Proceedings of the 2005 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 17–19 June 2005

A Visualization Paradigm for Network Intrusion Detection

Yarden Livnat, Jim Agutter, Shaun Moon, Robert F. Erbacher, Stefano Foresti

Abstract—
We present a novel paradigm for visual correlation of network alerts from disparate logs. This paradigm facilitates and promotes situational awareness in complex network environments. Our approach is based on the notion that, by definition, an alert must possess three attributes, namely: *What*, *When*, and *Where*. This fundamental premise, which we term w^3, provides a vehicle for comparing between seemingly disparate events. We propose a concise and scalable representation of these three attributes that leads to a flexible visualization tool that is also clear and intuitive to use.

Within our system, alerts can be grouped and viewed hierarchically with respect to both their type, *i.e.*, the *What*, and to their *Where* attributes. Further understanding is gained by displaying the temporal distribution of alerts to reveal complex attack trends. Finally, we propose a set of *visual metaphor extensions* that augment the proposed paradigm and enhance users' situational awareness. These metaphors direct the attention of users to many-to-one correlations within the current display helping them detect abnormal network activity.

I. Introduction

The spread of malicious network activities poses great risks to the operational integrity of many corporations, institutions, and organizations, in addition to imposing heavy economic burdens. Intrusion detection systems (IDS) analyze network traffic and host-based processes in an attempt to detect such malicious activities. The proliferation of different IDSs and the sophistication of attacks lead to a large number of alert types. This complexity is compounded by the sheer number of alerts these systems generate and a high rate of false positives.

There is a growing body of research that validates the role of visualization as a means for solving complex data problems. Visualization elevates the comprehension of information by fostering rapid correlation and perceived associations. To that end, the design of the display must support the decision making process: identifying problems, characterizing them, and determining appropriate responses. It is imperative that information be presented in a manner that facilitates the user's ability to process the information and minimize any mental transformations that must be applied to the data. Our goal is to enable users to quickly decide how pervasive and how severe problems are.

In this work, we focus on developing visualizations that take advantage of human perceptive and cognitive facilities in order to enhance users' situational awareness and support decision-making. We propose a visualization paradigm, Figure 1, for the correlation of various network- and host-based alerts from disparate IDS logs. The display shows the local network topology map in the center, with the various alert logs (divided into specific or group of alert types) on the surrounding ring. The ring width represents time and is divided into several history periods. Finally, the alerts themselves are shown as lines from the most recent history period to the local node being attacked. This visual system provides a holistic view of the local networks, promotes situational awareness, and supports the decision making process. This design thereby increases the likelihood that complex malicious attacks are detected early.

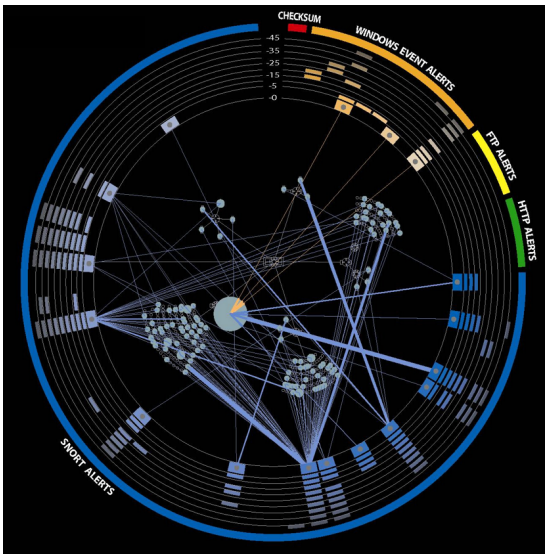


Fig. 1
VisAlert: A novel visualization paradigm for network intrusion detection.

We wish to emphasize that our tool does not perform any analysis on the alerts, nor does it process raw data, *i.e.*,

Y. Livnat: Scientific Computing and Imaging Institute, University of Utah
J. Agutter: College of Architecture+Planning, University of Utah
S. Moon: College of Architecture+Planning, University of Utah
R. Erbacher:Department of Computer Science, Utah State University
S. Foresti: Center for High Performance Computing, University of Utah

ISBN 0-7803-9814-9/$10.00

the network traffic and alerts. Our aim is to enhance situational awareness via visual correlation of existing alerts. In this respect, our approach can accommodate any past or future IDS, or any other system for that matter, which generates network alerts of any kind. Section III describes in detail the rationale behind this statement and the context within which it is valid. It is precisely this lack of additional analysis that frees us from any constraints and dependencies on the kind of alerts we visualize, and provides the flexibility and power of this paradigm.

This paper is structured as follows. Section II reviews earlier visualization work. Section III presents the basic premise of this work, *i.e.*, the notion of network alerts as space-time events of a given type. We then present our visualization paradigm in section IV, followed by a set of visual extensions in section IV-C. Section V presents initial results and we offer some conclusions and future work in section VI.

II. Previous Work

Historically, visualization has been applied extensively to network monitoring and analysis, primarily for monitoring network health and performance [1], [2]. Initial visualization techniques for IDS environments focused on simple scales and color representations to indicate state or level of threat [3], [4], [5], [6]. Other environments provide basic graphical user interfaces but fall short of providing the needed visual capabilities for analysis [7], [8], [9], [10].The few visualization techniques that have been developed for intrusion detection have been limited as to their applicability and effectiveness.

With the need for better analysis mechanisms for security and IDS-related data, more advanced visualization techniques have been increasingly explored over the last few years. Many of these techniques have been proven to be effective at allowing users to see malicious activities such as worm [11], [12] or DoS attacks [13], [11]. However, these visualization techniques tend to be focused on specific problems and not on general alert correlation for an entire enterprise. Other techniques, such as those by Secure Decisions [14], have focused on visual pattern matching, i.e., the representation of already-known attacks. Of the more advanced techniques, additional techniques of note include:

The work by Teoh *et al.* [12] focuses on Internet routing data which allows for the analysis of worms and other large-scale attacks within its limited focus. Similarly, McPherson *et al.* [13] developed a technique for the visualization of port activity that is geared toward monitoring of large scale networks for naive port scans and DoS attacks.

The work by Yin *et al.* [15] and Lakkaraju *et al.* [16] focuses on the representation of netflows and associated link relationships. Such techniques are critical for analyzing attacks and IDS data but these two techniques quickly suffer scalability issues and are limited as to the number of representable parameters.

Wood [17] describes basic graph-based visualization techniques, such as pie charts and bar graphs, and how these techniques can be applied to typical network data available to all system administrators. This work provides a fundamental description of how visualization can be implemented and its application to such data, as well as the meaning behind the identified results. The technique is limited only in the simplicity of the visualization techniques, which currently cannot analyze the high-volume, high-dimensional data currently being generated by today's environments. This remains a major challenge for IDS data analysis in general.

Traditional representations and network alert reporting techniques tend to use a single sensor - single indicator(SSSI) display paradigm. Each sensor has a unique way of representing its information (indicator) and does not depend on information gathered by other sensors. The benefit of such an approach lies in the separation of the various sensors. Each sensor's indicator can thus be optimized for the particular data produced by its sensor, and the user can pick and choose which sensors to use in an analysis. Furthermore, the failure of one sensor does not impact the capability of the rest of the system.

Consequently, the separation between the various sensors is also the weakness of this representation technique. Because each indicator is isolated, the user must observe, condense, and integrate information generated by the independent sensors across the entire enterprise. This process of sequential, piecewise data gathering makes it difficult to develop a coherent, real-time understanding of the interrelationship between the information being displayed, and particularly the identification of malicious attacks.

Our work is designed to expand the visualization capabilities to provide more robust general capabilities and tools to aid administrators and analysts in the analysis of IDS-related data.

III. Alerts, Resources and Instances

The various IDSs employ numerous complex approaches and techniques; yet fundamentally their only purpose is to generate alerts when such activity is detected. In addition, some IDSs generate alerts based on alerts logged by other IDSs. Automated response systems do exist; however, most decision-making is done by humans.

One approach to resolving these issues is to correlate various alerts by common attributes. This approach is based on the premise that while a false positive alert should not exhibit correlation to other alerts, a sustained attack will likely raise several alerts. Furthermore, real attack activities will most likely generate multiple alerts of different types. There exists a large body of work aimed at correlating these disparate alert logs based upon clustering, probability and similarity to predefined attacks [18], [19].

ISBN 0-7803-9814-9/$10.00

Alert correlation aims at establishing the validity or generating a confidence measure for the participating alerts. The main problem in correlating alerts from disparate logs is the *seeming* lack of mutual grounds on which to base any kind of comparison between the alerts. By its very nature, any alert, or any event for that matter, must possess what we term the w^3 premise, namely, the *When*, *Where*, and *What* attributes. *When*refers to the point in time when the alert happened. *Where*refers to the local network node, *e.g.*, an IP address, to which the alert pertains. Finally, the *What*refers to some global indication of the type of the alert, (*e.g.*, $log = snort, gid = 1, sid = 103$). Put another way, a network alert is a point in *space-time*, that has a generic *type* associated with it and may contain additional information specific to that alert type.

Typically, alerts are correlated based on either their *When* or *What* attributes. In the latter, the alerts are first grouped based on the *What* and then correlated, within each group, based on the additional attributes associated with that particular *What* attribute. However, the real value of an alert is with respect to the local resource(s) it pertains to. It is, in fact, the preservation of the resources' status and integrity that is the main focus of IDS to begin with. The alerts' (*What*, *When*) attributes by themselves have little if any inherent value. As a consequence, it is the correlation of alerts with respect to resources that is the focus of this work.

We also distinguish between an alert's *definition* and an alert's *instances*. An alert definition is static and describes the preconditions and meaning of an alert. Though there are collections of predefined and widely-used general definitions, each installation usually extends them with its own private sets of definitions. An alert instance, on the other hand, refers to a particular point in time when the alert preconditions were met. An alert instance may also include detailed information about how and what triggered a particular alert. Correlating alerts and resources, therefore, means correlating alert instances with respect to a particular resource.

IV. The Visualization Paradigm

The proposed visualization paradigm is based on the w^3 premise (Section III), and correlates disparate alerts based on their *What*, *When*, and *Where* attributes. What we need is a way to visually organize the alert instances in a clear and comprehensive fashion with respect to these attributes.

One possible approach is to use a three-dimension Cartesian coordinates system and map the *What*, *When*, and *Where* to the three axises. In this configuration, a network event is represented by a three-dimensional point, as seen in Figure 2. There are, however, several problems with this approach. First, the points do not exhibit any obvious correlation, for example, two nearby points may not share any attribute with each other. Second, the three-dimensional view introduce visual obstacles such as occlusion and depth perception. Finally, the visual perception will depend on the user's specific view point, which only adds another unnecessary degree of freedom to the already complicated situation.

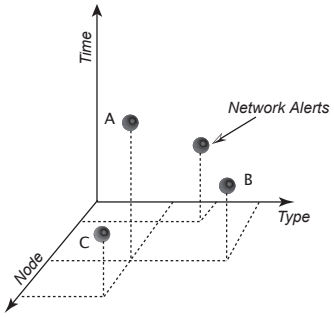


Fig. 2
Cartesian representation of the *What*, *When*, and *Where* attributes. Note that while alert A has the same *Where* as B, and the same *What* as C, it is not evident in this representation.

Alternatively, we base our approach on representing the network alerts as connections between two domains. These two domains are a one dimensional domain representing the *Where* attribute, and a two-dimensional domain representing the *When* and *What* attributes. Note that the *Where* and *What* spaces are finite while the *When* space is semi-infinite. A network alert instance, in this scheme, is thus a straight line from a point in the *What*-*When* domain to a point in the *Where* domain, as shown in Figure 3.

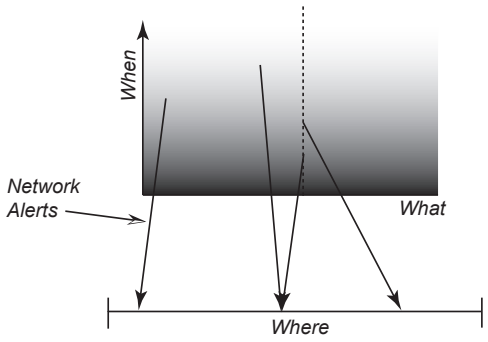


Fig. 3
Domains: Representing network alerts as lines between two domains. The design facilitates the correlation of alerts with respect to the same resources.

We choose to separate the *Where* attribute from the *What* and *When* as resources provide a more or less static

set of objects that we can use as visualization anchors for the transient alert instances. In addition, we can now expand the *Where* domain into a two-dimensional domain, which enables us to layout the resources in a more flexible and meaningful way for the user, such as the network topology map.

Our design layout, as shown in Figure 4, maps the *Where* domain onto a finite circle, while the *What-When* domain is wrapped around it in the shape of a ring. We maintain the orthogonality of the *What* and *When* spaces by mapping the *What* attribute to the angle around the circle and the *When* attribute to the radial direction from the center of the circle.

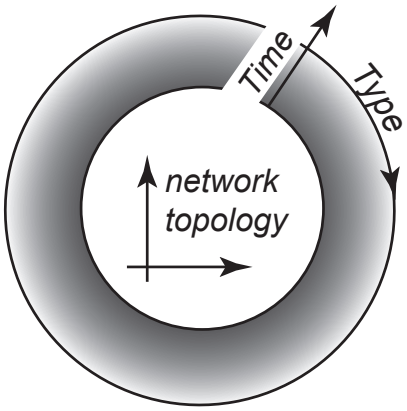


Fig. 4
Mapping: The finite one-dimensional *Where* domain is mapped into a central circle containing the network topology map, while the *When-Where* domain is mapped to a ring around it.

Alert instances can now be visualized, Figure 5, as lines from $\rho(what, when) \rightarrow (angle, radius)$ on the outer ring, to $\psi(where) \rightarrow (x, y)$ in the inner circle, where ρ and ψ are general projections of the alerts into our two domains. In our system we enable the user to dynamically control and configure these two projections as necessary. We discuss these projections in more detail in the rest of this section.

Finally, to reduce the possible cluttering when showing all the alerts simultaneously, we divide the *When* space into varying intervals and show the alerts instances for the most recent history period only. The rest of the history periods show only the number of alert instances that occurred during that period as described in Section IV-B.4.

A. Model and Presentation

The issues of organization and scalability with respect to the number of resources, alert definitions and alert instances can be addressed by providing hierarchical groupings, also known as *levels of detail* (LOD). However, in order to provide changing views of the logs and alerts, we employ the *model, view, control* (MVC) methodology. MVC is based on separating the *model*, namely the data and its organization, from the *presentation*, and using a *controller* to generate the visualization or presentation based on the model. This separation of the data and the presentation components provides a mechanism by which different visualizations can be applied to the data based on outside input.

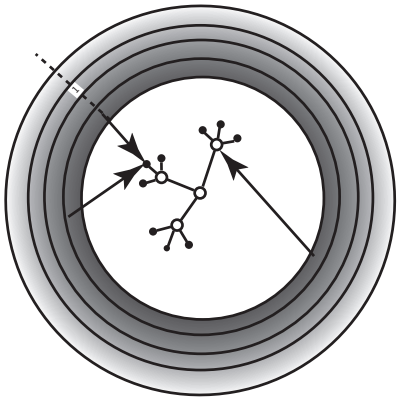


Fig. 5
The visualization paradigm of VisAlert.

B. Level of Detail

Level of detail refers to the notion of a *dynamic* representation on a model based on a set of constraints. For example, as an object moves farther away, the finer details fade away, and thus the object representation can be simplified without effecting how it is perceived. LOD frameworks are based on either discrete or continuous representation. In the discrete case, the framework contains an ordered set of different representation of the object, *e.g.*, coarser views with less details. At runtime, the framework chooses one of these representation based on external input. The discrete approach can therefore lead to abrupt changes in the display of the object. To alleviate this effect, care must be taken to ensure that these transitions occur only when the differences between two adjacent representations are not visible. The continuous framework, on the other hand, provides a representation that changes smoothly based on a continuous parameter. To this end the framework may have a continuous representation of the object or it may interpolate from a fixed collection of representations.

ISBN 0-7803-9814-9/$10.00

In our visual paradigm we employ LOD for each of the three components of the w^3 premise, namely the type, location and time. This provides us with the flexibility to restrict our view of the data to only subsections as in a drill-down or zoom-in operations. In addition, we gain the capability to get global views of the data from different perspectives.

B.1 Logs and Views

Conceptually, we tend to think of alerts as being grouped in two levels: the log they belong to and the kind of alert they represent, *e.g.*, *FTP, access-attempt, policy-violation*. Using this approach, alert instances are collected into bins based on their actual type and these bins are then organized into the various groups which comprise the log. This approach, however, is too restricted and rigid.

In this work we chose to separate between the *model* and its *presentation*. The *model* comprises a fixed two-level hierarchy, in which a log represents an unsorted collection of alert instances, or in our terminology, *events*. Our rationale for this fixed hierarchy is that the various logs are usually collected separately from each other using different IDSs and sensors (thus the *log* grouping). Furthermore, even for the same kind of alert log, *e.g.*, *Snort*, the actual alert types it contains can vary considerably between different organization (thus the notion of a log as an unordered set of alerts).

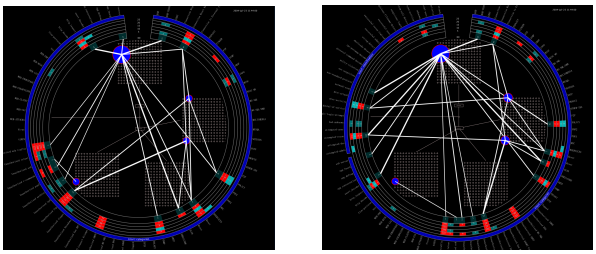
It is the responsibility of the *presentation* layer to organize the logs and the alerts in each log in a fashion that is meaningful to the local operator. This, in turn, allows us to use different presentation schemes for different logs. Furthermore, a single log can be presented in multiple ways-based on the current operator tasks, capabilities, clearance, or even as a filtering operation.

The different presentation schemes can differ in hierarchical structure as well. Each node in the hierarchy can have a different number of children, and the height of each branch can vary as well. The structure can be static or dynamic and can adapt to user needs and tasks. Finally, the graphic presentation can also differ and employ different visual motifs.

B.2 Transition

The visualization paradigm allows the user to zoom in or out of any group within any view, as well as open or close new views. Yet by their very nature, the log views are discrete, leading to a discrete LOD representation. In order not to confuse the user, we must take care of smoothing the transition between the various levels. Consider the transition from a single view to two views of a Snort log, as depicted in Figure 6. The first view organizes the alerts based on the typical Snort groups (*Attach, FTP, P2P, X11, etc.*) while the second view organizes the alerts based on their Snort classification (*attempted-recon, shellcode-detect, policy-violation, etc*). Note that while these two snapshots of the display look fine one next to each other, the user may get confused and disoriented if the left image is instantaneously replaced by the right. Not only will a whole new view popup, but the locations of the bins, groups and alerts instances of the first view will suddenly shift.

We can, however, achieve a smooth transition between the two levels of details via animation. In particular, whenever a group or view is opened or closed we animate it by smoothly varying the length of the arc occupied by the object. During the animation, we multiply the length of an object's arc by a parameter that varies between zero (the object is not seen) to one (the object is fully visible).



(a)Single View

(b)Multiple Views, Each view occupies part of the ring

Fig. 6
A single and multiple views of the same log. Note that the same alert may be displayed twice, once per view.

B.3 The resources map

In the scope of this work, the term *resources* refers to individual network nodes such as hosts, switches, and routers. Each such node may include additional information such as its name, IP address, mission(s), how critical it is to the organization, its operating system and even the OS patch level. However, an unordered collection of nodes can make it hard on the operator to comprehend the relationship between the various nodes. We, thus, use the local network topology map as the base for the resources presentation inside the inner circle. The use of the topology map has both pros and cons. On one hand, the topology map of an organization may not be available, or may not be up to date. On the other hand, the topology map can help to correlate attacks based on sub-group, such as a particular class C, or the physical location of the nodes (assuming the topology map is laid out in that way), or even a combination of the two.

The scalability issue arises when the topology map becomes too big to fit into the inner circle. To this end, we let the user shift the view (a pan operation) of the underlying topology map as well as zoom in and out. However, as the

user zooms out, the individual nodes may overlap or may become undistinguished from one to another. To address these issues, we create a collection of topology maps at various resolutions, essentially a discrete LOD representation. For example, a highly detailed map may show all the nodes in a particular class C, while a map at the next level may use a single node to represent all of the class C's 256 nodes. It is the responsibility of the *controller* (the visualization engine in this case) to group all the alerts that pertain to these 256 hosts and represent them as alerts going to a single class C node when the new map is used. By separating the node's model from the node's representation, *i.e.*, the topology maps, we gain the flexibility of using multiple LODs. Furthermore, the various maps can represent the nodes differently, *e.g.*, using different icons, based on their mission, operating system or criticality.

B.4 Time periods

In our visualization paradigm, time is represented, as seen in Figure 4, as the radial coordinate of a polar coordinate system. We could employ a hierarchical structure similar to a log view described in Section IV-B.1; however, time is represented in a continuous, semi-infinite space rather than in a discrete finite space. A second issue stems from our design, which projects the time domain on the same plane as the network topology. If we try to project alerts from a wide range of timestamps (the *When* attribute) then the resultant image will not be intelligible.

We address the semi-infinite and continuous issue by restricting the width of the ring to a window in time. We assign time t_0 to the inner side of the ring and t_1 to the outer side, where $t_0 > t_1$. In effect, we move back in history as we move from the inner side outward. In this scenario, a zoom operation amounts to changing either t_0, t_1 or both, while a shift (pan) operation is achieved by adding the same dt to both. In addition, we discretize this time window into several history periods, or sub-rings, as shown in Figure 1.

The history rings and the hierarchical structure of the log views divide the *What-When* ring into many cells. Each such cell, in turn, represents a collection of alerts of a particular type(s) which occurred during a specific history period. The second issue can now be resolved by displaying as lines only the alert instances that occurred in the most recent history period, as seen in Figure 5. For the cells in the other history periods, we only display the number of alerts that were collected in each cell. Furthermore, we can use the number of alert instances in each cell divided by the cell period length to compute the alert instance's *density* and use it to assign a background color to that cell. The background colors then in turn show trends in recent history.

C. Other visual indicators

We incorporated additional visual indicators that encode information that increase the situational awareness of the user. We have adopted a method of increasing the icon size for nodes experiencing unique alerts. The assumption is that a resource/node on the topology that is experiencing multiple unique alerts from both host- and network-based sources has a higher probability of malicious activity than one experiencing only one alert. An example of this would be a scan of a particular machine. While this may generate a Snort alert, the activity may or may not be benign; however, a standard IDS system will catch this simple probe and reject the traffic. If on the other hand a machine is receiving a Snort alert in addition to a Windows log alert, then that machine could be experiencing an intrusion attempt or a successful attack. The size of the node is a very clear indication and is easily distinguishable from other machines in order to focus the user's attention so he or she can take action and correct the problem on the suspect machine(s).

When multiple alerts of the same type are triggered with regard to the same node, the alert lines will override each other. To this end we replace these multiple lines with a single *alert beam* which encodes the additional information by its width and color. A beam's width is based on the number of alerts it represents and thus emphasizes the persistence of a problem over a defined interval of time. In this manner, continual or reccurring problems become evident very quickly, enabling the user to take quick action. The color of a beam encodes the severity of the problem. Oftentimes, *e.g.*, Snort alerts, a severity of the problem is associated with each particular alert. Thus severe problems become immediately distinguishable from other alerts.

V. Results

A prototype of our system, termed VisAlert, was deployed and tested at the Air Force Research Lab (AFRL) in Rome, New York, for one week in December 2004. During this time we collected information from analysts and other AFRL personnel about the usability of the tool and features that should be added. The response was very positive; users specifically noted the effectiveness, simplicity, and flexibility of the tool. The analysts stated that VisAlert provided increased situational awareness of network events and would be useful as a tool to help direct their inquiries into network anomalies. They further expressed a desire for formal testing in an operational environment. Additional features that were suggested included the ability to obtain detailed information from selected alerts in a pop-up window, the ability to generate a log of events that have been viewed and store a thumbnail of the visualization for future reference and communication, and the ability to automatically generate the topology map. The detail notes and the event recording have since been

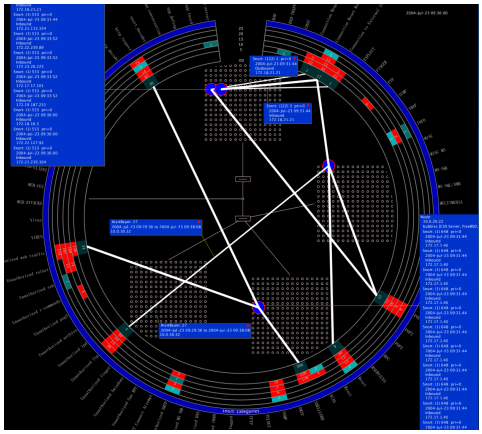


Fig. 7
NORMAL ACTIVITY

integrated as shown below.

For the purpose of testing the capabilities of our system with real attacks, we employed the Hackfest dataset from AFRL. The Hackfest dataset is the result of an AFRL internal exercise in which two red teams attacked (and defended from) each other over a 12 hour period. The dataset also includes background traffic simulated by a Skaion traffic generator. However, one drawback is that the dataset includes only Snort alerts. Nevertheless, for the purpose of illustrating the capabilities of VisAlert, the Hackfest dataset proves satisfactory.

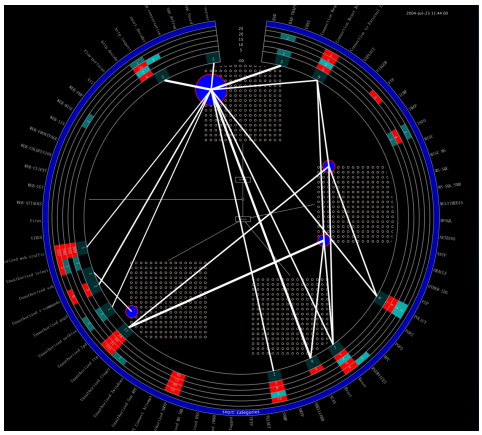


Fig. 8
ATTACK: LARGE NODES EMERGES FROM BACKGROUND TO SHOW MACHINES UNDER ATTACK. NOTE THE MULTIPLE ALERTS CONVERGING ON EACH OF THESE MACHINES

The images presented show different examples of the visualization in different scenarios. Figure 7 shows normal traffic at the beginning of the exercise. Notice that there are few machines that are experiencing alerts, due to the Skaion-simulated network traffic. Figure 8 shows an example of an attack on one local machine by one outside source. Figure 9 shows an example of a massive attack on a whole class C, where each node is attacked for a few minutes. In realtime, the attack is seen as a sweep from left to right until the whole class is under attack at the same time. Only then does the attack on the first nodes stop, again in a left to right fashion.

VisAlert also facilitates interrogation of any additional information associated with a node or an alert. Node information may include its *operating system, mission(s)* and *IP address*, to name a few. Alert information includes attributes specific to its particular *What* attribute, such as *source IP address, destination IP address, source port, protocol*, and *flags*. Figure 7 also shows the VisAlert display with several popup notes that show detailed information pertaining to some of the nodes and alerts. Figure 10 shows a web-based interface to previously logged incident reports.

VI. CONCLUSIONS

In this paper, we present a novel visualization paradigm for correlating network alerts generated by multiple sensors deployed across a network. The paradigm is based on the observation that every network alert must possess three fundamental attributes (the w^3 premise), which in turn provide a consistent basis for correlation. Using domain analysis and perceptive- and cognitive-based design principles we develop a unique and flexible two-dimensional display. We demonstrate a careful design process that leads to a two-dimensional display that is more informative than typical three-dimensional approaches and better facilitates the decision making process.

These visualization techniques are geared for analysis of large-scale distributed sensor environments and resolve critical problems within the IDS domain. The visualization

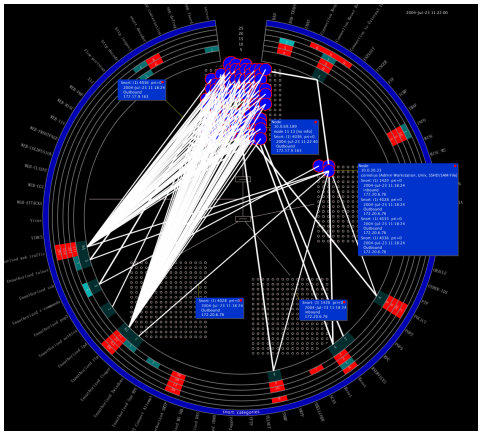


Fig. 9
MASSIVE ATTACK

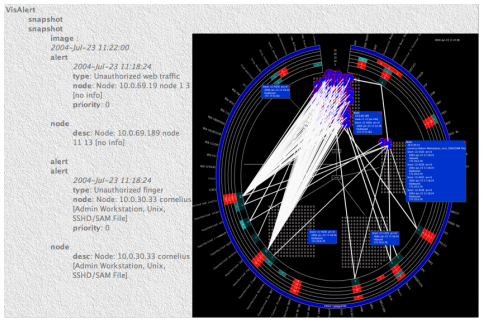


Fig. 10
INCIDENT REPORTS CAN BE VIEWED AND INTERROGATED DYNAMICALLY USING A WEB-BASE INTERFACE.

integrates the critical components of alerts, i.e., the *What*, *When*, and *Where* attributes, with additional specifics retrievable through probing. The visualization system provides analysts and system administrators a flexible platform for analyzing network alerts. While our environment performs no analysis itself, the transformation of the data to a visual form allows users to perform a perceptual analysis which augments that which can be performed automatically.

Our visualization paradigm is designed to improve users' ability to spot critical network anomalies more quickly in order to reduce the impact and severity of network attacks. The results from a field test at the Air Force Research Lab in Rome, New York, suggest that VisAlert has the ability to speed up the detection, diagnosis, and treatment of network attacks. In future work, we intend to expand the visualization capabilities of VisAlert and incorporate additional visualization displays for network flow traffic. We are conducting additional user studies and beta testing at a variety of operational sites.

ACKNOWLEDGMENTS

We would like thank the network security experts and managers from Battelle, the AFRL, and the University of Utah (Information Security Office, NetCom, Center for High Performance Computing, and Scientific Computing and Imaging Institute) that significantly contributed to the domain analysis work. We also thank AFRL's Rome labs for hosting a field test of the VisAlert system as well as providing the Hackfest dataset. This work was supported in part by a grant from the IC-ARDA.

REFERENCES

[1] S. E. Kenneth Cox and T. He, "3d geographic network displays," *ACM Sigmod Record*, vol. 25, 50 1996. December.
[2] D. Estrin, M. Handley, J. Heidermann, S. McCanne, Y. Xu, and H. Yu, "Network visualization with nam, the vint network animator," *IEEE Computer*, vol. 33, pp. 63–68, November 2000.
[3] D. Polla, J. McConnell, T. Johnson, J. Marconi, D. Tobin, and D. Frincke, "A framework for cooperative intrusion detection," in *21st National Information Systems Security Conference*, pp. 361–373, October 1998.
[4] J. M. Vert G. and D. Frincke, "Towards a mathematical model for intrusion," in *21st National Information Systems Security Conference*, pp. 329–337, October 1998.
[5] C. Ko, D. Frincke, and T. G. et al., "Analysis of an algorithm for distributed recognition and accountability," in *ACM conference on Computer and Communication Security*, vol. 1, 1993.
[6] S. e. a. Snapp, "Dids (distributed intrusion detection system) motivation, architecture and an early prototype," in *National Information Systems Security Conference*, 1993.
[7] http://www.networkice.com.
[8] http://www.iss.net/securing_e-business/security_products/intrusion_detec
[9] http://www.cisco.com/univercd/cc/td/doc/pcat/nerg.htm.
[10] http://www.esecurityinc.com.
[11] S. F. W. S.T. Teoh, K.L. Ma and X. Zhao, "Case study: Interactive visualization for internet security," in *In Proceedingsof the IEEE Conference on Visualization 2002*, pp. 505–508, 2002.
[12] K. M. S.T. Teoh and S. F. Wu, "Visual exploration process for the analysis of internet routing data," in *IEEE Conference on Visualization 2003*, pp. 523–530, 2003.
[13] J. McPherson, K.-L. Ma, P. K. T. Bartoletti, and M. Christensen, "Portvis: A tool for port-based detection of security events," in *CCS Workshop on Visualization and Data Mining for Computer Security*, October 2004.
[14] A. D'Amico and M. Larkin, "Methods of visualizing temporal patterns in and mission impact of computer security breaches," in *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, vol. 1, pp. 343–354, June 12–14 2001.
[15] X. Yin, W. Yurcik, M. Treaster, Y. Li, and K. Lakkaraju, "Visflowconnect: netflow visualizations of link relationships for security situational awareness," in *Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security,ACM Conference on Computer and Communications Security*, October 29 2004.
[16] A. J. L. Kiran Lakkaraju, William Yurcik, "Nvisionip: netflow visualizations of system state for security situational awareness," in *Proceedings of CCS Workshop on Visualization and Data Mining for Computer Security, ACM Conference on Computer and Communications Security*, October 29 2004.
[17] A. Wood, "Intrusion detection: Visualizing attacks in ids data," giac gcia practical, SANS Institute, February 2003.
[18] A. Valdes and K. Skinner, "Probabilistic alert correlation.," in *Recent Advances in Intrusion Detection*, pp. 54–68, 2001.
[19] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts.," in *Recent Advances in Intrusion Detection*, pp. 85–103, 2001.