

A novel visualization technique for network anomaly detection

Iosif-Viorel Onut
 Faculty of Computer Science
 University of New Brunswick
 Fredericton, Canada
 Email: onut.viorel@unb.ca

Bin Zhu
 Faculty of Computer Science
 University of New Brunswick
 Fredericton, Canada
 Email: bin.zhu@unb.ca

Ali A. Ghorbani
 Faculty of Computer Science
 University of New Brunswick
 Fredericton, Canada
 Email: ghorbani@unb.ca

Abstract—Visualized information is a technique that can encode large amounts of complex interrelated data, being at the same time easily quantified, manipulated, and processed by a human user. Our aim is to develop a novel graphical technique for network traffic visualization that will easily highlight anomalies that can arise within the network. In our work we are exclusively concerned with all the information that can be extracted at the network layer (e.g., from the TCP/IP datagram). We choose to use the Darpa 1999 database given the fact that all the intrusions are labeled and we can easily observe the visualization behavior while the network is under attack. Although applied to a dataset, the visualization technique can work on-line in a network because it only uses data that can be extracted in a real-time manner. Experiments show our visualization technique to be a good medium when trying to identify possible anomalies of the network such as: *DoS types of attacks* (e.g., *Smurf* and *Mailbomb*) as well as *probing attacks* (e.g., *Portsweep* and *IPsweep*).

I. INTRODUCTION

Data visualization is a technique that has been used for a long time to represent information. Although old, yet powerful, its main outcome is that it allows the representation of data by different formats and shapes, each one highlighting a particular group of features.

Visualization represents a powerful link between the most dominant information-processing systems, the human brain and the modern computer. It is a key technology for extracting information, and therefore it is becoming more and more necessary in the field of Network Security. The power of network visualization goes beyond the simple “illustration” of network behavior to help the analyst to discriminate between normal, and abnormal activities.

Our goal here is to develop an innovative graphical technique for network traffic visualization that will highlight the features of the network data most vulnerable to intrusions. In order to work on-line the visualization technique uses only data that can be captured from the network in a real-time manner, in our case all the relevant data that can be extracted from a TCP/IP datagram. Due to the computational time the payload of the datagram is ignored; the headers are the only information processed.

The network is viewed as a community of entities (hosts or servers that have an IP address) that interact by changing packets among them; thus, each entity has a correspondent graphical representation. Throughout our experiments we have

identified four main features of each entity that are highly sensitive to the anomalies that arise in a network: its load, the number of ports that it is using, the number of hosts that it is using, and the services that are used.

This paper is organized as follows: In Section II, some background review is presented concerning the state of the art of the visualization methods that have been already used. Section III describes in detail our visualization approach and debates the main outcomes and drawbacks of the representation. The implementation and deployment of our system is described in Section IV. Next, Section V is concerned with the experimental results. Finally the last Section summarizes the conclusions of the work, and discusses possible future improvements.

II. BACKGROUND REVIEW

Network traffic visualization represents one of the first directions to take when it comes to understanding data flow within a network. Its ultimate goal is fulfilled when the human user through that visualization is not only able to monitor the traffic, but can also discriminate between the normal and the abnormal behaviors. A lot of work has been done in this direction, starting from the base understanding of TCP/UDP(IP) traffic, and finishing with complex 3D network representations. C. Zhao and J. Mayo proposed a didactic visualization system which is intended to help students to understand the functionality of the protocols, displaying the network information under different formats: Packet List View, Connection Packet View, Topology View, and Timeline View to name a few [5]. The “QVision” product [9] (renamed in March 2004 as “QRadar”) is commercial software which has a variety of views for network traffic visualization such as Server Application View and Geographic View. All these views are based on the same concept, namely a 2D representation of a certain value in time.

In the same 2D representation category falls the work that has been done by R.F. Erbacher, who proposed a glyph based graph for displaying the topology and load of the network [2]. In this glyph representation each node represents a host, a router or a server. The graphical symbol for a node is a circle and its gray filling represents its current load. Each edge in the graph represents the connection between two nodes. The black borders of each edge represent the full capacity of that

particular connection, while the gray interior fill shows the percentage of currently used bandwidth. Despite the advantage of direct load monitoring, no temporal information or user information can be displayed. In order to overcome this, R. Erbacher et al. introduce some new graphical features capable of showing both types of information in an elegant way [4], [6].

The need for a third dimension is comprehensible when it comes to network monitoring, especially when a lot of features are to be displayed. The network monitoring system proposed by M. Fisk et al. proposes a novel 3D representation technique of the network. The main contribution of this work is the internal-external address model. By its use the network scans, and all the internal/external hosts can be easily identified in one screen [1].

All the previous visualization techniques use different tricks (like shapes, colors, and sizes) in order to add more features to the graphical image for representing information about connections, ports, territories, and load of the network entities.

The main problems encountered when creating a network visualization technique remain: feature selection and computational time. The former problem arises because there are a lot of features that can be computed (at all TCP/IP layers) in real-time for a particular datagram. Moreover some of them are platform independent (application layer), their extraction requiring a deep understanding of almost all platforms that can be encountered in a real network. Another factor that influences the feature selection is introduced by the size of the monitor, the fact that the more information is displayed the harder it is for the human user to distinguish between different features. Finally, the latter problem is encountered as a consequence of the high rate of which data must be processed in order to maintain a real-time representation of the network.

To address these problems various techniques have been adopted including the following: disregarding the datagram payload, and considering only the header information [7], [5]; introducing multiple view with the aim of displaying more information, and classifying it into views [3], [6]; constraining the display analysis to a particular protocol or some particular features [7], [2], [5].

III. APPROACH

This paper introduces two main types of graphical views: *Services Behavior View* and *Category View*. These views are defined for the *Network Layer* of the *Internet Architecture*. The former view relies on the fact that all hosts in the network can be clustered with respect to the services that they are using into normal and abnormal ones. The latter view creates a classification of the hosts with respect to the usage of a particular value (e.g., *No. of Ports* for Ports View, and *No. of Hosts* for Hosts View).

The basic elements of the graphical views are the spheres which represent the hosts. In order to distinguish between internal and external hosts different colors are used (e.g., red for external hosts, and blue for the internal ones). In addition, the depth of color implies the duration of a host's activity in

the network. For example if the host just appeared it has a dark blue or dark red color. As time passes the older hosts become lighter and lighter.

The main goal when designing the two views is the real-time issue. Once integrated in a system, the views should provide instantaneous information about the network. Consequently, the data collected from the network must be available without delay. Thus, the data being visualized are raw network packets that come directly from a network link. Since there is a large amount of information being animated over time, it is critical that appropriate data features are selected for visualization. Accordingly, five fields are selected from Ethernet header, IP header and TCP header (i.e. timestamp, source IP address, destination IP address, source port, destination port, length of packet, and type of protocol). Due to the computational time involved and the amount of information that must be considered in each instant of time, we defined a *basic time interval* τ which is used to collect the instantaneous data from the network, to process it, and to display it. Selecting a appropriate τ is a crucial task. If τ is too small, the position of the hosts might change dramatically from τ to τ making their behavior hard to interpret. In contrast, if τ is too large, two problems arise: firstly, there will be an obvious delay in displaying the information, and secondly, a short term abnormal behavior may be covered by other activities conducted by the same host.

A. Services Behavior View

The first premise when defining this view is that the system administrator is especially interested in the behavior of the internal/external hosts with respect to a certain set of services. Let Ψ represent a particular set of services. Assume a network that has two servers S_1 and S_2 , each one providing a set of services Ψ_{S_1} and Ψ_{S_2} respectively to the network. Consequently the network administrator will be especially interested in the reunion of those services $\Psi = \Psi_{S_1} \cup \Psi_{S_2}$.

Secondly, the reunion of the monitored services Ψ will have between 3 and 8 members. This constraint is motivated by the fact that most of the time one is interested in the most popular services (e.g., HTTP, FTP, DNS, to name a few).

Finally, another motivation is that the greater the number of services displayed the harder it is for the user to distinguish between them.

In our 3D graphical representation model, two of the dimensions are used to represent service usage for internal and external hosts. Accordingly, all the services from Ψ are arranged in this bi-dimensional plane called *Service Usage Plane*. Let us define θ as the origin of the view's axis. Consequently, all the services from Ψ set will be positioned equally distant from θ . At the same time the services are equally distanced among themselves. As a result, all the services will lie on a circle centered at θ named *the attraction circle*, Γ . The number of selected services will define the *base shape* of the *Service Usage Plane* where hosts will move (e.g., triangle, square, pentagon, hexagon, heptagon, octagon for 3,4,5,6,7, and 8 services, respectively). Let us define the

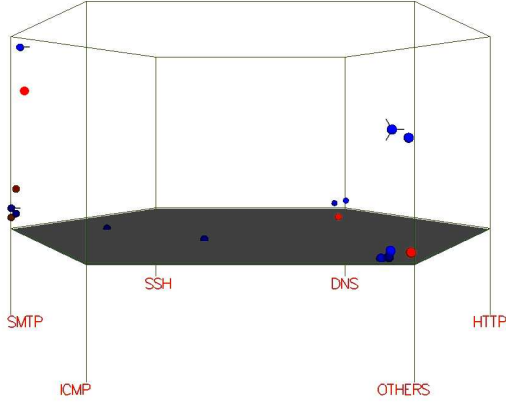


Fig. 1. The *Services Behavior* view. In this particular example the six selected services are arranged in a hexagonal base shape. The shadowed portion of the image represents the *Service Usage Plane* while the load is displayed in the third dimension.

service point as the place where a service is displayed on the Γ circle.

Having the *Service Usage Plane* defined, our host clustering strategy is based on the assumption that the more a host is using a particular service, the closer that host will be from that *service point* during a predefined time interval τ . For instance, if a host H_j uses only one service S_k for the last τ interval then its position will be the same with the S_k *Service Point*. Furthermore, if H_j is using two services at the same time, then its position will be on the line connecting the two services.

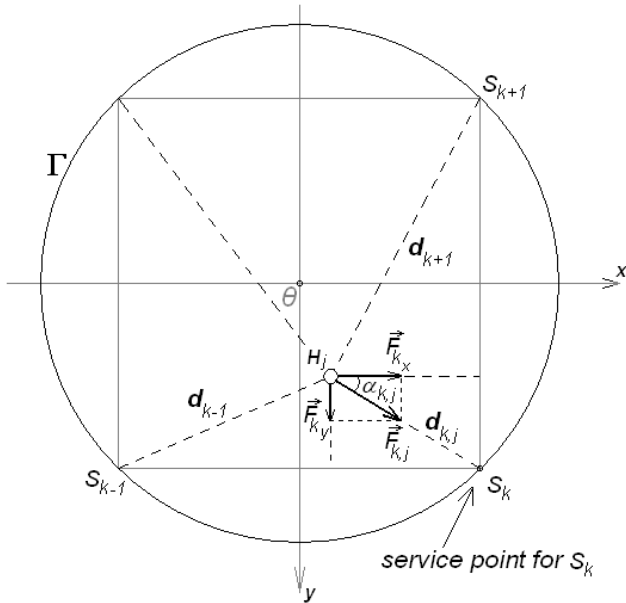


Fig. 2. The *Service Usage Plane* when four services are selected. For the simplicity, only one host H_j is displayed.

Fig. 2 illustrates an example where four services are selected. In this case the *base shape* of the *Service Usage Plane*

is a square, and $k \in [1, 4]$.

Let us define the *Attraction Force*, the force with which a particular service S_k attracts a host H_j , as follows:

$$\vec{F}_{k,j} = A_k \cdot L_{k,j} \begin{bmatrix} \cos(\alpha_{k,j}) \\ \sin(\alpha_{k,j}) \end{bmatrix}, \quad (1)$$

where $L_{k,j}$ is the load of the host H_j with respect to the service S_k , and A_k is a predefined *anomaly factor* for each service S_k with respect to its normal load. For instance hundreds of KB per second is a normal load for FTP and HTTP connections, but too high for ICMP. Consequently, the ICMP service will have a higher *anomaly factor* than the FTP service. Through this, if a host is heavily using a service like ICMP or DNS and at the same time is normally using a service like FTP or HTTP, the host will be attracted by those services which have a higher *anomaly factor* (e.g., ICMP and DNS).

The final position of a host H_j is influenced by all the *Attraction Forces* that exist in a scenario (e.g., in Fig. 2 there are four services, thus there will be four forces that will influence each host). Therefore, the formal mathematical formula to express the *balance position* constraint of host H_j is:

$$\sum_{\forall S_k \in \Psi} \vec{F}_{k,j} \cdot d_{k,j} = 0, \quad (2)$$

where $d_{k,j}$ represents the distance between the host H_j and the *Service Point* of the S_k .

From (1) and (2) we have:

$$\sum_{\forall S_k \in \Psi} A_k \cdot L_{k,j} \begin{bmatrix} \cos(\alpha_{k,j}) \\ \sin(\alpha_{k,j}) \end{bmatrix} \cdot d_{k,j} = 0 \quad (3)$$

Assume that H_j and S_k points are defined by the following coordinates $H_j(x_j, y_j)$ and $S_k(x_{S_k}, y_{S_k})$ in the *Service Usage Plane*. From Fig. 2 we can express the $\sin(\alpha_{k,j})$ and $\cos(\alpha_{k,j})$ as:

$$\cos(\alpha_{k,j}) = \frac{|x_j - x_{S_k}|}{d_{k,j}} \quad (4)$$

$$\sin(\alpha_{k,j}) = \frac{|y_j - y_{S_k}|}{d_{k,j}} \quad (5)$$

Replacing (4) and (5) into (3) we have:

$$\sum_{\forall S_k \in \Psi} A_k \cdot L_{k,j} \begin{bmatrix} |x_j - x_{S_k}| \\ |y_j - y_{S_k}| \end{bmatrix} = 0 \quad (6)$$

we can split (6) into the following two equations:

$$\sum_{\forall S_k \in \Psi} A_k \cdot L_{k,j} |x_j - x_{S_k}| = 0 \quad (7)$$

$$\sum_{\forall S_k \in \Psi} A_k \cdot L_{k,j} |y_j - y_{S_k}| = 0 \quad (8)$$

Finally, using equations (7) and (8) the coordinates (x_j, y_j) of the host H_j can be computed.

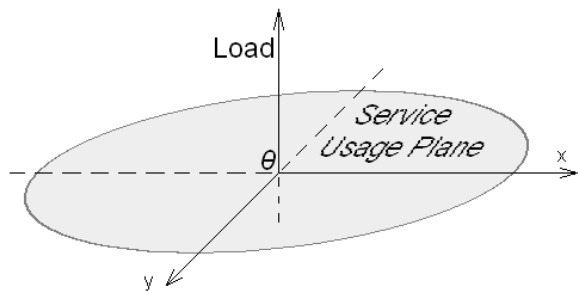


Fig. 3. The 3D space of the *Services Behavior View*

The two dimensional representation, which we have discussed so far, only reveals the proportional service usage of all hosts in the network. However, the real traffic load, which also encodes important information about network behavior, is still missing. Besides this disadvantage, a large number of hosts tend to overlap in this two dimensional space near the *Service Points* that they use (e.g., especially near commonly used services like HTTP, FTP, DNS). To avoid this problem, we introduced a third dimension to represent the overall traffic load of the hosts. Consequently, the hosts with higher traffic load are close to the ceiling of the 3D dimension, while the hosts with lower traffic load remain near the *Service Usage Plane* (see Fig. 1 and Fig. 3).

In order to improve this view, there are three main issues that are to be addressed, namely the *Inbound/Outbound* problem, the *sliding time window* problem, and the *ports/IPs* problem.

1) *Inbound/Outbound*: The *inbound/outbound* traffic with respect to a host is defined as the number of bytes received/sent by that particular host in a predefined time interval τ . Being able to show both *inbound* and *outbound* activities for a host is of great importance. The *outbound* traffic represents the *active behavior* of a host, by its use being easier to identify attackers (e.g., flooding attacks). Conversely, the *inbound* traffic is more important in terms of locating the victim reflecting its *passive behavior* (e.g., distributed DoS attack, *Smurf* attack, flooding attack, to name a few).

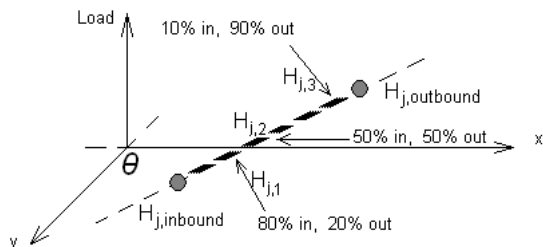


Fig. 4. The *Inbound/outbound* ratio

Assume the situation in Fig. 4. There $H_{j,outbound}$ represents the (x, y, z) position of j -th host in the *Services Behavior View* with respect to its *outbound* traffic. Furthermore, the $H_{j,inbound}$ represents the (x, y, z) position of the same host with respect to its *inbound* traffic. This type of scenario is

encountered for almost all hosts in the network (since most of the time the *inbound* and the *outbound* traffic are different). If we want to represent both of the behaviors in the same view (see Fig. 4) we have to consider and display both of the locations. Unfortunately this will introduce twice as many points as the normal number of hosts, making the information in the view hard to distinguish. In addition, if two locations are considered for each host, a new graphical mechanism must be inserted to link the two points.

It is worth noting that all the possible combinations of *inbound* and *outbound* activity for a particular host lie on the segment defined by the two points $H_{j,inbound}$ and $H_{j,outbound}$. The $H_{j,inbound}$ corresponds to 100% *inbound* and 0% *outbound* activity, while the $H_{j,outbound}$ corresponds to 0% *inbound* and 100% *outbound* activity. For example, the points $H_{j,1}$, $H_{j,2}$, $H_{j,3}$ correspond to (80% in, 20% out), (50% in, 50% out), and (10% in, 90% out) respectively. Let us define r_j as the *Inbound/Outbound Ratio* for the j -th host, where $r_j \in [0, 1]$. Accordingly, if $r_j = 0$ the current position H_j of the *host* j will coincide with $H_{j,inbound}$. Furthermore, if $r_j = 1$ then the current position will coincide with $H_{j,outbound}$. As a formal definition the H_j position will be computed as

$$\begin{aligned} x_j &= x_{j,in} + (x_{j,out} - x_{j,in}) \cdot r_j \\ y_j &= y_{j,in} + (y_{j,out} - y_{j,in}) \cdot r_j \\ z_j &= z_{j,in} + (z_{j,out} - z_{j,in}) \cdot r_j \end{aligned} \quad (9)$$

where $(x_{j,in}, y_{j,in}, z_{j,in})$, $(x_{j,out}, y_{j,out}, z_{j,out})$, and (x_j, y_j, z_j) are the coordinates for the $H_{j,inbound}$, $H_{j,outbound}$, and H_j points, respectively. In conclusion, by tuning r_j , the user has the ability to observe the morphing between the *active* and *passive* behaviors of the hosts.

2) *Sliding Time Window*: The behavior of a host in a network is time-dependent; consequently, it is important for the visualization system to show the changes in behavior over time. Storing and displaying all historical information for each host is an expensive and infeasible task. On the other hand, considering only one time interval does not allow the time-dependent features to be characterized.

The *sliding time window* represents a possible solution to this problem. By its use, we observe the behavior of a host in time as the *time window* moves forward at each *basic time interval* τ . We also define the size m of the *sliding time window* as a multiple of τ .

Recall that the *basic time interval* τ is the unit interval in which the real-time data is collected, processed, and displayed. Accordingly, for each τ a new value $\vec{F}_{k,j}$ will be computed. Let us define $\vec{F}_{k,j,n}$ as the *attraction force* that is computed in the n -th *unit interval* τ for the j -th *host* with respect to the k -th *service*.

To compute the *time-dependent features*, a *Short-Term Memory Mechanism* for each host was implemented. By its use, the position of a host is computed in terms of its traffic load in the past m time intervals. Different weights are applied to each time interval; e.g. the closer a τ interval is to the current time, the larger its weight will be. Formally, we can redefine the *attraction force* from Equation (1) as

$$\vec{F}_{k,j,n} = \sum_{t=n-m}^n A_k \cdot L_{k,j,t} \begin{bmatrix} \cos\alpha_{k,j,t} \\ \sin\alpha_{k,j,t} \end{bmatrix} \cdot e^{-|n-t|}, \quad (10)$$

where m is the number of τ intervals that are considered in the *short-term memory*; $\vec{F}_{k,j,n}$ is the *Attraction Force* at time n for the j -th host with respect to the k -th service; A_k is the anomaly factor for the k -th service; $L_{k,j,t}$ is the *Load* at time t for the j -th host with respect to the k -th service; $\alpha_{k,j,t}$ is the angle $\alpha_{k,j}$ at time t . Finally, e^{-1} is the *unit delay operator*; that is, e^{-1} operating on load $L_{k,j,t}$ at time t yields its delayed version $L_{k,j,t-1}$.

3) *Ports/IPs*: A valuable type of information is the number of *ports* and *IPs* being used by a particular host. This information helps us to identify some *probing attacks* such as *Portsweep* and *IPsweep*.

The easiest way to display this information is to introduce a new graphical element for each host, such as a *ray* attached to the host circle (see all figures from the *Experimental Results* Section). The interpretation of these rays is different between internal and external hosts. If the host is an internal one, then the rays represent the number of ports that have been opened or scanned by other hosts. On the other hand, if the host is an external one, then the rays represent the number of internal hosts it uses.

Because we are targeting slow scanning attacks, a *Long Term Incremental Update Mechanism* is implemented. By its use, the system memorizes for each host all the *ports* and all the *hosts* that it has been using within the last hour, those values being updated at each τ interval.

B. Category View

In this view the hosts are sorted with respect to a particular relevant attribute. For instance if we talk about the *port scanning* attack, the *number of ports* attribute is the one that is directly relevant. Furthermore, the *IP scanning* attack directly influences the *number of hosts* that are accessed by the attacker. Consequently, two views are defined that share the same graphical idea but sort the hosts by considering different attributes, namely *No. of Hosts View* and *No. of Ports View*.

The contribution of this view is not the host sorting process but the way in which the data is displayed. Throughout our study, we realized that normally the *number of hosts* and the *number of ports* that a particular host is accessing varies from network to network but stays relatively low. For instance, if we define the range of $[0, 100]$ in a 20 second time interval, most of the hosts will reside in the $[0, 20]$ interval, while only a few will exceed that limit. Since we are not interested in the majority of them we can split the $[0, 100]$ interval into a user defined number of intervals k (see Fig. 5 where $k = 7$). Consequently, all the hosts that exhibit an abnormal behavior will reside in the upper part of the view, while the rest of them will crowd together in the lower part.

Although the basic idea of the view is a simple one, this view proves to be a very suitable approach when it comes to fast *IP* or *Port* scans. The experimental results are not

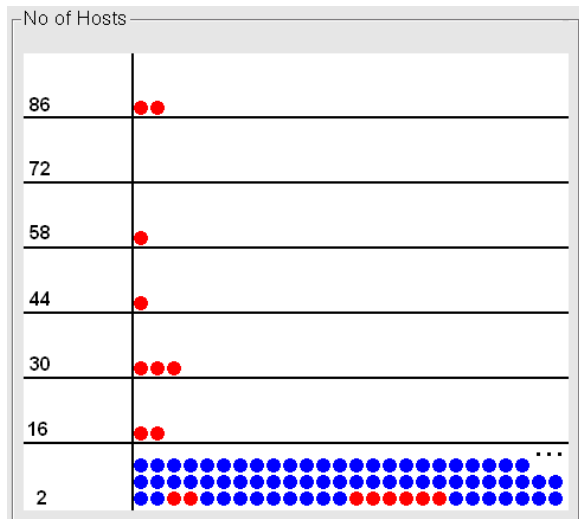


Fig. 5. The No. of Hosts View. The hosts are sorted by the number of IPs that they are using in a predefined time interval τ .

presented on the *category view* because the DARPA dataset has neither a fast *IP* scan nor a fast *Port* scan [8], but we are confident that in the real world this view is a very important source of information.

IV. IMPLEMENTATION

The views are implemented as a Distributed Network Visualization System (DNVS). This system is not intended to be a replacement for an IDS (Intrusion Detection System), being just a complementary tool used to visualize the behavior of hosts and detect anomalies within the network. The main goal of DNVS is to provide a coherent “image” of the network that will help the network administrator in identifying abnormal behaviors.

The main goal, as a system, is to be highly flexible when it comes to different network topologies. Accordingly, DNVS is composed of three major components: *Flow Generator Module*, *Visualization Module*, and *Communication Module*, each of which having its own importance.

A. The Flow Generator Module

This component is responsible for collecting the data from data sources, processing the collected data, and sending data in response to queries initiated by the *Visualization Module*. DNVS supports two types of data sources, namely network and database. The former data source collects real-time data directly from the network, while the later data source collects saved data from TCPdump files. The *Flow Generator Module* resides on a dedicated computer connected to the network. DNVS has support for multiple *flow generators* being adaptable to any kind of network topology.

B. The Visualization Module

The central component of the DNVS is the *Visualization Module*. This module receives information from all the *Flow Generators* in the network, centralizes and processes the data,

and renders the graphical information. Here are implemented all the views discussed in this article. By its use the administrator is able to select hosts, extract information about them, and apply requests to all *Flow Generators*.

C. The Communication Module

The backbone of the system is the *Communication Module*. Being implemented on top of the TCP layer, it provides reliable routines for handshaking between the two previous components, as well as support for data exchange. Here are also defined all the data filters that are to be sent from the *Visualization module* to the *Flow Generator Module*.

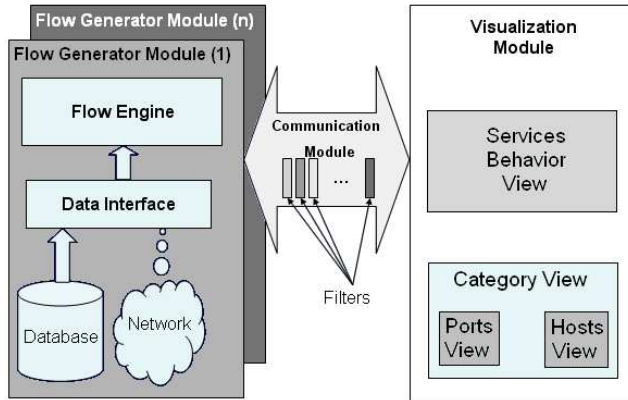


Fig. 6. The DNVS architecture.

As a developing platform, the DNVS is implemented in C++ under Linux. The Flow Generator Module also uses two other open source libraries (i.e. libpcap library [10] is used for capturing the network data, and MySQL++ API [12] is used for accessing the MYSQL [11] database). The *Visualization Module* involves many graphical elements for the GUI and the views. The main GUI is built using QT X11 free edition [13], and provides the basic tools that the user needs to interact with the system. Furthermore, the views are implemented using Mesa OpenGL [14] and GLUT library [15] for the 3D visualization. In addition to the graphical elements, DNVS employs multi-thread support in order to be able to communicate with multiple *Flow Generators* simultaneously.

Porting the DNVS between different operating systems is an easy job as long as the OS supports the POSIX-threads, and has a corresponding library implementation for each library used by our system.

V. EXPERIMENTAL RESULTS

In order to test our system, we used the DARPA 1999 Intrusion Detection Evaluation Dataset [8]. This dataset is a publicly available collection of network traffic collected during five weeks of simulation. The data is available in TCPDump format, which makes it possible to simulate real network traffic. Moreover, this data contains network based attacks in the midst of normal background data, and detailed information about each attack described in a file called *Identification Scoring Truth* under headings such as attack name, starting

time, duration, attacker, and victim. This information provides a good way for us to verify the abnormal behavior. In our experiment, two days of outside TCPDump data collected during the fourth week of simulation were replayed by the *Flow Generator Module*. Our experimental results showed that DNVS is able to highlight anomalies like *Portsweep*, *IPsweep*, and some *DoS* attacks (see [8] for documentation regarding these intrusions).

A. Portsweep

Fig. 7 shows an example of *Portsweep*. The victim is an internal host whose ports are being scanned. The scanned ports are represented in the figure by black rays. In this particular case the scanning process is a very slow one, but the system is able to detect it using its *Long Term Incremental Update Mechanism*. Our system can detect only the victim of the *Portsweep* attack, while it can detect only the attacker for an *IPsweep* attack.

A potential abnormal behavior can be detected in the upper-right corner of the figure. There, an external host is connecting to five internal IP addresses signaling a potential abnormality. There are two cases that can be encountered here: the external host can be a server (e.g., Yahoo, Google) that provides a service being accessed by a group of internal hosts, or the external host is not a server. While in the first case there is no anomalous behavior, in the second one the external host is performing IP scanning over the network. Although the two cases are similarly displayed, the network administrator will be able to discriminate between the two by exploring more detailed information about the anomaly.

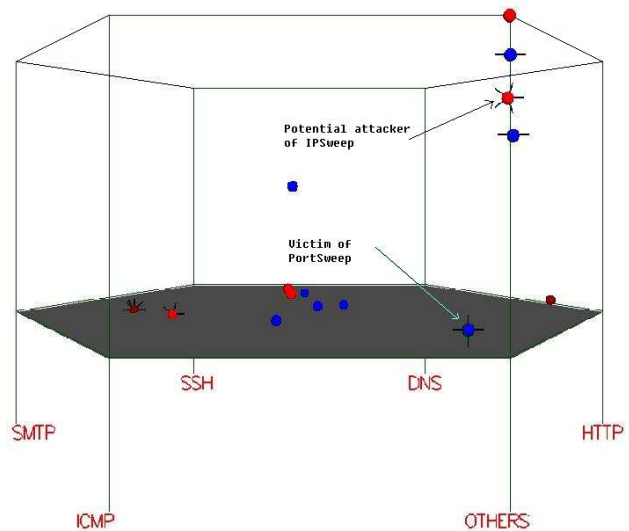


Fig. 7. The *Portsweep* attack.

There are two more potential anomalies in the bottom-left corner of the figure that turned out to be just overlaps between multiple hosts. In this case, by selecting that region, the administrator can determine how many hosts overlap. Furthermore, another reason to disregard that anomaly is the low load that those hosts exhibit.

B. Smurf

Smurf is a type of *DoS* attack which shows a high level of anomaly in the *Services Behavior View*, because the victim gets a large number of ICMP 'echo reply' packets in a short period of time. This behavior is very different from the normal situation. Even though the victim might also use other services at the same time (e.g., FTP, HTTP, DNS), and the traffic load with respect to those services could be higher than ICMP, the position of the victim will still be close to the ICMP *service point* because of the ICMP's *anomaly factor*. Furthermore, by changing the *inbound/outbound* rate, one can also identify the other hosts that are used by the attacker for the *Smurf* attack.

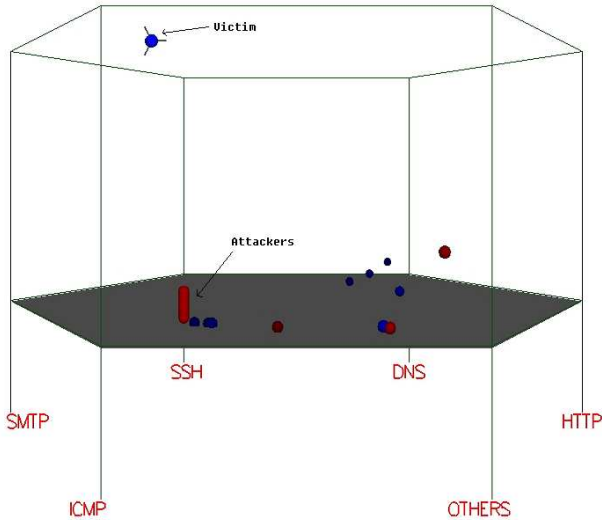


Fig. 8. The *Smurf* attack, when $r_j = 0.02$, showing the *inbound* activity of the hosts which can be associated with their *passive behavior*.

Fig. 8 shows the *Smurf* attack when the *Inbound/Outbound Ratio* is set to $r_j = 0.02$ meaning that the view shows the *inbound* activity. Here, the *inbound* of the victim is very high (*passive behavior*) while the attackers' *inbound* is very low. In Fig. 9, where $r_j = 0.95$ one can easily identify the *active behavior* of the attackers, while the *outbound* of the victim is almost zero.

C. Mailbomb

Fig. 10 illustrates a *Mailbomb* attack encountered in the second day of the fourth week. *Mailbomb* is a *DoS* attack in which the attacker sends many messages to a server overflowing that server's mail queue. As a result the server is slowed down being unable to promptly respond to clients and possibly crashing.

The first sign that an abnormal behavior is encountered is the high load of the external host identified near the SMTP *service point*. Furthermore, the external host (attacker) stays in that region for a fairly long period of time.

The reason that the suspicious attacker is easy to identify here is that all the emails sent to the mail server in the DARPA dataset are small size emails.

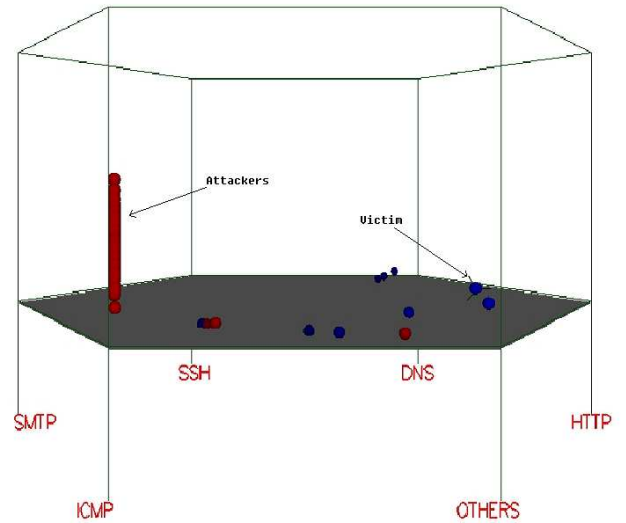


Fig. 9. The *Smurf* attack, when $r_j = 0.95$, showing the *outbound* activity of the hosts which can be associated with their *active behavior*.

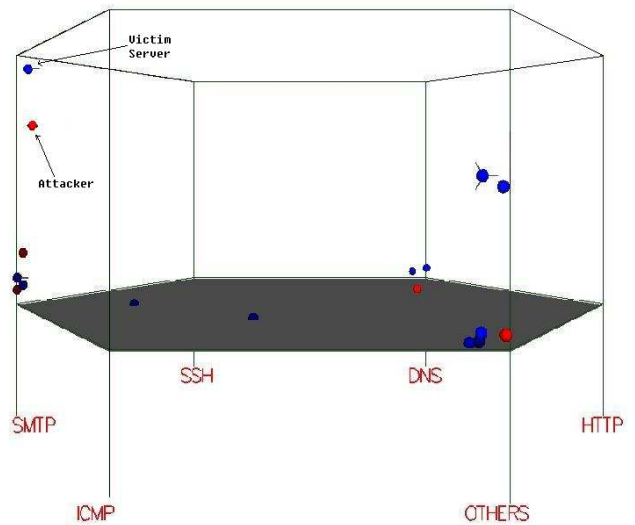


Fig. 10. The *Mailbomb* attack.

In real world scenarios a *Mailbomb* attacker would be harder to detect because the aim of this type of attack is not to produce a high traffic volume, but to create a large number of messages in the server's queue. Therefore, by looking only at the IP header, a normal host might exhibit behavior similar to that of an attacker since it could send mail with large attachments to the SMTP server. Consequently, our system may not be able in some cases to provide visualizations that will allow *Mailbomb* attack to be reliably identified.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, a novel graphical technique is proposed for network traffic flow visualization, which highlights the features of the network most vulnerable to intrusions. Two types of graphical views are presented here, namely the *Services*

Behavior View and the *Category View*. Both of the views rely on the fact that the hosts in the network can be clustered with respect to their behavior into normal and abnormal categories.

The experimental results produced using the DARPA 1999 dataset also are promising. The experiments show our visualization technique to be a good medium when trying to identify possible anomalies of the network such as DoS types of attacks (e.g., *Smurf* and *Mailbomb*) as well as *probing* attacks (e.g., *Portsweep* and *IPsweep*).

The Distributed Network Visualization System (DNVS), which implements this graphical technique, is a valuable tool for network administrators, helping them to monitor the network traffick and identify potential anomalies.

DNVS is an on-going project; we are still in the middle of its development. Our future plans include improving the performance of the system and transforming it from a passive display system to an interactive system where anomalies are signaled to the user and views are automatically changed with respect to the potential anomalies that are detected; to this end, an Anomaly Detection Module (ADM) will be incorporated into the DNVS, and more views will be developed in order to visualize not only the network data flow, but also the alerts generated by the ADM.

ACKNOWLEDGMENT

The authors would like to express their deepest thanks to Xinge Du for his work on the *data conversion* for the *Flow Generator Module*.

This work was funded in part by grant RGPNT227441 from National Science and Engineering Research Council of Canada.

REFERENCES

- [1] M. Fisk, S. A. Smithy, P. M. Webery, S. Kothapallyz, and T. P. Caudellz, *Immersive Network Monitoring*, 2003 Passive and Active Measurement Workshop.
- [2] R. F. Erbacher, *Visual Traffic Monitoring and Evaluation*, Proceedings of the Conference on Internet Performance and Control of Network Systems II, Denver, CO, August, 2001, pp. 153-160.
- [3] Nyarko, K., Capers, T., Scott, C., and Ladeji-Osias, K., *Network intrusion visualization with niva, an intrusion detection visual analyzer with haptic integration*, In Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, page 277. IEEE Computer, 2002.
- [4] R. Erbacher and D. Frincke *Visual Behavior Characterization for Intrusion and Misuse Detection*, Proceedings of the SPIE '2001 Conference on Visual Data Exploration and Analysis VIII, San Jose, CA, January, 2001, pp. 210-218.
- [5] C. Zhao and J. Mayo, *A TCP/UDP Visualization Tool: Visual TCP/UDP Animator(VTA)*, ICEE International Conference on Engineering Education UMIST, Manchester, UK 18 - 22 August 2002.
- [6] R. F. Erbacher and K. Sobylyak, *Improving Intrusion Analysis Effectiveness*, 2002 Workshop on Computer Forensics, Moscow, ID, September, 2002.
- [7] V. Paxon, *Automated Packet Trace Analysis of TCP Implementations*, In SIGCOMM, pages 167C179, 1997
- [8] Lincoln Laboratory, *1999 DARPA Intrusion Detection Evaluation Data Set*, 1999, http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html August 20, 2004, last access.
- [9] Q1Labs, *The "QVision" product* (renamed in March 2004 as "QRadar"), <http://www.q1labs.com/>, May 15,2004, last access.
- [10] *TCPDUMP public repository*, <http://www.tcpdump.org/>, May 15,2004, last access.
- [11] *MYSQL* <http://www.mysql.com>, May 15, 2004, last access.
- [12] *MSQL++ API* <http://dev.mysql.com/downloads/other/plusplus/index.html>, May 15,2004, last access.
- [13] *Qt/X11 Free Edition* <http://www.trolltech.com/download/qt/x11.html>, May 15,2004, last access.
- [14] *The Mesa 3D Graphics Library* <http://www.mesa3d.org/>, June 15,2004, last access.
- [15] *GLUT - The OpenGL Utility Toolkit* <http://www.opengl.org/resources/libraries/glut.html>, July 15,2004 , last access.