Visualizing Graph Dynamics and Similarity for Enterprise Network Security and Management

Qi Liao
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana, U.S.A.
qliao@nd.edu

Aaron Striegel
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana, U.S.A.
striegel@nd.edu

Nitesh Chawla
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana, U.S.A.
nchawla@nd.edu

ABSTRACT

Managing complex enterprise networks requires an understanding at a finer granularity than traditional network monitoring. The ability to correlate and visualize the dynamics and inter-relationships among various network components such as hosts, users, and applications is non-trivial. In this paper, we propose a visualization approach based on the hierarchical structure of similarity/difference visualization in the context of heterogeneous graphs. The concept of hierarchical visualization starts with the evolution of inter-graph states, adapts to the visualization of intra-graph clustering, and concludes with the visualization of similarity between individual nodes. Our visualization tool, *ENAVis* (Enterprise Network Activities Visualization), quantifies and presents these important changes and dynamics essential to network operators through a visually appealing and highly interactive manner. Through novel graph construction and transformation, such as network connectivity graphs, MDS graphs, bipartite graphs, and similarity graphs, we demonstrate how similarity/dynamics can be effectively visualized to provide insight with regards to network understanding.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*; H.5.2 [**Information Interfaces and Presentations**]: User Interfaces; K.6.5 [**Management Of Computing and Information Systems**]: Security and Protection

General Terms

Security

Keywords

Visualization, security, enterprise networks, graphs, visual graph data mining, local context, policy assessment

VizSec'10 September 14, 2010, Ottawa, Ontario, Canada.

1. INTRODUCTION

Complex systems such as large-scale enterprise networks are hard to manage in terms of security and troubleshooting. In cases of security investigation and performance troubleshooting, tracking down precisely *who* (users) and *what* (applications) are responsible for the generation of this network connectivity is a non-trivial task. Administrators need a tool that allows them to sift through massive amounts of traffic logs in a visually appealing and interactive manner that encourages data exploration with minimum effort.

Traditionally, network monitoring and logging schemes are focused on point-to-point communications that involve hosts' IP addresses and port numbers. Out of the few visualization tools [1–7] that exist, most rely on either per-packet information or NetFlow data. These visualizations often fall short in enterprise settings where users and applications are more important from a security policy perspective than the particular host IP and/or port [8–10].

Visualizing enterprise networks is challenging due to the increased number of data dimensions and finer granularity of information that occurs within enterprise networks. Understanding the enterprise networks involving hosts, users, and applications is especially hard due to the highly dynamic nature and inter-dependency causality relationships among users and applications. Although the network connectivity naturally condenses to well-connected graphs, the nodes and connection edges are constantly changing, making visual comparison and classic anomaly detection less effective due to lack of clean training data.

The key challenge is how to effectively visualize the *dynamics* and *similarity* (or conversely *difference*) among the heterogeneous network graphs consisting of hosts, users, and applications. Similarity visualization is important because it is usually the first step in understanding the patterns in networks and potentially finding abnormal behaviors. The ability to extract the meaningful knowledge from otherwise highly dynamic and noisy data and present them in a visually appealing manner that can provide insights to enterprise security management is non-trivial.

The security visualization approach that we propose ties together intelligent data analysis involving graph theory and data mining algorithms as well as interactive data exploration/querying mechanism to drill down the root cause of potential security issues. The visual analysis presented in this paper builds on a *hierarchy* of similarity visualization ranging from inter-graph to intra-graph clustering visualization to dynamics visualization in each individual nodes

based on four different types of graphs. Novel graph construction is formalized for host-user-application (*HUA*) connectivity graphs [10], bipartite (or multi-partite graphs), multi-dimensional scaling (MDS) graphs, and similarity graphs based on common properties shared by two nodes. Each type of graph has been illustrated to show what insight can provide and how clever algorithms and visualization can help security management and network monitoring.

2. BACKGROUND

Managing large scale enterprise networks is hard due to the complexity and dynamics of the *context* of a connection, i.e., the user (*who*) and application (*what*) responsible for the network activity, to be known rather than simply *where* (address) it came from and went to. The local context information (4W: *who*, *what*, *when* and *where*) associated with each network connection can be collected in a variety of ways. Tools such as [8–10] describe various mechanisms to collect the missing context data. The dataset to be visualized, explored, and analyzed by the network operators and administrators is nevertheless challenging due to the magnitude, temporal nature and heterogenous property of network graphs. Unlike end-to-end flow data, there is a wealth array of local *context* information involving hosts, users, applications and files. However, how to present and convey only essential knowledge is not easy. Besides the magnitude, the temporal nature of the network context data also makes visualizing the time evolution of network activities especially hard. Moreover, traditional data mining algorithms become less effective due to the lack of clean, noise-free data, which is unlikely in the high level of dynamics in user and application connectivity. Furthermore, few data mining algorithms can deal with the heterogenous graphs because the node types can be either hosts, users or applications. Other factors such as how to construct the graphs (there are usually numerous ways to do so) and how to decide the time windows are challenging and also interesting as different types of graph construction can lead to different interesting problems in terms of best suitability and most effective visualization.

With the proposed visual analytic framework, the administrator is presented with an array of connectivity graphs and statistics on how the network is being used. To assist the user in understanding the many possible visualization modes, we utilize a novel *meta-visualization* which compactly represents and controls how data is represented, as developed in our prior work [10]. By adjusting the Host-User-Application (*HUA*) control, the investigator can easily expand, contract, and explore a very rich data space in a visually appealing and highly interactive manner. At the top level, we have *H* denoting the *host* level chaining, which is similar to NetFlow data map. This is also the most common scenario, in which all the connectivity between physical end-host machines is constructed. At the lower levels, we have *U* and *A*, denoting the *user* and *application* level chaining; this is useful when we want to quickly know which users or applications have been communicating with each other. The user may switch between views and filter out *HUA* information in a customizable level of granularity. An example of *HUA* graphs is presented in Figure 1. For example, suppose an enterprise user *jdoe* logged onto a host *cselab01.domain.edu* and launched the application *firefox.exe* to visit the webserver *www.cnn.com*, a straightforward way of constructing an *HUA* graph would be a directed path: [*cselab01*] → [*jdoe*] → [*firefox*] → [*cnn*]. In the next few sections, we will describe other novel ways of graph construction as well.

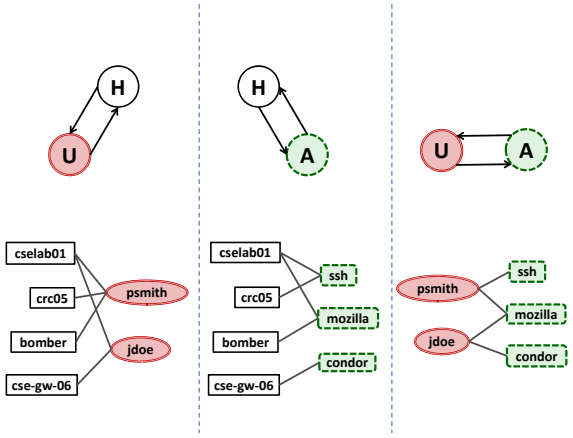


Figure 1: Examples involving only two network components (i.e., bipartite graphs): HU, HA and UA.

Our visualization tool suite (*ENAVis*) allows the network administrators to view their network activities at the user and application levels in addition to the topology created by the host connectivity. The tool allows network operators to visualize, explore and analyze the network activities among hosts/domains, users and applications, which is possible through the gathering of *local context* information. The tool offers interesting, ready-to-use, and invaluable functions for monitoring, visualizing, exploring, investigating and analyzing the activities on a network by real-world network administrators. The methodology of visual analysis on different graphs can have wide applications in network management such as security policy audit, forensics, and fault localization.

While being able to visually examine the network activity graphs augmented with the above *HUA* context information can be helpful, an even more powerful yet challenging task is how this wealthy set of data can be viewed and converted into human-understandable knowledge that can potentially help network operators to gain insights and to make high-level decisions on their network strategies. How can the analytical methodologies such as graph mining and graph theories can be combined with visualization techniques as well as interactive data exploration and querying mechanisms to provide intelligent assistance? Understanding enterprise networks requires a higher level of visualization since it is not simply to *visualize* the data literally, but how to present the data in a meaningful and insightful manner. Motivated by this, visual mining capability as an *intelligence* module has been built into the visualization tool for analyzing network related local-context data. The intelligence module quantifies the network changes (and similarities) in a scientific manner, and *guides* the exploration process by presenting only things that need further examination and investigation. We demonstrate how this visual analysis combined with various novel views and visualization techniques, graph theories and data mining techniques can aid human operators to gain such insights and achieve better understanding on their managed enterprise networks, and can provide ad-

vises for potential problems/anomlies that are otherwise not so obvious even with effective visualization process.

The next few sections focus on one important direction, i.e., how to visualize the *dynamics* and *similarities* in complex enterprise networks. We ask important questions in enterprise network monitoring and security management, such as *how different* (or conversely *how similar*) is from day-to-day network activities; what are the *variance* and *invariance*; what are the patterns; which changes are *normal* while others are *abnormal*; and how to visualize the evolution and the dynamics of changes.

3. SIMILARITY VISUALIZATION OF ENTERPRISE NETWORK GRAPHS

Enterprise network systems are represented as graphs where the dynamic interactions between network components such as hosts, users, applications and data naturally form a complex network. One of the most relevant features of graphs representing real systems is community structure, or clustering. Visualizing the community structures and clusters offers significant contributions in understanding the networks and has applications in network security including traffic classification and anomaly detection for abnormal user and application behavior. Therefore, cluster visualization will be an important component of similarity visualization.

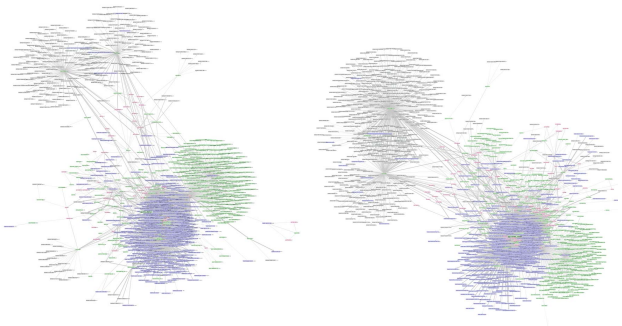
We illustrate how the tool can provide insight for understanding and solving practical network management and security problems through a *hierarchical similarity visualization*. We discuss the similarity visualization in a top-down manner by starting with the top level, where we want to visualize the similarity/differences among network graphs across multiple timelines. Next, we move down one level to each individual graph and try to identify the similar structure among groups of nodes. Novel ways were developed for dealing with the heterogeneous network graphs by transforming them into different types of graphs, i.e., network connectivity graphs, bipartite graphs and similarity graphs. Lastly, we show another aspect of similarity visualization at the bottom level, i.e., how dynamics of neighborhood change at each individual node.

3.1 Inter-graph Cluster Visualization

The first step of anomaly detection is to understand what is *similar* among all days' traffic pattern and what *changes* are abnormal. Visualization of similarity/changes is therefore essential to network monitoring and security. However, visual comparison of similarity/changes even between moderately sized graphs (Figure 2) becomes a challenging task due to physical constraint of human perceptual and cognitive limitations.

To quantify the changes of network graphs, appropriate metrics must be defined to suit the need of an organization. Generally, common graph properties such as graph sizes, diameters, degree distributions, etc. can be used as a first step of measuring the day-to-day changes happened in the managed networks. While these metrics are relatively coarse (e.g., graphs having exactly the same degree distribution can be very different), finer granular metrics such as graph edit distance may be used to quantify the changes at the nodes/edges level.

In this section, we focus on the top level of the similarity visualization by comparing the network graphs as a whole.



(a) *HUA* graph 1 (b) *HUA* graph 2

Figure 2: Examples of two moderately sized *HUA* graphs. Visual comparison of what changed and what not changed is infeasible.

Each node in an inter-graph cluster will represent one snapshot graph taken at a specific time. A concrete example is given to illustrate how to visualize the inter-graph clusters based on graph distance visualization.

3.1.1 Graph Distances

While general graph properties such as degree distribution and graph sizes can be useful in getting a rough idea about the changes in networks, they are sometimes too coarse to be useful in the setting of enterprise network management, where the administrator usually wants to know exactly what host or what user is causing the problem. Methods based purely on topological information while not differentiating node labels are not suitable in enterprise network management since even two topologically identical graphs consisting of two different sets of hosts will rarely be treated the same by an administrator. While a full description of the differences between two large networks is infeasible because it requires solving the subgraph isomorphism problem, many graph algorithms become very efficient due to the fact that each node in an enterprise network is uniquely labeled, either by its IP address, user ID or process ID.

While several graph distances are defined in the literature [11], without loss of generality the following general metrics are adopted to evaluate the graph similarity (or distance) between any pair of heterogeneous *HUA* graphs. However, it is understood that different weights may be put on the nodes/edges based on the needs of different organizations.

MCS based distance: The rationale behind this maximum common subgraph (MCS)-based metric [12] is based on the portion of subgraph structure that both graphs share, i.e.,

$$d(g_1, g_2) = 1 - \frac{|\text{mcs}(g_1, g_2)|}{\max(|g_1|, |g_2|)}$$

Essentially, it is the graph size of the common part divided by the graph size of original graphs. If two graphs are exactly the same, then the size of MCS equals the original graphs, and the distance will be 0. On the other hand, if two graphs are totally different, the size of MCS will be zero resulting a distance of 1.

Edit based distance: In information theory, the edit distance is the number of operations required to transform one of them into the other. For exact graph matching, graph edit distance (GED) [13] can be used for this purpose. The

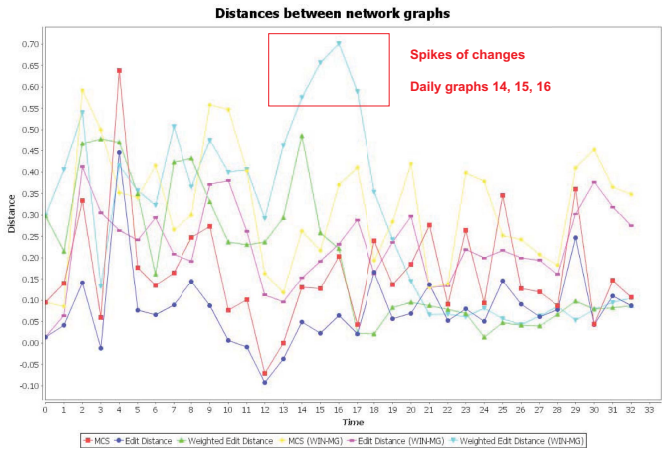


Figure 3: Similarity measurement of day-to-day network graphs using various distance functions.

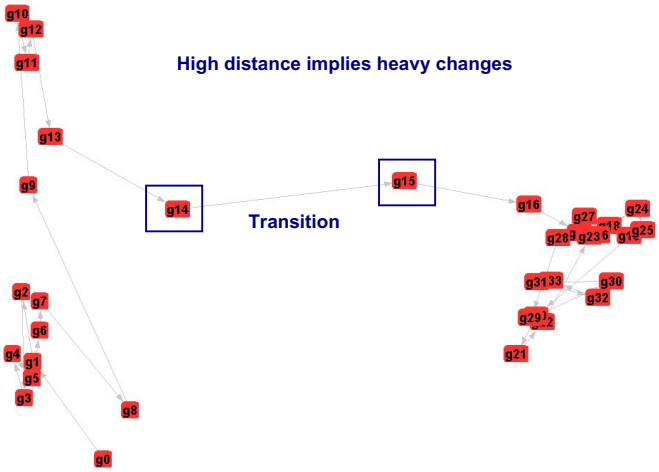


Figure 4: A multi-dimensional mapping (MDS) visualization illustrates the cluster evolution of network graphs. Network states transit at day 14 and 15.

basic idea of graph edit distance is the costs associated to modify a graph such that it becomes isomorphic to the other graph. The rationale behind this scheme is the more steps taken to transform from one graph to the other the larger the distance. One way to calculate the edit distance is to compute the *deletion* cost from g_1 to $MCS(g_1, g_2)$ and plus the *insertion* cost from $MCS(g_1, g_2)$ to g_2. If all cost functions equal to one, then the normalized distance function can be simplified to:

$$d(g_1, g_2) = \frac{|g_1| + |g_2| - 2|\text{mcs}(g_1, g_2)|}{|g_1| + |g_2|}$$

Intuitively, if two graphs are matched exactly the same, the numerator will be zero, resulting a zero distance. On the other hand, if two graphs do not share a single node, resulting a distance value of one.

3.1.2 Case Study

In this example, the administrator opens up the tool, quickly loads the data of the past month, which automatically generates network graphs involving the connectivity of hosts, users and applications. The first thing he does is to plot a graph distance chart (Figure 3) over the entire investigation period.

Figure 3 shows normalized pairwise distances of network connectivity graphs between consecutive days using six distance metrics: MCS-based, GED-based, edge-weighted GED based, and then for each metric a *median graph*[1] was computed over a sliding time window of five days and was used to compare against rather than immediate predecessors. It clearly shows spikes of changes in day 14 and 15. The administrator wants to know what are the main causes behind those big changes and whether all days after day 15 are converging to earlier states or moving further away, which cannot be directly derived from Figure 3.

In order to answer the above questions, a distance matrix composed all pairs of graphs is computed. Now, the administrator needs to plot and visualize the graphs' relative positions to each other. Generally speaking, knowing the exact locations of points, computing their distances is straightforward (e.g., Euclidean distance). However, the opposite way, i.e., knowing their pairwise distance, finding their exact x/y coordinates is not straightforward, and sometimes it may or may not be possible to find the exact points in lower dimensions (2D or 3D), which are the valid spaces for visualization. Multidimensional scaling (MDS) [14, 15] has been proposed to visualize high-dimensional data by mapping the graph nodes into lower-dimensional space.

Therefore, the administrator chooses the MDS menu from the visualization tool, which performs a mapping of the distance matrix into a 2D space, so the administrator can use the tool to visualize the inter-graph cluster evolution, presented in Figure 4. Each node represents a network graph on a specific day, and an edge is drawn to indicate the evolution of movement over one month's period. Condensing a daily network graph into just a node in a MDS graph reduces the dataset significantly for clearer and more effective visualization. While the temporal lines are just for illustration purpose, the lines can be optionally set invisible if one wants to examine the past year's graphs instead of just past month. The numbers in the node labels reflect the time sequences of network graphs. Note that such a global MDS view on the evolution of network graphs shows the relative relationships between *all HUA* graphs. The MDS view shows roughly three clusters: lower left (day 0-8), upper left (day 9-14), and right (day 15-33). Interestingly, graphs on day 14 and 15 are of particular interest because they act like important state transition nodes that bridge between two clusters. Clearly, the networks after day 15 form a cluster that does *not* go back to the original state, but instead moving further away from the state of previous graphs. This view greatly enhances administrators' understanding and insight on their networks.

One open question still remains as what actually causes the network state changes starting from day 14 and 15? In order to drill down to the root cause, the administrator uses the tool's distance-by-components function (Figure 5), which provides the top n network graph components that contribute most to daily changes. Two applications *sge_commd* and *sge_execd* run by the *root* user and user *1025*

[1]Median graphs are computed by taking the medians of edge weights of all graphs.

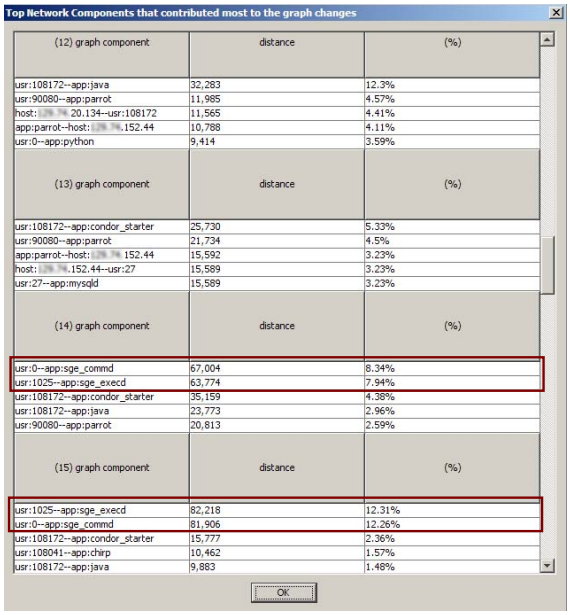


Figure 5: Drill down of the individual responsible hosts, users or applications through top n list pointing to two users: root and 1025, running Sun Grid Engine (SGE) related applications that contributed nearly 25% daily network graph changes.

are the top two heavy hitters which contribute almost 17% and 25% on both days.

Further investigation through dynamic interaction with the network graph of the 14th day is conducted by selecting and visually exploring the problem node facilitated through the sorting capability by node degree and edge weights. A 1-hop view from a node under investigation (user *1025*) is generated on-demand (Figure 6) showing the source and destination hosts and applications associated with that user. The administrator dynamically interacts with the network graph by clicking the node and choosing the context-aware menu with *trend* option to view the activity (by weighted degrees, which takes into consideration the magnitude of connectivity) of a node under investigation. The activity chart automatically generated next to the node clearly indicates that this user *1025* emerged on day 14 and stayed on for all following days. Thus the cause of network state changes is discovered that new user and new application, i.e., Sun grid engine communication agent run by this user are responsible for the shift of network communication patterns.

3.2 Intra-graph Cluster Visualization

In the previous section, we illustrated that the inter-graph cluster visualization can be a good starting point to find anomaly from a top-down fashion. Through interactive graph exploration and queries, the root cause can be quickly identified to specific users and applications with minimum effort. In this section, we move the hierarchical similarity visualization one level down and focus on the similar nodes within a graph rather than among graphs, namely why some users, applications and hosts should be grouped together and what potential insight of those structure changes can bring to us. Before visualizing clusters within graphs, graphs need to be constructed and/or transformed in ways that either

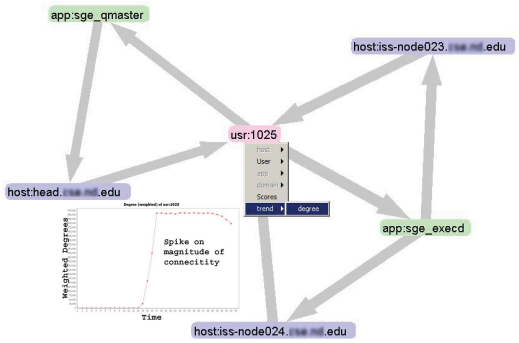


Figure 6: One-hop view from the node under investigation with automatic user activity view pins down the cause of transition during the days 14 and 15 to this user with two new applications.

topological-based or attribute-based methods can be developed. The graphs we consider here are: *HUA* connectivity graphs, *HUA* bipartite graphs, and *HUA* similarity graphs. We illustrate how each clustering method is suitable for different *HUA* graphs and what information gain over traditional IP/port connectivity graphs from a cluster visualization perspective.

3.2.1 HUA *connectivity graphs*

We start with grouping similar nodes within an enterprise network graph involving hosts, users and applications. The *HUA* connectivity graphs represent the actual connectivity among various network components with the augmented local context of network connections. It is ideal for providing an aggregate view and capturing the inter-relationships among hosts, users and applications. While the tool implements other alternative graph-based community detection methods, state-of-the-art graph clustering algorithm: Walktrap [16] is chosen for illustration purpose due to the fact that the similarity measurement of Walktrap is based on a simple yet effective assumption: a random walk tends to be trapped in highly connected or dense area. Walktrap has been widely recognized for the capture of the community structure in a network [17].

A screenshot of one example of intra-graph clustering is overviewed in Figure 7, where nodes belong to different clusters are colored differently for easy visualization. Clusters number one and two are web related communities, where cluster one shows all external domains connected by *firefox*, and cluster two shows an internal web server that has been accessed by a group of clients. Cluster number three shows seven enterprise users sharing a similar set of applications that have queried the campus directory server: *directory.domain.edu*. The giant cluster number four shows a well structured *condor* community formed by a few local and condor users that have launched batch jobs in the *condor* enabled computing nodes on campus. The nice thing of visualizing clusters on the entire *HUA* graph is the aggregate view of the grouping of both hosts, users and applications based on the topological information derived from the actual network connections.

Visualizing the intra-graph clusters can not only provide insight on monitoring the network activities occurred in a

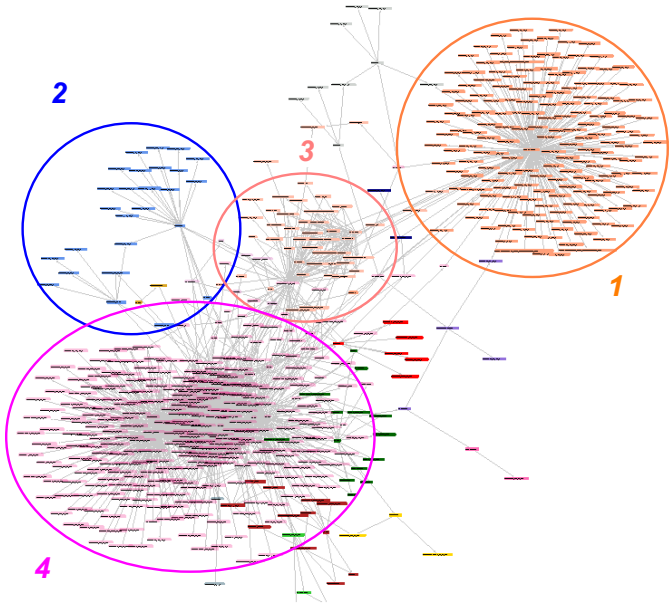


Figure 7: Intra-graph cluster visualization (different colors represent different clusters) using Walktrap algorithm helps understand the communities, e.g., firefox users (1) and web related traffic (2), condor-related research computing community (4), or enterprise users running a set of desktop applications that exhibit similar patterns of connectivity (3). This information has potential to identify anomalous user behaviors.

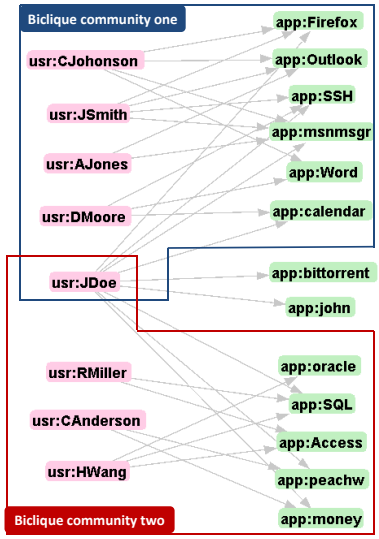


Figure 9: Example of biclique communities consisting of enterprise *users* running similar sets of network *applications*. The user *JDoe* migrating from an exclusive community (top) to overlapping communities (bottom) indicates possible violation of network usage policy.

busy enterprise network, but also naturally find anomaly. For example, by observing the cluster evolution over a certain time series, it is possible to detect users who changed their cluster memberships due to either shift of application usage patterns or changes of source/destination machines on which they conduct their network activities.

The visualization of cluster evolution involving the hosts, users and applications reveals the shift of usage pattern that can be readily spotted by the administrators. To facilitate the process, there must be a way to compute the distance between two sets of clusters. The normalized cluster distance function is defined as

$$dist(C_1, C_2) = 1 - \frac{SS + DD}{SS + SD + DD + DS}$$

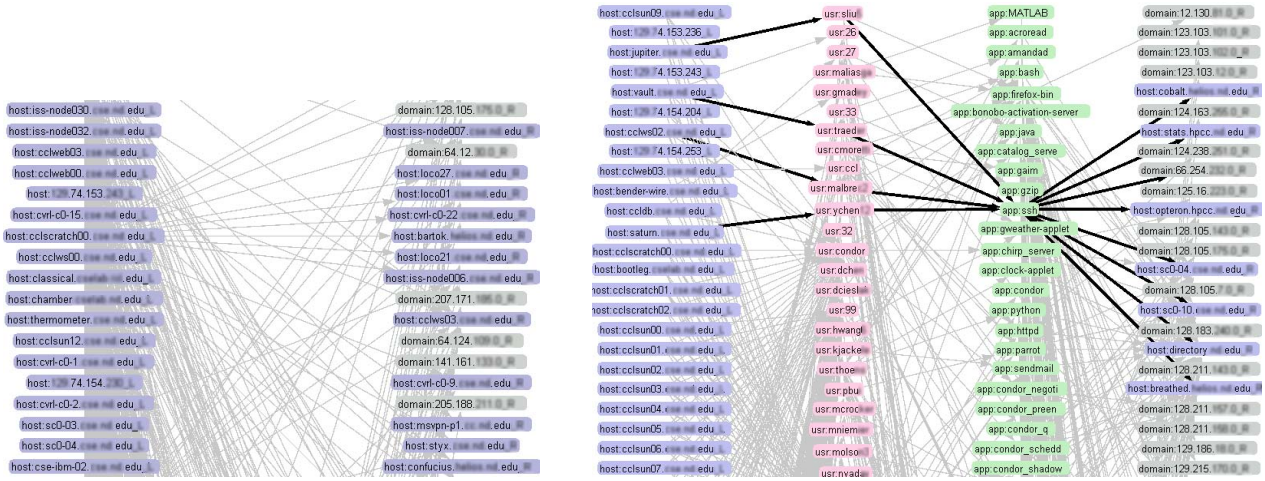
where SS and DD represent the number of nodes staying either in the same cluster or in different clusters in both C_1 and C_2, SD and DS represent the number of nodes that are in the *same* cluster in C_1 but belong to *different* clusters in C_2, and vice versa. This simple yet effective way to measure the changes of clusters is based on an idea similar to the Rand Index [18]. Once the similarity/distance between any two sets of clusters C_1 and C_2 can be computed, visualizing the cluster changes/evolution becomes straightforward.

3.2.2 Bipartite graphs

Visualizing clusters in the context of *heterogeneous* graphs that have different types of nodes (hosts, users and applications) is similarly challenging due to its complexity. An alternative approach to find such structures in HUA graphs is thus desirable. The idea we present here is to separate the general graph into bipartite graphs, which breaks down the heterogeneity property as well as the complexity of HUA connectivity graphs by considering each pair of hosts, users and applications at a time. Normally, a general graph is not guaranteed to transform into a bipartite graph, equivalent to the graph coloring problem. A graph is bipartite if and only if it does not contain an odd cycle, or in other words a bipartite graph cannot contain a clique of size three or more. However, the heterogeneity properties of HUA graphs allow vertices to be divided into disjoint sets H, U, and A. Possible types of bipartite graphs are host-to-host, user-to-user, application-to-application, host-to-user, user-to-application, and application-to-host, etc.

Visualizing network connectivity in bipartite or multi-bipartite graphs can provide a clearer view of the roles that either a host, user, or application plays in the network flow. The augmented $HUAH$ quadripartite view shown in Figure 8(b) demonstrates an expanded view over the HH bipartite in Figure 8(a) since Figure 8(b) involves the finer details of the *connectivity paths* involving users and applications.

It is important to notice the *information gain* from Figure 8(a) to Figure 8(b). For example, by clicking on a user node, all machines that the user has logged on and all applications that the user has launched will be highlighted. In this specific screenshot in Figure 8(b), one questionable application *ssh* is selected, and four users who ran *ssh* are highlighted, together with all source hosts those users have logged on and all destinations hosts connected via *ssh* are also highlighted. This is equivalent to showing the *critical paths of investigation* after a compromise or attack. It also helps to understand and find the patterns in the network, e.g., which users or applications usually connect to which set of machines. Multi-bipartite graphs are useful for modeling the matching problems and finding the bonds between hosts



(a) Bipartite View: Traffic flow between local hosts (left) and remote hosts (right).

(b) Quadripartite View: Gain insight information by expanding *host-host* relationship and visualizing continuous *HUAH* bipartite graphs.

Figure 8: Rather than having a relatively simple point-to-point view (left), administrators can expand the dimensions of network connectivity with multi-bipartite view with extra *information gain* in users and applications (right) for elevated insight. Clicking on a questionable node will highlight the critical paths of investigation that go through that node.

and users, users and applications, etc.

The biclique communities detection algorithm [19] is used in our visualization tool to find all bicliques of hosts, users and applications in the above bipartite graphs. Biclique communities detection algorithm is based on k-clique community detection algorithm. Essentially, k is subdivided into two values: $k_{a,b}$ meaning the clique (a complete graph) consists of a of left nodes and b of right nodes. A $k_{a,b}$ community is defined as a union of all $k_{a,b}$ cliques that are adjacent. The communities detected by clique detection are strong communities because they are the maximal connected graphs that one can ever get. One uniqueness of biclique clustering is that distinct communities can overlap by sharing nodes. The concept of *overlapping* communities is usually not available in most other community detection algorithms, and is interesting in the context of enterprise network management and security. For example, a user can belong to multiple communities due to different tasks required by job functions. Visualizing these network of communities linked by bridging nodes can be of particular interests from policy compliance perspective.

Figure 9 provides a hypothetical example of biclique community visualization. The view shows two biclique communities for a *user-application* type of bipartite graph. The top community is characterized by the usage pattern of typical enterprise users, which involves primarily desktop applications that have network connections. The bottom bipartite community includes a set of applications that according to the policy can be accessed only by employees in human resource (HR) and finance department. The enterprise user *JDoe* who has a role of normal enterprise user changes his exclusive cluster membership to an overlapping community with the HR and financial cluster. Furthermore, the additional file sharing applications (*bittorrent* and *john-the-ripper*) are clearly against the network usage policy.

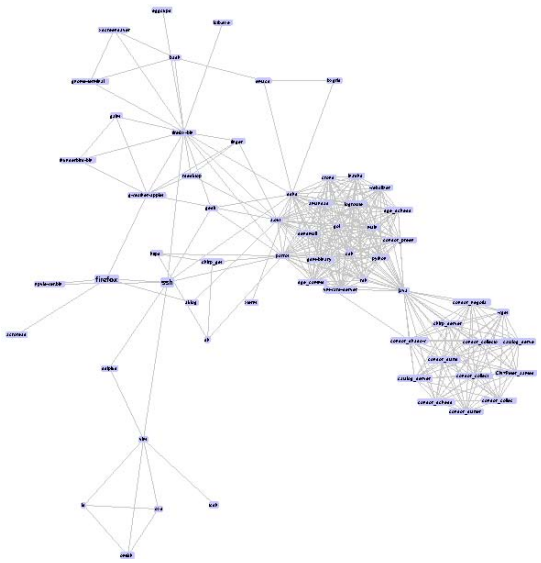
While this example only shows the *user-application* bipartite graph, many other options are available. For example, we can observe the bicliques that involve the *hosts* and *users* and tell which users change usage behaviors by logging onto a different set of machines. Another example would be an *application-host* type of bipartite graph. This type of bipartite community visualization can easily tell the evolution of the set of target machines that each application contacts. This would be helpful to investigate the virus/worm infections, botnet activities or other maliciously abnormal application behaviors. Insights can be derived through observing the evolution of biclique communities. The intelligence behind visualization makes network monitoring a more scientific approach.

3.2.3 Similarity Graphs

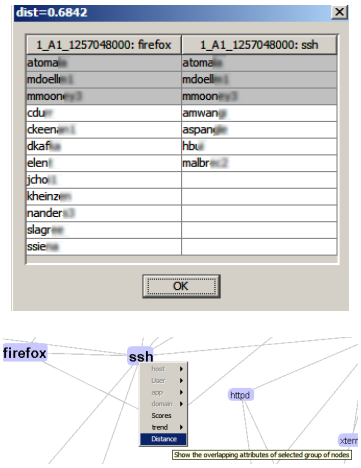
While network connectivity graphs are ideal for visual exploration of what is going on in the network, another interesting type of graphs we can visualize is called *similarity graphs*. Similarity graphs break down the *heterogeneity* by constructing a *homogeneous* graph in a novel and interesting way: edges representing the common properties that any pair of nodes may share. For example, users A and B would be connected if they share the same applications or logged onto the same machines. Note that the edges between nodes are no longer the network connections but the degrees of similarity among them. It is called a similarity graph because we essentially push the similarity level into the edge weights. *ENAVis* performs the visualization of homogeneous similarity graphs in the following manner.

Host graphs: The nodes represent the hosts while the weight of the edge between any pair of host nodes represents one or more of the following:

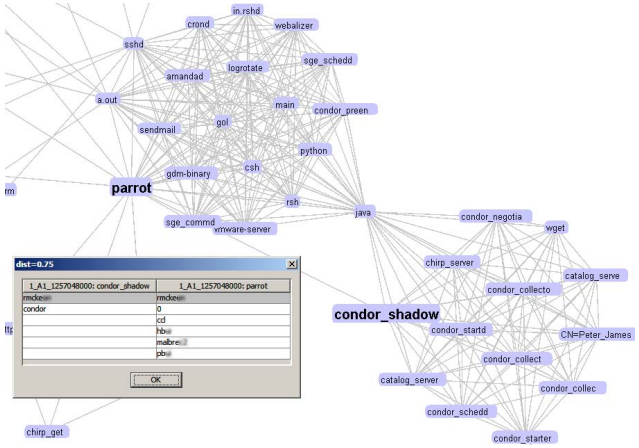
- Hosts: the number of same destination hosts to which the two hosts connected;
- Users (local): the number of same users who logged on and had network activities on both hosts;



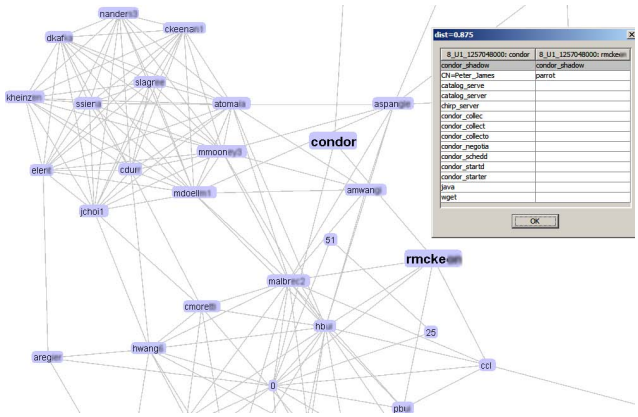
(a) Similarity graph of applications by users



(b) Group selection of application nodes, and highlights of similar users



(c) Bridges between clusters



(d) Similarity graph of users by applications

Figure 10: Visual exploration of clusters of similarity *HUA* graphs provide an interesting alterative view of network graphs and provide insights of how similar and what relationship among hosts, users, or applications.

- Users (remote): the number of same users who connected to both hosts;
- Applications (local): the number of same applications running on the two hosts;
- Applications (remote): the number of same remote applications that connected to both hosts.

User graphs: The nodes represent the users while the weight of the edge between any pair of user nodes represents one or more of the following:

- Applications: the number of same applications run by the two users;
- Hosts (local): the number of same local hosts on which the two users logged and made network connections;
- Hosts (remote): the number of same remote hosts to which the two users contacted.

Application graphs: The nodes represent the applications while the weight of the edge between any pair of application nodes represents one or more of the following:

- Users: the number of same users who ran both applications;
- Hosts (local): the number of same hosts on which both applications ran;
- Hosts (remote): the number of same hosts to which both applications connected.

3.2.4 Case Study

Figure 10 shows examples of similarity graphs, e.g., an application similarity graph with the edges representing the degrees of similarity based on the users who ran those applications (Figure 10(a)). If two applications do not have an edge connecting them, that means they have two disjoint sets of users. The higher the edge weights (and thicker the edges), the more common users they share and therefore the more similar the nodes are. At a zoom-in view (Figure 10(b)), multi-selection is enabled to allow an administrator to select a group of nodes (whose labels automatically highlighted in a bold font) under investigation and to perform

context-aware, on-demand queries. A table view shown on top of the investigated nodes displays all overlapping users (highlighted in grey colors) that the two applications (*SSH* and *firefox*) share. Furthermore, a normalized distance score (e.g., 0.6842) is shown on the table's title. The score is computed based on *edit distance* of two feature vectors to measure the similarity quantitatively among the two selected nodes.

As shown in Figure 10(a), clearly there are well connected clusters. Zoom-in view provided in Figure 10(c) shows the upper-left cluster is the applications mainly run by the *root* users while the lower-right cluster is the applications used by the *condor* users. Interestingly, there are two types of *bridges* connecting these two clusters: one through the edge (*parrot–condor_shadow*), and the other through the node (*java*). A further exploration suggests that the bridging edge between the two applications *parrot–condor_shadow* is made possible by one enterprise user: *rmckexx*, who ran both applications that resulted in network connections. Not only application *parrot* serves as the bridge node between the *root* cluster and the *condor* cluster, *parrot* also serves as the gateway to the rest of graph as it was run by many other users as well.

The converse part of an application similarity graph by users is a user similarity graph by applications (Figure 10(d)), which is a quite different view since instead of measuring how similar of applications, it measures how similar of users. A focus view of such a graph (Figure 10(d)) shows user *rmckexx* and user *condor* share one common application *condor_shadow*. The fully connected upper-left cluster is the *firefox* community, in which all users used *firefox* at least once that connected to somewhere. A further investigation can easily continue by generating a similarity graph of applications by the destination hosts, so what common destinations that applications such as *firefox* connected to can be visualized.

The interactive exploration interface provided by this visualization tool allows an administrator to quickly extract insightful information from his/her network that is otherwise obscure in network connectivity graphs. By removing the heterogeneity of *HUA* graphs (i.e., only homogeneous nodes of either hosts, users, or applications remain) and using the novel graph construction method via pushing similarity degrees into edge weights, it becomes easier to visualize the inter-relationships and similarities among the host, user, or application nodes in network graphs.

3.3 Node Similarity Visualization

In the previous sections, we discussed the similarity visualization on the evolution of between graphs (inter-graphs), and group structures within graphs (intra-graphs). We present the bottom part in the hierarchy of similarity visualization: the dynamics/similarity of each individual node.

The challenging part of the data visual analysis is the highly dynamic properties of the enterprise network graphs. The network components of graphs such as the host, user or application nodes tend to be different at every moment. We need an efficient way to visualize the neighborhood changes over time.

A natural process of analyzing these dynamics is to examine how the nodes' *neighborhood* changes over time. Given a *HUA* graph, the following metrics are used to investigate the dynamic properties of nodes:

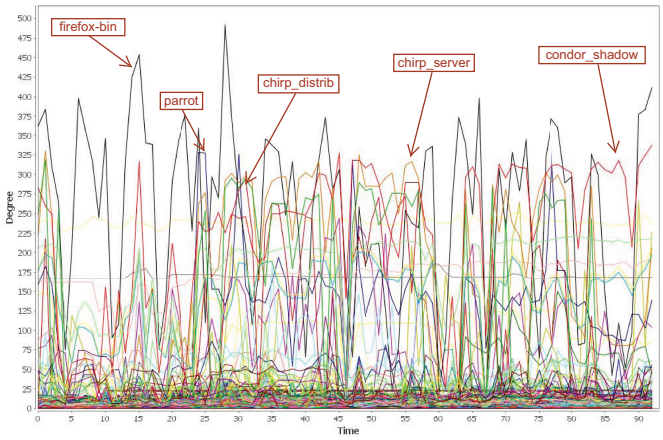


Figure 11: Dynamics of node degrees over a three-month period. Each line represents a node. Applications are among the top nodes having largest standard deviations of degree changes.

- *Hosts*: How does the composition of *users* change over time? How consistent is the set of users logging on the machines?
- *Users*: How does the composition of *hosts* on which they logged change? How does the set of network *applications* they run change over time?
- *Applications*: How does the group of *users* who run those applications change over time? How do the target *machines* contacted by these applications change?

Manual comparison of each element in two sets is time consuming and error prone. Visualization can make this process much quicker and easier. To that end, another visualization function is developed in our tool that can assist a network administrator to quickly browse through each node and easily get the answers for each of the above questions with just a few mouse clicks.

Figures 12 and 13 show screenshots of visualizing the dynamics of nodes. There are four components for each view. First, the upper left is the options for filtering and ordering nodes. Administrators can select whether to view host, user or application nodes, and select whether to sort the nodes by labels or by the dynamic scores (explained later). Second, the lower left part is the list of node names. By clicking on any node in the list, two additional views are rendered automatically on the right panels: a table view (lower right) and a chart view (upper right). For example, if a *host* node in the left list is selected, all *users* that have logged on that host during the past month are shown on the lower right table. The exact date and day-of-week are also listed in the table column headers to help identify any potential usage pattern over weekdays vs. weekends. If the users appear every day during the inspection period, the table cell elements are highlighted in gray color indicating they are overlapping across all sets of nodes. Finally, for easier visualization, a scatter view is plotted (upper right), where each dot of different shape and color represents a unique node of specific type (e.g., user). The x-axis represents the time while the y-axis represents the total number of unique neighbors appeared over the entire time period. Through this visualization, what hosts, users or applications that appear consistently all days

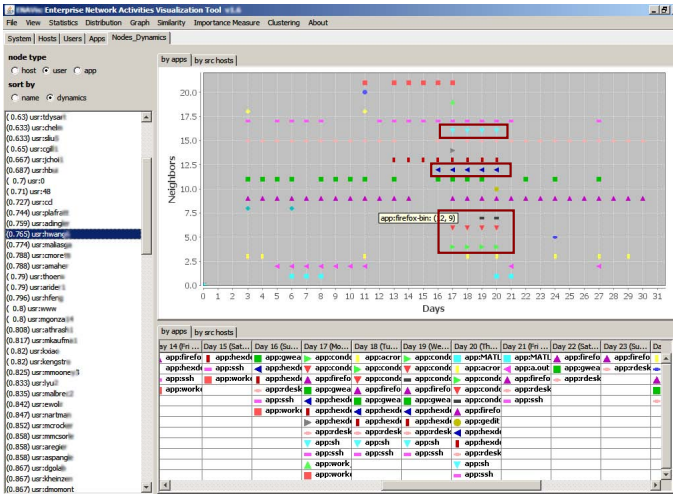


Figure 12: Nodes dynamics/similarity visualization. Selecting a questionable user node (sorted by dynamic scores) in the left list automatically plots all the applications that user ran over an entire month's period. Dots of different shapes and colors represent unique applications. New applications emerged from day 17 to 20 (highlighted).

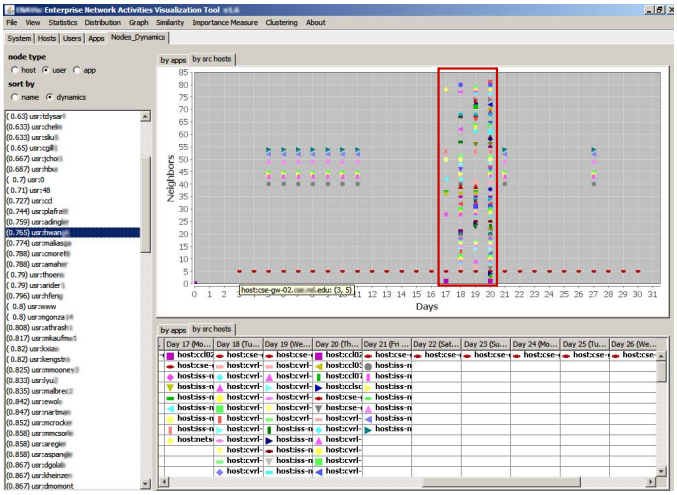


Figure 13: Changes of the source hosts on which the same user have logged and had network activities. The significant increase of the number of different hosts from day 17 to day 20 is caused by the new applications (*condor*, *hexdoku*, etc) highlighted in Figure 12.

are easy to observe as well as the outliers which appear only once or twice over the entire month. Moving the mouse pointer over the dots in the chart reveals the host addresses, user identities or application names. For easy correlation, the shapes and colors used in the charts are also prepended to the identity names in the table view, much like legends.

While the visualization provides a straightforward way of examining how dynamic each node is, in order to quantify the degree of dynamics, a *dynamic score* is computed for each node to rank them in an ascending order in the node list on the left. By this way, a busy administrator can simply rank the nodes by the scores and look at the top dynamic nodes for possible anomalies. The dynamic scores of nodes are normalized as $DS(n) = 1 - \frac{N_a}{N_u * T}$, where N_u is the number of unique labels of neighbors of node n spanning over the entire time T (e.g., days) and N_a is the actual appearance of all nodes over T.

3.3.1 Case Study

We start by visualizing the dynamics of node degrees across a three-month period in a regular fall semester (Sep.1 - Dec. 1, 2009), as shown in Figure 11. The rest experiments in this section are consistently based on the same data. Each line in the chart represents one unique node. While most nodes (at bottom) have small variations as expected, there exist a few nodes exhibiting highly dynamic behaviors. These nodes mainly belong to web browsers and *condor*-related applications such as *parrot* and *chirp*. For example, *firefox-bin* can have as high as 500 neighbors on one day and have only 75 neighbors on some other day. The dynamics were mainly due to the number of unique external domains that the *firefox-bin* had visited on each day. A distributed system like *condor* [20] is also highly dynamic due to the nature of the scientific computation jobs being queued, scheduled and dispatched to run on a pool of cooperative research computing nodes. For example, application *condor_shadow* was run by only three users and connected to 19 destination hosts, but on the next day 11/4/2009 (Wed), it had six users and connect to as many as 290 hosts covering nodes from *ccl*, *cvrl*, *iss*, *sc0*, *helios*, *cheg*, and *netscxxx* machines.

Figure 12 shows the change of the set of applications that users launched to make network connections over one month's period. The view clearly shows the patterns for each user's activities over time. Among the most used applications by a selected user *hwanxx* are *firefox-bin*, *gweather-applet*, and *ssh*. Some applications used for a short period of time (day 17, Mon, Nov 30,2009 to day 20, Thu, Dec 03, 2009) are batch job related, i.e., *condor_exec*, *condor_shadow*, *sh*, *hexdoku*. Similarly, Figure 13 shows the source hosts on which the same user *hwanxx* has logged. The user *hwanxx*'s office computer *cse-gw-02.domain.edu* is consistently listed everyday. Suspiciously, from day 17 to day 20 there are much more number of source hosts on which *hwanxx* had run programs than any other days. The anomaly indicates further investigation. By comparing Figure 12 and 13, there is reason to believe the new applications (*condor_exec*, *condor_shadow*, *sh*, *hexdoku*) on days 17 to 20 caused the abnormal activities. Examining the original log data files further confirm this. Further visual exploration on the changes of the users who ran the application (*LifeParallel*) shows this application has been run by the *root* user consistently, except by one user *psempoxx* on day 2 and 3. Three target machines to which the same application have contacted are consistent over time except on day 2 and 3, which coincides the time frame with the new user *psempoxx*. It is verified from the data that the new additional target hosts were caused by this user. These above examples show that how such a simple visualization for the node dynamics can help human investigators quickly browse through each node, find activity patterns, detect potential anomalies, and find the underlying correlations of activities and causes.

4. RELATED WORK

Visualization is useful for enterprise network management and security [21–23]. However, existing solutions have involved tie-ins of network flow data or packet analysis. For example, src/dst IPs and ports can be analyzed in 2D scatter plots (PortVis [24]), 3D scatter plots (InetVis [25]), and stacked histograms [26]. Fields of packet headers can be visualized from libpcap data (TNV [1], Rumint [2], etc). Critically, these existing systems are not geared towards real-world system administration. Packet-based and network flow data will only detail the *where* of a connection, but not *who* (users) and *what* (applications). Out of existing visualization and data exploration tools (NetFlow Visualizer [3], NFlowVis [4], ISIS [5], NVisionIP [6] and VisFlowConnect-IP [7]), they primarily rely on chaining together network connections based on the NetFlow or sFlow data. However, multiple hop connections are typically obfuscated due to the nature of network flows; the level of detail supplied is traditionally limited to the IP addresses and port numbers involved. As stated earlier, the key weakness of packet/flow data is the missing *user* and *application* information, which is critical for enterprise network management [8–10].

The network data can be naturally converted into graphs. Graph-based network traffic visualization [27] has been used to monitor host behavior. Detection of intrusion and network anomalies can be analyzed and visualized through graph drawing and graph clustering [28], pixel luminance based histographs (IDGraphs [29]), or animated glyphs [30]. While some of graphs can be replayed via animation, they are still statically generated and both interactive exploration and intelligent analysis modules are missing. The proposed visualization intends to provide insight through both automatic intelligent analysis and manual interactive visual exploration and queries to understand host, user and application behaviors. Additionally, administrators usually do not know what type of anomaly expected to see, making definition in advance impractical. Rather than focusing on specific threat model based on the magnitude of attack (e.g., port scanning, SYN flood, worm outbreaks, etc.), we seek a general visualization framework with no restriction in any specific anomaly or thread model on enterprise network graphs that can convey important knowledge of similarity/difference patterns of network activities.

In this paper, we focus on one important aspect of intelligent visualization, namely the similarity and dynamics visualization in the context of heterogeneous graphs involving hosts, users and applications. In a primitive stage, ASCII-based visualization [31] uses a series of different symbols such as '.', '+', to indicate the up/down states of IP addresses for efficiency consideration. To quantify network changes, graph edit distance [13] has been suggested to measure the topological evolution. We illustrate how patterns of similarities can be effectively visualized on a hierarchical structure ranging from inter-graphs clustering to intra-graphs clustering to dynamics visualization on individual nodes. Most importantly, how network connectivity graphs can and should be transformed into various forms in order to achieve this goal.

Finally, tools such as Cytoscape, Pajek, Gephi, and Titan/VTK exist for general graph visualization and analysis. While these programs are designed to visualize and analyze the general networks, there exist several differences. First, we focus on the dynamics of enterprise networks with heterogeneous graphs involving hosts, users and applications. Many generic graph algorithms would fail to produce meaningful results in this setting. Our tool is tailored toward solving practical network security and management problems faced by busy administrators. For example, the above general graph visualization tools will not show the evolution of clusters by comparing different graphs and cleverly suggest which changes are abnormal and need further investigation. Second, we provide *dynamic* visualization in which the interactive exploration part (e.g., by clicking on nodes, performing query on demand, etc.) is essential to pin down the root cause of anomaly quickly. Static graph plots and visualization fall short in this case. Lastly, although Pajek and Cytoscape have extensive graph analytic functions, the analysis strictly focuses on *pure* graph properties and topological information while neglecting node labels. In the enterprise network settings, nodes are uniquely identified by either their network addresses, user IDs or process names. Since there are also many ways to construct graphs, what are the most suitable ways of transforming and interpreting various types of graphs needs careful study. Therefore, general graph visualization is not enough in this case.

5. CONCLUSION

Traditionally, the network monitoring and visualization consists of end-to-end flow or per packet information that involves primarily IP addresses and/or port numbers. In enterprise network setting, two other important and most dynamic network components, i.e., the enterprise users and applications, provide a rich set of local context information associated with each network flow. The increasing dimensions, complexity, dynamics, and causality inter-relationships create a challenge in network visualization and analysis. In this paper, we study how to effectively visualize the network activities that involve hosts, users and applications. We demonstrate the important role that similarities/distance metrics and visualization can play in our understanding of networks and practical network management and security problems. Novel ways of transforming context data into different types of graphs focusing on network connectivity, bipartite, multidimensional scaling and similarity graphs are also discussed. Through a hierarchal structure of similarity visualization ranging from inter-graph, intra-graph clustering to node dynamics visualization, this visual analytical framework provides significant insight to researchers, network operators and administrators.

6. REFERENCES

[1] J.R. Goodall, W.G. Lutters, P. Rheingans, and A. Komlodi. Focusing on context in network traffic analysis. *IEEE Computer Graphics and Applications*, 26(2):72–80, March/April 2006.

[2] Gregory Conti. Rumint – open source network and security visualization tool. `http://www.rumint.org`.

[3] P. Minarik and T. Dymacek. Netflow data visualization based on graphs. In *Proc. of 5th International Workshop on Visualization for Computer Security (VizSec'08)*, pages 144–151, Cambridge, MA, September 15 2008.

[4] F. Fischer, F. Mansmann, D.A. Keim, S. Pietzko, and M. Waldvogel. Large-scale network monitoring for visual analysis of attacks. In *Proc. of 5th International*

Workshop on Visualization for Computer Security (VizSec'08), pages 111–118, Cambridge, MA, September 15 2008.

[5] D. Phan, J. Gerth, M. Lee, A. Paepcke, and T. Winograd. Visual analysis of network flow data with timelines and event plots. In *Proc. of Workshop on Visualization for Computer Security (VizSEC '07)*, pages 85–99, Sacramento, CA, Octoboer 29 2007.

[6] K. Lakkaraju, W. Yurcik, and A.J. Lee. NVisionIP: netflow visualizations of system state for security situational awareness. In *Proc. of the 2004 ACM workshop on Visualization and data mining for computer security (VizSEC/DMSEC)*, pages 65–72, New York, NY, 2004.

[7] W. Yurcik. Visualizing netflows for security at line speed: The SIFT tool suite. In *Proc. of 19th Large Installation System Administration Conference (LISA '05)*, page 16, San Diego, CA, December 4-9 2005.

[8] P. Hertzog. Visualizations to improve reactivity towards security incidents inside corporate networks. In *Proc. of the 3rd international workshop on Visualization for computer security (VizSec '06)*, pages 95–102, Alexandria, Virginia, November 3 2006.

[9] D. Lalanne, E. Bertini, P. Hertzog, and P. Bados. Visual analysis of corporate network intelligence: Abstracting and reasoning on yesterdays for acting today. In *Proc. of Workshop on Visualization for Computer Security (VizSec'07)*, pages 115–130, Sacramento, CA, October 29 2007.

[10] Q. Liao, A. Blaich, A. Striegel, and D. Thain. ENAVis: Enterprise network activities visualization. In *Proc. of the USENIX 22nd Large Installation System Administration Conference (LISA '08)*, pages 59–74, San Diego, CA, November 9-14 2008.

[11] A. Schenker, M. Last, H. Bunke, and A. Kandel. Comparison of distance measures for graph-based clustering of documents. In *Graph Based Representations in Pattern Recognition, Springer LNCS*, volume 2726, pages 187–263, 2003.

[12] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3-4):255–259, March 1998.

[13] H. Bunke, P.J. Dickinson, M. Kraetzl, and W.D. Wallis. *A Graph-Theoretic Approach to Enterprise Network Dynamics (Progress in Computer Science and Applied Logic (PCS))*, volume 24. A Birkhäuser Boston book, 2007.

[14] T.F. Cox and M.A.A. Cox. *Multidimensional Scaling, Second Edition.* Chapman & Hall/CRC, September 2000.

[15] H. Bunke, P. Dickinson, A. Humm, Ch. Irniger, and M. Kraetzl. *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52, chapter Graph Sequence Visualisation and its Application to Computer Network Monitoring and Abnormal Event Detection, pages 227–245. Springer Berlin / Heidelberg, 2007.

[16] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[17] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010.

[18] W.M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, Dec. 1971.

[19] S. Lehmann, M. Schwartz, and L. K. Hansen. Biclique communities. *Physical Review E*, 78(1):016108, 2008.

[20] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: The condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, February-April 2005.

[21] R. Marty. *Applied Security Visualization.* Addison Wesley Professional, 2008.

[22] R. McRee. Security visualization: What you don't see can hurt you. *Information Systems Security Association (ISSA) Journal*, June 2008.

[23] D. Schweitzer and W. Brown. Using visualization to teach security. *Journal of Computing Sciences in Colleges*, 24(5):143–150, May 2009.

[24] J. McPherson, K. Ma, P. Krystosk, T. Bartoletti, and M. Christensen. Portvis: a tool for port-based detection of security events. In *Proc. of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSEC'04)*, pages 73–81, Washington DC, 2004.

[25] B.V.W. Irwin and J.P. Riel. Inetvis: a graphical aid for the detection and visualisation of network scans. In *Proc. of Workshop on Visualisation for Computer Security (VizSec'07)*, Sacramento, CA, October 29 2007.

[26] K. Abdullah, C. Lee, G. Conti, and J.A. Copeland. Visualizing network data for intrusion detection. In *Proc. of the 2002 IEEE Workshop on Information Assurance and Security*, pages 30–38, United States Military Academy, West Point, NY, June 17-19 2002.

[27] F. Mansmann, L. Meier, and D. Keim. Graph-based monitoring of host behavior for network security. In *Proc. of Workshop on Visualization for Computer Security (VizSec'07)*, pages 187–202, Sacramento, CA, October 29 2007.

[28] J. Tolle and O. Niggemann. Supporting intrusion detection by graph clustering and graph drawing. In *Proc. of RAID 2000 Third International Workshop on the Recent Advances in Intrusion Detection*, pages 51–62, Toulouse, France, October 2-4 2000.

[29] P. Ren, Y. Gao, Z. Li, Y. Chen, and B. Watson. Idgraphs: Intrusion detection and analysis using histographs. In *Proc. of the IEEE Workshops on Visualization for Computer Security (VizSec'05)*, Minneapolis, MN, October 26 2005.

[30] R.F. Erbacher, K.L. Walker, and D.A. Frincke. Intrusion and misuse detection in large-scale systems. *IEEE Computer Graphics and Applications*, 22(1):38–47, Jan/Feb 2002.

[31] A. Stewart. Efficient visualization of change events in enterprise networks. In *Proc. of IEEE Workshop on Enterprise Network Security*, pages 1–6, Baltimore, MD, Aug.28-Sep.1 2006.