Computer Networks 54 (2010) 2809–2824

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Managing networks through context: Graph visualization and exploration

Qi Liao *, Andrew Blaich, Dirk VanBruggen, Aaron Striegel

Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, United States

ARTICLE INFO

Article history:
Available online 10 August 2010

Keywords:
Enterprise network management
Security
Visualization
Context
Graphs
Visual mining
Interactive exploration
Forensics

ABSTRACT

With the increasing prevalence of multi-user environments in distributed systems, it has become an increasingly challenging task to precisely identify *who* is doing *what* on an enterprise network. Current management systems that rely on inference for user identity and application are not capable of accurately reporting and managing a large-scale network due to the coarseness of the collected data or scaling of the collection mechanism. We propose a system that focuses data collection in the form of local context, i.e. the precise user and application associated with a network connection. Through the use of dynamic correlation and novel graph modeling, we developed a visualization tool called *ENAVis* (the work appeared in earlier form in [1] and received USENIX best paper award). (Enterprise Network Activities Visualization). *ENAVis* aids a real-world administrator in allowing them to more efficiently manage and gain insight about the connectivity between *hosts*, *users*, *applications* and *data access* offering significant streamlining of the management process.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Distributed systems are inherently complex and hard to manage due to the large scale of inter-related heterogeneous system components. In addition to hosts, other components such as users, applications and data also play an important role in the overall picture of network activities. In contrast to traditional point-to-point network flow monitoring, these user/application components are highly dynamic. The sheer number of network connections in large scale enterprise networks form ever evolving interdependency graphs. Tracking down precisely *who* (users) and *what* (applications) are responsible for the generation of this network connectivity is a non-trivial task. The causes for this problem are due to the specific data not being available to capture the user and application level of network activities and/or the inability to capture the interrelationships of such data that can bridge daily network monitoring and high level decision making [2].

Understanding one's own network, such as knowing *what is going on in the network*, is usually the first step towards improved network management. Despite the abundant amount of data available from point-to-point logging, the coarseness of the data does not make it particularly useful. For example, current logging schemes such as Cisco NetFlow data provide activity details in terms of IP addresses and ports, but are unable to tell which *users* and what *applications* are running on the managed network. In a multi-user environment with increasing port dynamics of distributed applications, the many-to-many relationships between IPs and users and also between ports and applications make network addresses and port numbers poor identifiers for network activities [3]. While host or network intrusion detection systems (HIDS/NIDS) are useful as a first step to identify attacks, these alerts normally need additional context data for further analysis [4]. Since it is the actual applications run by users that are the most active components in networks, we argue that the right context of network connections (i.e. which users and what applications are responsible, what data they access, etc.) is essential for network monitoring.

Context monitoring nevertheless increases the complexity of network dependency graphs due to the dynamic interactions between users and applications as well as the scale of the data collected. The key challenge becomes how

* Corresponding author. Tel.: +1 5746318720; fax: +1 5746319260.
E-mail address: qliao@nd.edu (Q. Liao).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.07.015

the value of context can efficiently help us gain insight and knowledge. With the complexity of the context data, 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 rather than hindering it. In the network management domain, however, the gap between the daily network monitoring and the high level of decision making is currently not well bridged by any interactive tool [2]. Smart visual analysis is the key to solving the above problem. A properly designed visual representation and human–computer interaction can expedite data understanding and improve the exploration process [5,6].

To facilitate solving the above problems, we present *ENAVis* (Enterprise Network Activities Visualization). *ENAVis* is a tool for visualizing the network activities among hosts/domains, users and applications, which is possible through the gathering of *local context* information. *ENAVis* offers interesting, ready-to-use, and invaluable functions for monitoring, correlating, exploring, visualizing and analyzing the activities on a network by real-world network administrators. Through the use of a highly detailed local context data collection system spanning over a wide diversity of machines on our campus since 2007, we have collected terabytes of context information and developed *ENAVis* to allow an administrator to explore this informative data set.

With *ENAVis*, the administrator is presented with a wealth of user and application connectivity graphs, statistical charts and reports on how the network is being used. To assist the investigator in understanding the many possible visualization modes, we provide a novel *meta-visualization* which compactly represents and controls how data is represented. Unlike most flow visualizations which are only capable of plotting the entire static host-only graphs, *ENAVis* allows the user to easily expand, contract, and explore a very rich data space in a visually appealing and highly interactive manner via the Host-User-Application-File (HUAF) interactions.

In addition, being able to *interactively explore* the causal relationships of the user and system events, i.e. gaining insight on what is occurring, is also important. For example, if an account on a network is compromised then it needs to be known what hosts that user account attempted to log into, along with the applications and programs they attempted to run, and files that may have been modified or touched. Knowing exactly *who* (users) and *what* (applications), not inferring from IP and port, at *both* sides of connections is of particular interest in policy compliance auditing. Being able to present all of this information in a single visually appealing and manageable view would be a tremendous asset for network administrators. The key highlights of this paper include:

- *Data collection*: The light-weight, easy-to-deploy monitoring agent collects the *local context* information (who, what, when, and where) associated with each network connection in an enterprise network.
- *Graph model*: Our novel hierarchical graph representation of context data in terms of domain/hosts, users, applications and files (HUAF) captures the dynamic relationship and interaction between machines, user applications and data flows.
- *Visualization and interactive exploration*: Rather than static graph visualization, an easy-to-use yet powerful graphical interface makes exploration of large amounts of network connectivity interactive and manageable and helps administrators quickly drill down to the root cause of connectivity and security problems.
- *Visual mining*: Clever combination of interactive visual exploration with automatic data mining, machine learning algorithms and statistical graph theory bridges day-to-day network monitoring and high level decision making. The significant improvement over our prior work [1] includes a new, intelligent module for visual mining and graph analysis, streamlining the visual exploration, and the inclusion of files to context information in addition to hosts, users, and applications. Most significantly, the data mining algorithms and graph theory incorporated in *ENAVis* facilitate and *guide* the manual visualization process to provide more insight and detect the root cause of network management problems more efficiently.

2. Objectives

It is usually a good practice for administrators to log system events and network activities [7]. However, the large amount of data accumulated each day is difficult for human beings to understand and explore. Visualization is therefore an important topic in network management and system adminstration since visualization can not only ease the manual process of going through log data and correlating events but can also present the relationships in a meaningful and easy-to-understand way. Therefore, the objectives of *ENAVis* are threefold. First, *ENAVis* should correlate and visualize the events and the local context (hosts, users, applications, data accesses, etc.) of network connections. The tool should also plot various combinations of the feature/attribute vectors in the log data for a quick overview of the network activities within the enterprise network. Second, the visualization tool provides a customizable and interactive interface for human auditors to explore and investigate the activities that occurred on their networks. The exploration feature of *ENAVis* would allow the investigator to easily drill down to the root cause of abnormal or suspicious activities. Most importantly, a unique inter-hosts/users/processes matching capability included in *ENAVis* provides the administrator with intuitive information on the dependant relationships, which may help many other important problems such as security tracing and fault localization. Third, an intelligent graph data mining module should ideally be built into *ENAVis* that can automate part of the manual examination process and guide the human investigator to look at only things of most interest and consequently facilitate his decision making.

2.1. Problem statement and solution

There are two problems that we tackle in this paper. First, there is a lack of tools and data to capture the user and application level of network activities. Second, there is also a lack of interactive tools to visualize and capture

the inter-relationships of such data that can bridge daily network monitoring and high level decision making [2].

In addressing the first problem, administrators do not usually lack log data for security measurement [8]. However, administrators are facing a dilemma that on one side there is an overwhelming amount of data, but on the other side much of the data is not at the level of detail administrators would like. Although there are tools to log network activities in either packet or flow format, there is no lightweight mechanism in current practice to monitor the network at a finer granularity than host-to-host. For example, the network IP addresses included in the packet header only serve as *locators* for the machines. They tell nothing about the *identities* of the end-users. On the other hand, the transport layer's port numbers are also less meaningful in determining the actual end-processes [3]. Using deep packet inspection (DPI) [9] requires an understanding of all known protocols, but this expensive practice is still of little help because which users and applications are sending those data still remains unknown.

Motivated by the observation that the end host has full *visibility* of the user's processes, our approach to the first problem is to deploy a simple agent on the end hosts to collect the missing local context data for each network connection. The agent is easy to deploy and lightweight in that it is a `bash` script that calls commonly available system tools such as `netstat` and `ps` and requires absolutely no changes to the underlying system. Through careful mapping between each TCP/UDP socket with the user ID and process ID, we associate users and applications with each network connection. The data is then sent securely via `scp` from each host to a central database server for correlation, analysis and audit.

The second problem, independent of the data collection mechanism, is how to understand and interpret the data. The natural question to ask is how should we correlate and visualize the rich context information associated with each network connection in a more intuitive and easy-to-understand manner? With the amount of workload on a busy system administrator, being able to quickly browse through the data, view summary statistics and charts, and interact with connectivity graphs can be very helpful.

Visualization is the key to solve the second problem, which is the focus of this paper. It is commonly recognized that many of the human errors are due to the lack of domain knowledge [10]. A properly designed human–computer interaction can expedite data understanding and improve knowledge and insight. Our solution is to develop a powerful yet friendly graphic user interface that 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 design principles of our system are described in the next section.

2.2. Design principles

The target of the system, namely what is to be achieved by this tool, is detailed below:

- *Know who, what, when and where (4 W)*: Know what is happening on the network, i.e. who (which users) are running what (applications) on where (which hosts) at when (what time). Context information relevant to the connection needs to be recorded.
- *Compute, trace, and visualize heterogeneous graphs*: In order to visualize the 4W aspects of the data, the tool needs to transform the raw data into an animated graph topology view. The graph is considered *heterogenous* because each node in the graph can be either a domain, host, user, application or data. Fig. 1 shows an example of such a graph.
- *Investigate interactively*: Static graph visualization is less flexible and desirable in network management. Based on user events (such as clicking, querying or applying filtering rules), the graph is instantly regenerated to reflect the changes. Through only a few mouse operations, the administrator is able to make queries to the database, Domain Name System (DNS), and Lightweight Directory Access Protocol (LDAP) servers for more detailed information, analogous to "please tell me more about this".
- *Make it simple, efficient and customizable*: The tool should be simple yet powerful, usable for real-world administrators. While most users will not need to modify the base set of views, the ability to customize via a modular viewer is a powerful feature. Ideally, users would be able to customize their configuration and build an environment in which they are most interested (e.g. top 10 applications, current connectivity of human resource (HR) users, status of grid compute nodes, etc.).
- *Make it intelligent*: To delineate with traditional visualization techniques, this visualization tool will not simply try to "visualize" the data. It should be intelligent meaning the tool can automatically "learn" and "guide" the human operator to visually explore only things to which attention should be paid. We will bring visualization and data mining together into the daily network monitoring and management practice. One example would be to build decision trees to classify network events, or compute and visualize clusters for understanding similar behaviors by users or applications and identifying potential problems.

Fig. 2 shows how *ENAVis* brings the above parts together to form a powerful tool for administrators. In summary, the design objective of the framework of *ENAVis* is to seek clever ways to collect, correlate, visualize, explore and automatically analyze the dynamic interactions among important network components, i.e. hosts, users, applications and files that can maximize the insight for network operators and the management team.

3. Network context graphs

This section describes the first two important components of the system, i.e. local context data collection and novel network graph model. First, we introduce and define the meaning of *local context* followed by a description of what type of data we have been collecting and an overview

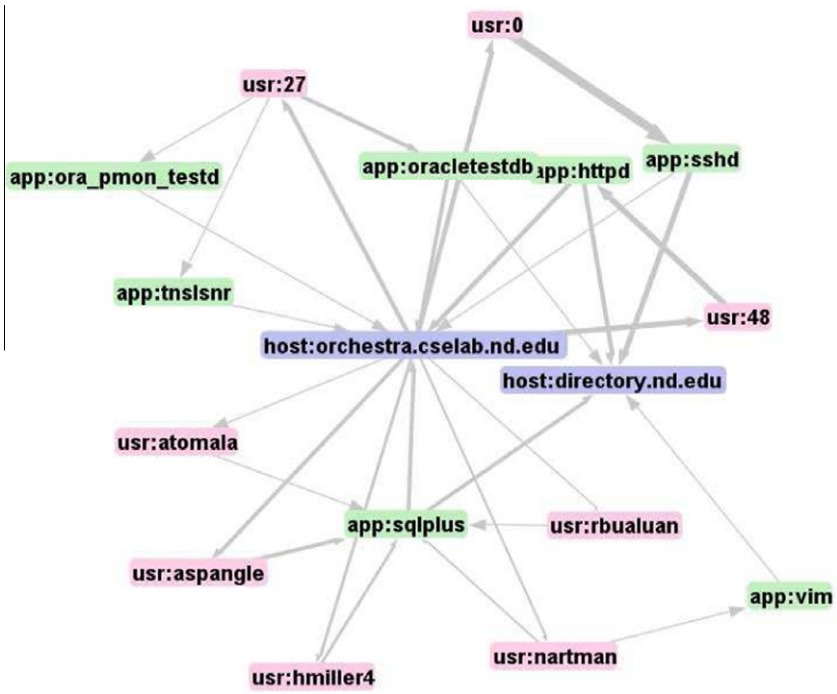


Fig. 1. A screenshot from *ENAVis* showing connectivity between hosts, users, and applications.

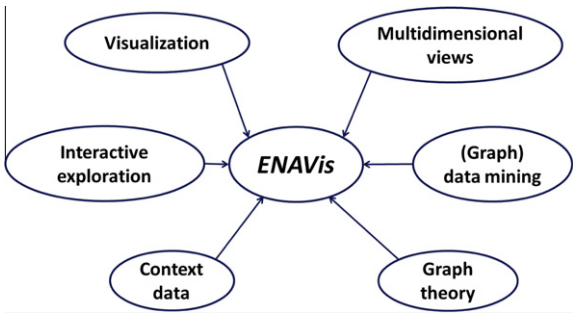


Fig. 2. Overview of the components of *ENAVis* that performs visual analysis on the context data.

of the entire system. As discussed earlier in Section 2, the first problem we are trying to solve is how to collect the missing context information, i.e. to capture the user and application level of network activities (4W). The system we propose ties the user and application identities into the enterprise network management by utilizing existing tools (`netstat`, `ps`, `lsof`), which together build a hierarchical gathering of local context related to network connectivity.

Second, also in this section, we lay out the theoretical foundation for the graph representations of the data we collected. We make a unique contribution using a heterogenous graph model that involves mappings between hosts, users and applications (HUA). This interesting graph model can have applications in the area of enterprise network management, security, auditing, problem debugging and fault localization.

3.1. A hierarchy for gathering local context

We now briefly describe the three major tools used in our data gathering system, what each supplies, and how the supplied information can be fused together to provide a complete view of the local context associated with each network connection. The *local context* is defined as the information fully detailing a network connection (protocol, src/dst IP/port), time, user, application, application arguments, and network-related file accesses. While the above definition is the "standard" version of the local context, the powerfulness of the idea of local context lies on the easy expandability. It is very important to note that the local context associated with each network connection can be extended to include *any additional context information that is relevant* to the network connection depending on the changing needs of organizations.

The data gathering component utilizes commonly available tools in order to take advantage of development robustness and administrator familiarity. The tools should augment the existing data significantly, i.e. not just another method to report *IPflows* or Simple Network Management Protocol (*SNMP*) data. A natural fit for these criterion is the `netstat` tool, in essence the equivalent of *whois* for network connectivity. In the base tier, `netstat` is the most important command utilized to capture each instance of network connectivity occurring on the monitored system. In comparison to the standard rules in the firewall, `netstat` provides the additional user and process ID associated with the network sockets as well as similar information with regards to the connection tuple (protocol, src/dst IP and src/dst port). `netstat` can be coupled with other

tools such as the process table via `ps` (linking process ID to the application and arguments). As the second tier command, `ps` is used to supplement the application information from `netstat` as well as the process tree. Moreover, the context information can be further correlated with the open file handles via the tier 3 command (linking the application to files). The most important insight offered by the `lsof` command is regarding potential information flow, i.e. what files a connected process is touching. Another interesting aspect of `lsof` is the discernment of an application's location. From a policy management standpoint, centrally served (e.g. Network/Andrew File System (NFS/AFS) mount) or validated local versions (e.g. MD5, SHA1 hash) can reduce the ambiguity associated with applications. The notion of classifying according to *application location* can offer an additional mechanism for extracting characteristics such as versions of applications. In a broad sense, one could view applications as existing in one of three forms, user local (local directory or user path), machine local (root-level install, e.g./usr/bin), and enterprise served (root-level mounted). The concept of the fusion of the data forming the context information for each network connection conceptually shown in Fig. 3.

The data collection agent was implemented as a `bash` script that calls Unix commands `netstat`, `ps`, `lsof`, and `diff` on the current and previous set of outputs at a customized interval (T_{diff} = 5 s in our case for a good balance of granularity and system overhead) and every T_{out} = 15 - min pushes out the collected data securely to the central server. In addition, when the agent component initializes for the first time, it collects an array of system-wide information such as OS version, iptables rules, network interfaces and other hardware info. The system has been deployed throughout our campus with a mix of faculty and student office computers, scientific grid computing nodes, and engineering lab machines since April 2007. The cost of agent deployment is minimal in terms of CPU, file size and bandwidth [1]. It is important to note that the lightweight nature of the system comes from the fact that it provides local *context* with regards to the presence of connectivity (network and files), not the *content* passed in the connectivity itself (data payloads, packet headers, etc.). The benefit of implementing the agent as a script is its immediate deployability without any special changes to the network or hosts. While it is understood that the data collected in this polling scheme may not be perfect and could miss some transient events such as TCP connection state changes or file system accesses, it is possible that the reports from the end hosts can be combined with the NetFlow data for more accuracy in connection time, direction, packet size, etc. Moreover, we note that the full visibility at the end hosts provides a richer context (in terms of users and applications) of network connectivity that is not readily available from inline monitoring.

3.2. Heterogenous graph model (HUA control)

With *ENAVis*, the administrator is presented with a variety of choices for connectivity graphs. While we have a rich

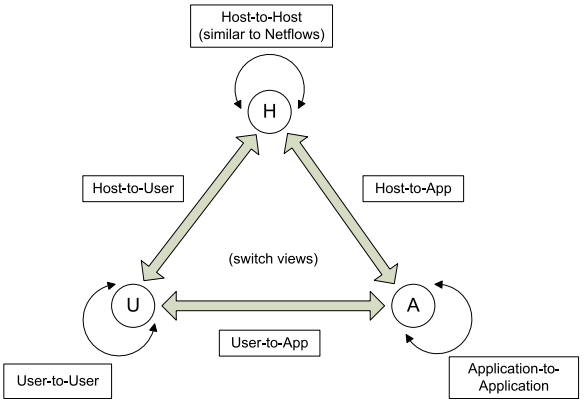
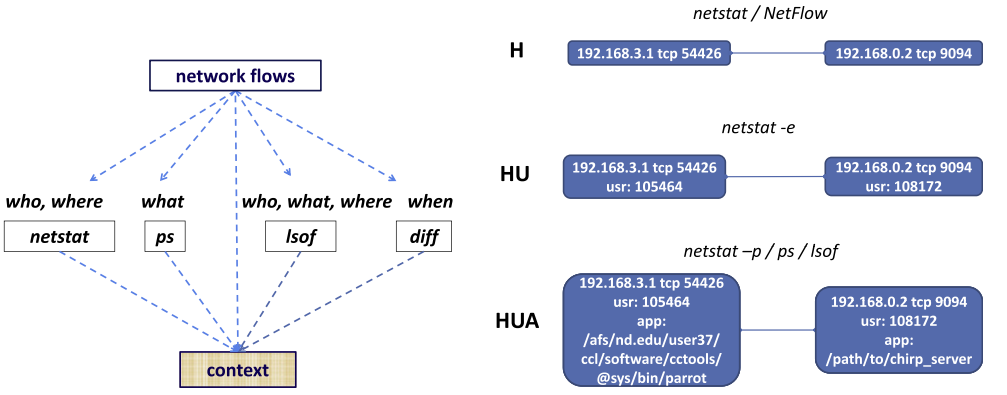


Fig. 4. Meta graph showing state transitions among *H*osts, *U*sers, and *A*pplications (HUA) for modeling and constructing network connectivity graphs. Being able to visualize the various combinations of HUA is one of contributions of *ENAVis*.



(a) Forming 4W context (b) An example

Fig. 3. A hierarchical structure of local *context* (users, applications, data, etc.) associated with each IP/port pair from network connections (flows) can be readily achieved through correlating a set of widely available, robust (and free) system tools.

pool of data containing the entirety of host, user, and application connectivity, it is not always desirable to view all the data at once. To assist the investigator in understanding the many possible visualization modes, we propose a novel *meta-graph* visualization, which compactly represents and controls how data is represented. By adjusting the Host-User-Application (HUA) control (Fig. 4), the user may easily expand, contract, and explore a very rich data space in a visually appealing and highly interactive manner. We can imagine a 4D space, where the time, host, user and application interact with each other. The administrator may quickly switch between views and filter out any irrelevant information in a customizable level of granularity. A concrete example of the HUA graph model is illustrated in Fig. 5.

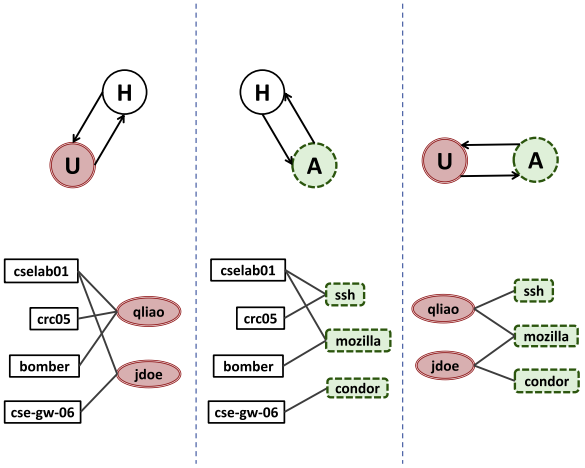


Fig. 5. Examples involving only two network components (i.e. biparite graphs): HU, HA and UA.

At the top layer in Fig. 4, we have *H* denoting the *host* connectivity, basically described by traditional IP/port pairs among servers and clients. Using *H* only is analogous to a connectivity view offered by NetFlow data. At the middle layer, we have *U* denoting the *user* connectivity chaining, in which we can observe the connectivity relationships solely among the users. Because there are many-to-many relationship between hosts and users, namely multiple users can log onto the same machine and a single user can log onto multiple machines, by treating an enterprise user (no matter how many physical hosts they have logged on) as one single entity node, we are able to observe the overall network activities for any specific user. At the bottom layer, we have *A* denoting the *application* connectivity, in which we can observe the connectivity relationships solely among applications. A simple example would be which browsers are interact-

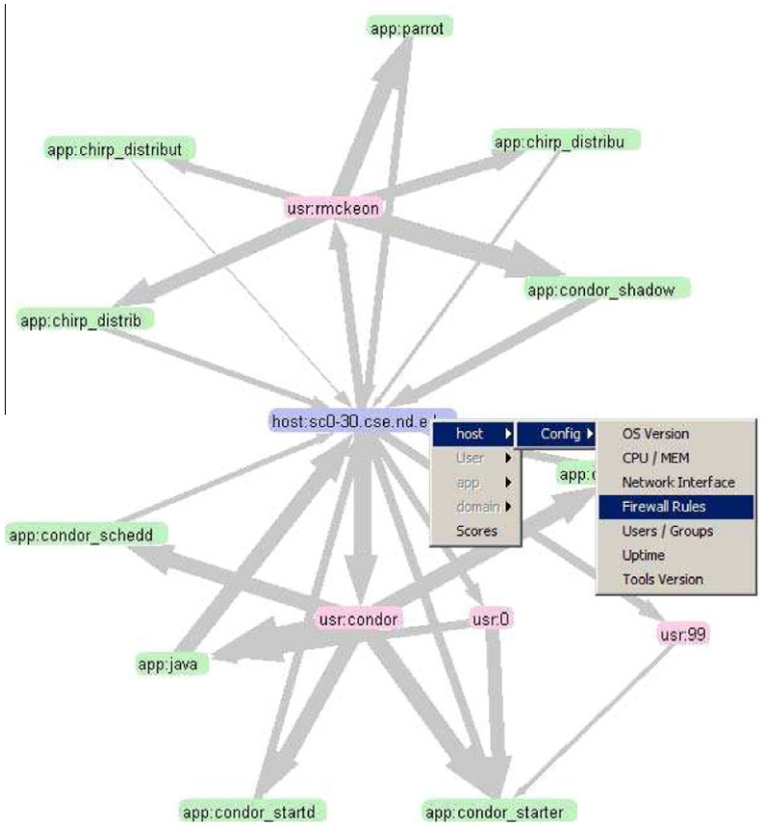


Fig. 6. Popup implemented to display node properties and to provide detail-on-demand and "please tell me more" function by querying database.

ing on my intranet web server (i.e. Firefox 2.0, Internet Explorer 7, etc.) without worrying about user-agent spoofing. Similarly, what applications (and their versions) are checking out licenses from my license server? As stated earlier, since an user can log onto multiple hosts and a host has multiple users simultaneously logged in, a mix-mode (*HU*) of interaction between users and hosts can provide insights on *who* is responsible for the traffic. An edge is added between the user and the host only if that user has made at least one connection on that host. The user is then connected with another user on another host. Similarly, with *HA*, hosts and applications can be combined to construct a connectivity graph when users are of less concern. Also, *H* can be temporarily filtered out and only leave *UA* if we are more interested in who (users) and what (applications) are running on the network and less concerned about the physical location of hosts. Lastly, with *HUA*, building hosts, users and applications into one graph provides the most comprehensive view as we show in later case studies.

Beyond HUA, the *local context* can be extended to include *any* relevant information associated with each network connection. An interesting consideration is the *data files* accessed by networked applications. Fig. 7 conceptually illustrates an augmented graph containing hosts, users, applications and files (HUAF), in which normal and malicious users can be visually identified by comparing the data access patterns on any given host. This is especially useful for those organizations that care most about their data flows and sensitive data access across networks such as financial and credit card information, classified documents, etc.

With the HUA graph model, the identity of parties at both ends of network connection can be linked together. The motivation for doing user and application level chaining comes from the question: *what are the foreign applications and users behind the other side of the connection*? It is of particular interest as the traditional packet analysis is not of any usefulness in knowing the identity of applications or users. With our system, the identity (user/application) of *both* sides of the end-to-end connection can be linked together assuming both hosts are monitored. In its simplest form, a bipartite matching is found if an established connection recorded on Host A with src_A and dst_B matches another established connection record on Host B with src_B and dst_A within the same time frame. The time frame can be from a single hour to several days depending on the granularity requirement. Each chaining record begins with the *start* and *stop* time of each connection and is further divided into the *local identity/context* and *foreign identity/context* in terms of source and destination host names, IP/Port pairs, users, and applications associated with both ends of the connection. Without *ENAVis*, the identity of *who* is connecting to *whom* is vaguely inferred from the *IP/Port* pair. With the context information at hand, the identity can now be precisely tracked down through the bipartite matching, i.e. which *user* and what *application* are revealed at *both sides* of connection. This is useful in evaluating the effectiveness of the enforcement of the existing policy on the enterprise network.

The *ENAVis* visualization tool[1] was implemented using Java. The function of graph visualization and exploration of the context information is built on top of *Prefuse* [11], a graph rendering library for Java. Rather than static graph plots, Prefuse provides easy extendibility for *highly interactive* graph data exploration, queries and visualization. Some of the graph algorithms are from the Java Universal Network/graph framework (*JUNG*) and *JFreeChart* is utilized to plot the trend and statistical charts by taking advantage of active open-source projects and relative robustness and scalability of those libraries. Various data analysis algorithms and interactive exploration mechanisms were built in and Fig. 6 shows an interactive feature by querying graph nodes.

4. Application discussion and case studies

We discuss several case scenarios in which *ENAVis* can be helpful in local network management. The graphical exploration reduces the tedious, error-prone nature of log checking and mapping down to a few mouse clicks, which makes administrators' lives much easier. With the capability of central correlating hosts, users and applications through interacting with HUA graphs and straightforward statistical charts offered by *ENAVis*, the investigation carried out by the system and network adminstration can be confined to $O(1)$ steps and does not have to hop through $O(n)$ hosts in scale of a distributed system. The investigation supported by *ENAVis* is a quick and convenient process with mouse-click driven exploration. In this section, we will study two cases in detail with supporting graphs and data from using the tool: user and application-level policy compliance check and investigate and cleanup after user account compromise. Additional case studies can be found in [1].

4.1. Scenario 1: policy compliance

The management needs to know whether their employees have complied with the company's network usage policy with regards to finance information compliance. Specifically, the administrator is requested to provide a report of whether the mechanisms are adequate for enforcing the current policy. For this case study, consider a financial intranet server whose access policy is defined such that only authorized users can access or even see the financial system. To that end, a set of host-based firewall rules are put in place on the finance server (*finance.nd.edu*) with restrictions to the hosts of authorized finance personnel (*concert.cse.nd.edu*, *striegel*) on the company campus.

4.1.1. Current approach

First, the admin checks that the firewall rules (IP/port) settings are correct for the finance server through the application of a policy rule visualization tool such as [12]. Once the rules are validated, the administrator checks the *ipfilter* log and NetFlow log data to ensure that only

[1] More information and download available at http://netscale.cse.nd.edu/LockDown

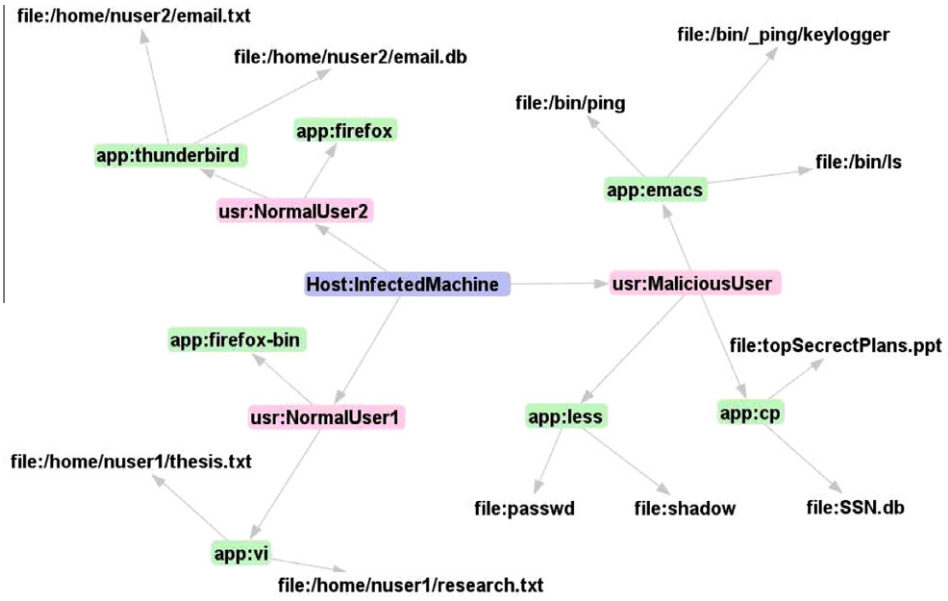


Fig. 7. Local context of network connections can be extended to include hosts, users, applications, and files (*HUAF*) to identify normal and malicious users.

authorized hosts accessed the server. Upon only seeing authorized hosts on the list (*concert*), the admin concludes that the policy is sound and not violated.

4.1.2. ENAVis approach

Unfortunately, the earlier approach is only sufficient if the host to user mapping stays consistent, i.e. only user *striegel* uses the host *concert.cse.nd.edu*. If host to user mapping is dynamic or unclear, the notion of host as identity quickly breaks down (see Fig. 8). Suppose in the same environment that *ssh* connectivity was enabled on the network. In the scenario of Fig. 8, an unauthorized user *qliao* connects from *IrishFB.nd.edu* with X11 forwarding to *concert.cse.nd.edu* and launches an instance of *Firefox* to access the finance web server. In a multi-user environment, where multiple users are logged onto the same machine and make network connections, other tools have no way of differentiating those connections because the connections all have the same source IP. Similarly, the legitimate user *striegel* may carelessly connect from a *Starbucks* shop to his office desktop *concert* in order to access a financial account just for convenience. Neither of these two cases is desirable and is a violation of the policy because the original intent of the policy was that anyone not part of the finance department should not be able to access the finance host.

The tiered graph created with *ENAVis* includes nodes representing hosts, users and applications, taking advantage of our data which records every user ID (UID) and process ID (PID) associated with each network socket created.

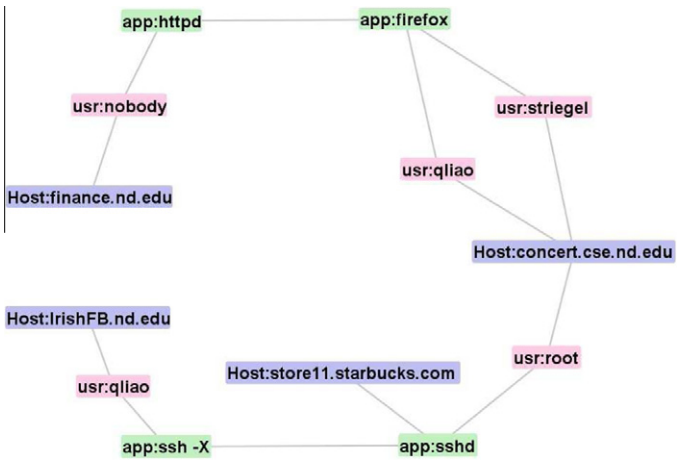


Fig. 8. An *ENAVis* HUA graph captures two possible host IP ACL policy violations caused by the unintentional configuration on host *concert* without `ssh` restriction.

Since each connection tuple now is expanded to be {time, proto, src_ip/port, dst_ip/port, usr, app}, we have finer granularity on the policy control on the user and application level in addition to the host level, which can be clearly seen from the *ENAVis* graph (Fig. 8). The admin is able to find the problem which is not offered by other tools, namely the unintentional configuration of *concert* with no `ssh` restriction causes the violation of the policy.

In addition to policy compliance checks at the host and user level, another side benefit is to check the policy compliance at the application level. Consider the case when the admin wants to make sure only the most up-to-date version of applications are approved for use on the network (see Fig. 9 for an example). There have been known vulnerabilities in an earlier version (1.5.x) of Firefox and the policy states users must use the properly patched version. A simple HA graph would reveal any non-compliance with this policy by looking at the applications connecting to the web server. In addition to vulnerability control, it is also useful for *license management*. Usually, the organization buys a fixed number of licenses from the software vendors. The license server should only check out a license to legitimate users and newest version of the application software. Our tool makes it possible to track this type of compliance as well.

4.2. Scenario 2: cleanup after compromise

A phishing email pretending to be from the IT department claims they are updating the system and require all users to send in their passwords, or their accounts will be suspended. A naïve user believes this scam and therefore his password is suspected to have been compromised.

4.2.1. Current approach

The system administrator needs to find out which hosts the compromised user account has used. Have those hosts been compromised as well? What applications did that user invoke? What data files did this user account touch during the past two weeks since the user revealed his password? The admin must make sure the student/faculty's sensitive information and intellectual property was not leaked from the network. In order to do this, the admin checks a centralized server such as an Active Directory or Kerberos 5's log file. Fortunately, the log file is still there, and the admin can then manually search and find all hosts that user has been trying to log into via the `ssh` pluggable authentication modules (PAM). The admin logs into each machine and makes sure they are clean. However, the admin has no idea what files have been read/modified or been sent out to external hosts. The admin also does not know what applications have been run by that user account because the data is not available.

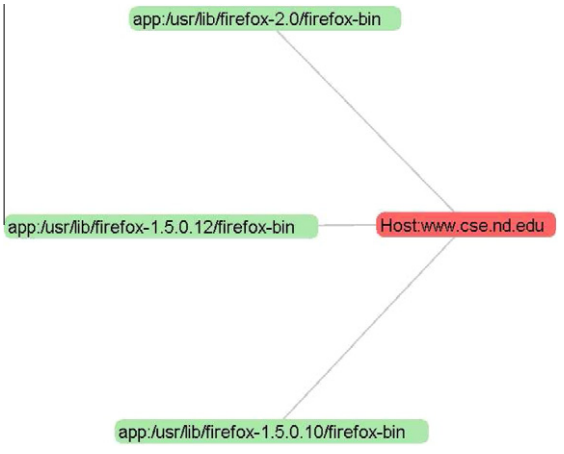


Fig. 9. A simple example showing policy compliance control on application versions for vulnerability avoidance and license management.

4.2.2. ENAVis approach

The admin simply generates a network graph by selecting the HUA from the graph menu. The admin highlights the problem user node (*jdoe*) (as in Fig. 10). It is straightforward to see which hosts the user has touched during the time frame and what applications the user used. The file access information logged by `lsof` is not available to other tools, neither in a centralized authentication server nor in the normal end-host's access logs. Although we do not normally plot the `lsof` data, each file accessed by that user is kept in the master database. Therefore, a single query would reveal all files that user ID has touched among all the hosts. In this case, a visual graph is very helpful to see what hosts and users that a compromised user account has contacted and which applications it has attempted to launch. This helps expedite significantly such an investigation should it occur.

5. Visual mining on context graphs

As shown earlier, context visualization and exploration is an effective way for network monitoring and solving management and security problems. However, visualization is only useful if an investigator knows exactly what he/she is looking for. Wandering randomly in an attempt to find suspicious activities in vast amounts of data is less likely to be productive. There must be a *starting point* for visualization and analysis [13]. Therefore, there is a need for an intelligent module to be built into the *ENAVis* visualization tool. On the other hand, however, pure automatic computer analysis cannot substitute for visualization because there are certain patterns or knowledge that can be missed by traditional automatic data mining methods [14]. Ideally, during the visualization process, some graph statistical and data mining or machine learning algorithms can *augment* the domain knowledge of human investigator and *guide* the visual exploration process to only things that are important (e.g., by highlighting the part of the graph that is most suspicious and need further investigation). In this section, we briefly discuss a few functions in *ENAVis*, i.e. how to visualize the variance (and invariance) of HUA network graphs without manual comparison of graphs themselves; and how the users and applications communities can be visualized to provide insights.

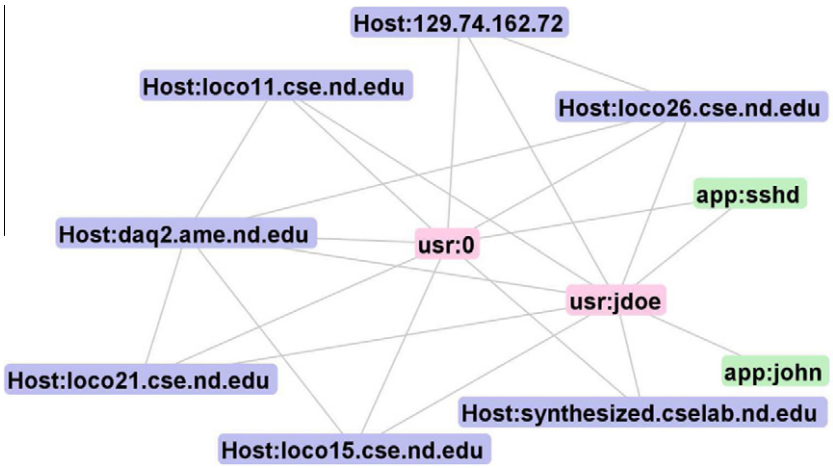


Fig. 10. The HUA network graph reveals a highlighted user (jdoe) has logged on seven machines via `ssh` and has used the application John the Ripper to crack password files on those machines.

5.1. Visualizing variance and invariance of network graphs

In order to help the administrators/researchers understand their networks and consequently detect abnormal activity, *ENAVis* attempts to find the *patterns* occurring on the network. It is important to determine the set of *variants* and *invariants* of the network graphs. We show that this can be answered by visualizing the maximum common subgraph (MCS) and minimum common supergraph (MCP). The maximum common subgraph (MCS) of graphs is defined as the largest subgraph that appears in all supergraphs. It is important to note that the MCS of all n network graphs is the *invariant* spanning the entire monitored period. In other words, the host, user and application nodes and the connection edges between them always appear. Concerning the computation complexity, *maximum common subgraph-isomorphism* is an optimization problem that is known to be NP-hard [15]. However, in an enterprise network setting, since each node is uniquely labeled (by its IP address, user ID or process binary path), in fact, many NP-hard problems for *general graphs* can be solved very efficiently (usually in linear time) [16]. The minimum common supergraph (MCP) of network graphs is defined as the smallest graph that includes all graphs as subgraphs. Note that the *variants* of network graphs are the differences between MCP and MCS, i.e. $VAR_n = MCP_n - MCS_n$.

The subgraph MCS and the supergraph MCP are important in network monitoring and management because while MCPs measure the maximum possible activities that can grow in the network, MCSs as invariants imply the strong tie and consistent relationships between long lived nodes which constitute the stability of networks. While the above subgraphs and supergraphs exhibit discrete properties (i.e. 0 for not appearing or 1 for appearing), it is probably a better and more useful approach if we can denote the appearance of nodes/edges with probabilities. The minimum common supergraph with probabilities (MCPP) is an extension to MCP in that the probability is computed as the edge weights, i.e. $W_{(u,v)} = \frac{F(u,v)}{|G|},\ \forall u,\ v$, where $F(u,v)$ is the appearance frequency of edge (u,v) and $|G|$ denotes the number of snapshot graphs. The relationships among appearance probabilities for nodes/edges are $P_{MCS} = 1 > P_{\{P_i \ldots P_j\}} > P_i \nwarrow\!\!\!\nwarrow P_j > 0$. MCPP is also important because building such a probability graph is helpful to *predict* network links and *detect* possible anomalies. For example, suppose an user U has a connection probability *0.9* with application A_1 and host H_1, and zero probability with application A_2 and host H_2. It is reasonable to conjecture any missing edges in future graphs among U, A_1 and H_1, or a sudden new connection edge among U, A_2 and H_2, is suspicious and needs investigation.

5.2. Expected graph distance variances

Manually comparing the difference of daily network graphs is painful and time consuming (see Fig. 11). While visual examination of a network consisting of only a few dozens of nodes might be feasible, visually comparing networks as large as thousands of nodes is nearly impossible because of human perceptual and cognitive limitations. Therefore, an automated view is provided in *ENAVis* to give a quick overview of how the network is different from how it is expected to be. Fig. 12 shows such an alternative view to measure how *different* (or equivalently how *similar*) each snapshot graph is from the "expected graphs". To achieve that, *ENAVis* builds three graphs, i.e. MCS, MCP and MCPP, which contains the connectivity probabilities of links, and then generates a statistical chart on the distance variances among all graphs over time. Generally speaking, the higher the variance score, the more "abnormal" a graph is. With MCS being the minimum and MCP as the maximum graphs that a graph can possibly grow, the distance variances have the opposite relation between them. It is interesting to observe that MCS is essentially a MCCP with a threshold of 1 and similarly the MCP is a MCPP with an infinitely small

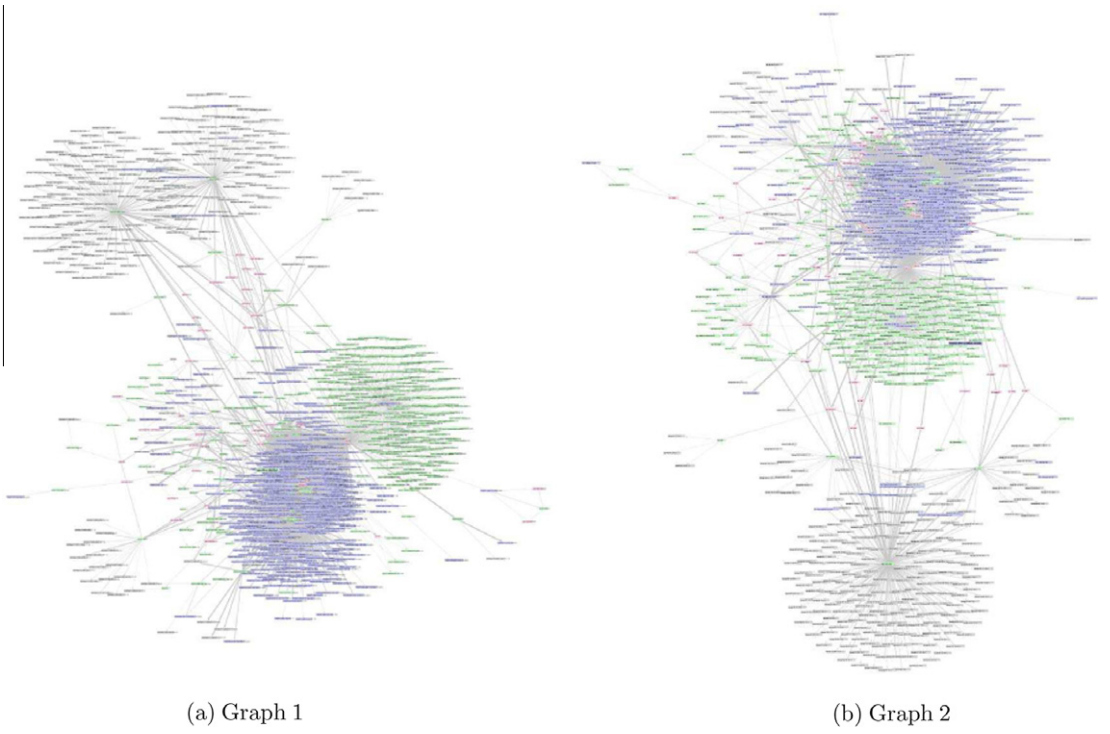


Fig. 11. Visual comparison for similarity among large network graphs is difficult for humans, therefore need clever data mining process to assist visualization.

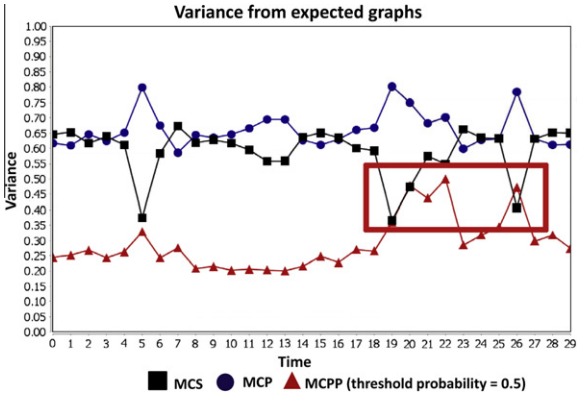


Fig. 12. The normalized distance variances from the *expected graphs*, i.e. MCS, MCP, and MCPP, can be easily and visually examined over one month period rather than visually comparing the graphs directly. The higher the score the more *abnormal* that day's network is.

number approaching 0. When the threshold is set to 0.5, MCPP smoothes out its curve and more accurately points out the network graphs on days 19–22 and 26 (highlighted in red in Fig. 12) that need further investigation.

To calculate the distance from the expected graphs, we adopt a distance metric based on the idea of edit distance. In information theory, the *edit distance* is the number of operations required to transform one of them into the other. To quantify the similarity of network graphs, *graph edit distance* (GED) [16] has been suggested to measure the topological change. The basic idea of graph edit distance is the costs associated to modify a graph such that it becomes isomorphic to the other graph. There are usually three operations for the transition: insertion, deletion and substitution. Since vertex label substitution is not a valid edit operation because each node in an enterprise network denotes a unique host, user or application, label substitution is an essential two-step operation, i.e. remove the old one and insert a new one. 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. *ENAVis* uses the following equation to compute the graph edit distance between two graphs:

$$\frac{C^V_{del} * |V_{g_1} - V_{\mathrm{mcs}(g_1,g_2)}| + C^E_{del} * |E_{g_1} - E_{\mathrm{mcs}(g_1,g_2)}| + C^V_{add} * |V_{g_2} - V_{\mathrm{mcs}(g_1,g_2)}| + C^E_{add} * |E_{g_2} - E_{\mathrm{mcs}(g_1,g_2)}|}{C^V_{del} * V_{g_1} + C^V_{add} * V_{g_2} + C^E_{del} * E_{g_1} + C^E_{add} * E_{g_2}} \quad (1)$$

where C^V_{del} and C^E_{del} are the deletion cost and C^V_{add} and C^E_{add} are the insertion cost for vertices and edges respectively. The choice of these cost functions need to be further studied to reflect the best interests of each individual network. If all cost functions are equal to one, then the above equation can be simplified to:

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

Intuitively, if two graphs are 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 or edge, the result will be a distance value of one.

Lastly, in Fig. 13 we present one alternative multidimensional scaling (MDS) view that shows the relative relation between all graphs and the expected graph. Once we have computed a distance matrix for all pairs of graphs, we need to plot and visualize the graphs' relative positions. Generally speaking, knowing the exact locations of points, computing their distances is straightforward (e.g., Euclidean distance). However, in the opposite way, i.e. knowing their pairwise distance, finding their exact X/Y coordinates in a 2D Euclidean space is not quite straightforward. In fact, while it is deterministic to find their locations in $(n-1)$ dimensional space given n snapshot graphs and their pairwise distance matrix, it may nor may not possible to find the exact points in lower dimensions. For visualization purpose, the dimension is usually only 2D or 3D. Multidimensional scaling (MDS) [17,18] has been proposed to visualize high-dimensional data by mapping them into lower-dimensional space. *ENAVis* incorporates a MDS module for visualizing the relative locations (inexact) by mapping network graphs into a 2D space. In Fig. 13, 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. The expected graph (EG) is a MCPP with a threshold connection probability set to be 0.5. In this alternative view, not only can we see the distance of our network graphs from the expected graph (highlighted), but the distance variances among all network graphs themselves are shown as well. Although the EG sits in the center of all graphs, it is positioned closer towards the majority of graphs to the right, which clearly isolates the abnormal ones on the left.

5.3. Graph clustering visualization

We conclude this section with another capability of *ENAVis*: graph cluster visualization. Generally speaking, clustering can be categorized as intra-graph clustering and inter-graph clustering. Intra-graph clustering looks at grouping similar nodes and/or edges within a graph while inter-graph clustering seeks similarity among different graphs at different times. No matter whether intra or inter-graph clustering, the notion of *similarity* must be defined in order to group items. *ENAVis* graphs are more challenging since the graphs are *heterogeneous* in which the nodes can be either hosts, users, applications or files. For intra-graph clustering, we must utilize the important network connectivity information between nodes and for inter-graph clustering, we must define an appropriate similarity metric between any pair of graphs. Traditional clustering algorithms range from simple k-means to Bayesian clustering to Expectation–Maximization (EM). Unless we can map the nodes into Euclidean space (as done in Fig. 13), alternative graph-based community detection methods are needed. While Fig. 13 illustrates to some degree the inter-graph clustering, in this section, we focus on intra-graph clustering.

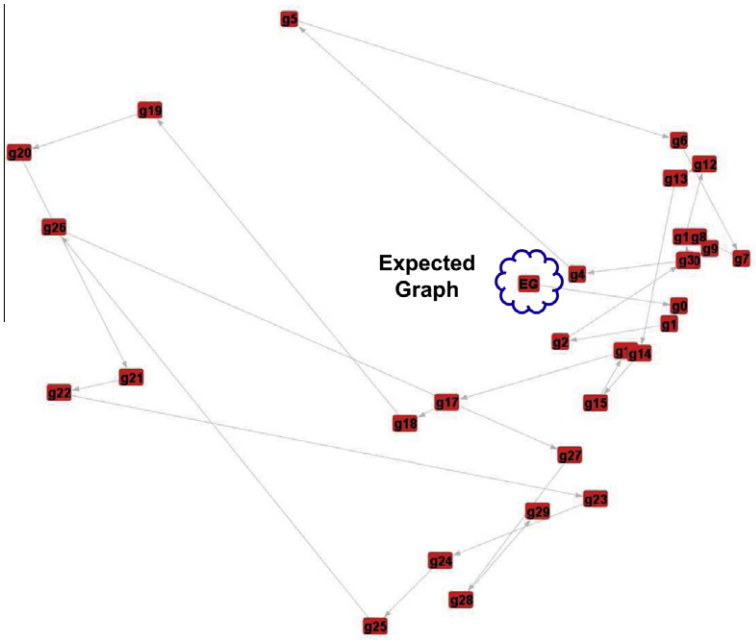


Fig. 13. An alternative MDS view showing the evolution and the relative relationships between network graphs. The expected graph (EG), a MCPP with a threshold probability of 0.5, locates in the center of all graphs. The further from EG, the more abnormal of a network graph.

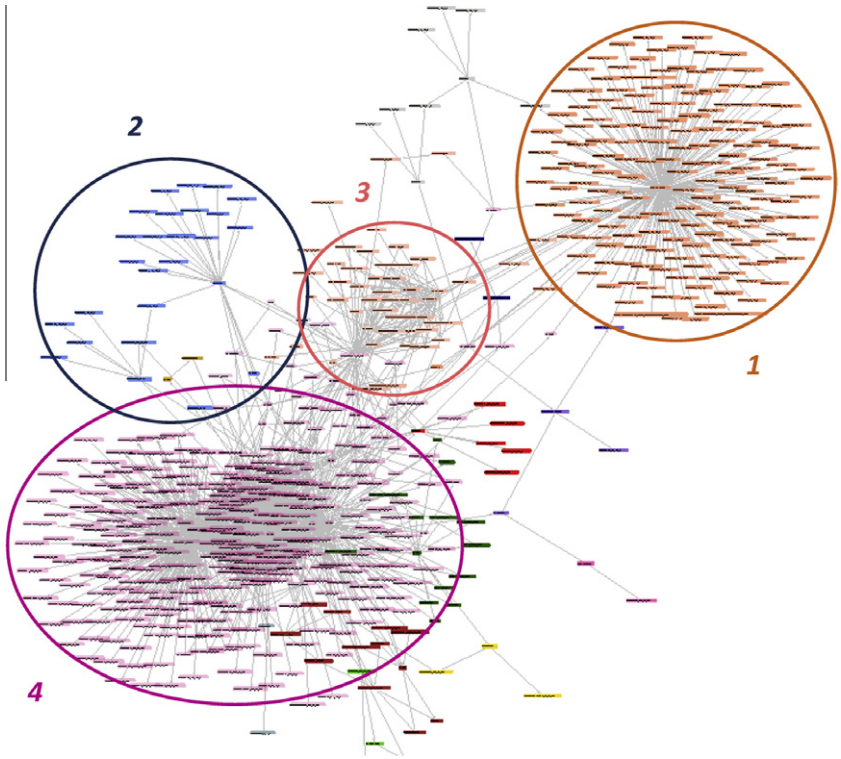


Fig. 14. A cluster view (different colors represent different clusters) using the Walktrap algorithm. The cluster visualization helps understanding the communities (e.g., firefox users and web related traffic, condor-related computing community, or users using similar set of applications). It also helps to identify potential anomalous user behaviors.

Fig. 14 shows one such example of clustering views using *ENAVis*. The Walktrap algorithm [19] is selected as it takes the approach of random walk. The similarity measurement is based on a simple yet effective assumption that a random walk tends to be trapped in a highly connected or dense area. In Fig. 14, nodes belonging to different clusters are colored differently for easy visualization. Cluster number one (1) and two (2) are web related communities, where cluster one (1) shows all external domains connected by Firefox, and cluster two (2) shows an internal web server that has been accessed by a group of hosts. Cluster number three (3) shows seven enterprise users sharing a similar set of applications that have queried the campus directory server: *directory.nd.edu*. The giant cluster number four (4) shows a well structured Condor [20] community formed by a few local users and condor users that launched condor-related batch jobs in the condor enabled workstations.

5.4. Privacy and security implication

While *ENAVis* provides the power of traceability and accountability in terms of users, applications and data files for security and management benefits, it is up to the administrators to decide which privacy policy is appropriate for their organizations. This raises possible debate from normal users' points of view as to how much information should be revealed to the administration. On the other hand, the anonymity on the Internet to some extent contributes to the increasing cybercriminal activities. Within the premise of a private organization, it is reasonable to assume the network and system administration should have the full visibility on the activities on their networks for various good faith reasons, such as troubleshooting, intrusion detection, malware or malicious user behavior monitoring. Future research will be developing graph mining algorithms for aggregate view and abstraction analysis by grouping similar nodes (privacy preservation through hiding in the crowd) as well as considering a number of different cryptographic anonymization algorithms to be combined to achieve a reasonable level of user privacy.

6. Related work

Broadly speaking, network monitoring and analysis can be categorized into two models. In the first type, in-network devices record and collect data using tools such as `tcpdump` or Cisco's NetFlow profiling. The other type is end-host monitoring using an agent mechanism. The end-host monitoring approach has the advantage of being able to see more information than inline monitoring since it has full visibility of the network activities occurring on each host. We adopted the latter model for our data collection system. While sFlow uses agents on switches/routers to log packets and send the logs to a central collector for analyzing, the traffic monitoring is at the packet level, thus missing the local context information for each connection. It is necessary for the *context* of a connection, i.e. the user and application responsible for the network activity, to

be known rather than simply *where* (address) it came from and went to. Existing solutions to this problem have involved tie-ins of network flow data and authentication systems such as Active Directory and Kerberos. Critically, these existing logging systems are not geared towards real-world system administration. Network flow data will only detail the *where* of a connection, where as an Active Directory and Kerberos tie-in can explain the *who*.

A few visualization and data exploration tools (ISIS [21], NetFlow Visualizer [22], OverFlow [13], NVisionIP [23] and VisFlowConnect-IP [8]) that exist, primarily rely on chaining together network connections based on either packet headers or the flow data. For example, ISIS [21] is a tool that visualizes temporal relationships among network flow data by plotting time in combinations with IPs to find correlations between events to aid investigations regarding network intrusion. IP addresses and port activities can be visually analyzed by PortVis [24] and histograms [6] while packet headers can also be visually analyzed by Ethereal, Rumint and TNV [4]. 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 visualizing NetFlow or packet header data is the missing *user* and *application* information, which we posit is critical for enterprise network management.

Visualization is useful for enterprise network management and security [25]. Graph visualization can be performed on various log data such as service dependance in large web sites like Amazon.com (Maya [26]). Rather than focusing on specific predefined anomaly (e.g., Severity 1 failure type), we seek a more general visualization framework through analysis of evolution of network graphs with no restriction in any specific anomaly or thread model. The visual analysis on the similarities/changes of network activity graphs provides insight through both automatic intelligent analysis and manual interactive exploration within local-context graphs to understand *hosts*, *users* and *application* behaviors in enterprise networks.

Visualization techniques have been applied to view *static* data, such as distributed firewall rules to detect potential conflicts or anomalies. For example, PolicyVis [12] is a visualization tool for plotting IP addresses and port numbers specified by the firewall rules. Instead of visualizing policy rules, we visualize the *dynamic* data, which is the actual network activities made by users' applications. The visual analysis done on the empirical data is a substantial and necessary supplement to the static rules inspection as a proof of correctness to the policy rules. Graph-based network traffic visualization [27] can be used to monitor host behavior, suspicious behavior and network anomalies can be detected through graph drawing [25], graph clustering [28], tree-views on hierarchical clustering (NetADHICT [29]), animated glyphs [30], pixel luminance based histographs (IDGraphs [31]), and scatter plot of IPs/ports in Service Usage Plane in 2D/3D Cartesian coordinates for the flow data [32]. However, lack of the ability for either *interactive exploration* of *context* data or intelligent misuse detection modules in these tools make less efficient for insight/knowledge acquisition and root cause identification of various management and security problems. Our visualization is dynamic and intelligent as opposed to static visualization in that the users can explore the data in a highly interactive manner. Users can click on nodes, perform queries to database on demand, automatically compute and generate statistical charts (e.g., distances), selectively run from a rich pool of algorithms (e.g. clusters evolution), and intelligently *guide* the administrator to pin down the problem source nodes which require further investigation.

Finally, in [33,34] the authors propose capturing the inter-dependencies among network components in 'Leslie graphs,' based on the original dependency work of Lamport. The "black-box" approach relies on the correlation of observed network traffic to infer system dependencies. The agents in their system (called *AND*) perform temporal correlation of the packets sent and received by the hosts; where the central server engine performs Bayesian inference from the reports generated by the agents. While these works mainly focus on computing the dependency graphs for fault localization (i.e. debugging the location of network failure or sluggish performance), our system focuses on the lightweight aspects of information gathering and how to visualize not only connectivity but, the context of the connectivity itself. In short, while these tools help to locate dependency-related performance problems at the host-level in a theoretical sense, *ENAVis* provides a robust platform for exploring and visualizing the connectivity data for a much wider assortment of security and performance-related issues.

7. Conclusion

It is desirable, yet difficult, to know exactly *who* and *what* is running on an enterprise network. In current network architecture, the identity of *user* and *application* in network flows is inferred from a packet's *content* (i.e. IP addresses and port numbers) rather than directly from the *context* (user processes, file accesses) that actually make those connections. In this paper, we describe a network local context data collection system and *ENAVis*, an Enterprise Network Activities Visualization and analysis tool.

In addition to the regular analysis functions provided similarly by NetFlow and packet monitoring tools, *ENAVis* offers interesting new features of visual analysis on the user's and application's level. Connectivity graphs of combinations of hosts, users, applications and data accesses capture the dynamic interactions among these essential and most dynamic components in the network. The interactive exploration of the *context* data makes finding the root cause of various network management and security problems takes less time and effort. While the current intelligent module of *ENAVis* provides some degree of clever guidance to manual exploration, our work-in-progress focuses on combining the visualization techniques and graph mining process to maximize the knowledge/insight acquisition and anomaly detection.

Acknowledgement

This work was supported in part by the National Science Foundation (CNS-03-47392).

References

[1] Q. Liao, A. Blaich, A. Striegel, D. Thain, ENAVis: enterprise network activities visualization, in: Proceedings of the USENIX 22nd Large Installation System Administration Conference (LISA'08), San Diego, CA, 2008, p. 5974.

[2] D. Lalanne, E. Bertini, P. Hertzog, P. Bados, Visual analysis of corporate network intelligence: abstracting and reasoning on yesterdays for acting today, in: Workshop on Visualization for Computer Security (VizSec'07), Sacramento, CA, 2007, pp. 115–130.

[3] A. Blaich, Q. Liao, A. Striegel, D. Thain, Simplifying network management with lockdown, in: Symposium on Usable Privacy and Security (SOUPS'08), Pittsburgh, PA, 2008.

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

[5] S.K. Card, J. Mackinlay, B. Shneiderman, Readings in Information Visualization: Using Vision to Think, first ed., Morgan Kaufmann, 1999, ISBN:1558605339.

[6] K. Abdullah, C. Lee, G. Conti, J.A. Copeland, Visualizing network data for intrusion detection, in: Proceedings of the 2002 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 2002, pp. 30–38.

[7] T. Takada, H. Koike, MieLog: a highly interactive visual log browser using information visualization and statistical analysis, in: Proceedings of the 16th USENIX Conference on System Administration (LISA'02), Philadelphia, PA, 2002, pp. 133–144.

[8] W. Yurcik, Visualizing netflows for security at line speed: the SIFT tool suite, in: 19th Large Installation System Administration Conference (LISA'05), San Diego, CA, 2005, p. 16.

[9] M. Becchi, P. Crowley, A hybrid finite automaton for practical deep packet inspection, in: Proceedings of the 2007 ACM CoNEXT Conference, New York, NY, 2007.

[10] J.R. Gersh, J.A. McKneely, R.W. Remington, Cognitive engineering: understanding human interaction with complex systems, Johns Hopkins APL Technical Digest 26 (4) (2005) 377–382.

[11] J. Heer, S.K. Card, J.A. Landay, Prefuse: a toolkit for interactive information visualization, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Portland, Oregon, 2005, pp. 421–430.

[12] T. Tran, E. Al-Shaer, R. Boutaba, PolicyVis: Firewall security policy visualization and inspection, in: 21st Large Installation System Administration Conference (LISA'07), Dallas, TX, 2007, pp. 1–16.

[13] J. Glanfield, S. Brooks, T. Taylor, D. Paterson, C. Smith, C. Gates, J. McHugh, Overflow: an overview visualization for network analysis, in: The Sixth Internationsl Workshop on Visualization for Cyber Security (VizSec 09), in conjunction with VisWeek 2009, Atlantic City, NJ, 2009, pp. 11–19.

[14] S.T. Teoh, K.-L. Ma, S.F. Wu, T. Jankun-Kelly, Detecting flaws and intruders with visual data analysis, IEEE Computer Graphics and Applications 24 (5) (2004) 27–35.

[15] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, 1979.

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

[17] T.F. Cox, M. Cox, Multidimensional Scaling, second ed., Chapman & Hall/CRC, 2000.

[18] H. Bunke, P. Dickinson, A. Humm, C. Irniger, M. Kraetzl, Applied graph theory in computer vision and pattern recognition, Ch Graph Sequence Visualisation and its Application to Computer Network Monitoring and Abnormal Event Detection, vol. 52, Springer, Berlin, Heidelberg, 2007.

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

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

[21] D. Phan, J. Gerth, M. Lee, A. Paepcke, T. Winograd, Visual analysis of network flow data with timelines and event plots, in: Workshop on Visualization for Computer Security (VizSEC), Sacramento, CA, 2007, pp. 85–99.

[22] P. Minarik, T. Dymacek, Netflow data visualization based on graphs, in: Proceedings of Fifth International Workshop on Visualization for Computer Security (VizSec'08), Cambridge, MA, 2008, pp. 144–151.

[23] K. Lakkaraju, W. Yurcik, A.J. Lee, NVisionIP: Netflow visualizations of system state for security situational awareness, in: Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC), New York, NY, 2004, pp. 65–72.

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

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

[26] P. Bodik, Advanced tools for operators of internet services, Master's Thesis, University of California, Berkeley, May 2006.

[27] F. Mansmann, L. Meier, D. Keim, Graph-based monitoring of host behavior for network security, in: Workshop on Visualization for Computer Security (VizSec'07), Sacramento, CA, 2007, pp. 187–202.

[28] J. Tolle, O. Niggemann, Supporting intrusion detection by graph clustering and graph drawing, in: RAID 2000 Third International Workshop on the Recent Advances in Intrusion Detection, Toulouse, France, 2000, pp. 51–62.

[29] H. Inoue, D. Jansens, A. Hijazi, A. Somayaji, NetADHICT: a tool for understanding network traffic, in: 21st Large Installation System Administration Conference (LISA'07), Dallas, TX, 2007.

[30] R.F. Erbacher, Intrusion behavior detection through visualization, in: IEEE International Conference on Systems, Man and Cybernetics, Albany Univ., NY, 2003, pp. 2507–2513.

[31] P. Ren, Y. Gao, Z. Li, Y. Chen, B. Watson, Idgraphs: intrusion detection and analysis using histographs, in: Proceedings of the IEEE Workshops on Visualization for Computer Security (VizSec'05), Minneapolis, MN, 2005.

[32] I.-V. Onut, B. Zhu, A.A. Ghorbani, A novel visualization technique for network anomaly detection, in: Proceedings of Second Annual Conference on Privacy Security and trust, Fredericton, Canada, 2004, pp. 167–174.

[33] P. Bahl, P. Barham, R. Black, R. Chandra, M. Goldszmidt, R. Isaacs, S. Kandula, L. Li, J. MacCormick, D.A. Maltz, R. Mortier, M. Wawrzoniak, M. Zhang, Discovering dependencies for network management, in: ACM SIGCOMM Fifth Workshop on Hot Topics in Netowrks(Hotnets-V), Irvine, California, 2006, pp. 97–102.

[34] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D.A. Maltz, M. Zhang, Towards highly reliable enterprise network services via inference of multi-level dependencies, ACM SIGCOMM Computer Communication Review 37 (2007) 13–24.

Qi Liao is a Ph.D. candidate at the Computer Science & Engineering department of the University of Notre Dame. His current research interests include visualization for network management and security, graph and data mining and economics for computer networks and security. He was awarded a master's degree in computer science and engineering (*MSCSE*) from the University of Notre Dame, Indiana. Qi received his BS and graduated with Departmental Distinction in *Computer Science* from Hartwick College, New York, with minor concentration in *Mathematics*. Qi is a member of *Kappa Mu Epsilon* (national mathematics honor society), *Upsilon Pi Epsilon* (international honor society for the computing and information disciplines), and *Tau Beta Pi* (national engineering honor society). Reach him at qliao@nd.edu.

Andrew Blaich is a Ph.D candidate at the University of Notre in the Computer Science and Engineering department. His current research interests include wireless networks (802.11), security, network management, and mobile devices. He received his BS and MS in Computer Engineering at Villanova University. Andrew can be reached at ablaich @nd.edu.

Dirk VanBruggen received the BS degree in computer science and mathematics from Hope College in 2009. He is currently working toward the Ph.D. degree at the University of Notre Dame. His research focuses on networking, visualization and security.

Dr. Aaron Striegel is currently an associate professor in the Department of Computer Science & Engineering at the University of Notre Dame. He received his Ph.D. in December 2002 in Computer Engineering at Iowa State University under the direction of Dr. G. Manimaran. His research interests include networking (bandwidth conservation, QoS), computer security, grid computing, and real-time systems. During his tenure as a student at Iowa State, he worked for various companies in research and development that included Sun Microsystems, Architecture Technology Corporation, and Emerson Process. He has received research and equipment funding from NSF, DARPA, Sun Microsystems, Hewlett Packard, Architecture Technology Corporation, and Intel. Dr. Striegel was the recipient of an NSF CAREER award in 2004. Reach him at striegel@nd.edu.