FireViz: A Personal Firewall Visualizing Tool

by

Nidhi Sharma

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Masters of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2005

© Massachusetts Institute of Technology 2005. All rights reserved.
FireViz: A Personal Firewall Visualizing Tool

by

Nidhi Sharma

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Computer Science and Engineering

Abstract

In this thesis, I present FireViz, a personal firewall visualizing tool. FireViz visually
displays activities of a personal firewall in real time. The primary goal of FireViz is
to educate typical computer users of the security threats their computers are exposed
to when connected to a network and expose any potential loop holes in the firewall’s
security policies. To this end, FireViz presents a novel visualization paradigm that
provides users with an informative yet non-intrusive interface to their network and
firewall. FireViz achieves this by incorporating a peripheral mapping of the network
on the user’s screen and displaying network events along this periphery. Information
about network events is encoded visually to help users develop a more intuitive model
of the network at a low cost. The peripheral nature of FireViz along with the use of
non-modal visual displays allows users to easily understand network events without
obstructing their primary tasks. The visualization is also capable of highlighting
exceptional events that may represent potential threats without relying on the end
user to understand any threat model. Thus, FireViz aims to provide users with
a superior framework for understanding the network security model and achieving
improved system security in the process.

Thesis Supervisor: Robert C. Miller
Title: Assistant Professor
Acknowledgments

I would like to thank my outstanding thesis advisor, Robert Miller, for his advice, support and encouragement throughout the project. I am also very grateful to him for keeping faith in my work even through times of slow progress. Your passion and commitment have served as a source of great inspiration. Thank you very much.

I would like to thank members of the User Interface Design Group at MIT’s Computer Science and Artificial Intelligence Laboratory especially, Maya Dobuzhskaya, Michael Bolin, Simson Garfinkel, Phil Rha, Victor Costan, Vineet Sinha, David Tsai, Matthew Webber and Min Wu; David Nelson, David Broniatowski, Yue Hann Chin, Charuleka Varadharajan and Miguel Ferreira for their helpful comments.

I am grateful to the MIT Financial Aid Office and the EECS Department for providing me financial support; Daniel Jackson, George Verghese and Tony Eng for giving me opportunities to teach at this remarkable institution; Sanjoy Mitter and Marc Baldo for being remarkable academic advisors and for providing me invaluable opportunities in research and academics; many members of the extended Gaur and Sharma families and my friends and teachers at home and at MIT for teaching me so much I know and for standing by me through my metamorphoses.

I am very thankful to Ed and Paulette Schwartz for being excellent mentors and for being my family away from home; Manu Nellutla for his indispensable support and my brother, Abhinav for being the best older brother one could ask for. Your support and affection has given me immeasureable strength.

Finally, I would like to thank my parents, Usha and J.C. Sharma for giving me everything that I have. I am so grateful for your innumerable sacrifices for giving me this incredible opportunity, for knowing me and believing in me more than anyone, for giving me the strength to continue my education in spite of many challenges, for giving me faith when I needed it most and most importantly for always being there for me. Your faith and love have provided me the only truth I have known in my life. Thank you.
## Contents

1 Introduction ................................................................. 15
   1.1 Motivation .............................................................. 15
      1.1.1 The Need for User Education ................................. 16
      1.1.2 The Need for Visualization and Usability Engineering .... 19
   1.2 FireViz .................................................................... 20
   1.3 Thesis Contributions ................................................... 24
   1.4 Thesis Overview .......................................................... 26

2 Related Work .................................................................. 27
   2.1 Network Traffic Analysis ............................................. 27
   2.2 Network Traffic Visualization ........................................ 28

3 User Interface ............................................................... 31
   3.1 User Interface Design Goals ......................................... 31
   3.2 UI Features and Rationale ............................................. 32
      3.2.1 Visual variables in User Interface Design .................... 34
      3.2.2 Encoding Network information in FireViz .................. 35
   3.3 Usability Engineering .................................................. 41
      3.3.1 User Analysis ....................................................... 41
      3.3.2 Task Analysis ....................................................... 44
      3.3.3 Paper Prototypes ................................................... 45
      3.3.4 Computer Prototype ............................................... 46
      3.3.5 Hi-fidelity Prototypes .............................................. 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>A Mapping of the Network</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>IP based mapping</td>
<td>52</td>
</tr>
<tr>
<td>4.2</td>
<td>Application based Mapping</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>Frequency based Mapping</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>Geographical Location based Mapping</td>
<td>57</td>
</tr>
<tr>
<td>4.5</td>
<td>Domain based mapping</td>
<td>59</td>
</tr>
<tr>
<td>4.6</td>
<td>Adaptive Domain based mapping</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>Implementation</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>Implementation Overview</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>TCP Connection Module</td>
<td>69</td>
</tr>
<tr>
<td>5.3</td>
<td>The Firewall Module</td>
<td>70</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Choice of Firewall</td>
<td>70</td>
</tr>
<tr>
<td>5.4</td>
<td>The Display Engine</td>
<td>71</td>
</tr>
<tr>
<td>5.5</td>
<td>The Mapping Module</td>
<td>71</td>
</tr>
<tr>
<td>5.6</td>
<td>The DNS Module</td>
<td>72</td>
</tr>
<tr>
<td>5.7</td>
<td>FireViz Display Objects</td>
<td>72</td>
</tr>
<tr>
<td>5.8</td>
<td>Implementation Challenges</td>
<td>75</td>
</tr>
<tr>
<td>5.8.1</td>
<td>Choice of Firewall</td>
<td>75</td>
</tr>
<tr>
<td>5.8.2</td>
<td>Stroke Model in Display</td>
<td>76</td>
</tr>
<tr>
<td>6</td>
<td>Evaluation</td>
<td>77</td>
</tr>
<tr>
<td>6.1</td>
<td>Testing Methodology</td>
<td>77</td>
</tr>
<tr>
<td>6.2</td>
<td>Test Results</td>
<td>78</td>
</tr>
<tr>
<td>6.3</td>
<td>Subjective Feedback on the Interface</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion and Future Work</td>
<td>83</td>
</tr>
<tr>
<td>7.1</td>
<td>Thesis Contributions</td>
<td>83</td>
</tr>
<tr>
<td>7.2</td>
<td>Applications</td>
<td>85</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Work</td>
<td>86</td>
</tr>
<tr>
<td>7.3.1</td>
<td>Complete Integration with a Firewall</td>
<td>86</td>
</tr>
</tbody>
</table>
7.3.2 Context-Aware Activity Logs ........................................ 86
7.3.3 Host Intrusion Detection with FireViz ............................. 87
7.3.4 Adapting FireViz to Multi-Screen Monitors ...................... 87
7.3.5 Incorporating Geographical Location ............................... 87
7.3.6 Long-term User Testing ............................................. 88

A User Study Briefing ..................................................... 89
List of Figures

1-1 Zone Alarm Alert Dialogs .................................................. 17
1-2 Sygate Dialog requesting user action ................................. 18
1-3 FireViz displaying connections made by AOL Instant Messenger . . 24
1-4 FireViz System Diagram ...................................................... 25

3-1 Zone Alarm Connection Alert ............................................. 32
3-2 Visual Variables .............................................................. 34
3-3 FireViz’s Line and Label display from Internet Explorer ........... 37
3-4 FireViz’s Activity lumps displaying Connection Tooltips ........... 38
3-5 FireViz displaying a flashing starburst for a background process . . 40
3-6 FireViz’s Starburst activity lump displaying Connection Information . 41
3-7 FireViz’s displaying a red starburst for a blocked activity ........... 42
3-8 First Hi-Fi Prototype of FireViz displaying a Network Connection . 47
3-9 Tooltip in the First FireViz Prototype ................................. 48

4-1 IP Address based mapping ................................................. 53
4-2 Application based mapping ................................................. 54
4-3 Connection Frequency based mapping ................................. 56
4-4 Geographical location based mapping ................................. 58
4-5 Domain based mapping ...................................................... 60
4-6 Visual cues in Domain based Mapping ................................. 61
4-7 State Stored within an Adaptive Algorithm ......................... 63

5-1 FireViz Module Dependency Diagram ................................. 68
List of Tables

3.1 FireViz User Characteristics ........................................... 43
Chapter 1

Introduction

The internet is playing an increasingly important role in business, education and communication. While the internet is a powerful means to establish connections to other remote hosts and effectively share useful data, it also serves as a medium to quickly and widely spread malicious code. Any host connected to a network is liable to be compromised by various means. Given this hostile internet environment, it is imperative that every computer connected to the internet be protected by appropriate means. While organizations have a greater incentive in securing their hosts and networks, ordinary home users are generally unaware of both the risks and the measures for preventing attacks. However, few personal security tools focus on educating users about the threats they are exposed to.

This thesis describes a personal network visualization tool called FireViz. FireViz is designed to provide real-time visualization of the network processes on a given host and reveal potential threats and holes in the firewall’s security policies. This chapter provides the motivation and an introduction to FireViz.

1.1 Motivation

The need for network visualization has been fueled by the need to gain a better understanding of the network security model. A concrete understanding of a network security model can only arise from gaining education about the network character-
sitics and creating means for effectively providing users such education. Thus, Fire-Viz’s motivation comes from providing computer users a more intuitive and helpful understanding of the network.

1.1.1 The Need for User Education

Almost every computer connected to a network is constantly scanned for various security vulnerabilities. Malware such as worms are self-replicating programs that scan the network for vulnerable hosts and infect them. However, unlike viruses, worms need no user initiation, such as opening a file, and hence spread very quickly and easily. Attacks launched by viruses and worms, among others, quickly compromise the integrity of the affected host and spread to many others. Over the last few years, both the intensity and frequency of such network-based attacks have increased rapidly. In the year 2004 alone, internet users have been confronted by an estimated 100,000 forms of malicious attacks [1].

The frequency however is not the only cause for concern today. The increasing intensity of these attacks and the shrinking time lag between when vulnerabilities are announced and when they are exploited are magnifying the security problem. A study of network worms launched over the last 24 months shows the time lag shrinking from 330 days for the Nimda Worm in 2002 [2, 3], to 16 days for the Sasser in April 2004 [4, 5]. According to Gartner research [6], this pattern will only get worse. The study projects a 25% increase over the next several years in 'Zero Day' attacks, which exploit software vulnerabilities that have no known fixes [7]. Given these statistics, network users have to stay current with the existing security mechanisms that are already struggling to keep up with the sophistication of attackers today. As a result, the tolerance for any laxity in maintaining computer security is diminishing quickly. Statistics report that the average survival time for an unprotected computer fell from 40 minutes in 2003 to a mere 20 minutes in 2004 [8].

Many commercial developers are promoting tools to monitor and protect individual computers. Personal network firewalls such as Zone Alarm [9], Kerio [10] and Sygate [11] succeed in detecting and blocking numerous unfriendly network probes.
However, few personal security applications focus on informing the user of the extent or nature of these threats. This is particularly evident when examining the user interfaces of these firewalls. Zone Alarm, for example, allows users to grant internet access to programs on application granularity. Once applications are deemed as trusted, the user is not given any further feedback on their network activity. This eliminates all feedback for users about potential attacks launched through these trusted applications. Zone Alarm does, however, provide users with information on port scanning activities. However, this information is provided to the users in very intrusive ways. As a result, many users turn off feedback from port scans, further reducing the information they may receive.

Figure 1-1: Zone Alarm Alert Dialogs

(a) Zone Alarm Information Alerts are Modal Dialogs that stack on top of each other

(b) Zone Alarm dialog requesting user action with limited information
Figures 1-1 and 1-2 highlight some of the drawbacks in the user interfaces of contemporary security software. As evident from this sampling, most security software provide users with little useful information while requiring them to make decisions on the spot. These information dialogs are almost always modal, mandating user input in some form or another and are extremely distracting. The modality of these dialogs also distracts the users’ attention from the main task that they are performing and consumes resources such as CPU time and screen estate that could be used for users’ primary goals. All these drawbacks either lead users to turn off all feedback features, thus defeating their very purpose, or lead them to making decisions that may be incorrect - such as allowing access to potentially malicious code. Thus, most security software today fail to provide useful information to users regarding the security of their computers.

Security is a complicated and important aspect of using computers today. It is therefore essential for all users to understand the vulnerabilities their computers are exposed to. Given current trends, awareness is irreplaceable for survival. To this end, many network security applications have singularly failed to educate the very users they hope to protect. This lack of awareness also provides negative feedback by reducing users’ motivation to run personal security software at all times. It is no surprise then that a majority of internet users today fail to appreciate the reality of the security threats they are exposed to. In a study conducted by the NCSA, more than a third of the users surveyed said that they had a greater chance of winning the
lottery than being hit by malicious code [12]. This is not just characteristic of novice users. Many sophisticated computer users are also unaware of the sheer volume of such threats. At the SC03 conference in 2003, many expert computer users were surprised at the number of malicious attempts at the conference’s high bandwidth network [13]. All these statistics suggest that there is immediate need for computer users to be educated of the potential threats they are exposed to.

1.1.2 The Need for Visualization and Usability Engineering

While many personal security applications do not provide any feedback or visual cues about network activity, they do maintain activity logs to a certain extent. These logs can, in principle, be used to find unexpected or anomalous network behavior that may be potentially harmful. The analysis of these logs can either be automated or conducted manually by an expert user. Automatic analysis is based on statistical modeling and machine learning. However such data mining techniques minimally engage human interaction and visualization and are likely to miss important features in the data. Manual inspection on the other hand is extremely tedious and both approaches are unreasonable to be expected from typical network users. Most importantly, these logs lack the situational context about what the user was doing at the time of the network activity. Such situational context is often more important than nominal connection information to detect anomalous or unexpected activity.

The human perceptual processor is capable of very fast visual processing; therefore, information depicted visually is easier to process and make out interesting patterns and features [14]. Consequently, tools that use visualization can leverage this human capability to enable users to easily discern anomalous network events. FireViz is a personal network firewall analyzer that provides information visualization in ways that take advantage of the huge bandwidth of the human vision sensory system. It provides quick, real-time network information for users and uses visual cues to aid in detecting anomalous activities. The use of visual cues is especially useful when the users are unaware of what information to particularly look for, which is typical of network activity monitoring. Thus effective network visualization tools can therefore
come a long way in educating users of their specific security needs.

1.2 FireViz

FireViz is an attempt at solving both these problems - showing users their network characteristics and doing so effectively. FireViz works in conjunction with the user’s personal firewall and provides a more intuitive window to the network on which the firewall acts. It detects all network activity in real-time and visually displays the information highlighting the most crucial pieces (such as host application, remote location and the firewall security policy), while allowing the user to analyze any subset of the events more closely.

The visualization of the Firewall’s actions can help users get a sense of the volume and nature of network events. For instance, if the user sees that a large number of connections are being blocked by the firewall then she may get a sense that her computer is trying to get broken into. This way, FireViz helps to create a more intuitive model of the network weather for the user. Similar to the climate of a place, users can deduce if the weather on their network is calm (little activity) or stormy (excessive, potentially harmful activity).

The visualization employed in FireViz is capable of highlighting exceptional events especially when they occur along side more expected events. FireViz can be used to display all traffic seen by the firewall or only filtered traffic (such as TCP connections only). FireViz is therefore capable of highlighting the following attacks:

- **Attacks launched through trusted applications:** Most firewalls allow users to create rules specifying which applications should be allowed network access. However, once applications are trusted it is extremely difficult for users to detect malicious activities launched through these applications. For instance, if a user checks her email at mailsite.com through her web browser (which is a trusted application in the firewall), a connection to mailsite.com is expected. However if a certain email (possibly spam) makes an unauthorized connection to badsite.com, FireViz will display this as well. The user should then be able
to recognize this an unexpected activity and create a new rule in her firewall. The power of FireViz’s visualization is its ability to provide a contextually rich description of such events. A mere log entry of the web browser making an access to badsite.com encapsulates little meaning. The fact that the access occurred while the user was using her browser to read a specific piece of email makes the event exceptional. FireViz preserves this situational context in its display.

- **Spyware activity:** Since FireViz displays all activities, whether or not the applications involved in them are trusted or not, it is easy to detect when an application initiates many network events. Thus spyware processes become more and more visible as their activities increase. This is particularly useful since spyware often sneak past firewall rules. Again users can recognize such processes as they make multiple connections throughout the internet and can change their firewall rules to disable such applications.

- **Port Scans:** Port scans help attackers exploit open ports to send malicious data. Essentially, a port scan consists of sending a message to each port, one at a time. Since such scans may not be associated with specific applications, firewalls either just allow or block such events without asking the user for a decision. Moreover, fragmented packet port scans\(^1\) and stealth port scans\(^2\) are quite capable of bypassing firewall rules. The ability of FireViz to visualize these events is an extremely useful technique in highlighting such loop holes in the firewall.

FireViz is based on a simple philosophy - *You cannot protect when you can not see the threat.* FireViz is used in conjunction with tools that enforce security policies

---

\(^1\)some scanners split the TCP headers into multiple IP packets. Since firewalls need a complete header to check data against their rules, fragmented TCP headers are often ignored by the firewall.

\(^2\)A stealth port scan is designed to be undetectable by firewalls. A common characteristic of port scans is sending many similar packets to multiple ports on the same host. However when such packets are distributed over a period of time, they generally do not resemble most port scans and are undetected.
- such as personal firewalls. However, no security system is perfect and hence there is constant need to update not just the policy-enforcing tools but also the policies themselves. However, users cannot be expected to update their security policies unless they know what the security policies should be. Furthermore, users cannot know what the security policies should be unless they have an understanding of which aspects of network behavior are expected and which ones are potential threats. This issue is further complicated by the fact that most users, when using the network, do not have security as a primary goal. This insight forms the basis of FireViz’s emphasis on the visualization of network security at a low cost to the user.

FireViz is designed to be useful to even novice computer users. It aims to achieve the following main goals -

- **Educate users of their network conditions.** The primary goal of FireViz is to educate users to keep up their defense mechanisms at all times by providing them with a more concrete network security model.

- **Provide users with immediate feedback on all network activity.** This is to allow users to gain a deeper understanding of the network to which their computer is connected and develop a model of expected behavior. Such a model is complemented by the situational context around each network activity. Such knowledge is helpful in detecting potentially malicious behavior.

- **Provide users with the right knowledge to effectively utilize their defense mechanisms.** The hope is that FireViz will help reveal weak security policies to its users. We further hope that the users will take the right steps to remedy such problems using the defense mechanisms they already use.

- **Employ effective visual techniques to display important information.** The most important user-centric goal of FireViz is to provide a novel means of viewing security information by incorporating a visually rich display. This will allow even novice users to easily discern exceptional events and examine them more closely. In order to achieve this goal, we followed these main principles -
– Since network activity is very frequent, all information must be provided in the most non-intrusive fashion as possible.

– The cost on the user to retrieve this information should be minimal. This means that costly operations such as a complete context switch of users’ attention and current application should be avoided.

– Since displaying all relevant information related to a specific connection is infeasible (since displaying more information also requires more space), the most easily available information should highlight only the most interesting features of the connection. All other information should be easily accessible if the user chooses to retrieve it.

These goals are intended to provide the following benefits to the users -

• Increased network awareness

• Reduced cost on users to gain network awareness

• Improved system security

Figure 1-3 shows an instance of FireViz displaying network accesses by AOL Instant Messenger. Figure 1-3 also shows the situational context preserved in FireViz’s display. While I sign into my AIM account, the AIM client makes a connection to buc2-vip-d.blue.aol.com but it also makes a rather unexpected connection to 64.12.25.28 at the same time. Note that, the display is not a dialog and hence does not require any user input and automatically disappears in a few seconds.

A firewall monitors all incoming and outgoing traffic on the network to which the computer is connected. FireViz works alongside the firewall and provides a visual window to the network and how the firewall, by means of its policies, manages traffic to the computer. FireViz detects network activity and the firewall’s actions and non-intrusively displays it on the user’s desktop in real time. This display may help reveal potential holes in the firewall’s policies which a user can fix by creating new rules. Figure 1-4 shows a system diagram for FireViz. FireViz provides a novel way of visualizing network events. We believe that intelligent visualization techniques will
help achieve increased network awareness for users at a low learning cost and provide improved system security.

1.3 Thesis Contributions

This thesis project, FireViz, makes the following contributions:

- First and foremost, it presents a means to visualize and understand network behavior, i.e., the activity patterns, nature of traffic and the frequency of such events in real time.

- It presents a framework that collaborates with personal security software such as a firewall and presents a visualization of the enforced security policies. This visualization can help expose potential security holes in the firewall and lead...
the user to close them and strengthen their security.

- It presents a novel visualization paradigm that focuses primarily on users’ expectations from security and provides a new direction in the development of UIs for personal security software. FireViz presents a UI that is peripheral and encodes information visually.

- Finally, it provides a framework for testing the usability of the system and presents a discussion of results from executed tests.

FireViz allows users to watch the firewall’s activities on a much finer granularity. By allowing their firewalls to trust certain applications, users are exposing their computers to a large array of threats that can exploit the vulnerabilities in these applications, which may themselves be malicious. FireViz allows users to watch every action performed by such applications and in the process find open doors to their system. This visualization can help users create stronger, more refined security rules for their firewalls and achieve a greater level of security.
1.4 Thesis Overview

Chapter 2 provides an overview of related work in the field. Chapters 3 and 4 provide a description of the design features and the UI of FireViz, Chapter 5 provides an implementation overview, Chapter 6 presents an evaluation of the UI and Chapter 7 concludes.
Chapter 2

Related Work

The motivation behind the development of FireViz intersects two important research areas - network traffic analysis and network traffic visualization, both of which have a substantial body of current research. This chapter provides an overview of relevant research in the two areas.

2.1 Network Traffic Analysis

Many users secure their computers using software mechanisms such as anti-viruses and personal network firewalls. Commercial firewalls such as ZoneAlarm [9], Kerio [10], Sygate [11] and Norton [15] make sure that the doors providing entry to the computer are not left wide open. Such doors may include vulnerable or buggy applications running on the computer that listen for network data. Firewalls isolate the host computers by intercepting each packet of data, incoming or outgoing, and selectively allowing some packets to continue, based on the security policies. The challenge for firewalls is to maintain accessibility while maintaining security. FireViz relies on both the system firewall and its own network scanning module to monitor network activity on the host it runs.

Tools such as Netstat [16] and TCPView [17] provide the current state of network connections on the host. FireViz uses such data for displaying established TCP connections in real time. However such tools may not be used to display activity that
may already have been blocked by the firewall.

FireViz creates a mapping of the network around the edges of the user’s screen. This mapping is chosen in such a way that it makes it easy to recognize safe and frequent activities and highlight exceptional events that could be potentially harmful. The design of the mapping is discussed in greater detail in Chapter 4. Users can learn to visually identify frequent events thus making it easy to discern any unexpected events. This idea is similar to Anomaly detection systems (ADS) and Intrusion Detection Systems (IDS) such as Symantec Advantage [18] and Cisco Trellis [19] that attempt to identify exceptional network patterns for intrusions. Anderson et al. [20] describe such ADS systems in greater detail.

The various ADSes and tools such as NetFlows [21] record information on unidirectional end-to-end transactions, aggregating packets into larger flows of data. However these tools best operate on whole network systems rather than focusing on individual hosts on the network. Erbacher [22] describes visualizations of collections of individual transactions on a single machine. A similar tool, NVisionIP [23] spans multiple levels of network abstractions including the entire network, a subnet or a single machine.

In addition to high level applications that monitor network traffic for individual hosts and entire networks, there exist very low level packet capture libraries as well. Packet sniffing libraries such as WinPcap [24] and Libpcap [25] provide raw access to network data without regard to the specifics of network protocols. Such libraries are used in the design of network or protocol analyzers, Network Intrusion Detection Systems (NIDS) [26] and traffic loggers. However, since these libraries only provide a protocol-independent view of the network traffic, they cannot be used to filter or manipulate such traffic and cannot be used in applications such as personal firewalls.

### 2.2 Network Traffic Visualization

The use of information visualization to display network traffic is an idea that is being widely experimented with. Information visualization mechanisms such as parallel
coordinates [27] have been specifically designed for this purpose. Parallel coordinates represent multidimensional data (such as network accesses) in two dimensions. Each dimension is represented as a vertical line and a whole event is displayed as a line connecting the values across all the dimensions. PortVis [28], which uses port-based detection of security activities, uses a visualization system that depicts the network traffic by choosing axes for important features of the connection and creating cells in a grid which represent the network activity at that point. SeeNet [29] uses a colored grid, where each point represents the amount of traffic between the hosts represented by the x and y coordinates of the point. NVisionIP [30] uses a graphical matrix representation to show relationships between events on a network. VisFlowConnect [31] uses a parallel axes representation to display network traffic both within and between domains. However, all of these tools are designed to facilitate anomaly detection in whole network systems intended to be used by expert system administrators. FireViz, on the other hand, is intended to be used by any users, regardless of their proficiency level and without any special training.

The Spinning Cube of Potential Doom [13] provides an animated display of network traffic within a 3D cube that users can spin at will. The cube is intended to be used by novice users as well. However, unlike the cube and other network visualization tools, FireViz is a real-time display - it provides a display of the activity as it happens, without the user having to explicitly request the information or context switch to the visualization tool.

Many of the tools presented here provide network level information and finding host-specific information is relatively harder. Network Eye [32] attempts to provide a more balanced Host/Network picture by preserving the context when displaying the whole network at once and showing the interactions within the hosts and their programs. It is nevertheless still meant to be used by network administrators to detect potential threats.

One of the goals of FireViz is to provide the user with situational context along with network traffic. Most security applications today have next to no means of providing a situational context (such as what the user and the application were doing
at the time of the event) when indicating activities. Security software like firewalls, merely stack up dialogs one on top of another when providing the user any information. Furthermore, these dialogs say little about how or when the access was made (or blocked) especially when multiple instances of the same application may be running.

Some work has been accomplished in wrapping more context around network activities by means of enhanced logging features. VisFlowConnect [31] employs animation to replay events recorded in the data logs as they occurred. Teoh et al. [33] describe a focus + context radial layout to manage screen real estate by showing snapshots of activity in adjacent time periods in a circle around a larger focal image. However FireViz preserves the context in its visualization. FireViz’s display engages the application presence on the user’s screen, when possible, to provide more information on specific events. Thus it is easy to see that an unknown remote host with IP 64.236.41.63 was contacted by AIM when the user was signing in as in Figure 1-3.

Network traffic visualization has numerous applications. However, little focus has been provided to using visualization for personal security software. FireViz provides a novel way to display and identify network events and is specifically targeted to ordinary computer users.
Chapter 3

User Interface

3.1 User Interface Design Goals

FireViz aims to achieve three important goals, as described in the introduction in Chapter 1:

- Educating users about their network conditions
- Providing users immediate feedback on all network activity
- Employing proper visual techniques to provide such information in the most efficient means possible.

Choosing the right visualization emerged as the single most important aspect of the development of FireViz. To facilitate the design choices of specific features, the third goal above was further designed to achieve the following:

- Since network activity may be very frequent, all information must be provided in the most non-intrusive fashion. This is essential in order to prevent FireViz’s displays from annoying users.
- The cost on users to retrieve such information must be minimal.
- The information displayed should highlight only the most important features related to any activity.
• The visualization should be able to highlight exceptional events easily.

These formed the key motivation behind a peripheral interface for FireViz. In this chapter, I provide an overview of the UI features of FireViz and how these features meet the aforementioned goals.

3.2 UI Features and Rationale

Traditional security software UIs pay little attention to the information that is displayed to the user, especially when mandating the user to make a decision based on such information. Figure 3-1 shows a Zone Alarm Alert (which is in fact more of a dialog box) asking if Internet Explorer should access 127.0.0.1 at port 1066. It is
easy to see that while this dialog asks the user to make a decision, it provides little useful information to the user. It attempts to provide some validation information, which is unavailable. Therefore the very purpose of such information is unclear. More importantly, it suggests that the program is trying to access the internet and the remote host is 127.0.0.1. For users who recognize this IP address as their local host, the information provided is contradictory. It appears that Internet Explorer is just talking to a different service on the same computer. Then, how could it be accessing the network and why is it different from regular Inter-Process Communication and being blocked by the firewall? For users who do not recognize this IP address, little help is offered to help them make the right decision. Displaying the host name might be more useful in such cases. There is no visible means to get more information on the activity. Moreover, the default option in the dialog is “Deny” and it is unclear if that is the right default for this particular connection.

Dialogs like these drive home the point that little attention is paid to the needs of the user when trying to provide information about network connections. In designing FireViz, a thorough user analysis revealed that most users care about the following information on network activity:

- The application requesting or receiving the connection
- The remote end of the connection, specifically a host name.
- The action taken by the firewall
- The direction of communication - incoming versus outgoing
- The time of the activity.
- Additionally, users care more about established connections than blocked connections.
- Users care most about activities related to applications they most commonly use.
- Finally, users like to get an overview of the network weather.
Since FireViz neither advocates, nor implements security policies, it need not ask users to make any decisions. FireViz merely provides information on what the firewall does with any given connection and help the user judge if such action is appropriate or not.

With this information, our goal was to design a visualization that met the users’ expectation in the most efficient way.

### 3.2.1 Visual variables in User Interface Design

Since the human perceptual processor is capable of fast information processing, the main focus of FireViz’s interface is to provide information visually as opposed to textually. Thus, the central theme in the design of the UI was to encode information through the use of proper visual contrast.

Contrast is a central concept in the use of visual cues. It provides the irregularities in designs that help elements stand out. Once we decided what information was to be displayed, the next step in the design of the UI was to choose the appropriate means for representing it. Bertin [34] presents seven visual dimensions of contrast. Figure 3-2 shows seven different visual variables.

Each visual variable has different characteristics such as its *scale* (the values that
it can take), \textit{length} (the number of distinguishable levels), \textit{selectivity} (can attention be focussed on one value of the variable, ignoring all others) and \textit{associativity} (can the variable be ignored when focussing on other variables). Additionally, variables can have values that are nominal, ordered or quantitative. An understanding of the strengths and weaknesses of these individual variables was key to choosing the right representation of FireViz’s information.

An additional consideration in the design of the UI involved using a natural mapping for representing information. For example, when the firewall allows a connection, it is shown in green color and red otherwise. This allows users to easily understand and learn certain aspects of the visualization.

\subsection{Encoding Network information in FireViz}

For each network activity it displays, FireViz provides important information about the event. Information representing the application involved in the activity is encoded using position. A line appears from a point within the window of the application to a point on the screen periphery, indicating exactly which instance of the an application caused the event. In the case when the application window is hidden or does not exist, a different visualization is used. This is explained further later in this section.

In order to make the remote ends of connections more meaningful to the user, we decided to use hostnames instead of IP addresses, where possible. This information is also displayed using position. The network is mapped around the periphery of the screen and the remote host is displayed at the position dictated by the mapping. In the final prototype of FireViz, we used an adaptive domain-based mapping i.e., all top level domains (such as COM, EDU, IN and UK) were mapped along the edges of the screen. The span of each domain was adapted to the volume of traffic received from a given domain. Thus when a line appears on the user’s screen, one end of the line is located within an application window and the other points to where the remote host is located on the screen. The design of the mapping is discussed in greater detail in Chapter 4.

For every network event, the firewall takes one of two actions - allow it or block
it. In choosing how to display this information, we considered cultural metaphors to provide a natural mapping to the user. The color green is associated with success and the color red is associated with prohibition. Thus the action was encoded with hue - green representing allowed connections and red representing blocked connections.

Similar to the firewall’s action, the direction of connection is also a binary variable. Connections are either incoming or outgoing. We used the orientation of the lines to represent this direction. This also provided a direct mapping to the user. Thus, incoming connections pointed to the application window and outgoing connections pointed away from application windows.

Since FireViz provides a real time display, the time of the activity is not redundantly encoded by means of any visual variables. The rest of the information specified in the user analysis was communicated through the use of the adaptive domain-based network mapping. The mapping algorithm incorporated visual variables such as position (of displays) and size (allocation of screen estate to network domains depending on the composition of the traffic) to present qualitative aspects of the network weather as a whole. Our hope is that over time, users will learn to identify events just based on where they appear on the screen.

We now provide examples of FireViz’s visualization of network events. Figure 3-3 shows Internet Explorer making an access to www.slashdot.org. The display immediately makes the following information available to the user:

- The application making the connection is Internet Explorer, as evident from the lines from the application window.
- The connection firewall allows the connection, as evident from the green color.
- The same user action (i.e., accessing www.slashdot.org) caused three different connections, evident from the 3 lines emerging from the Internet Explorer Window. This also provides an example of how FireViz preserves the situational context when displaying events.
- The remote hosts contacted are specified in the labels that accompany the lines.
The time of the access is also known as the display is in real time.

The direction of the access is outgoing, represented by the lines pointing (the narrow end) to the network (which is the periphery of the screen).

Therefore, a single label and line display of FireViz conveys many features of the activity through the use of proper visual encoding. However, there are additional aspects of this visualization that are not easily evident from Figure 3-3.

The lines and labels are transient and appear on the screen for just a few seconds. Additionally, most of the display is concentrated along the periphery of the screen. These features make the display unobstructive to what the user is currently working on. The display does not mandate any user input (as opposed to a modal dialog box) and the cost to the user of retrieving the information is merely glancing to the area of the screen where the connection is displayed and reading the label. This avoids a
complete context switch of the user’s attention and application.

After the lines and labels disappear, they are replaced by little green (or red, if the connections were blocked) lumps. The little green lumps provide another level of visibility and feedback to the users. These lumps remain on the screen for a longer period of time after the labels and lines have disappeared. These lumps maybe clicked at any time to review a summary of the activity in the form of a tooltip. An example is shown in the Figure 3-4.

![Figure 3-4: FireViz’s Activity lumps displaying Connection Tooltips](image)

The tooltip provides the same information as evident with the line and label display. For example, the tooltip in Figure 3-4 shows the Application (IEXPLORE), the action (Allowed) taken by the firewall, the direction (“to” implying outgoing) and the remote end of the connection. These lumps also disappear in a short period of time (about 30 seconds) thus allowing the user to access the information while
not interfering with her work and applications. The presence of these lumps helps FireViz’s interface be more invisible while at the same time helping users by offering them multiple opportunities to retrieve the same information. This is especially useful given the transient nature of FireViz’s activities.

The tooltips themselves provide the relevant information in a compact yet smart fashion that enable users to easily scan them.

Another extremely important feature highlighted in this display is its ability to highlight exceptional events. In this context, for instance, when the user made a request for www.slashdot.org, she was expecting the web browser to establish a connection with www.slashdot.org. The web browser established the said connection, however it established two other connections as well - to 64.233.161.147 and ads.osdn.com, which were quite unexpected. The former connection was made to an unregistered host and hence got displayed at the bottom of the screen, clearly establishing the contrast with the expected activity. Thus the choice of the network mapping helps FireViz effectively display exceptional events.

A different visual display is provided for network activities that are either associated with background processes or processes that do not have any visual presence on the screen. These appear as flashing starbursts along the periphery of the screen. These starbursts can be clicked to view a tooltip that provides more information on the activity. Figure 3-5 shows a starburst display. After a display of about 8 seconds, these starbursts are also replaced by similar lumps described above. This is displayed in Figure 3-6. Network activities that do not have any processes associated with them (such as random port scans) are also displayed in a similar fashion.

The starburst display also uses color to indicate the firewall action. Figure 3-7 shows FireViz displaying a connection attempt that was blocked by the firewall. Since the event was blocked, it is shown as a red starburst.

As evident in Figures 3-6 and 3-7, the mapping shows all events from hosts on the MIT network close together, thus once a user has seen a single event, she can deduce that all other events are from hosts on the MIT network. This helps the user gain a better understanding of the nature of traffic on the network she is connected
Figure 3-5: FireViz displaying a flashing starburst for a background process

to (in this case the MIT network). Such understanding is quite useful as users can judge how safe their immediate network surroundings are. This is again achieved by the network mapping which places all MIT hosts in close proximity to one another. For this reason, it is important for the users of FireViz to learn the mapping it uses. To that end, FireViz displays a high-level map of the network when a user zooms in on one specific event (such as viewing a tooltip for an event). This display shows the main top level domains on the network and shows their ranges and demarkations using arrows and the labels as seen in Figure 3-7.

In summary, FireViz attempts to provide a usable interface that provides a useful visualization technique for network traffic while also providing extensive situational context surrounding each activity. This interface was designed after careful consideration to the needs of the users and was developed incrementally with many prototypes.
3.3 Usability Engineering

FireViz’s user interface was developed iteratively by building successively higher fidelity prototypes and user testing them. This section provides an overview of the usability engineering of FireViz.

3.3.1 User Analysis

Performing a thorough user analysis was the first step in the development of FireViz [36]. FireViz is intended to be used by all users who use a computer connected to
some network. These users have varying degrees of computer experience. While some users understand that the internet is an unsafe medium for their hosts, many others may be completely unaware of these security issues. This knowledge served as an important factor when designing an interface that would be used by most people for extended periods of time. I envision two different classes of FireViz users:

- **User Group 1: Novice Users** - The first group includes novice home users and school students for whom security is not a primary goal or concern. A majority of these users can be expected to know little about computer security and threats. This group includes school students, kids and housewives.

- **User Group 2: Experienced Users** - The second group includes relatively more computer savvy individuals such as college students or computer professionals who understand the threats that their computers are exposed to and
regard computer security as an important goal, even though not their primary goal.

Table 3.1 further characterizes the expected demographics of these two user groups.

In evaluating the user characteristics, I assumed some demographic attributes for all users of FireViz. These include the physical ability to work with computers, familiarity with the English language, some prior computer experience and literacy. All these characteristics are most likely to be true for many computer users. FireViz will likely not serve as a very useful tool for users in whose case the above assumptions may be false. Additionally, demographics such as gender, ethnicity and user skill-sets did not have any significant impact on the task analysis or UI design decisions of FireViz.

The single most important characteristic which forms the basis of the division of the users into the two different groups is prior computer experience. This influences users’ expectations and eagerness to use FireViz. After talking to some users from the MIT student community, it appeared that higher and broader computer experience correlated positively with awareness of network security issues. More experienced users are generally more willing to run firewall software and more interested in knowing about the network weather in general. Home users or novice users who

<table>
<thead>
<tr>
<th>Demographics</th>
<th>Novice Users</th>
<th>Experienced Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>12-60</td>
<td>18-70</td>
</tr>
<tr>
<td>Gender</td>
<td>M/F</td>
<td>M/F</td>
</tr>
<tr>
<td>Spoken Languages</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>Ethnicity</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>Physical Abilities</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>Computer Experience</td>
<td>Minimal - Moderate</td>
<td>Moderate - High</td>
</tr>
<tr>
<td>Skills</td>
<td>Various</td>
<td>Various</td>
</tr>
<tr>
<td>Literacy</td>
<td>Literates</td>
<td>Literates</td>
</tr>
</tbody>
</table>

Table 3.1: FireViz User Characteristics
may mostly use the internet to surf the web, chat with buddies and share files are
sometimes aware that network connections may be unsafe but do not understand the
exact nature and mechanisms of the threats. Such users seemed willing to use software
that would provide them more insight into personal computer security. Consequently,
more experienced users are generally aware of the exact information that they may
be looking for and novice users are not. This knowledge had important implications
on the user interface design of FireViz.

3.3.2 Task Analysis

FireViz aims to provide objective information about computer security to users. After
a thorough user analysis, the following emerged as the major tasks users would need
to perform when using FireViz:

- **Accessing the network as FireViz runs in the background.** This is an
  extremely important and central task to FireViz. Most users will not work
  on FireViz, they will work on applications that help them with their primary
task - such as an email client or a web browser. Consequently, FireViz’s design
  philosophy always envisioned FireViz as a peripheral application and not a
  primary application. The task for users therefore is to work on their primary
  applications with FireViz running in the background. The hope for FireViz is
  that it will not interfere with the users’ main tasks while running.

- **Retrieving information on network activities.** The challenge for FireViz
  is to provide information while being invisible and non-intrusive at the same
time. In order for the UI design to be effective, users should be able to easily
  retrieve the information that FireViz presents while continuing to work on their
  primary applications and avoiding costly context switches.

- **Learning the Visualization.** Users will get the most efficient use out of
  FireViz when they learn the visualization. This includes, learning how different
  events are shown in FireViz, how the information is encoded in the display and
how the network is mapped along the screen. Thus, the task for users is to learn this visualization and the hope for FireViz is that it is easily learnable.

### 3.3.3 Paper Prototypes

FireViz’s UI was developed using the iterated design model [35]. Once the user and task analyses were done, the next step in the UI development was to test the UI features with multiple prototypes with increasing fidelity in look and feel. The very first prototype included design sketches of what the UI should look like. These were shown to a few users and the feedback was incorporated when designing the paper prototype. The line and label display along with the vestigial lumps were incorporated in the design sketches themselves and survived all successive iterated tests.

The next iteration was the development of a paper prototype for FireViz. Users were given a brief overview of FireViz and were presented with a few scenario tasks. The prototype was executed by a third person while I silently observed the users perform the tasks and make note of exceptional events.

The scenario presented to the users was as follows - “You are Joe User. You are expecting to receive an email from your friend Emaily so you log on to your Gmail account. As you sign in, FireViz detects this connection and displays some activity on the screen. Your task is to find the information on the frequency of connections to the Gmail server”.

Users were then asked to retrieve information on this activity.

This task was given to three different users who had no prior information about FireViz and had limited knowledge about how personal security firewalls work. All users successfully completed the task. However, the time taken by each user varied to a certain extent. In this scenario, FireViz displays a line from the browser window to a spot on the periphery of the screen. This line appears only for a few seconds and then disappears. Mousing over the line provides information about the specific connection. However, drawing the line in real time on the paper prototype completely confused two users who were clueless as to what that line meant. It was also hard to convey that the line was a transient display. After some exploration through the system tray...
menu for FireViz and not finding the requested information, the users finally decided to act on the line and that revealed the requested information. From this observation, we decided to change the design such that these lines were accompanied by a label indicating the remote end of the connections. This would allow users to receive the information without needing (and therefore knowing) to act on it.

After the completion of the task, users were asked to provide subjective feedback on the user experience with FireViz. One user suggested that FireViz should show network activity at just one spot on the screen instead of all around the periphery because she feared that it may get annoying quickly and make it difficult for her to focus on her work. However, it was evident from the task analysis that users would benefit from a semantic mapping of the network. The success of this theory however relied on real network traffic and could not be tested effectively by paper prototyping alone.

The feedback received from paper prototype testing was incorporated in the next iteration of user testing i.e., the computer prototype.

### 3.3.4 Computer Prototype

User testing with a computer prototype provided the opportunity to get more objective feedback on FireViz’s most important UI features. To facilitate a complete redesign of the UI, if so required, the computer prototype itself was developed using a different technology from what I intended to use for the actual interface. The prototype was developed in Java and continued to be a shallow prototype i.e., without a working backend. This prototype was then distributed to three users, who then heuristically evaluated [37] the prototype. The users were also provided an overview of FireViz, instructions for running the prototype and tasks. The overview and the tasks were the same as those used for the paper prototype.

The heuristic evaluation revealed numerous problems with the graphic design of the prototype. While the computer prototype was useful in getting feedback on the graphic design of the UI, it still could not be used to test the mapping aspect due to its lack of a working backend. The feedback was incorporated in the development of
3.3.5 High-fidelity Prototypes

The high-fidelity prototypes for FireViz were developed using the Microsoft .NET framework for the Windows platform. The backend for the prototype involved a separate Network Module that detected established TCP connections on the host. Figure 3-8 provides a screenshot of the prototype.

Figure 3-8: First Hi-Fi Prototype of FireViz displaying a Network Connection

This prototype also provided an opportunity to test the network mapping algorithm. This prototype employed a frequency based mapping of the network as described in section 4.3. Note that the direction of the connection was not displayed. This prototype also displayed a Tooltip when the user clicked on the green lump. Figure 3-9 shows an image of the tooltip.
The evaluation of this prototype revealed several problems with the graphic design of the tooltips. The lumps were designed to stay for a long period of time after the lines had disappeared. After running FireViz for some time, these lumps started interfering with the user’s tasks by hiding scrollbar arrows and buttons to maximize windows. The frequency-based mapping proved to be learnable but not very useful in providing information such as where the bulk of the traffic was coming from (for example the MIT network).

The development of this frequency based prototype was therefore followed by more prototypes with different mapping algorithms - geographical mapping, static domain based mapping and finally an adaptive domain based mapping. The prototypes and their results are discussed in Chapter 4. The final prototype of FireViz featured a domain based mapping. The results from the user testing of this prototype are
discussed in Chapter 6.
Chapter 4

A Mapping of the Network

One of the most crucial goals of FireViz is to provide a non-intrusive means of indicating the network weather to the users. This is consistent with the users’ expectations because most users do not concern themselves primarily with security. Most security applications are built with a user model where users expect security “to just work”. Keeping this in mind, FireViz is designed with a peripheral interface. This allows users to work on their primary tasks without being obstructed.

A task analysis of FireViz also indicated that users liked to know about relevant aspects of network events such as the nature of traffic on their local network (if any), which applications were making or receiving these events and if any event was unexpected. Thus, in addition to defining the peripheral feature of the interface, the choice of mapping also aimed to achieve the following goals -

- **Provide a feature rich representation of the network.**

  The choice of mapping can provide additional useful information about a particular network event such as the network proximity and expectedness of such an event. For instance, a geographical location based mapping or a mapping with a notion of a local network could easily indicate the volume of traffic from the local MIT network. During the user analysis for FireViz, it became evident that users cared to know more about the immediate network around them, rather than care equally about the whole internetwork.
• Provide easily identification of exceptional events.

The selected mapping should be able to display extraordinary events in a way that can be easily recognized. As a motivating example consider the following scenario. If an application made an access to www.microsoft.com, users would expect the remote end to be located in Washington, USA. However, if the connection appeared to be made to a remote host located in Russia, it would be remarkably unexpected and also easily identifiable with a mapping based on geographical locations.

• Provide easy recognizability and learnability of the mapping.

The mapping should allow common activities to be easily recognized. This would enable users to identify events just by the location of the displays without having to view the tooltips. Thus an easily learnable mapping will help users get a better sense of the network with little effort.

Thus the choice of mapping emerged as the single most crucial decision in the implementation of FireViz. As a result, the development of FireViz involved experimenting with a number of different mapping choices and evaluating the interfaces. A discussion of the mapping alternatives is presented.

4.1 IP based mapping

One of the preliminary ideas that we considered in the development of FireViz was mapping the network based on IP addresses. There exist a finite number of IP addresses and these would be mapped around the periphery of the screen in increasing order. Figure 4-1 shows the schematic of an IP based mapping. However this idea was discarded because it failed to meet the three goals outlined earlier.

IP addresses often encapsulate less information and meaning than hostnames (for example, webmail.mit.edu vs. 18.7.22.79). IP addresses are also hard to recognize and memorize. Moreover, IP addresses on the same network could potentially appear far away from one another thus failing to provide any indication to the user.
that the two remote hosts could potentially be on the same network. For instance, *webmail.mit.edu* with an IP address of 18.7.22.79 would show up at a very different place from my computer on the MIT CSAIL network with an IP address of 128.30.2.24. Thus, an IP based mapping failed to provide a feature rich representation of the network. For the same reason, it would also fail to selectively highlight exceptional network events by failing to provide the necessary contrast for genuinely extraordinary events. Thus, the idea for using an IP based mapping quickly failed to meet the objectives of FireViz.

### 4.2 Application based Mapping

Another mapping algorithm that I considered for FireViz was an application based mapping. In this mapping, all applications on the users’ computer would get allocated a portion of the screen and all connections made by a single application would appear
in that application’s region alone. The motivation for considering such a mapping came from the task analysis. Users indicated that they cared to learn more about the activities of their most commonly used applications, such as their browsers and mail or chat clients. Figure 4-2 shows an instance of an Application based mapping.

Figure 4-2: Application based mapping

This application based mapping, however, also failed to provide considerable success with the other goals outlined in this chapter. One of the problems with this mapping was the issue of scalability. As users added or removed applications, the mapping would need to change. Also, it was unclear if applications should be allocated screen locations beforehand or only as FireViz ran and discovered applications that explicitly made or requested connections. In the former case, it would be next to impossible for FireViz to detect all possible processes on the user’s computer that may ever access a network. The latter approach on the other hand would make it hard for users to identify where events would be displayed since this could change drastically.
between multiple runs of FireViz. This requirement of constantly having to change
the mapping conflicted with providing a stable and easily learnable mapping to the
users.

More importantly, an application-based mapping would also fail to indicate ille-
gitimate accesses by trusted applications in the context of making legitimate accesses.
For instance, if while visiting a Gmail account, the web browser made an access to
an unknown web server (due to the presence of a malicious piece of email in the
mailbox), both of those connections would appear on the region of the screen that
belongs to the browser. This fails to provide sufficient contrast to users to identify
this unexpected event.

An application based mapping would also have to specially deal with random
probes on the network that are not specifically targeted towards an application. This
is very common when a computer is connected to a LAN. Thus an MIT user for
example, would not be able to easily recognize that a non-negligible amount of traffic is
coming from hosts located on the MIT network. Thus, an application based mapping
also emerged as an unsuitable choice for FireViz.

4.3 Frequency based Mapping

The first high fidelity prototype of FireViz employed a frequency based mapping of
the network. The network was mapped according to the frequency of connections
made to the remote ends. Frequency indicated the total number of times a particular
event had occurred. The frequency was mapped starting at value 1 on the lower right
corner of the screen and increased upwards until some maximum value which was
mapped to the upper left corner of the screen.

The total frequency of a given connection could range in values starting from 0
to $N$ accesses, where $N$ may be an arbitrarily large or small positive integer. For
this prototype, I chose $N = 100$ at which point, a given connection became frequent
enough to be shown on the upper left corner.

Based on their actual frequency, connections were described as - “First access”,
“Rare”, “Frequent” or “Expected”. This mapping made it very easy to recognize exceptional events since they are often rare. For example, a web bug in an email message making a connection would show up as a rare event (on the lower right corner) while the connection to the mail server would show close to the upper left corner of the screen (since it is likely more frequently accessed). Figure 4-3 shows a schematic of this mapping.

Figure 4-3: Connection Frequency based mapping

In spite of all these positive features, a frequency based mapping was not the perfect solution to all the goals that FireViz aimed to achieve. A mapping based on frequency alone, seemed like a naive approach. Most users experience periods of increased bursts of activities for short timeframes\(^1\) but beyond a threshold, these would all become frequent or expected events. This problem could be solved by

\(^1\)Examples of such activities may be related to seasonal or periodic events such as annual tax refund deadlines and the corresponding use of tax software
adding a decay factor to all accesses, such that events that had not occurred for a while would be accorded a lower level of expectation than mere absolute frequency alone. However, the more significant drawbacks emerged from the use of frequency as the primary basis for the mapping.

Once FireViz has been running for extended periods of time, most activities would seem expected. For instance, software such as spyware, if allowed internet access, could quickly generate traffic that would seem normal and expected. Additionally, a mere frequency based mapping would again fail to convey information that suggests traffic behaviors from a local network. As a result, we decided to experimenting with different mapping algorithms.

4.4 Geographical Location based Mapping

The next prototype of FireViz involved a new mapping algorithm, this time based on geographical location. The left and top edges of the screen were used to display locations in alphabetical order (A through Z) for hosts located in the USA. The right edge of the screen alphabetically displayed locations outside the USA and the bottom edge displayed hosts by IP addresses for whom a location could not be obtained.

The location included a state and city if in the US (e.g. MA, Cambridge) and a country for all other locations. For US locations, the states were mapped alphabetically from the bottom left corner to the top right corner. Cities within the states were again arranged alphabetically within the state. Thus, all locations within the same state would be displayed close together. Figure 4-4 shows the geographical location based mapping used in an early prototype of FireViz.

The hope with using such a mapping was that this would again succeed in providing a learnable and rich representation of the network. It would make it extremely easy for users to recognize activity from networks closer to them and also recognize exceptional events. If a web browser made a connection to www.somesite.jp while making a connection to www.microsoft.com, the two activities would appear at very different locations on the user’s screen.
In spite of the richness of its representation, certain implementation challenges made this mapping infeasible for a stable prototype. The source of the information was either a reverse DNS look-up to locate hosts outside the USA (hostnames with a country domain e.g. yahoo.co.in) or a Whois lookup from sources such as www.completewhois.com. Whois databases generally contain data collected by DNS authorities at the time of domain registration. As a result, this information is not entirely up to date. Moreover, such information is only provided at a high level - for instance for a company, this information may only include the location of the company headquarters and not of the specific host in question. Thus even if a specific Gmail server were located in Salt Lake City, Utah; FireViz would only know that it is located in California because that is where Google is located.

The prototype implemented caching of such location information. However, in
cases where such information was not available in the cache, huge delays were accumulated in retrieving the data from the whois servers. Whois servers generally recursively query numerous DNS servers to receive such information thus adding to the delay. This adversely impacted the real-time nature of FireViz’s display. Moreover, since Whois information is not retrieved in a well defined format (such as DNS responses), parsing the information to retrieve the location was also not straightforward. Therefore, even though a geographical location based mapping could potentially be an excellent choice for a network map, various implementation challenges made it infeasible for FireViz to employ such a mapping.

Other alternatives to consider in a future iteration of this project include Akamai’s EdgeScape [39] or Netcraft [40] that uses its database to find locations of hosts. Recently, Google produced a GeoDisplay [41] that provides a three dimensional geographical representation of the globe. Colored dots on this globe represent where Google is being used at any given moment of time. The color of the dots (red, orange or yellow) also encodes information on the quantity of the usage as well. Such displays are testimonies to the fact that a geography based visualization could provide significant amounts of information, if they can be reliably implemented.

4.5 Domain based mapping

The final prototype of FireViz employed a domain based mapping. In this mapping, all top-level domains were allocated a span of the screen periphery including the left, top and right edges. Specifically, the five top level domains (COM, EDU, GOV, NET and ORG) were allocated on the left and top edges of the screen in that order. All other top-level domains, such as country domains were mapped alphabetically on the right edge and all unresolved hosts were mapped in increasing IP order on the bottom edge of the screen. Subdomains within these top domains were arranged alphabetically within their top domain. Figure 4-5 shows the domain based mapping used in the current implementation of FireViz.

The lengths of the screen regions for the top level domains were chosen based on
average traffic patterns on a small sampling of hosts connected to an MIT network.

This mapping allows hosts in the same sub-domain to be close together. Thus, as shown in Figure 4-5, each subdomain in within COM is arranged contiguously. Hosts within these individual subdomains are similarly arranged contiguously. This also enables FireViz to conveniently display events that span remote hosts over different domains and subdomains with visible contrast. This contrast can however be reduced if remote hosts within the same domain are located close together in name space (for instance www.paypal.com and www.paypai.com). This mapping is however very well able to highlight events when the remote hosts have the same name but on different domains (for instance www.paypal.com and www.paypal.net). Thus, a domain based mapping can be useful in detecting phishing attacks [42].

This domain-based mapping also helps retain information on network locality. For
instance, all traffic from the MIT network would appear under the EDU domain thus making it easy for users to recognize that the source of such activity could be the local MIT network.

One of the challenges with using a domain based mapping however was that it is not easily learnable by just watching a few events displayed. Thus, to make the mapping more easily recognizable, we added several visual cues in the UI to help the user identify the mapping. An example is shown in Figure 4-6.

![Figure 4-6: Visual cues in Domain based Mapping](image)

This prototype was tested with multiple users. A summary of the results is provided in Chapter 6.
4.6 Adaptive Domain based mapping

We built another version of FireViz with an adaptive variant of the domain based mapping described in the previous section. This algorithm adapts the network mapping to the traffic on the host. Thus domains and subdomains that are accessed more frequently are accordingly allocated a longer span on the screen than those that are less frequently accessed. This section describes the adaptive algorithm in greater detail.

For the purpose of presenting the argument we assume that all domains and subdomains in the network can be arranged as a tree. We further assume that all the subdomains within a domain are sorted alphabetically. Each of the nodes in this tree has a weight associated with it. The span of any subdomain is a function of its weight. Consequently, the span of the top-level domains is a function of the weight of all their children. The adaptive nature of this algorithm emerges from the adjustment of the weights of the child nodes according to the traffic received from the nodes.

In our implementation, the absolute frequency of connections to remote hosts was used as the weight on the nodes. For instance, let’s consider that the host log.launch.yahoo.com has been accessed 10 times. log.launch.yahoo.com is located at the fourth level (the domain is com, the subdomain is yahoo, the sub-subdomain is launch and the host is called log). Therefore the frequency of all its parent nodes is at least 10. The absolute span of launch.yahoo.com is the same fraction of the total screen estate as its fraction of the total frequency and log.launch.yahoo.com is located within that span. As the frequency of the traffic changes, the span and location of log.launch.yahoo.com also changes. Thus even if a host has not been accessed over a period of time, its span will decrease due to its reduced weight relative to other hosts that are accessed more frequently.

To illustrate how this adaptive algorithm works to find the location of display for a host, let us consider the state of the tree as shown in Figure 4-7. For the sake of simplicity, only a part of the tree is shown here. The numbers above the nodes represent the weights which are absolute frequencies for the given hosts or
subdomains. Let us assume that given this state, FireViz detects another connection
made to web.mit.edu.

The Adaptive Algorithm, goes through the root node and finds the top-level do-
main, i.e., EDU. It then looks for edu.mit within the children of EDU and then for
edu.mit.web. It finds that edu.mit.web currently has a frequency of 4. The fre-
quency of this node is updated to 5 and consequently, the weights of its parent nodes
increase by 1 as well. The algorithm now calculates that EDU gets \( \frac{20}{85} \) of the total span
of the domain mappings (which includes the left, top and right edges of the screen).
Similarly, edu.mit gets \( \frac{15}{20} \) of EDU’s span and edu.mit.web gets \( \frac{10}{15} \) of edu.mit’s span.

In order to calculate the exact position of the web.mit.edu, the algorithm now
needs to calculate the span of all children that appear before it. In this case, the
algorithm calculates the lengths of COM, which is \( \frac{30}{85} \), edu.bu (\( \frac{3}{85} \)) and edu.mit.is (\( \frac{5}{85} \)).
These are then scaled by the total available span (which is twice the screen height
plus the screen width) and the event is displayed within the range for edu.mit.web
which was previously calculated. This provides the actual position of the event on
the screen. If FireViz is restarted at any point of time, the state of the tree is restored
before any new events are displayed.
It is evident from this algorithm that as more and more traffic is generated to a particular host, the length of the subdomains increases proportionately.

Since FireViz would adapt to the network traffic over a course of time, it is initially configured with weights that correspond to the static domain based mapping. These weights were chosen to reflect typical traffic patterns seen on an MIT host. Over time, these initial weights would steadily decay causing the actual node weights to be more and more biased to the weights that reflect network traffic patterns (based on frequency). Thus, an important design tradeoff here was choosing the right speed of learning. A more sensitive algorithm would be able to quickly adapt to the traffic patterns, however it could potentially reduce the learnability of the interface as the positions and domain ranges are susceptible to changing quite frequently. The right decay rate should be chosen according to the amount of traffic seen by the host. This would also require testing the prototype over longer periods of time to monitor the effectiveness of both the adaptability of the mapping and its impact on user experience. This adaptive prototype of FireViz could not be tested extensively due to limitations of time.

One drawback of using absolute frequencies of connections as the weights on the host is that applications that make periodic connections to particular hosts can easily bias the algorithm. For instance, an application like Kerberos periodically connects to an MIT server to renew authentication tokens for secure shell applications. This would cause the algorithm to place a higher weight on the MIT server and hence on the whole EDU subtree. This is not a very useful adaptation to the network. One solution to this problem could bias the weight adjustments also to the number of different hosts that are accessed within a subdomain in addition to frequency of accesses.

Using an adaptive algorithm in the visualization can help users easily monitor events from domains that generate greater traffic. It can also help users get a qualitative sense of their network weather. For instance, if the EDU span grows relative to all other domains, then it is easy to guess that a majority of the traffic is received from the immediate MIT network. This may also help FireViz highlight extraordinary events as they would have much smaller weights relative to more common events and
create a noticeable visual contrast.
Chapter 5

Implementation

FireViz was developed using the .NET Framework and is designed to work on most Windows Operating Systems. Since Windows is the most popular operating system used on home PCs and laptops, this choice is consistent with the intent of making FireViz accessible to a large population of computer users. On an extremely high level, FireViz is composed of two modules - one that detects network activities on the host and the other that analyzes these events and displays them appropriately on the screen. This chapter provides an overview of the implementation of FireViz.

5.1 Implementation Overview

FireViz consists of the following modules.

- **TCP Connection Monitor** - This module monitors and detects all established TCP connections on the host. While this module is capable of detecting all types of connections (including UDP), the current implementation of FireViz only detects TCP connections.

- **Firewall Log Reader** - This module detects connections that are blocked by the firewall, since these cannot be detected by the TCP Monitoring module.

- **The Display Engine** - This module receives events when a new activity is detected. It then analyzes the event and displays it appropriately.
• **The Mapping Module** - This module uses the mapping algorithm to calculate the location on the screen where the activity should be displayed. The Display Engine uses this module when processing display events.

• **DNS Resolver** - Since FireViz employs a domain based network mapping algorithm, this module abstracts the reverse DNS look ups for remote IP addresses. This module also employs caching of resolved names for efficiency.

Each of these modules are discussed in greater detail in the following sections. Figure 5-1 shows a module dependency diagram for FireViz. The rectangles represent the modules and the arrows represent the direction of the dependencies. The name of the dependency is represented as the labels on the dependency arcs. Note that the Firewall Log Reader depends on the Firewall to log all events correctly. However, the firewall itself, shown in a dashed rectangle, is an external component to the system.

![FireViz Module Dependency Diagram](image)

**Figure 5-1: FireViz Module Dependency Diagram**
5.2 TCP Connection Module

FireViz implements its own connection detection algorithm. The motivation behind this design choice emerged in the task analysis. Most users were more interested in knowing about established connections as opposed to blocked connections. This is understandable since every established connection can be viewed as a potential security threat to the computer. Thus, a successful display of established connections is very crucial to FireViz’s proper functioning. Additionally, since FireViz provides a real-time display of activities, quick detection of established connections is extremely crucial.

As an alternative to implementing its own module, FireViz could rely on the firewall logs to detect established connections. However, firewalls often batch network events for efficiency, before logging them. This firewall optimization is not conducive to the real-time nature of FireViz. Thus, by relying on its own connection detection module, FireViz is able to quickly and effectively detect network activity by eliminating delays caused while the firewall logs such events.

FireViz uses the API exported by libraries such as IpHlpApi.dll [43] to constantly poll the TCP Connection Tables on the host. These TCP tables hold information on the status of each established or in-process connection along with auxiliary information such as the port numbers on the two ends, the protocol, and the application name. Programs such as Netstat [16] and TCPView [17] are also implemented similarly. When FireViz detects a new established connection, it encapsulates the information in a state object (called a NetworkEvent) and notifies the Display Engine.

It is noteworthy that even though FireViz is projected as a firewall visualizing tool, it does not depend on the presence of a firewall for its functioning. The independent activity detection engine of FireViz allows a user to just use FireViz without actually using a firewall. Another way of looking at this is running FireViz with a firewall that allows all accesses. Our hope in such a scenario is that FireViz will reveal the sheer volume of unexpected traffic and encourage the user to install security software such as a firewall.
5.3 The Firewall Module

Network events such as port scans or blocked accesses are not visible in the TCP tables on the computer. For detecting such events, FireViz has to rely on the firewall logs. Thus the Firewall Module parses the firewall logs and retrieves any new events. A majority of blocked events occur at random points in time. Moreover, some events such as random port scans do not have any visible screen presence on the local host. Therefore, the logging delay is not critical in these cases.

When the Firewall Module detects a new event, it again encapsulates the information in the NetworkEvent state object and notifies the Display Engine.

5.3.1 Choice of Firewall

An important decision in the implementation of this module was the choice of firewall. The current prototype of FireViz is implemented for the Sygate Personal Firewall [11]. A couple of different firewalls were considered for FireViz before choosing Sygate Personal Firewall.

ZoneAlarm [9] emerged as a suitable candidate for the firewall since it provided easily parsable activity logs. However, these logs only recorded activity for applications that were not trusted. Thus there were no records for accesses made or received by trusted applications. This was a major drawback of ZoneAlarm because trusted applications pose a greater threat to the security of the system (because any vulnerabilities in such applications can be exploited to compromise the system without being questioned by the firewall). Therefore, ZoneAlarm was not the right choice for the firewall.

Besides ZoneAlarm, firewalls such as the Windows Firewall [44], Norton [15] and Kerio [10] were also considered. However, most of these firewalls either did not provide complete machine readable logs or failed to expose information on their state (such as trusted applications).

Sygate personal firewall on the other hand, provided comprehensive logs in a well-defined format and could be easily parsed to detect network activity. It also
exposed information on the rules used by the firewall to define the action taken for each activity. For all its features, Sygate was chosen to implement this prototype. In spite of its advantages, the delayed writing optimization used when logging the events prevented us from depending solely on the logs to detect established connections in real time. This motivated the development of a separate TCP connection detection module as described in the previous section.

5.4 The Display Engine

FireViz is implemented according to the Model-View-Controller (MVC) design philosophy. The MVC design helps separate the backend (network monitoring) from the frontend (display of network events). It also helps to make FireViz easily extensible by providing means to easily add new views to the backend.

The TCP Monitoring module and the Firewall module encapsulate the model that detects new network events. The display module registers itself as an observer with both the backend modules. When a new activity is detected, the observer (the display engine) is notified by means of a NetworkEvent object. The Display engine analyzes the event and creates a \textit{DisplayObject}. The FireViz window then paints the display object appropriately.

The display engine relies also on the Mapping module to derive the location on the screen where the display object must be displayed. This location is stored in the DisplayObject instance and is used by the FireViz event loop when displaying the activity.

5.5 The Mapping Module

FireViz uses a domain based mapping for the network. The mapping module uses the hostnames of the remote ends to calculate their location on the screen. In case of an adaptive domain based mapping, the Mapping module also uses the frequency of accesses to calculate the position.
Separating the Mapping module from the display allows the display module to be used without any changes if a different mapping algorithm is used.

5.6 The DNS Module

Since FireViz employs a domain based mapping, the DNS module abstracts the resolution of hostnames from IP addresses. The DNS module maintains a cache of resolved hostnames for efficiency. This optimization is crucial to reduce delays in displaying events while a DNS query is being executed. The DNS module therefore, either returns a cached value or executes a DNS query if one is not found in the cache. This result is then propagated to the Mapping Module so that it may compute the display location of the event.

5.7 FireViz Display Objects

Another design feature implemented in FireViz is inheritance. This is evident in the implementation of the Display Objects. A Display Object encapsulates all the information that FireViz needs to paint it. Thus the Display Object dictates whether a particular event should be shown as a line and label or a flashing starburst. All Display Objects exhibit some common properties - such as an associated NetworkEvent object and a location on the screen where they are displayed. This abstraction of similar properties allows FireViz to easily replicate common behavior from different objects without having to know the specific type of the Display Object.

Figure 5-2 provides an object model of the various classes implemented in FireViz. Also shown in the figure is the subclassing hierarchy within the Display Objects.

FireViz implements four different kinds of Display Objects. These are visible in Figure 5-3.

- **NetLine** - A NetLine object represents information that is displayed by drawing lines from an application window to a location on the edge of the screen. This
Figure 5-2: FireViz Object Model
Figure 5-3: FireViz Display Objects

is used when a connection is established by an application whose window is the current window on the screen.

- **NetFlash** - A NetFlash object represents information that is displayed by a flashing starburst around the screen edge. Such a representation is used to display network activity associated with applications running in the background or network events (such as port scanning packets) that do not correspond to any specific applications. These events are displayed on a separate component (called FlashWindow). This separation allows only the NetFlash objects to flash on the screen while letting other Display Objects display steadily.

- **NetLump** - A NetLump is a visual vestige on the user’s screen after a Netline or NetFlash has appeared. Since the lines and the flashes only appear for a very short period of time, sometimes users may be unable to click them to see further
information. However, once they have disappeared, these lines or flashes are replaced by small lumps that remains on the screen for a longer period of time. These lumps can then be accessed to receive the same connection information, in the form of a tooltip.

- **NetTooltip** - A NetTooltip represents connection information available from a visual activity lump.

### 5.8 Implementation Challenges

FireViz’s implementation involved several crucial design choices that had to be made in lieu of a conducive framework for the development of such an application. These design choices have had a significant impact on the implementation and are worth mentioning.

#### 5.8.1 Choice of Firewall

A major challenge in the development of this application was the lack of a firewall that could be easily integrated with FireViz. Few reliable open-source firewalls exist for Windows systems. As a result, an event-driven architecture could not be implemented for FireViz. This led us to develop two independent modules - one that polls the host’s TCP tables to detect established network connections and one that parses the firewall logs.

The tradeoff of a polling based design is between accuracy and efficiency. As the polling interval increases, the accuracy of connection detection decreases. Short lived connections may therefore never be detected in real time. The accuracy can be increased by making the polling interval arbitrarily small but at the cost of efficiency. As a result, FireViz is extremely CPU-intensive. This not only has the adverse effect of slowing down the system as a whole and hence interfering with the users’ primary tasks but also reducing the accuracy of FireViz as more applications compete for CPU time.
The ideal design for FireViz is to have the firewall raise events whenever it detects a new event, thus eliminating the need to poll. This would help FireViz be less obstructive to users.

5.8.2 Stroke Model in Display

Since FireViz’s display widgets consist of lines, starbursts and custom labels, it is implemented using the Stroke model instead of using full-blown widgets such as buttons. By not using the component model, FireViz thus needed to implement its own mouse input handling and component hierarchy. This was further complicated by the fact that FireViz has two top level containers on which these objects are displayed (one for the flashing starbursts and one for all other Display Objects). Therefore both these containers needed to be aware of the objects currently being displayed so that the mouse events would work as expected without regard to which window happened to be on top of the other. This added an additional cyclic dependency in the design. Creating widgets using the stroke model also made it hard to provide the right affordances to the users about the actions they could perform.
Chapter 6

Evaluation

A final stable prototype of FireViz was again user tested to receive feedback on the user interface design. This chapter provides a summary of the results.

6.1 Testing Methodology

All users for the tests were recruited with their consent. Users were provided a brief overview of FireViz (see attached Appendix) and informed that they had the right to stop the test at any point of time. Users were given a task and asked to think out aloud while I took notes of their observations. At the end of the test, users were asked to provide comments and feedback on various aspects of the UI in order to get a sense of their subjective satisfaction from using the UI. The tests lasted about 30 minutes each and all measures were taken to maintain the anonymity of the users.

During the test, users were asked to perform regular tasks such as browsing the web or checking their email while FireViz was running in the background. Users were asked to interact with FireViz and think out aloud as they were doing so.

Users were intentionally given such an open-ended task. Our hope was to see how users were able to perform on the following aspects:

- **Interacting with FireViz.** We wanted to see how well users were able to retrieve information from the visual display of FireViz. It also aimed to assess
how well the visual cues were recognized by the users and the information encoded therein.

- **Learning the Visualization.** We also wished to know if users were able to understand the visualization used in FireViz i.e., the lines and labels, the starbursts and the mapping.

- **Performing primary tasks while FireViz was running in the background.** We wished to assess how well FireViz succeeded in not obstructing users’ primary tasks. Specifically, we wished to see if users were distracted or annoyed by FireViz and were unable to focus on their work.

- **Identifying exceptional events.** Finally, we wished to see if the visualization in FireViz actually succeeded in highlighting exceptional events. Even though this is harder to test, we hoped that whenever there were any exceptional events, users would recognize them to be so.

8 different users who had no prior experience with FireViz tested this prototype. A summary of the results is provided.

### 6.2 Test Results

Through the tests, we aimed to see whether users succeeded in understanding the following aspects of the visualization:

- The color of the display events (red or green)
- The direction of the connection in line displays (incoming or outgoing)
- The application, if any, associated with the connection
- The mapping of the network (domain-based)
- The visualization itself (lines-and-labels or starbursts)
Figure 6-1 indicates the rate of success the users had for each of the five aspects of the visualization. All users were easily able to associate the right application with the events in both displays (line-and-label and starburst). The origin of lines within application windows served as an intuitive association between the applications and the events. Even when users clicked the starbursts, the tooltips provided them with the application information. Most users also succeeded in associating the color with the firewall action. One user however interpreted the green color as meaning good or safe and the red color as dangerous. Thus users were able to retrieve information from FireViz’s display without much difficulty.

By comparison, users had a harder time learning the visualization. The five users who understood the mapping were able to do so after watching several events. All five users who recognized the mapping utilized the domain labels and ranges displayed on the screen thus indicating that such visual cues were helpful. The hardest task for the users was to learn what kind of activity was displayed with which event. As
a result only three out of eight users were able to correctly recognize this aspect of the visualization. In addition to possible UI design flaws, this result can also be attributed to other factors. Users only spent 20-25 minutes using the system, which is an extremely short period of time to learn the visualization.

Among other observations, most users found FireViz to be quite unobstructive to work with. However, some users were unsure if that would continue to be true if FireViz were used extensively. Most users expressed surprise when FireViz first displayed any event and were unsure of how to act on it. However, they quickly learnt to click on the displays to get some information. Users also had a tendency to click on the domain labels, with no response. Both of these observations indicate the lack of proper affordances in the UI.

A few unexpected events also occurred while some users were testing FireViz. For instance, FireFox visited a site in the UK (.uk) when the user visited www.yahoo.com. In other instances, the browser detected multiple connections to unresolved hosts when the user visited either www.ebay.com, www.sourceforge.net or www slashdot.org. All users found those additional events interesting and acted on them to receive more information.

### 6.3 Subjective Feedback on the Interface

Users were also asked to provide subjective feedback on their experience with the UI. The subjective feedback from users was very helpful in evaluating many individual features of FireViz’s UI. Some are presented here.

The transience of most activities helped users to focus on their work and forget about events that were not very important and which they had already seen.

While most users found the graphic design of the UI pleasing, some expressed concern that the display might get annoying quickly in the presence of significant amounts of traffic. However this aspect of FireViz can only be effectively tested through long-term deployment.

Most users found the visualization useful in learning the firewall’s actions. A
particular user also commented that a particular action allowed by the firewall should in fact be blocked. Another user commented that the allowed accesses were more of what they wanted to see. All users were surprised to see the sheer volume of traffic they received on the MIT network and expressed concern on that. However, some users also felt frustrated at not knowing exactly how to stop such events. Since FireViz is not completely integrated with a firewall, it is impossible to add a new rule or change existing firewall rules through FireViz.

With regards to the UI itself, there is a definite problem with the lack of proper affordances. Since most objects in FireViz could be clicked to extract more information, users were inclined to do the same with static labels too, only to find no response from the UI. Some users also felt frustrated at their inability to move the tooltips around and to choose when the widgets should disappear. The little green lumps were sometimes hard to click because of their small size in spite of their position along the edges of the screen.

Finally, some users commented that the evident drop in the responsiveness of the system, due to the compute-intensive nature of the TCP module of FireViz, adversely affected the non-intrusiveness of FireViz. I believe that this is a valid criticism of the design and implementation of FireViz, however it is a side-effect of our solution to the implementation challenges, as discussed in Chapter 5. Consequently, this feedback cannot be generalized or extended to the UI. The purpose of these user tests aimed at evaluating the visualization employed in FireViz.

In summary, user testing helped us evaluate how well the current implementation of FireViz succeeded in meeting its goals. Early tests indicate favorable results for FireViz’s potential to educate users of potential firewall holes and network threats. While initial user tests corroborate the rationale and decisions made in the design for various features of the visualization, there is still plenty of room for improvement. Long term testing can also provide answers and more constructive feedback on the riskier and more involved aspects of FireViz’s UI.
Chapter 7

Conclusion and Future Work

7.1 Thesis Contributions

This thesis project, FireViz, makes the following contributions:

• First and foremost, it presents a means to visualize and understand network behavior, i.e., the activity patterns, nature of traffic and the frequency of such events in real time.

• It presents a framework that collaborates with personal security software such as a firewall and presents a visualization of the enforced security policies. This visualization can help expose potential security holes in the firewall and lead the user to close them and strengthen their security.

• It presents a novel visualization paradigm that focuses primarily on users’ expectations from security and provides a new direction in the development of UIs for personal security software. FireViz presents a UI that is peripheral and encodes information visually.

• Finally, it provides a framework for testing the usability of the system and presents a discussion of results from executed tests.

At this juncture of summarizing the contributions of FireViz, I would like to take a moment to restate the very motivation behind FireViz in new light. A skeptic may
argue that if a user asks her firewall to trust an application, presumably she knows what she is actually doing and would not bother to know about what the application does everytime she uses it. In answer to this question, I present Figures 7-1 and 7-2.

Figure 7-1: Internet Explorer accessing a website

I believe that the question to ask here is not so much “what a trusted application does?” as it is “what is the user really giving permission to when she permits her firewall to trust an application?” or “who and what is it that a user is trusting when doing so?”. Given that a significant source of attacks such as identity thefts and phishing are launched through applications such as peer-to-peer shareware clients and web browsers, such retrospection is critical. The focal contribution then, of FireViz, is compelling users to ask the right questions as a first step to understanding and improving security.
Figure 7-2: Yahoo Messenger signing in a user

7.2 Applications

The ideas implemented in FireViz can find applications in related fields:

- FireViz can be integrated with firewall software to provide users with a powerful framework of not merely visualizing but also managing their security. Not only will this integration benefit the development of the tool by eliminating various system-level design issues, it can significantly empower the usage of both FireViz and the firewall. Users will have the ability to identify security holes (with the use of FireViz) and be able to immediately close such holes by reconfiguring their firewall. They can also easily bias the visualization to their needs (such as only exceptional events) by specifying trusted connections (in the firewall) to not be displayed, thus allowing FireViz to provide a more optimistic visualization.
• The visualization tools in FireViz can be enhanced by adding intelligence to pick out patterns in traffic and help users identify suspect activity. To present a motivation for this, periodically repeated activities may be identified as malware traffic configured to trigger at a specific period of time. Thus visualization techniques in FireViz can be employed to differentiate between attacks and scans.

### 7.3 Future Work

FireViz presents a peripheral interface to effectively display network activities as seen by a host. This prototype can be enhanced and furthered in various ways:

#### 7.3.1 Complete Integration with a Firewall

A new iteration of FireViz should be developed alongside a firewall. Integrating FireViz with the firewall’s UI will help eliminate many of the implementation challenges presented in Chapter 5. This will also make it easier to deploy FireViz and collect feedback on the long term effectiveness of the visualization employed in FireViz. We also hope that this integration will make it easier for users to add new rules to their firewall and improve system security.

It is also our hope that such an integration will help further the development of next generation security software. By enabling users to see both their network behavior and the capabilities of their firewall, end users can find new areas for the development of security software that is more capable of protecting computers against their network-specific threats. This will enable a more user-centered iterated development cycle in the development of security tools.

#### 7.3.2 Context-Aware Activity Logs

The dimension of time can be more fully integrated in the visualization of FireViz. This could include animated logging features, such as described in VisFlowConnect
Another possibility for FireViz is to have visual (as opposed to textual) logs which incorporate screenshots of activities. As an example, a screen-shot of FireViz’s line display can clearly communicate to the user all relevant information about the connection while preserving the context (such as which website or piece of email the user was accessing at the time of the event).

Even the presentation of such logs proposes itself as an interesting design issue. Teoh et al. [33] present a focus + radial layout of log information to show events from contiguous days. Another interesting design issue here is to find effective means of storing such logs due to their potential of being storage-intensive.

7.3.3 Host Intrusion Detection with FireViz

FireViz’s goal is to provide information on network traffic as seen by a host and its visualization is geared towards meeting this goal. However, FireViz can be enhanced with pattern recognition to help identify known attack models or threats. Thus FireViz will not only provide the user with what is going on the network but also what some data on the network may mean.

7.3.4 Adapting FireViz to Multi-Screen Monitors

An interesting exercise could involve generalizing FireViz to multi-screen monitors. This will allow FireViz to utilize more screen estate or allow it to exclusively display certain kinds of events on certain screens. Multi-screen monitors also propose some interesting issues in the UI development. This may lead us to rethink how users focus their attention when working on such displays and how potentially large amounts of data can be displayed effectively yet non-intrusively.

7.3.5 Incorporating Geographical Location

Another future improvement opportunity for FireViz is to incorporate a geographical mapping or display and achieve the benefits discussed in Section 4.4. By employing
more reliable means of retrieving geographical data about network hosts, FireViz can provide more qualitative information about potential threats - not just in the domain of immediate network security from malicious hosts but also in the domain of phishing attacks.

### 7.3.6 Long-term User Testing

Last, but not the least, FireViz should continue to be extensively user tested. For a more objective evaluation of FireViz’s visualization, FireViz should be deployed for extended periods of time on the computers of a subset of the representative user population. We should also conduct controlled experiments to see if there was any perceptible change in the subjects’ security or network awareness after their interaction with FireViz. This will help us better evaluate the effectiveness of the visualization in both being non-intrusive and educative of the network security model.
Appendix A

User Study Briefing

The following briefing was provided to all users who evaluated the final prototype of FireViz.

Dear User,

This is a user testing for the UI of FireViz. FireViz is a personal firewall visualizing tool which works with a personal firewall and graphically presents network activity. It monitors all incoming and outgoing connections involving the host and shows them by displaying some activity around the periphery of the screen. The kind of activity displayed will depend upon the connection and other state on the computer. Additionally, the position of such activity depends on a network mapping on the screen periphery as defined to FireViz.

The idea of running FireViz is that it will provide greater transparency to the user in terms of the possible security threats on the network and the actions taken by the firewall.

The aim of this test is to evaluate the UI and find any problems with its design. You are free to stop the test at any time.
Bibliography

[1] Study: Consumers take cyberattacks lightly

[2] Nimda worm strikes Net, email

    http://securityresponse.symantec.com/avcenter/venc/data/w32.nimda.a@mm.html

    http://www.microsoft.com/security/incident/sasser.mspx

    http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm.html

    http://www3.gartner.com/research/focus_areas/asset_48267.jsp
    #Research

[7] InfoSec 2003: 'Zero-day' attacks seen as growing threat
    http://www.computerworld.com/securitytopics/security/story/0,10801,88109,00.html?SKC=news88109

[8] Study: Unpatched PCs compromised in 20 minutes
[9] ZoneAlarm Security Suite
   http://www.zonelabs.com

[10] Kerio Personal Firewall
   http://www.kerio.com/us/kpf_home.html


[12] AOL/NCSA Online Safety Study


[15] Norton Personal Firewall
   http://www.symantec.com/sabu/nis/npf/

[16] Microsoft Windows XP - Netstat

[17] Sysinternals TCPView
   http://www.sysinternals.com/ntw2k/source/tcpview.shtml


[19] Cisco Trellis IDS


[23] W. Yurcik, J. Barlow, K. Lakkaraju, and J. Rosendale. A prototype tool for visual data mining of network traffic for intrusion detection. 3rd *IEEE International Conference on Data Mining (ICDM), Workshop on Data Mining for Computer Security (DMSEC)*, 2003

   http://winpcap.polito.it

[25] Libpcap Packet Capture
   http://sourceforge.net/projects/libpcap/

[26] Network Intrusion Detection Systems
   http://www.digitaldefense.net/white_papers/nids.html


[38] Whois Service.
http://www.whois.net/


[40] Netcraft.
http://toolbar.netcraft.com

[41] Google GeoDisplay.

94

    http://www.liutilities.com/products/wintaskspro/dlllibrary/iphlpapi