. The direction of the repulsion force received by p_1 from p_2 is same as the vector $\boldsymbol{p_1} - \boldsymbol{p_2}$.
- **Attraction:** If there is an edge between p_1 and p_2, there is an attraction force between them. The magnitude of the attraction force is proportional to the weight of the edge between p_1 and p_2.

$$Attr(p_1, p_2) = C_{attr} \times w(p_1, p_2) \tag{3}$$

  in which C_{attr} is a constant. The direction of the attraction force received by p_1 from p_2 is the same as the vector $\boldsymbol{p_2} - \boldsymbol{p_1}$.

In each iteration, each particle node p_i receives a force F_i that is the sum of all the repulsion and attraction forces it receives. The movement of p_i in this iteration is δF_i, where δ is the step length. The stability of an arrangement is defined as the average movement of each particle node in an iteration. In most cases, a fairly stable arrangement can be achieved in ten to fifty iterations, and each iteration usually takes several seconds on a graph of a few thousand nodes.

4.3 Visualization

VisFlowCluster-IP assigns a color[1] to each host, which is based on its IP address. The color is determined in a way that hosts with similar IP addresses have similar colors. As the horizontal location of a host is determined by the first three bytes of its IP, and the vertical location by the last byte, we also assign a unique color for each host. The hue of the color is determined by the first three bytes of its IP, and the brightness is determined by the last byte. In this way all hosts from same class C will have similar colors (same hue) but with different brightnesses.

We use an example to show how we determine colors of hosts. The network in NCSA is a class B, which contains 65536 distinct IP addresses. Suppose the IP of a host is *a.b.c.d.* Its hue is set to $360 \cdot c/256$, which ranges from 0 to 360. Its brightness is set to $d/512 + 0.5$, which ranges from 0.5 to 1. Its saturation is set to 1, so that each host has a vivid color (white, gray, and black will not appear here).

In the visualization, each particle node is represented by a circle filled with its color. Each hub node is also represented by a circle, but with the color of light gray. We define the weight of a node n, $w(n)$, as the logarithm of the total traffic volume involving that node. The diameter of a circle is proportional to the weight of the corresponding node. There is a line between two nodes if there is traffic between the two corresponding hosts. The location of each hub node h_i is a weighted average of locations of all particle nodes connected to it, based on the edge weights between h_i and the particle nodes.

[1] We note that some visualizations may be hard to understand if this paper is printed in black-and-white. They are clear in electronic versions of this paper.

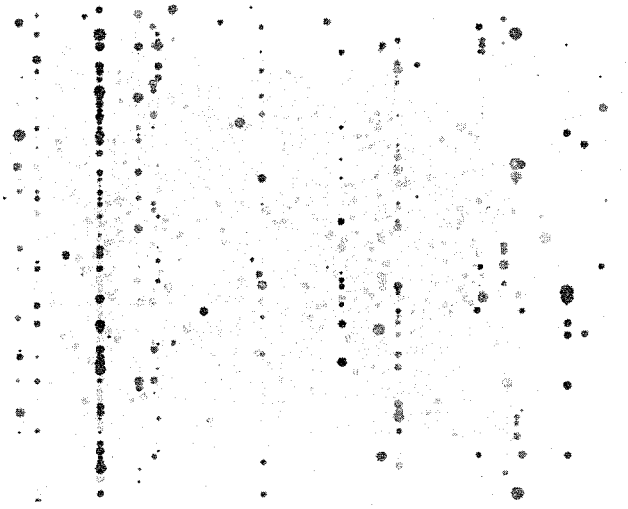

Fig. 1. Initial Arrangement of VisFlowCluster-IP.

The initial assignment of an example NetFlow log file is shown in Fig. 1. This file is an Argus log file that only contains traffic between internal hosts and external hosts. *VisFlowCluster-IP* repeatedly rearranges the particle nodes in Fig. 1. A stable arrangement is achieved after 32 iterations, which takes 11.52 seconds in total. The final arrangement is shown in Fig. 2, in which two clusters of hosts are automatically detected. The approach for detecting clusters will be introduced in Section 4.4, and the sematic meanings of the visualization results will be discussed in Section 5.

4.4 Finding Clusters

During the iterative arrangements of nodes, the sets of particle nodes that are closely related to each other will be grouped together because of the attractions among them, and the particle nodes that are not related to each other will seldom be located close to each other because of the repulsions. Therefore, the dense regions on the plane of visualization correspond to groups of closely related nodes.

VisFlowCluster-IP uses DBSCAN [12], a popular density-based clustering approach to find clusters of related nodes. DBSCAN considers every point as a node in a graph. An edge exists between two nodes if their corresponding points are very close to each other. A cluster is a connected component in such a graph. Because we are only interested in dense regions on the plane of visualization, we slightly modify the algorithm to ignore sparse regions. We first divide the space into many small grids. The density of a grid is defined as the total weights of particle nodes in this grid. We only consider grids whose densities are above a certain threshold, which is β times the average density of all grids ($\beta > 1$). Then we use each dense grid as a point and use DBSCAN to identify clusters of dense grids. Because we are only interested in clusters of significant sizes, we ignore a cluster if the total weight of its particle nodes is less than γ times the

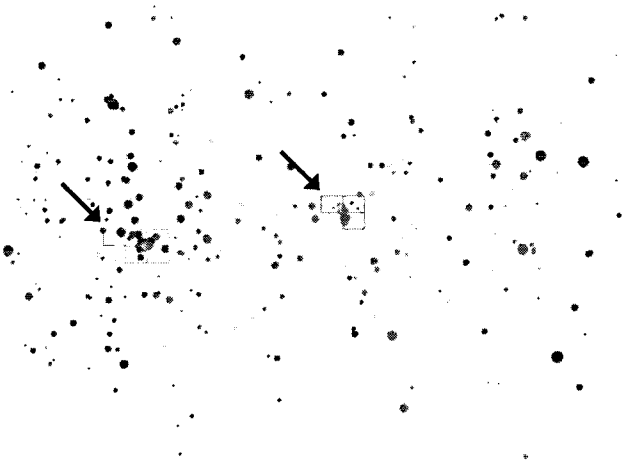

Fig. 2. Stable Arrangement of VisFlowCluster-IP.

total weight of all particle nodes ($0 < \gamma < 1$). Fig. 2 shows two clusters identified by *VisFlowCluster-IP*, which are highlighted by black arrows.

5 Experiments

Experiments are performed to show the effectiveness of *VisFlowCluster-IP* in assisting network administrators to detect abnormal behaviors in a network. We ran *VisFlowCluster-IP* on an Intel PC with 2.4GHz Pentium 4 CPU, 1GB memory, and running Windows XP Professional. In order to show its capability of visual clustering based on communications between internal and external hosts, we set $\alpha = 0$ in *VisFlowCluster-IP* to ignore internal traffic. After trying different values for parameters, we use $\beta = 5$ and $\gamma = 0.02$ when detecting clusters. Thus a region will be detected as a cluster if its density is at least five times the average density and the total weight of its particle nodes is at least 2% of the total weight of all particle nodes.

5.1 Blaster Worms

The Blaster worm spreads quickly between hosts. Once a host is infected, it will send out packets to all hosts known to it. Fig. 3 shows the pattern of Blaster worms, from a real Cisco NetFlow log file. The abnormal behavior is already apparent without rearranging nodes. The infected hosts can be easily detected using *VisFlowCluster-IP* (which are highlighted by an arrow in Fig. 3).

5.2 Communications with RIPE NCC

In one Cisco NetFlow log file we find that a large number of hosts in NCSA have high-volume communication with a small number of external hosts. This is shown in

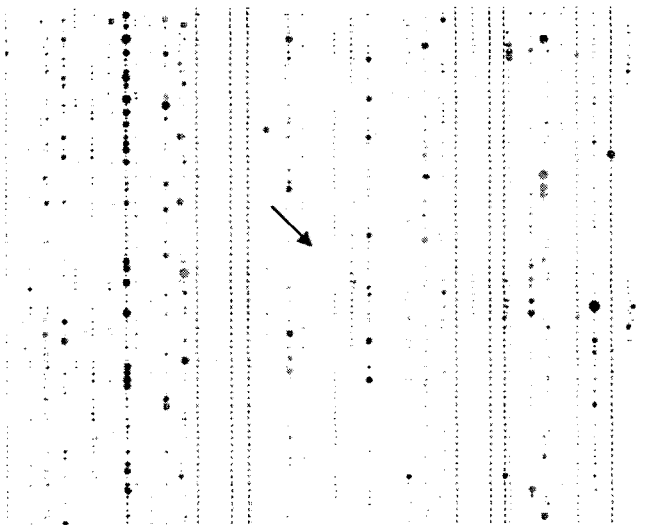

Fig. 3. Pattern of Blaster Worms.

Fig. 4, in which the biggest cluster shows these hosts in NCSA. We find that these hosts are from several clusters of computers, and they connect intensively to the following external hosts as shown in Fig. 4: 131.188.3.221, 153.107.47.81, 192.136.143.150, 192.136.143.151. We used the 'whois' command to check the domains of the above three hosts and we found that they are all from RIPE NCC, a Regional Registry (RIR) providing global Internet resources and related services. We feel that it is not surprising that many hosts connect to RIPE NCC, and this traffic pattern is benign. Therefore, we use filters to filter out traffic involving the domain of RIPE NCC in future visualizations.

5.3 Web Servers

From another Cisco NetFlow log file, we found a group of hosts with high traffic volume that are clustered together, as highlighted in Fig. 5. We found that they are all web servers in NCSA. It is not strange that the web servers are grouped together. However, we find that almost every server in that visual cluster is accessed by about ten hosts from the class C network of 64.68.82.*. This is strange because usually traffic to web servers should come from hosts whose IP addresses are quite random. Again with the "whois" command we find that 64.68.82.* is owned by Google.com. It is clear that Google is crawling our web servers.

6 Conclusions

In this paper we present *VisFlowCluster-IP*, a powerful tool for visualizing network traffic flows. It models the network as a graph by modeling hosts as nodes. It utilizes the force model [9, 13] to arrange graph nodes on a two-dimensional space, so that groups

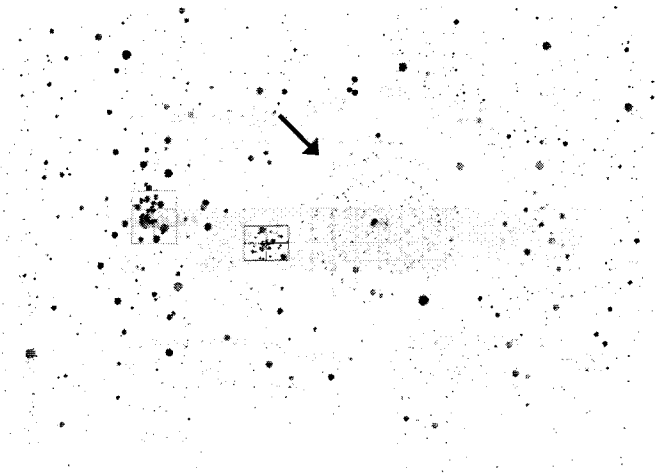

Fig. 4. Communications with RIPE NC.

of related nodes can be visually clustered in a manner apparent to human eyes. We also propose a method based on DBSCAN [12] to automatically detect dense regions on the plane of visualization. We present three real cases that validate the effectiveness of *VisFlowCluster-IP* in identifying abnormal behaviors.

We believe *VisFlowCluster-IP* will be useful for intrusion detection based on preliminary experiments in a laboratory environment. In the short-term we will continue testing *VisFlowCluster-IP* in laboratory environments to identify the types of behaviors it can detect and leverage machine learning techniques for models of normal behavior as well as deviate behaviors. In the future we seek to employ this tool in real networks where accuracy can be statistically measured. *VisFlowCluster-IP* will be distributed at: <http://security.ncsa.uiuc.edu/distribution/VisFlowCluster-IPDownLoad.html>

7 Acknowledgement

We thank fellow members of SIFT research group at NCSA who contributed indirectly to this work: (in alphabetical order) Ratna Bearavolu, Charis Ermopoulos, Kiran Lakkaraju, Yifan Li, and Ramona Su. We thank the anonymous reviewer whose insightful feedback we have incorporated to improve this paper.

References

1. S. Axelsson. Visualisation for Intrusion Detection - Hooking the Worm. *Eighth European Symposium on Research in Computer Security (ESORICS)*, Lecture Notes in Computer Science (LNCS) 2808 , Springer, 2003.
2. R. Ball, G. A. Fink, C. North. Home-Centric Visualization of Network Traffic for Security Administration. *ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

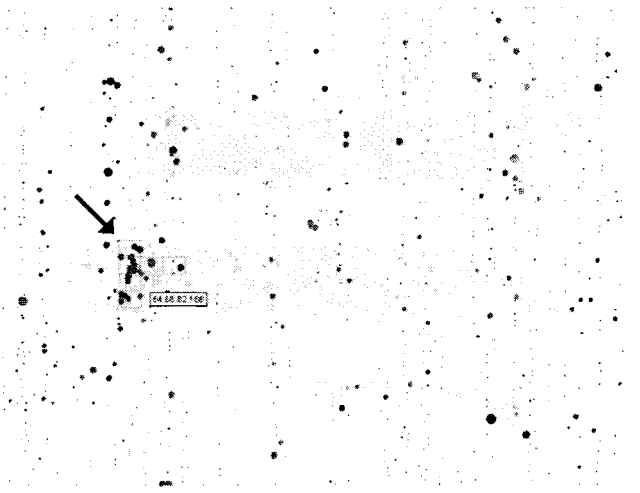

Fig. 5. Patterns with Web Servers.

3. R. Becker, S. Eick, A. Wilks. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics*, 1(1):16–28, 1995.
4. G. Conti, K. Abdullah. Passive Visual Fingerprinting of Network Attack Tools. *ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.
5. K. Cox, S. Eick, T. He. 3D Geographic Network Displays. *ACM SIGMOD Record*, 25(4):50–54, 1996.
6. F. Cuppens, A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. *IEEE Symp. on Security and Privacy*, 2002.
7. H. Debar, A. Wespi. Aggregation and Correlation of Intrusion Detection Alerts. *Int'l. Symp. on Recent Advances in Intrusion Detection (RAID)*, 2001.
8. M. Dodge, R. Kitchin. *Atlas of Cyberspace*. Addison-Wesley, 2001.
9. P. Eades. A Heuristic for Graph-Drawing. *Congressus Numerantium*, Vol 42, pp 149-160, 1984.
10. S. Eick, G. Wills. Navigating Large Networks with Hierarchies. *IEEE Visualization*, 1993.
11. R. Erbacher, K. Walker, D. Frincke. Intrusion and Misuse Detection in Large-Scale Systems. *IEEE Comp. Graphics and Applications*, 22(1):38–48, 2002.
12. M. Ester, H.-P. Kriegel, J. Sander, X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *ACM Int'l. Conf. on Knowledge Discovery and Data Mining (KDD)*, 1996.
13. T. Fruchterman, E. Reingold. Graph-Drawing by Force-Directed Placement. *Software-Practice and Experience*, Vol 21, pp 1129-1164, 1991.
14. C. Krugel, T. Toth, C. Kerer. Decentralized Event Correlation for Intrusion Detection. *Int'l. Conf. on Info. Sec. and Cryptology (ICISC)*, 2001.
15. C. Krugel, T. Toth, E. Kirda. Service Specific Anomaly Detection for Network Intrusion Detection. *ACM Symp. on Applied Computing*, 2002.
16. K. Lakkaraju, W. Yurcik, A. J. Lee, R. Bearavolu, Y. Li, X. Yin. NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness" *ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

17. W. Lee, D. Xiang. Information-Theoretic Measures for Anomaly Detection. *IEEE Symp. on Security and Privacy*, 2001.
18. W. Lee, S. J. Stolfo, K. W. Mok. A Data Mining Framework for Building Intrusion Detection Models. *IEEE Symp. on Security and Privacy*, 1999.
19. A. Noack. An Energy Model for Visual Graph Clustering. *Graph Drawing*, 2003.
20. S. T. Teoh et al. Elisha: a Visual-based Anomaly Detection System. *Int'l. Symp. on Recent Advances in Intrusion Detection (RAID)*, 2002.
21. S. T. Teoh, K. Ma, S. F. Wu. A Visual Exploration Process for the Analysis of Internet Routing Data. *IEEE Visualization*, 2003.
22. S. T. Teoh, K. Ma, S. F. Wu, X. Zhao. Case Study: Internet Visualization for Internet Security. *IEEE Visualization*, 2002.
23. J. Tölle, O. Niggemann. Supporting Intrusion Detection by Graph Clustering and Graph Drawing. *Int'l. Symp. on Recent Advances in Intrusion Detection (RAID)*, 2000.
24. F. van Ham, J. J. van Wijk. Interactive Visualization of Small World Graphs. *IEEE InfoVis*, 2004.
25. X. Yin, W. Yurcik, A. Slagell. The Design of VisFlowConnect-IP: a Link Analysis System for IP Security Situational Awareness. *IEEE Int'l. Workshop on Information Assurance (IWIA)*, 2005.